



UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

H.264

(05/2003)

SÉRIE H: SYSTÈMES AUDIOVISUELS ET
MULTIMÉDIAS

Infrastructure des services audiovisuels – Codage des
images vidéo animées

**Codage vidéo évolué pour les services
audiovisuels génériques**

Recommandation UIT-T H.264

RECOMMANDATIONS UIT-T DE LA SÉRIE H
SYSTÈMES AUDIOVISUELS ET MULTIMÉDIAS

CARACTÉRISTIQUES DES SYSTÈMES VISIOPHONIQUES	H.100–H.199
INFRASTRUCTURE DES SERVICES AUDIOVISUELS	
Généralités	H.200–H.219
Multiplexage et synchronisation en transmission	H.220–H.229
Aspects système	H.230–H.239
Procédures de communication	H.240–H.259
Codage des images vidéo animées	H.260–H.279
Aspects liés aux systèmes	H.280–H.299
Systèmes et équipements terminaux pour les services audiovisuels	H.300–H.349
Architecture des services d'annuaire pour les services audiovisuels et multimédias	H.350–H.359
Architecture de la qualité de service pour les services audiovisuels et multimédias	H.360–H.369
Services complémentaires en multimédia	H.450–H.499
PROCÉDURES DE MOBILITÉ ET DE COLLABORATION	
Aperçu général de la mobilité et de la collaboration, définitions, protocoles et procédures	H.500–H.509
Mobilité pour les systèmes et services multimédias de la série H	H.510–H.519
Applications et services de collaboration multimédia mobile	H.520–H.529
Sécurité pour les systèmes et services multimédias mobiles	H.530–H.539
Sécurité pour les applications et services de collaboration multimédia mobile	H.540–H.549
Procédures d'interfonctionnement de la mobilité	H.550–H.559
Procédures d'interfonctionnement de collaboration multimédia mobile	H.560–H.569
SERVICES À LARGE BANDE ET MULTIMÉDIAS TRI-SERVICES	
Services multimédias à large bande sur VDSL	H.610–H.619

Pour plus de détails, voir la Liste des Recommandations de l'UIT-T.

Recommandation UIT-T H.264

Codage vidéo évolué pour les services audiovisuels génériques

Résumé

Alors qu'ont diminué les coûts aussi bien de la puissance de traitement que de la mémoire, que se sont diversifiés les réseaux de traitement des données vidéo codées, et qu'ont progressé les avancées de la technique du codage vidéo, est apparu le besoin d'une norme industrielle pour la représentation vidéo compressée avec une efficacité de codage substantiellement augmentée et une robustesse améliorée aux environnements de réseau.

La présente Recommandation représente une évolution des normes de codage vidéo existantes (H.261, H.262 et H.263) et elle a été développée pour répondre à un besoin croissant de plus forte compression des images animées pour diverses applications telles que la visioconférence, les moyens de stockage numérique, la diffusion télévisuelle, le trafic Internet et la communication. Elle est aussi conçue pour permettre une utilisation souple de la représentation vidéo codée dans une large gamme d'environnements de réseaux. L'utilisation de la présente Recommandation permet de manipuler la vidéo animée comme des données informatiques et de la mémoriser sur des moyens de stockage variés, de la transmettre et de la recevoir sur les réseaux existants et futurs et de la distribuer sur les canaux de diffusion actuels et de l'avenir.

Source

La Recommandation UIT-T H.264 a été approuvée le 30 mai 2003 par la Commission d'études 16 (2001-2004) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8.

Cette édition comporte les modifications introduites par le Corrigendum 1 de la Rec. UIT-T H.264 (2003) approuvé par la Commission d'études 16 (2001-2004) de l'UIT-T selon la procédure définie dans la Recommandation UIT-T A.8 le 7 mai 2004.

AVANT-PROPOS

L'UIT (Union internationale des télécommunications) est une institution spécialisée des Nations Unies dans le domaine des télécommunications. L'UIT-T (Secteur de la normalisation des télécommunications) est un organe permanent de l'UIT. Il est chargé de l'étude des questions techniques, d'exploitation et de tarification, et émet à ce sujet des Recommandations en vue de la normalisation des télécommunications à l'échelle mondiale.

L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, détermine les thèmes d'étude à traiter par les Commissions d'études de l'UIT-T, lesquelles élaborent en retour des Recommandations sur ces thèmes.

L'approbation des Recommandations par les Membres de l'UIT-T s'effectue selon la procédure définie dans la Résolution 1 de l'AMNT.

Dans certains secteurs des technologies de l'information qui correspondent à la sphère de compétence de l'UIT-T, les normes nécessaires se préparent en collaboration avec l'ISO et la CEI.

NOTE

Dans la présente Recommandation, l'expression "Administration" est utilisée pour désigner de façon abrégée aussi bien une administration de télécommunications qu'une exploitation reconnue.

Le respect de cette Recommandation se fait à titre volontaire. Cependant, il se peut que la Recommandation contienne certaines dispositions obligatoires (pour assurer, par exemple, l'interopérabilité et l'applicabilité) et considère que la Recommandation est respectée lorsque toutes ces dispositions sont observées. Le futur d'obligation et les autres moyens d'expression de l'obligation comme le verbe "devoir" ainsi que leurs formes négatives servent à énoncer des prescriptions. L'utilisation de ces formes ne signifie pas qu'il est obligatoire de respecter la Recommandation.

DROITS DE PROPRIÉTÉ INTELLECTUELLE

L'UIT attire l'attention sur la possibilité que l'application ou la mise en œuvre de la présente Recommandation puisse donner lieu à l'utilisation d'un droit de propriété intellectuelle. L'UIT ne prend pas position en ce qui concerne l'existence, la validité ou l'applicabilité des droits de propriété intellectuelle, qu'ils soient revendiqués par un Membre de l'UIT ou par une tierce partie étrangère à la procédure d'élaboration des Recommandations.

A la date d'approbation de la présente Recommandation, l'UIT [avait/n'avait pas] été avisée de l'existence d'une propriété intellectuelle protégée par des brevets à acquérir pour mettre en œuvre la présente Recommandation. Toutefois, comme il ne s'agit peut-être pas de renseignements les plus récents, il est vivement recommandé aux responsables de la mise en œuvre de consulter la base de données des brevets du TSB.

© UIT 2005

Tous droits réservés. Aucune partie de cette publication ne peut être reproduite, par quelque procédé que ce soit, sans l'accord écrit préalable de l'UIT.

TABLE DES MATIÈRES

		Page
0	Introduction	xi
	0.1 Prologue.....	xi
	0.2 Objet	xi
	0.3 Applications.....	xi
	0.4 Profils et niveaux.....	xii
	0.5 Aperçu général sur les caractéristiques de base	xii
	0.6 Comment lire la présente spécification	xiv
1	Domaine d'application	1
2	Références normatives.....	1
3	Définitions	1
4	Abréviations.....	11
5	Conventions	12
	5.1 Opérateurs arithmétiques.....	12
	5.2 Opérateurs logiques	13
	5.3 Opérateurs relationnels.....	13
	5.4 Opérateurs binaires.....	13
	5.5 Opérateurs d'affectation.....	14
	5.6 Notation de gamme.....	14
	5.7 Fonctions mathématiques	14
	5.8 Variables, éléments syntaxiques et tableaux	15
	5.9 Description textuelle des opérations logiques	16
	5.10 Processus	17
6	Formats de données source, codés, décodés et de sortie, processus de balayage et relations de voisinage	17
	6.1 Formats de flux binaires	17
	6.2 Formats d'image source, décodés et de sortie.....	18
	6.3 Subdivision spatiale des images et des tranches.....	20
	6.4 Processus de balayage inverse et processus de déduction pour le voisinage ..	21
7	Syntaxe et sémantique	33
	7.1 Méthode de description de la syntaxe en forme tabulaire	33
	7.2 Spécification de fonctions, catégories et descripteurs syntaxiques	35
	7.3 Syntaxe en forme tabulaire.....	37
	7.4 Sémantique	51
8	Processus de décodage.....	95
	8.1 Processus de décodage d'unité NAL	96
	8.2 Processus de décodage de tranche.....	97
	8.3 Processus d'intraprédiction	121
	8.4 Processus d'interprédiction	133

	Page	
8.5	Processus de décodage du coefficient de transformée et processus de construction d'image antérieurement au processus de filtrage de démontage de blocs.....	159
8.6	Processus de décodage pour macroblocs P dans des tranches SP ou des macroblocs SI.....	167
8.7	Processus de filtrage de démontage de blocs	173
9	Processus d'analyse grammaticale.....	184
9.1	Processus d'analyse grammaticale pour les codes Exp-Golomb.....	184
9.2	Processus d'analyse grammaticale CAVLC pour les niveaux de coefficient de transformée.....	188
9.3	Processus d'analyse grammaticale CABAC pour données de tranche.....	195
Annexe A – Profils et niveaux.....		239
A.1	Exigences sur la capacité du décodeur vidéo.....	239
A.2	Profils.....	239
A.3	Niveaux.....	241
Annexe B – Format de flux d'octets.....		251
B.1	Syntaxe et sémantique d'unité NAL de flux d'octet.....	251
B.2	Processus de décodage d'unité NAL de flux d'octet.....	252
B.3	Récupération du verrouillage d'octet du décodeur (pour information).....	253
Annexe C – Décodeur fictif de référence.....		254
C.1	Fonctionnement de la mémoire tampon d'image codée (CPB).....	257
C.2	Fonctionnement de la mémoire tampon de l'image décodée (DPB).....	259
C.3	Conformité de flux binaire.....	261
C.4	Conformité du décodeur.....	263
Annexe D – Informations d'amélioration supplémentaires.....		268
D.1	Syntaxe de la charge utile SEI.....	269
D.2	Sémantique de charge utile SEI.....	277
Annexe E – Informations d'utilisation vidéo (VUI).....		300
E.1	Syntaxe VUI.....	301
E.2	Sémantique VUI.....	302

LISTE DES FIGURES

	Page
Figure 6-1 – Localisations nominales verticales et horizontales d'échantillons 4:2:0 luma et chroma dans une trame	19
Figure 6-2 – Localisations d'échantillonnage nominales verticales et horizontales des échantillons des champs supérieur et inférieur.....	20
Figure 6-3 – Image avec 11 sur 9 macroblocs qui est subdivisée en deux tranches.....	20
Figure 6-4 – Subdivision de la trame décodée en paires de macroblocs	21
Figure 6-5 – Divisions de macrobloc, divisions de sous-macrobloc, balayages de division de macrobloc et balayages de division de sous-macrobloc	22
Figure 6-6 – Balayage pour les blocs luma 4x4.....	23
Figure 6-7 – Macroblocs voisins pour un macrobloc donné.....	24
Figure 6-8 – Macroblocs du voisinage pour un macrobloc donné dans des trames MBAFF..	25
Figure 6-9 – Détermination des macroblocs, blocs et subdivisions du voisinage (pour information).....	26
Figure 7-1 – Structure d'une unité d'accès ne contenant aucune unité NAL avec nal_unit_type égal à 0, 7, 8 ou dans la gamme de 12 à 31 inclus.....	59
Figure 8-1 – Directions du mode de prédiction Intra_4x4 (pour information).....	122
Figure 8-2 – Exemple de déduction de vecteur cinétique temporel en mode direct (pour information).....	144
Figure 8-3 – Prédiction de segmentation directionnelle (pour information)	145
Figure 8-4 – Echantillons entiers (blocs ombrés avec lettres majuscules) et positions d'échantillons fractionnaires (blocs non ombrés avec lettres minuscules) pour interpolation de quarts d'échantillons luma.	152
Figure 8-5 – Variables selon la position d'échantillon fractionnaire en interpolation chroma avec les échantillons A, B, C, et D de position entière qui les entourent.....	155
Figure 8-6 – Allocation des indices de dcY à luma4x4BlkIdx	160
Figure 8-7 – Allocation des indices de dcC à chroma4x4BlkIdx	161
Figure 8-8 – a) Balayage en zigzag. b) Balayage de champ.....	162
Figure 8-9 – Frontières dans un macrobloc à filtrer (frontières luma montrées avec lignes continues et frontières chroma avec des pointillés).....	173
Figure 8-10 – Convention pour décrire les échantillons sur une frontière de bloc 4x4 horizontale ou verticale.....	177
Figure 9-1 – Illustration du processus d'analyse grammaticale CABAC pour un élément syntaxique SE (pour information)	197
Figure 9-2 – Vue générale du processus de décodage arithmétique pour un seul segment (pour information)	226
Figure 9-3 – Diagramme pour le décodage d'une décision.....	228
Figure 9-4 – Diagramme de renormalisation	230
Figure 9-5 – Diagramme du processus de décodage d'aiguillage.....	231
Figure 9-6 – Diagramme de décodage d'une décision avant achèvement	232

	Page
Figure 9-7 – Diagramme pour le codage d'une décision.....	234
Figure 9-8 – Diagramme de la renormalisation dans le codeur	235
Figure 9-9 – Diagramme de PutBit(B).....	235
Figure 9-10 – Diagramme de l'aiguillage de codage	236
Figure 9-11 – Diagramme de codage d'une décision avant achèvement	237
Figure 9-12 – Diagramme du vidage à achèvement	237
Figure C-1 – Structure des flux d'octets et des flux d'unités NAL pour les vérifications de conformité par le HRD	254
Figure C-2 – Modèle de mémoire tampon de HRD.....	256
Figure E-1 – Localisation des échantillons chroma pour les champs supérieurs et inférieurs en fonction de chroma_sample_loc_type_top_field et chroma_sample_loc_type_bottom_field.....	308

LISTE DES TABLEAUX

	Page
Tableau 6-1 – Valeurs de ChromaFormatFactor.....	18
Tableau 6-2 – Spécification des allocations d'entrée et de sortie pour les § 6.4.7.1 à 6.4.7.5 .	25
Tableau 6-3 – Spécification de mbAddrN	30
Tableau 6-4 – Spécification de mbAddrN et yM.....	32
Tableau 7-1 – Codes de type d'unité NAL.....	53
Tableau 7-2 – Signification de primary_pic_type.....	67
Tableau 7-3 – Association du nom au slice_type	70
Tableau 7-4 – Opérations de reordering_of_pic_nums_idc pour la remise en ordre des listes d'images de référence	77
Tableau 7-5 – Interprétation de adaptive_ref_pic_marking_mode_flag.....	80
Tableau 7-6 – Valeurs d'opération de commande de gestion de mémoire (memory_management_control_operation)	81
Tableau 7-7 – Types collectifs de macrobloc admis pour slice_type	84
Tableau 7-8 – Types de macrobloc pour tranches I.....	85
Tableau 7-9 – Type de macrobloc avec la valeur 0 pour les tranches SI.....	86
Tableau 7-10 – Valeurs de 0 à 4 de type de macrobloc pour les tranches P et SP	87
Tableau 7-11 – Valeurs de type de macrobloc de 0 à 22 pour tranches B.....	88
Tableau 7-12 – Spécification des valeurs de CodedBlockPatternChroma	90
Tableau 7-13 – Relations entre intra_chroma_pred_mode et modes de prédiction spatiale ..	90
Tableau 7-14 – Types de sous-macrobloc dans les macroblocs P	91
Tableau 7-15 – Types de sous-macrobloc dans les macroblocs B.....	92
Tableau 8-1 – Type précis de correspondance de groupe de tranches.....	102
Tableau 8-2 – Spécification de Intra4x4PredMode[luma4x4BlkIdx] et des noms associés...	122
Tableau 8-3 – Spécification de Intra16x16PredMode et des noms associés	128
Tableau 8-4 – Spécification des modes d'intraprédiction chroma et des noms associés	130
Tableau 8-5 – Spécification de la variable colPic.....	138
Tableau 8-6 – Spécification de PicCodingStruct(X).....	138
Tableau 8-7 – Spécification de mbAddrCol, yM et vertMvScale.....	139
Tableau 8-8 – Allocation des fanions d'utilisation de prédiction.....	141
Tableau 8-9 – Déduction de la composante verticale du vecteur chroma dans le mode de codage de champ.....	148
Tableau 8-10 – Localisation différentielle d'échantillons luma entiers	152
Tableau 8-11 – Allocation de l'échantillon de prédiction luma predPartLX _L [x _L , y _L].....	154
Tableau 8-12 – Spécification de la correspondance des indices à c _{ij} pour balayage en zigzag et de champ.....	162
Tableau 8-13 – Spécification de QP _C en fonction de qP _I	163

Tableau 8-14 – Calcul de indexA et indexB à partir des variables de seuil dépendantes du décalage α et β	181
Tableau 8-15 – Valeur de la variable d'écrêtage de filtre t_{c0} en fonction de indexA et bS	183
Tableau 9-1 – Chaînes binaires avec des bits de "préfixe" et de "suffixe" et allocation aux gammes de codeNum (pour information).....	185
Tableau 9-2 – Chaînes binaires Exp-Golomb et codeNum en forme explicite et utilisées comme $ue(v)$ (pour information)	185
Tableau 9-3 – Allocation d'élément syntaxique à codeNum pour les éléments syntaxiques codés en Exp-Golomb algébrique $se(v)$	186
Tableau 9-4 – Allocation de codeNum aux valeurs de $\text{coded_block_pattern}$ pour les modes de prédiction de macrobloc.....	186
Tableau 9-5 – Correspondance de coeff_token avec $\text{TotalCoeff}(\text{coeff_token})$ et $\text{TrailingOnes}(\text{coeff_token})$	190
Tableau 9-6 – Tableau des mots de code pour level_prefix	193
Tableau 9-7 – Tableaux de total_zeros pour blocs 4x4 avec $\text{TotalCoeff}(\text{coeff_token})$ de 1 à 7.....	194
Tableau 9-8 – Tableaux de total_zeros pour blocs 4x4 avec $\text{TotalCoeff}(\text{coeff_token})$ de 8 à 15.....	194
Tableau 9-9 – Tableaux de total_zeros pour blocs chroma DC 2x2.....	195
Tableau 9-10 – Tableaux pour run_before	195
Tableau 9-11 – Association des ctxIdx et des éléments syntaxiques pour chaque type de tranche dans le processus d'initialisation	199
Tableau 9-12 – Valeurs des variables m et n pour ctxIdx de 0 à 10	199
Tableau 9-13 – Valeurs des variables m et n pour ctxIdx de 11 à 23	200
Tableau 9-14 – Valeurs des variables m et n pour ctxIdx de 24 à 39	200
Tableau 9-15 – Valeurs des variables m et n pour ctxIdx de 40 à 53	200
Tableau 9-16 – Valeurs des variables m et n pour ctxIdx de 54 à 59	200
Tableau 9-17 – Valeurs des variables m et n pour ctxIdx de 60 à 69	201
Tableau 9-18 – Valeurs des variables m et n pour ctxIdx de 70 à 104.....	201
Tableau 9-19 – Valeurs des variables m et n pour ctxIdx de 105 à 165	202
Tableau 9-20 – Valeurs des variables m et n pour ctxIdx de 166 à 226	203
Tableau 9-21 – Valeurs des variables m et n pour ctxIdx de 227 à 275	204
Tableau 9-22 – Valeurs des variables m et n pour ctxIdx de 277 à 337	205
Tableau 9-23 – Valeurs des variables m et n pour ctxIdx de 338 à 398	206
Tableau 9-24 – Eléments syntaxiques et types de binarisation associés, maxBinIdxCtx et ctxIdxOffset	208
Tableau 9-25 – Chaîne de segments de la binarisation unaire (pour information).....	209
Tableau 9-26 – Binarisation pour les types de macrobloc dans les tranches I	212
Tableau 9-27 – Binarisation pour les types de macrobloc dans les tranches P, SP, et B.....	213

Tableau 9-28 – Binairisation pour les types de sous-macrobloc dans les tranches P, SP, et B.....	214
Tableau 9-29 – Allocation de ctxIdxInc à binIdx pour toutes les valeurs de ctxIdxOffset excepté celles se rapportant aux éléments syntaxiques coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1	217
Tableau 9-30 – Allocation de ctxIdxBlockCatOffset à ctxBlockCat pour les éléments syntaxiques coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1	218
Tableau 9-31 – Spécification de ctxIdxInc pour des valeurs spécifiques de ctxIdxOffset et binIdx.....	225
Tableau 9-32 – Spécification de ctxBlockCat pour les différents blocs.....	225
Tableau 9-33 – Spécification de rangeTabLPS en fonction de pStateIdx et qCodIRangeIdx.	229
Tableau 9-34 – Tableau de transition d'état	230
Tableau A-1 – Limites de niveau.....	243
Tableau A-2 – Limites de niveau du profil de base	245
Tableau A-3 – Limites de niveau du profil principal.....	245
Tableau A-4 – Limites de niveau du profil étendu	246
Tableau A-5 – Débits de trame maximums (trames par seconde) pour quelques exemples de tailles de trame	247
Tableau D-1 – Interprétation de pic_struct	279
Tableau D-2 – Correspondance de ct_type avec le balayage de l'image source.....	280
Tableau D-3 – Définition des valeurs de counting_type.....	280
Tableau D-4 – Valeurs de scene_transition_type	290
Tableau E-1 – Signification de l'indicateur de ratio d'aspect d'échantillon	303
Tableau E-2 – Signification de video_format.....	304
Tableau E-3 – Couleurs primaires	305
Tableau E-4 – Caractéristiques de transfert	306
Tableau E-5 – Coefficients matriciels.....	307
Tableau E-6 – Diviseur pour le calcul de $\Delta t_{fi,dpb}(n)$	309

Avant-propos

L'Union internationale des télécommunications (UIT) est l'agence spécialisée des Nations Unies dans le domaine des télécommunications. Le Secteur de la normalisation des télécommunications de l'UIT (UIT-T) est un organe permanent de l'UIT. L'UIT-T est responsable de l'étude des questions techniques, de fonctionnement et de tarification et de la production de Recommandations sur ces questions dans le but de normaliser les télécommunications au plan mondial. L'Assemblée mondiale de normalisation des télécommunications (AMNT), qui se réunit tous les quatre ans, arrête les sujets d'étude des Commissions d'études de l'UIT-T, qui à leur tour produisent des Recommandations sur ces sujets. L'approbation des Recommandations de l'UIT-T est traitée selon les procédures établies dans la Résolution 1 de l'AMNT. Dans certains domaines des technologies de l'information qui tombent dans le champ de compétence de l'UIT-T, les normes nécessaires sont préparées en collaboration avec l'ISO et la CEI.

L'ISO (Organisation internationale de normalisation) et la CEI (Comité électrotechnique international) constituent le système spécialisé de la normalisation mondiale. Les entités nationales qui sont membres de l'ISO et de la CEI participent au développement des normes internationales dans les comités techniques établis par les organisations qui traitent respectivement des domaines particuliers d'activité technique. Les comités techniques de l'ISO et de la CEI collaborent dans les domaines d'intérêt mutuel. D'autres organisations internationales, gouvernementales et non gouvernementales, participent également aux travaux, en liaison avec l'ISO et la CEI. Dans le domaine des technologies de l'information, l'ISO et la CEI ont constitué un comité technique conjoint, le JTC1 ISO/CEI. Les projets de norme internationale adoptés par le comité technique conjoint sont soumis à l'approbation des organismes nationaux. La publication comme norme internationale exige l'approbation par au moins 75% des organismes nationaux ayant exprimé leur voix.

La présente Recommandation | Norme internationale a été préparée conjointement par le Groupe de travail 16 Q.6 de l'UIT-T, connu aussi sous le nom de Groupe d'experts de codage vidéo (VCEG, *video coding experts group*), et par le groupe de travail ISO/CEI JTC1/SC29/WG11, connu aussi sous le nom de Groupe d'experts pour les images animées (MPEG, *moving picture experts group*). Le VCEG a été constitué en 1997 pour assurer la maintenance des anciennes normes de codage vidéo de l'UIT-T et développer une ou plusieurs nouvelles normes de codage vidéo convenables pour une large gamme de services conversationnels ou non conversationnels. Le MPEG a été constitué en 1988 pour établir des normes de codage des images animées et des sons qui leur sont associés pour diverses applications telles que les moyens de stockage numériques, la distribution et la communication.

Dans la présente Recommandation | Norme internationale, les Annexes A à E contiennent des exigences normatives et constituent une partie intégrante de la présente Recommandation | Norme internationale.

0 Introduction

Le présent article ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

0.1 Prologue

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

Alors qu'ont diminué les coûts aussi bien de la puissance de traitement que de la mémoire, que se sont diversifiés les réseaux de traitement des données vidéo codées, et qu'ont progressé les avancées de la technique du codage vidéo, est apparu le besoin d'une norme industrielle pour la représentation vidéo compressée avec une efficacité de codage substantiellement augmentée et une robustesse améliorée aux environnements de réseau. A cette fin, le Groupe d'experts de codage vidéo de l'UIT-T (VCEG) et le Groupe d'experts pour les images animées de l'ISO/CEI (MPEG) ont formé une équipe vidéo conjointe (JVT, *joint video team*) en 2001 pour le développement d'une nouvelle Recommandation | Norme internationale.

0.2 Objet

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

La présente Recommandation | Norme internationale a été élaborée pour répondre au besoin croissant d'une plus forte compression des images animées pour diverses applications comme la visioconférence, les moyens de stockage numériques, la radiodiffusion télévisuelle, le trafic Internet et la communication. Elle est aussi conçue pour permettre l'utilisation de la représentation vidéo codée de façon souple dans une large variété d'environnements de réseaux. L'utilisation de la présente Recommandation | Norme internationale permet de manipuler la vidéo animée comme une forme de données informatiques et de la mémoriser sur des moyens de stockage variés, de la transmettre et de la recevoir sur les réseaux existants et du futur et de la distribuer sur les canaux de radiodiffusion existants et de l'avenir.

0.3 Applications

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

La présente Recommandation | Norme internationale est destinée à couvrir une large gamme d'applications contenant de la vidéo, y compris les suivantes, mais sans exclusive:

- CATV Télévision par câble sur réseaux optiques, cuivre, etc. (*cable television*)
- DBS Services vidéo de diffusion directe par satellite (*direct broadcast satellite*)
- DSL Services vidéo sur ligne d'abonné numérique (*digital subscriber line*)
- DTTB Radiodiffusion télévisuelle numérique par voie hertzienne de Terre (*digital terrestrial television broadcasting*)
- ISM Support d'enregistrement interactif (disque optique, etc.) (*interactive storage media*)
- MMM Messagerie multimédia
- MSPN Services multimédias sur réseaux par paquets (*multimedia services over packet networks*)
- RTC Services conversationnels en temps réel (visioconférence, visiophone, etc.) (*real-time conversational services*)

RVS Télévidéosurveillance (*remote video surveillance*)

SSM Support d'enregistrement en série (VTR numérique, etc.) (*serial storage media*)

0.4 Profils et niveaux

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

La présente Recommandation | Norme internationale est conçue pour être générique en ce sens qu'elle sert à une large gamme d'applications, de débits binaires, de résolutions, de qualités et de services. Les applications devraient couvrir, entre autres, les supports d'enregistrement numériques, la radiodiffusion télévisuelle et les communications en temps réel. En créant la présente Spécification, des exigences variées d'applications typiques ont été prises en compte, les éléments algorithmiques nécessaires ont été développés et ont été intégrés en une syntaxe unique. Ainsi, la présente Spécification facilitera les échanges de données vidéo entre les différentes applications.

Cependant, en considérant l'implémentation pratique de la totalité de la syntaxe de la présente Spécification, un nombre limité de sous-ensembles de la syntaxe sont également stipulés au moyen de "profils" et de "niveaux". Ceux-ci et d'autres termes s'y rapportant sont formellement définis au § 3.

Un "profil" est un sous-ensemble de la syntaxe entière du flux binaire qui est spécifiée dans la présente Recommandation | Norme internationale. Dans les limites imposées par la syntaxe d'un profil donné, il est toujours possible d'exiger une très grande variation des performances des codeurs et décodeurs selon les valeurs prises par les éléments de la syntaxe dans le flux binaire, tels que la taille spécifique des images décodées. Dans de nombreuses applications, il n'est habituellement ni pratique ni économique d'implémenter un décodeur capable de traiter toutes les utilisations possibles de la syntaxe au sein d'un profil particulier.

Afin de régler ce problème, des "niveaux" sont spécifiés au sein de chaque profil. Un niveau est un ensemble spécifié de contraintes imposées aux valeurs des éléments syntaxiques dans le flux binaire. Ces contraintes peuvent être de simples limites des valeurs. Elles peuvent aussi prendre la forme de contraintes sur les combinaisons arithmétiques des valeurs (par exemple, une largeur d'image multipliée par une hauteur d'image multipliée par le nombre d'images décodées par seconde).

Les contenus de vidéo codés conformes à la présente Recommandation | Norme internationale utilisent une syntaxe commune. Afin de réaliser un sous-ensemble de la syntaxe complète, des fanions, des paramètres et autres éléments syntaxiques sont inclus dans le flux binaire, qui signalent la présence ou l'absence des éléments syntaxiques qui surviennent ultérieurement dans le flux binaire.

0.5 Aperçu général sur les caractéristiques de base

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

La représentation codée spécifiée dans la syntaxe est conçue pour permettre une forte capacité de compression pour une qualité d'image désirée. L'algorithme n'est pas sans perte, car les valeurs exactes d'échantillonnage de source ne sont pas en principe conservées pendant les processus de codage et de décodage. On peut utiliser un certain nombre de techniques pour réaliser une compression de haute efficacité. Les algorithmes de codage (qui ne sont pas spécifiés dans la présente Recommandation | Norme internationale) peuvent choisir entre l'intercodage et l'intracodage pour les parties structurales de chaque image. L'intercodage utilise des vecteurs cinétiques pour la prédiction interne fondée sur le bloc pour exploiter les dépendances temporelles statistiques entre différentes images. L'intracodage utilise différents modes de prédiction spatiale pour exploiter les dépendances statistiques spatiales dans le signal source pour une image unique.

Les vecteurs cinétiques et les modes d'intraprédiction peuvent être spécifiés pour des tailles de bloc variées dans l'image. La prédiction résiduelle est alors encore compressée en utilisant une transformée pour éliminer la corrélation spatiale à l'intérieur du bloc transformé avant qu'il ne soit quantifié, ce qui produit un processus irréversible qui élimine en principe les informations visuelles les moins importantes lorsqu'est formée une approximation fidèle des échantillons de la source. Finalement, les vecteurs cinétiques ou les modes d'intraprédiction sont combinés avec les informations de coefficient de transformé quantifiées et encodées en utilisant des codes à longueur variable ou un codage arithmétique.

0.5.1 Codage prédictif

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

A cause d'exigences contradictoires d'accès aléatoire et de compression à haute performance, deux types principaux de codage sont spécifiés. L'intracodage est fait sans référence aux autres images. L'intracodage peut fournir des points d'accès à la séquence codée lorsque le décodage peut commencer et se poursuivre correctement, mais en principe, il ne donne qu'une efficacité de compression modérée. L'intercodage (prédictif ou biprédictif) est plus efficace car il utilise l'interprédiction pour chaque bloc de valeurs d'échantillon à partir de certaines images précédemment décodées choisies par le codeur. A la différence de certaines autres normes de codage vidéo, les images codées à l'aide de l'interprédiction biprédictive peuvent aussi être utilisées comme références pour l'intercodage d'autres images.

L'application des trois types de codage aux images d'une séquence est souple, et l'ordre du processus de décodage n'est généralement pas le même que le processus de capture de l'image source dans le codeur ou l'ordre de sortie du décodeur pour l'affichage. Le choix est laissé au codeur et dépendra des exigences de l'application. L'ordre de décodage est spécifié de telle façon que le décodage des images qui utilise la prédiction interimage suive plus tard dans l'ordre de décodage les autres images qui sont référencées dans le processus de décodage.

0.5.2 Codage de vidéos progressives et entrelacées

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

La présente Recommandation | Norme internationale spécifie une syntaxe et un processus de décodage pour les vidéos qui ont été générées par balayage progressif ou entrelacé, qui peuvent être entremêlés dans la même séquence. Les deux champs d'une trame entrelacée sont séparés au moment de la capture, tandis que les deux champs d'une trame progressive partagent le même temps de capture. Chaque champ peut être codé séparément ou bien les deux champs peuvent être codés ensemble comme une trame. Les trames progressives sont en général codées comme une trame. Pour la vidéo entrelacée, le codeur peut choisir entre le codage de trame et le codage de champ. Le codage de trame ou le codage de champ peuvent être choisis de façon adaptative image par image et aussi sur une base plus locale au sein d'une trame codée. Le codage de trame est généralement préféré lorsque la scène vidéo contient des détails significatifs avec un mouvement limité. Le codage de champ fonctionne généralement mieux lorsqu'il y a un mouvement rapide d'image à image.

0.5.3 Partage d'image en macroblocs et en partitions plus petites

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

Comme dans les Recommandations et Normes internationales antérieures sur le codage vidéo, un macrobloc, consistant en un bloc d'échantillons luma de 16x16 et deux blocs correspondants

d'échantillons chroma, est utilisé comme unité de base de traitement du processus de décodage vidéo.

Un macrobloc peut encore être partagé pour l'interprédiction. Le choix de la taille des partitions d'interprédiction est le résultat d'un compromis entre le gain du codage procuré par l'utilisation de la compensation du mouvement avec de plus petits blocs et la quantité de données nécessaires pour représenter les données pour la compensation du mouvement. Dans la présente Recommandation | Norme internationale, le processus d'interprédiction peut former des segmentations pour une représentation du mouvement aussi petites en taille que des échantillons luma de 4 x 4, en utilisant une précision de vecteur cinétique d'un quart du déplacement spatial de la grille de l'échantillon luma. Le processus d'interprédiction d'un bloc d'échantillon peut aussi impliquer le choix d'une image à utiliser comme image de référence pour un certain nombre d'images mémorisées précédemment décodées. Les vecteurs cinétiques sont codés de façon différentielle par rapport aux valeurs prévues formées à partir des vecteurs cinétiques codés proches.

Généralement, le codeur calcule les vecteurs de mouvement appropriés et les autres éléments de données représentés dans le flux de données vidéo. Ce processus d'estimation du mouvement dans le codeur et le choix entre l'utilisation de l'interprédiction pour la représentation de chaque région du contenu de la vidéo n'est pas spécifié dans la présente Recommandation | Norme internationale.

0.5.4 Réduction de la redondance spatiale

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

Les images sources et les résidus de prédiction ont tous deux une forte redondance spatiale. La présente Recommandation | Norme internationale se fonde sur l'utilisation de la méthode de la transformée à base de bloc pour supprimer la redondance spatiale. Après l'interprédiction à partir d'échantillons décodés précédemment dans d'autres images ou la prédiction fondée sur l'espace à partir d'échantillons précédemment décodés au sein de l'image en cours, le résidu de prédiction résultant est partagé en blocs de 4x4. Ceux-ci sont convertis dans le domaine de transformation où ils sont quantifiés. Après quantification, de nombreux coefficients de transformation sont des zéros et ont une faible amplitude. Ils peuvent alors être représentés par un faible nombre de données codées. Le processus de transformation et de quantification dans le codeur n'est pas spécifié dans la présente Recommandation | Norme internationale.

0.6 Comment lire la présente spécification

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

On suggère que le lecteur commence à l'article 1 (Domaine d'application) et passe à l'article 3 (Définitions). L'article 6 comporte les relations géométriques de la source, l'entrée et la sortie du décodeur. L'article 7 (Syntaxe et sémantique) spécifie l'ordre de l'analyse grammaticale des éléments syntaxiques dans le flux de données. Se reporter aux § 7.1 à 7.3 pour l'ordre syntaxique et au § 7.4 pour la sémantique; c'est-à-dire, la portée, les interdictions et les conditions qui sont exigées pour les éléments syntaxiques. L'analyse grammaticale réelle pour la plupart des éléments syntaxiques est spécifiée à l'article 9 (Processus d'analyse grammaticale). Finalement, l'article 8 (Processus de décodage) spécifie comment les éléments syntaxiques sont mappés dans les échantillons décodés. Tout au long de la lecture de la présente spécification, le lecteur devrait se référer aux articles 2 (Références normatives), 4 (Abréviations), et 5 (Conventions) en tant que de besoin. Les Annexes A à E font également partie intégrante de la présente Recommandation | Norme internationale.

L'Annexe A définit trois profils (de base, principal et étendu), chacun d'eux étant adapté à certains domaines d'application, et elle définit ce qu'on appelle les niveaux des profils. L'Annexe B spécifie la syntaxe et la sémantique d'un format de flux d'octets pour la fourniture de la vidéo codée en flux

ordonné d'octets. L'Annexe C spécifie le décodeur de référence type et son utilisation pour vérifier la conformité du flux binaire et du décodeur. L'Annexe D spécifie la syntaxe et la sémantique pour les charges utiles, le message d'information ayant des améliorations supplémentaires. Finalement, l'Annexe E spécifie la syntaxe et la sémantique des paramètres d'information d'utilisation de la vidéo de l'ensemble de paramètres de la séquence.

Tout au long de la présente spécification, les remarques précédées de la mention "NOTE –" sont informatives et ne constituent pas une partie intégrante de la présente Recommandation | Norme internationale.

Recommandation UIT-T H.264

Codage vidéo évolué pour les services audiovisuels génériques

1 Domaine d'application

Le présent document spécifie la Recommandation UIT-T H.264 | Norme internationale ISO/CEI 14496-10 sur le codage vidéo.

2 Références normatives

Les Recommandations et autres références suivantes contiennent des dispositions qui, par suite de la référence qui y est faite, constituent des dispositions valables pour la présente Recommandation | Norme internationale. Au moment de la publication, les éditions indiquées étaient en vigueur. Toute Recommandation et normes sont sujettes à révision; et tous les utilisateurs de la présente Recommandation | Norme internationale sont donc invités à rechercher la possibilité d'appliquer les éditions les plus récentes des Recommandations et normes indiquées ci-après. Les membres de la CEI et de l'ISO tiennent des registres des normes internationales en vigueur. Le Bureau de normalisation des télécommunications de l'UIT tient à jour une liste des Recommandations de l'UIT-T en vigueur.

- Recommandation UIT-T T.35 (2000), *Procédure d'attribution des codes définis par l'UIT-T pour les facilités non normalisées*.
- ISO/CEI 11578:1996, Annexe A, *Identificateur unique universel*.
- ISO/CIE 10527:1991, *Observateurs de référence colorimétriques CIE*.

3 Définitions

Pour les besoins de la présente Recommandation | Norme internationale, les définitions suivantes s'appliquent.

3.1 unité d'accès: ensemble d'unités *NAL* qui contient toujours exactement une *image codée primaire*. En plus de l'*image codée primaire*, une unité d'accès peut aussi contenir une ou plusieurs *images codées redondantes* ou d'autres *unités NAL* ne contenant pas de *tranches* ou de *division de données par tranche* d'une *image codée*. Le décodage d'une unité d'accès a toujours pour résultat une *image décodée*.

3.2 coefficient de transformée AC (*AC transform coefficient*): tout *coefficient de transformée* pour lequel l'*indice de fréquence* dans une ou les deux dimensions est différent de zéro.

3.3 processus de décodage arithmétique binaire adaptatif: *processus de décodage entropique* qui calcule les valeurs des *segments* d'un *flux binaire* produits par un *processus de codage arithmétique binaire adaptatif*.

3.4 processus de codage arithmétique binaire adaptatif: *processus de codage entropique*, non spécifié de façon normative dans la présente Recommandation | Norme internationale, qui code une séquence de *segments* et produit un *flux binaire* qui peut être décodé en utilisant le *processus de décodage arithmétique binaire adaptatif*.

3.5 ordre arbitraire par tranches: *ordre de décodage* en *tranches* dans lequel l'*adresse de macrobloc* du premier *macrobloc* de certaines *tranches* d'une *image* peut être plus petite que l'*adresse de macrobloc* du premier *macrobloc* de certaines autres *tranches* précédentes de la même *image codée*.

3.6 tranche B: *tranche* qui peut être codée en utilisant l'*intraprédiction* à partir d'échantillons décodés au sein de la même *tranche* ou l'*interprédiction* à partir d'*images de référence*

précédemment décodées, en utilisant au plus deux *vecteurs cinétiques* et des *indices de référence* pour *prédire* les valeurs d'échantillon de chaque *bloc*.

3.7 segment: bit d'une *chaîne de segments*.

3.8 binairisation: ensemble de *chaînes de segments* pour toutes les valeurs possibles d'un *élément syntaxique*.

3.9 processus de binairisation: processus de mise en correspondance unique de toutes les valeurs possibles d'un *élément syntaxique* avec un ensemble de *chaînes de segments*.

3.10 chaîne de segments: une chaîne de *segments* est une représentation binaire intermédiaire des valeurs des *éléments syntaxiques* obtenues par la *binarisation* de l'*élément syntaxique*.

3.11 tranche biprédicative: voir *tranche B*.

3.12 flux binaire: séquence de bits qui forme la représentation d'*images codées* et des données associées qui constituent une ou plusieurs *séquences vidéo codées*. Le flux binaire est un terme collectif utilisé pour désigner soit un *flux d'unités NAL* soit un *flux d'octets*.

3.13 bloc: matrice MxN (M colonnes sur N rangées) d'échantillons, ou matrice MxN de *coefficients de transformée*.

3.14 champ inférieur: un *champ* parmi les deux que comporte une *trame*. Chaque rangée d'un champ inférieur est localisée spatialement immédiatement en dessous de la rangée correspondante d'un *champ supérieur*.

3.15 macrobloc inférieur (d'une paire de macroblocs): c'est, au sein d'une *paire de macroblocs*, le *macrobloc* qui contient les échantillons de la rangée inférieure d'échantillons pour la *paire de macroblocs*. Pour une *paire de macroblocs de champ*, le macrobloc inférieur représente les échantillons provenant de la région du *champ inférieur* de la *trame* qui se tient dans la région spatiale de la *paire de macroblocs*. Pour une *paire de macroblocs de trame*, le macrobloc inférieur représente les échantillons de la *trame* qui se tient dans la moitié inférieure de la région spatiale de la *paire de macroblocs*.

3.16 rupture de liaison: localisation dans un flux binaire à laquelle il est indiqué que certaines images suivantes dans l'*ordre de décodage* pourraient contenir des distorsions visuelles sérieuses dues à des opérations non spécifiées effectuées dans la génération du *flux binaire*.

3.17 octet: séquence de 8 bits, écrite et lue avec le bit de plus fort poids à gauche et le bit de plus faible poids à droite. Dans la représentation d'une séquence de bits de données, le bit de plus fort poids est le premier.

3.18 aligné par les octets: une position d'un *flux binaire* est aligné par les octets lorsque la position est un multiple entier de 8 bits à partir de la position du premier bit dans le *flux binaire*. Un bit ou *octet* ou *élément syntaxique* est dit aligné par les octets lorsque la position où il apparaît dans un *flux binaire* est alignée par les octets.

3.19 flux d'octets: incorporation d'un *flux d'unités NAL* contenant des *préfixes de code de démarrage* et des *unités NAL* comme spécifié à l'Annexe B.

3.19.1 can: terme utilisé pour désigner un comportement autorisé mais pas nécessairement obligatoire.

3.20 catégorie: nombre associé à chaque *élément syntaxique*. La catégorie sert à spécifier l'allocation des *éléments syntaxiques* aux *unités NAL* pour le partage des *données en tranches*. Elle peut aussi servir d'une façon déterminée par l'application pour se référer aux classes d'*éléments syntaxiques* d'une façon non spécifiée dans la présente Recommandation | Norme internationale.

3.21 chroma: adjectif spécifiant qu'un tableau d'échantillons ou un simple échantillon représente un des deux signaux de différence de couleur se rapportant aux couleurs primaires. Les symboles utilisés pour un tableau ou échantillon chroma sont Cb et Cr.

NOTE – Le terme chroma est utilisé de préférence au terme chrominance afin d'éviter l'implication de l'utilisation des caractéristiques linéaires de transfert de lumière qui est souvent associée au terme chrominance.

3.22 champ codé: *représentation codée d'un champ.*

3.23 trame codée: *représentation codée d'une trame.*

3.24 image codée: *représentation codée d'une image.* Une image codée peut être soit un *champ codé* soit une *trame codée*. Image codée est un terme collectif se rapportant à une *image codée primaire* ou à une *image codée redondante*, mais pas aux deux à la fois.

3.25 mémoire tampon des images codées (CPB, *coded picture buffer*): mémoire tampon en premier entré premier sorti contenant des *unités d'accès* dans l'*ordre de décodage* spécifié dans le *décodeur fictif de référence* de l'Annexe C.

3.26 représentation codée: élément de données tel que représenté sous sa forme codée.

3.27 séquence vidéo codée: séquence d'*unités d'accès* qui consiste, en ordre de décodage, en une *unité d'accès IDR* suivie de zéro ou plusieurs *unités d'accès* non IDR y compris toutes les *unités d'accès* suivantes jusqu'à, mais non compris, toute *unité d'accès IDR* suivante.

3.28 composant: matrice ou simple échantillon de l'une des trois matrices (une *luma* et deux *chroma*) qui forment un *champ* ou une *trame*.

3.29 paire de champs complémentaires: terme collectif pour une *paire de champs de référence complémentaires* ou une *paire de champs de non-référence complémentaires*.

3.30 paire de champs de non-référence complémentaires: deux *champs de non-référence* qui sont dans des *unités d'accès* consécutives en *ordre de décodage* en tant que deux *champs codés* de parité opposée lorsque le premier *champ* n'est pas déjà un *champ apparié*.

3.31 paire de champs de référence complémentaires: deux *champs de référence* qui sont dans des *unités d'accès* consécutives en *ordre de décodage* en tant que deux *champs codés* et qui partagent la même valeur de l'*élément syntaxique* *frame_num* (numéro de trame), où le second *champ* dans l'*ordre de décodage* n'est pas une *image IDR* et n'inclut pas un élément syntaxique *memory_management_control_operation* (opération de commande de gestion de la mémoire) égal à 5.

3.32 variable de contexte: variable spécifiée pour le *processus de décodage arithmétique binaire adaptatif* d'un *segment* par une équation contenant des *segments* récemment décodés.

3.33 coefficient de transformée DC: *coefficient de transformée* pour lequel l'*indice de fréquence* est zéro dans toutes les dimensions.

3.34 image décodée: une *image décodée* est déduite en décodant une *image codée*. Une *image décodée* est soit une *trame décodée*, soit un *champ décodé*. Un *champ décodé* est soit un *champ supérieur décodé* soit un *champ inférieur décodé*.

3.35 mémoire tampon d'image décodée (DPB): mémoire tampon contenant des *images décodées* servant de référence, de réorganisation de sortie et de retard de sortie spécifiée pour le *décodeur fictif de référence* de l'Annexe C.

3.36 décodeur: matérialisation d'un *processus de décodage*.

3.37 ordre de décodage: ordre dans lequel les *éléments syntaxiques* sont traités par le *processus de décodage*.

- 3.38 processus de décodage:** processus spécifié dans la présente Recommandation | Norme internationale qui lit un *flux binaire* et calcule des *images décodées* à partir de celui-ci.
- 3.39 prédiction directe:** *interprédiction* pour un *bloc* pour lequel il n'y a pas de décodage de *vecteur cinétique*. Deux modes de *prédiction* directe sont spécifiés qui sont appelés mode de *prédiction* directe spatiale et mode de *prédiction* temporelle.
- 3.40 décodeur soumis aux essais (DUT):** *décodeur* dont on vérifie la conformité à la présente Recommandation | Norme internationale en faisant délivrer au *décodeur* et au *décodeur fictif de référence* un *flux binaire* conforme par l'*ordonnanceur fictif de flux binaires* et en comparant les valeurs et la temporisation de sortie des deux *décodeurs*.
- 3.41 octet de prévention d'émulation:** octet égal à 0x03 qui peut être présent au sein d'une *unité NAL*. La présence d'octets de prévention d'émulation garantit que dans l'*unité NAL*, aucune séquence d'octets consécutifs alignés par les octets ne contient de *préfixe de code de début*.
- 3.42 codeur:** matérialisation d'un *processus de codage*.
- 3.43 processus de codage:** processus, non spécifié dans la présente Recommandation | Norme internationale, qui produit un *flux binaire* conforme à la présente Recommandation | Norme internationale.
- 3.44 champ:** assemblage de rangées alternées d'une *trame*. Une *trame* se compose de deux *champs*, un *champ supérieur* et un *champ inférieur*.
- 3.45 macrobloc de champ:** macrobloc contenant des échantillons provenant d'un seul *champ*. Tous les *macroblocs* d'un *champ codé* sont des macroblocs de champ. Lorsqu'on utilise le *décodage de trame/champ de macrobloc adaptatif*, certains macroblocs d'une *trame codée* peuvent être des macroblocs de champ.
- 3.46 paire de macroblocs de champ:** paire de *macroblocs* décodés comme deux *macroblocs de champ*.
- 3.47 balayage de champ:** ordonnancement séquentiel spécifique des *coefficients de transformée* qui diffère du balayage en *zigzag* en balayant les colonnes plus rapidement que les rangées. Le balayage de champ est utilisé pour les *coefficients de transformée* dans les *macroblocs de champ*.
- 3.48 fanion:** variable qui peut prendre une des deux valeurs possibles de 0 et 1.
- 3.49 trame:** une *trame* contient un assemblage d'échantillons luma et deux assemblages correspondants d'échantillons chroma. Une *trame* consiste en deux *champs*, un *champ supérieur* et un *champ inférieur*.
- 3.50 macrobloc de trame:** *macrobloc* représentant des échantillons des deux *champs* d'une *trame codée*. Lorsqu'on n'utilise pas le *décodage de trame/champ macrobloc adaptatif*, tous les macroblocs d'une *trame codée* sont des macroblocs de trame. Lorsqu'on utilise le *décodage de trame/champ macrobloc adaptatif*, certains macroblocs d'une *trame codée* peuvent être des macroblocs de trame.
- 3.51 paire de macrobloc de trame:** *paire de macroblocs* décodés comme deux *macroblocs de trame*.
- 3.52 indice de fréquence:** indice à une ou deux dimensions associé à un *coefficient de transformée* avant une partie de *transformation inverse* du *processus de décodage*.
- 3.53 décodeur fictif de référence (HRD):** modèle de *décodeur* fictif qui spécifie les contraintes en matière de variation des *flux d'unités NAL* conformes ou de *flux d'octets* conformes que peut produire le processus de codage.

3.54 ordonnanceur fictif de flux binaires (HSS): mécanisme de fourniture fictive de la temporisation et du flux de données à l'entrée d'un *flux binaire* au *décodeur fictif de référence*. Le HSS est utilisé pour vérifier la conformité d'un *flux binaire* ou d'un *décodeur*.

3.55 tranche I: *tranche* autre qu'une *tranche SI*, qui est décodée en effectuant uniquement la *prédiction* à partir d'échantillons décodés au sein de la même *tranche*.

3.55.1 informatif: terme utilisé pour désigner les éléments de la présente Recommandation | Norme internationale qui ne font pas partie intégrante de celle-ci. Les éléments informatifs ne créent aucune obligation de se conformer à la présente Recommandation | Norme internationale.

3.56 unité d'accès à rafraîchissement de décodage instantané (IDR): *unité d'accès* dans laquelle l'*image codée primaire* est une *image IDR*.

3.57 image à rafraîchissement de décodage instantané (IDR): *image codée* ne contenant que des *tranches* appartenant aux types de *tranche I* ou *SI* qui amène le *processus de décodage* à marquer toutes les images de *référence* comme "non utilisé pour référence" immédiatement après le décodage de l'*image IDR*. Après le décodage d'une *image IDR*, toutes les images codées suivantes en ordre de décodage peuvent être décodées sans *interprédiction* à partir de toute image décodée antérieurement à l'*image IDR*. La première *image* de chaque *séquence vidéo codée* est une *image IDR*.

3.58 intercodage: codage d'un *bloc*, *macrobloc*, *tranche* ou *image* qui utilise l'*interprédiction*.

3.59 interprédiction: *prédiction* déduite d'échantillons décodés d'*images de référence* autres que l'*image décodée* en cours.

3.60 intracodage: codage d'un *bloc*, *macrobloc*, *tranche* ou *image* qui utilise l'*intraprédiction*.

3.61 intraprédiction: *prédiction* déduite des échantillons décodés de la même *tranche décodée*.

3.62 intratranche: voir *tranche I*.

3.63 transformée inverse: partie du *processus de décodage* par lequel un ensemble de *coefficients de transformée* sont convertis en valeurs de domaine spatial, ou par lequel un ensemble de *coefficients de transformée* sont convertis en *coefficients de transformée DC*.

3.64 couche: élément d'un ensemble de structures syntaxiques dans une relation hiérarchique sans embranchement. Les couches supérieures contiennent les couches inférieures. Les couches de codage sont les couches de *séquence vidéo codée*, d'*image*, de *tranche* et de *macrobloc*.

3.65 niveau: ensemble défini de contraintes sur les valeurs que peuvent prendre les *éléments syntaxiques* et variables de la présente Recommandation | Norme internationale. Le même ensemble de niveaux est défini pour tous les *profils*, la plupart des aspects de la définition de chaque niveau étant communs à travers les différents *profils*. Les mises en œuvre individuelles peuvent, avec des contraintes spécifiques, accepter un niveau différent pour chaque *profil* géré. Dans un contexte différent, le niveau est la valeur d'un *coefficient de transformée* avant la *normalisation*.

3.66 vecteur cinétique de liste 0 (liste 1): *vecteur cinétique* associé à un *indice de référence* portant sur une *liste 0 (liste 1) d'image de référence*.

3.67 prédiction de liste 0 (liste 1): *interprédiction* du contenu d'une *tranche* utilisant un *indice de référence* portant sur une *liste 0 (liste 1) d'image de référence*.

3.68 luma: adjectif spécifiant qu'un tableau d'échantillons ou un simple échantillon représente le signal monochromatique qui se rapporte aux couleurs primaires. Le symbole ou l'indice utilisé pour luma est Y ou L.

NOTE – L'utilisation du terme luma est préférée à celle du terme luminance afin d'éviter les implications de l'utilisation des caractéristiques linéaires de transfert de la lumière qui sont souvent associées au terme luminance. Le symbole L est parfois utilisé au lieu du symbole Y pour éviter toute confusion avec le symbole Y utilisé sur l'axe des ordonnées.

3.69 macrobloc: un *bloc* d'échantillons *luma* de 16x16 et deux *blocs* correspondants d'échantillons *chroma*. La division d'une *tranche* ou d'une *paire de macroblocs* en macroblocs est une *subdivision*.

3.70 décodage de trame/champ de macrobloc adaptatif: *processus de décodage* pour *trames codées* dans lequel certains *macroblocs* peuvent être décodés comme *macroblocs de trame* et d'autres peuvent être décodés comme *macroblocs de champ*.

3.71 adresse de macrobloc: lorsqu'on n'utilise pas le *décodage de trame/champ de macrobloc adaptatif*, une adresse de macrobloc est l'indice d'un macrobloc dans un *balayage de trame de macrobloc* de l'*image* commençant par zéro pour le *macrobloc* supérieur gauche dans une *image*. Lorsqu'on utilise le *décodage de trame/champ de macrobloc adaptatif*, l'adresse de macrobloc du *macrobloc supérieur* d'une *paire de macroblocs* est deux fois l'indice d'une *paire de macroblocs* dans un *balayage de trame de paire de macroblocs* de l'*image*, et l'adresse de macrobloc du *macrobloc inférieur* d'une *paire de macroblocs* est l'adresse de macrobloc du *macrobloc supérieur* correspondant plus 1. L'adresse de macrobloc du *macrobloc supérieur* de chaque *paire de macroblocs* est un nombre pair et l'adresse de macrobloc du *macrobloc inférieur* de chaque *paire de macroblocs* est un nombre impair.

3.72 localisation de macrobloc: coordonnées bidimensionnelles d'un *macrobloc* dans une *image* notée (x, y). Pour le *macrobloc* supérieur gauche de l'*image* (x, y) est égal à (0, 0). x est incrémenté d'un pour chaque colonne de *macrobloc* de gauche à droite. Lorsqu'on n'utilise pas le *décodage de trame/champ de macrobloc adaptatif*, y est incrémenté de 1 pour chaque rangée de *macrobloc* du haut en bas. Lorsqu'on utilise le *décodage de trame/champ de macrobloc adaptatif*, y est incrémenté de 2 pour chaque rangée de *paire de macroblocs* du haut en bas, et est incrémenté de 1 en plus lorsqu'un *macrobloc* est un *macrobloc inférieur*.

3.73 paire de macroblocs: paire de *macroblocs* verticalement contigus dans une *trame* qui est couplée pour être utilisée dans le *décodage de trame/champ de macrobloc adaptatif*. La division d'une *tranche* en paires de macroblocs est une *subdivision*.

3.74 subdivision de macrobloc: un *bloc* d'échantillons *luma* et deux *blocs* d'échantillons *chroma* correspondants résultant de la *subdivision* d'un *macrobloc* pour l'*interprédiction*.

3.75 correspondance de macrobloc à un groupe de tranches: moyen de répartir les *macroblocs* d'une *image* en *groupes de tranches*. La correspondance de macrobloc à un groupe de tranches consiste en une liste de numéros, un pour chaque *macrobloc codé*, spécifiant le *groupe de tranches* auquel appartient chaque *macrobloc* codé.

3.76 correspondance d'unité de répartition à un groupe de tranches: moyen de faire correspondre les *unités de répartition de groupe de tranches* d'une *image* avec des *groupes de tranches*. La correspondance d'unité de répartition à un groupe de tranches consiste en une liste de numéros, un pour chaque *unité de répartition de groupe de tranches*, spécifiant le *groupe de tranches* auquel appartient chaque *unité de répartition de groupe de tranches*.

3.76.1 peut: terme utilisé pour désigner un comportement autorisé, mais pas nécessairement obligatoire. Parfois, lorsqu'on souhaite souligner le caractère facultatif du comportement décrit, on utilise à cet effet le membre de phrase "peut ou non".

3.77 opération de commande de gestion de mémoire: sept opérations qui commandent le *marquage d'image de référence*.

3.78 vecteur cinétique: vecteur bidimensionnel utilisé pour l'*interprédiction* qui donne le décalage à partir des coordonnées dans l'*image décodée* pour les coordonnées dans une *image de référence*.

- 3.78.1 doit:** terme utilisé pour exprimer une observation relative à une obligation ou une conséquence d'une obligation qui est spécifiée ailleurs dans la présente Recommandation | Norme internationale. Ce terme est utilisé exclusivement dans un contexte *informatif*.
- 3.79 unité NAL:** structure syntaxique contenant une indication du type des données qui suivent et des *octets* contenant ces données sous la forme d'une charge utile *RBSP* entrecoupée, autant que de besoin, d'*octets de prévention d'émulation*.
- 3.80 flux d'unités NAL:** séquence d'*unités NAL*.
- 3.81 champ non apparié:** terme collectif pour un *champ de référence non apparié* ou un *champ de non-référence non apparié*.
- 3.82 champ de non-référence non apparié:** *champ de non-référence* décodé qui ne fait pas partie d'une *paire de champs de non-référence complémentaires*.
- 3.83 champ de référence non apparié:** *champ de référence* décodé qui ne fait pas partie d'une paire de *champs de référence complémentaires*.
- 3.84 champ de non-référence:** *champ* codé avec *nal_ref_idc* égal à 0.
- 3.85 trame de non-référence:** *trame* codée avec *nal_ref_idc* égal à 0.
- 3.86 image de non-référence:** *image* codée avec *nal_ref_idc* égal à 0. On n'utilise pas d'*image de non-référence* pour l'*interprédiction* d'autres *images*.
- 3.86.1 note:** terme utilisé pour introduire des observations *informatives*. Ce terme est utilisé exclusivement dans un contexte *informatif*.
- 3.87 parité opposée:** la *parité opposée* du *haut* est le *bas* et vice versa.
- 3.88 ordre de sortie:** ordre dans lequel les *images décodées* sont extraites de la *mémoire tampon d'images décodées*.
- 3.89 tranche P:** *tranche* qui peut être décodée à l'aide de l'*intraprédiction* à partir d'échantillons décodés au sein de la même *tranche* ou à l'aide de l'*interprédiction* à partir d'*images de référence* précédemment décodées, en utilisant au plus un *vecteur cinétique* et un *indice de référence* pour *prédire* les valeurs d'échantillon de chaque *bloc*.
- 3.90 paramètre:** *élément syntaxique d'un ensemble de paramètres de séquence* ou d'un ensemble de paramètres d'*image*. Paramètre est aussi utilisé comme partie de l'expression définie de *paramètre de quantification*.
- 3.91 parité:** la *parité* d'un *champ* peut être *supérieure* ou *inférieure*.
- 3.92 subdivision:** division d'un ensemble en sous-ensembles de telle sorte que chaque élément de l'ensemble soit exactement dans un seul des sous-ensembles.
- 3.93 image:** terme collectif pour un *champ* ou une *trame*.
- 3.93.1 ensemble de paramètres d'image:** *structure syntaxique* contenant des *éléments syntaxiques* s'appliquant à zéro, une ou plusieurs *images codées* entières, telles que déterminées par l'*élément syntaxique pic_parameter_set_id* (identification d'ensemble de paramètres d'*image*) figurant dans chaque *en-tête de tranche*.
- 3.94 compte d'ordre d'image:** variable ayant une valeur non décroissante avec une position d'*image* croissante dans l'ordre de sortie par rapport à l'*image IDR* précédente dans l'*ordre de décodage* ou par rapport à la précédente *image* contenant l'*opération de commande de gestion de mémoire* qui marque toutes les *images de référence* comme "non utilisé comme référence".
- 3.95 prédiction:** matérialisation du *processus de prédiction*.
- 3.96 processus de prédiction:** utilisation d'un *prédicant* pour fournir une estimation de la valeur d'échantillon ou de l'élément de données en cours de décodage.

3.97 tranche prédictive: voir tranche P.

3.98 prédicant: combinaison de valeurs spécifiées ou de valeurs d'échantillons décodés antérieurement ou d'éléments de données utilisées dans le *processus de décodage* de valeurs d'échantillons suivants ou d'éléments de données.

3.99 image codée primaire: représentation codée d'une *image* à utiliser dans le *processus de décodage* pour un flux binaire conforme à la présente Recommandation | Norme internationale. L'image codée primaire contient tous les *macroblocs* de l'*image*. Les seules *images* qui ont un effet normatif sur le *processus de décodage* sont des images codées primaires. Voir aussi l'*image codée redondante*.

3.100 profil: sous-ensemble défini de la syntaxe de la présente Recommandation | Norme internationale.

3.101 paramètre de quantification: variable utilisée par le *processus de décodage* pour l'étalonnage des *niveaux de coefficient de transformée*.

3.102 accès aléatoire: action de démarrage du processus de décodage pour un *flux binaire* à un point autre que le début d'un flux.

3.103 balayage de trame: mise en correspondance d'un schéma rectangulaire bidimensionnel avec un schéma unidimensionnel, de telle sorte que les premières entrées dans le schéma unidimensionnel proviennent de la première rangée supérieure du schéma bidimensionnel balayé de gauche à droite, suivi de la même façon par la seconde, la troisième, etc., rangée du schéma (en descendant), chacune d'elle balayée de gauche à droite.

3.104 charge utile de séquence d'octets bruts (RBSP, raw byte sequence payload): structure syntaxique contenant un nombre entier d'octets, qui est encapsulée dans une *unité NAL*. Une charge utile RBSP est soit vide soit a la forme d'une *chaîne de bits de données* contenant des *éléments syntaxiques* suivis par un *bit d'arrêt de charge utile RBSP* et suivie par zéro ou plus bits suivants égaux à 0.

3.105 bit d'arrêt de charge utile de séquence d'octets bruts (RBSP): bit égal à 1, présent au sein d'une *charge utile de séquence d'octets bruts (RBSP)* après une *chaîne de bits de données*. La localisation de la fin de la *chaîne de bits de données* au sein d'une charge utile RBSP peut être identifiée en cherchant à partir de la fin de la charge utile RBSP le *bit d'arrêt RBSP*, qui est le dernier bit non à zéro dans la charge utile RBSP.

3.106 point de rétablissement: point dans le *flux binaire* auquel le rétablissement d'une représentation exacte ou approximative des *images décodées* représentées par le *flux binaire* est réalisé après un *accès aléatoire* ou une *rupture de liaison*.

3.107 image codée redondante: représentation codée d'une image ou d'une partie d'image. Le contenu d'une image codée redondante ne doit pas être utilisé par le *processus de décodage* pour un *flux binaire* conforme à la présente Recommandation | Norme internationale. Une *image codée redondante* ne doit pas nécessairement contenir tous les *macroblocs* de l'*image codée primaire*. Les images codées redondantes n'ont pas d'effet normatif sur le *processus de décodage*. Voir aussi à *image codée primaire*.

3.108 champ de référence: un *champ de référence* peut être utilisé pour l'*interprédiction* lorsque des tranches P, SP et B d'un *champ codé* ou des *macroblocs de champ* d'une *trame codée* sont décodés. Voir aussi à *image de référence*.

3.109 trame de référence: on peut utiliser une *trame de référence* pour l'*interprédiction* lorsque des tranches P, SP et B d'une *trame codée* sont décodées. Voir aussi à *image de référence*.

3.110 indice de référence: indice dans une *liste d'images de référence*.

3.111 image de référence: *image* avec *nal_ref_idc* non égal à 0. Une *image de référence* contient des échantillons qui peuvent être utilisés pour l'*interprédiction* dans le *processus de décodage* des *images* suivantes dans l'*ordre de décodage*.

3.112 liste d'images de référence: liste de numéros d'*image* à court terme et à long terme qui sont alloués aux *images de référence*.

3.113 liste 0 d'images de référence: *liste d'images de référence* utilisée pour l'*interprédiction* d'une *tranche P, B* ou *SP*. Toute *interprédiction* utilisée pour des tranches *P* et *SP* utilisent une liste 0 d'images de référence. Une liste 0 d'images de référence est une des deux listes d'*images de référence* utilisées pour l'*interprédiction* pour une *tranche B*, l'autre étant une *liste 1 d'images de référence*.

3.114 liste 1 d'images de référence: *liste d'images de référence* utilisée pour l'*interprédiction* d'une *tranche B*. Une liste 1 d'images de référence est une des deux listes d'*images de référence* utilisées pour l'*interprédiction* pour une *tranche B*, l'autre étant la *liste 0 d'images de référence*.

3.115 marquage d'image de référence: spécifie, dans le flux binaire, comment sont marquées les *images décodées* pour l'*interprédiction*.

3.116 réservé: le terme réservé, lorsqu'il est utilisé dans les clauses spécifiant des valeurs d'un *élément syntaxique* particulier, indique une utilisation future par l'UIT-T | ISO/CEI. Ces valeurs ne doivent pas être utilisées dans des *flux binaires* conformes à la présente Recommandation | Norme internationale, mais peuvent être utilisés dans des extensions futures de la présente Recommandation | Norme internationale par l'UIT-T | ISO/CEI.

3.117 résidu: différence décodée entre une *prédiction* d'un échantillon ou élément de données et sa valeur décodée.

3.118 exécution: nombre d'éléments de données consécutifs représentés dans le processus de décodage. Dans un certain contexte, le nombre de *niveaux de coefficient de transformée* de valeur zéro précédant un niveau de *coefficient de transformée* de valeur différente de zéro dans la liste de *niveaux de coefficient de transformée* générés par un *balayage en zigzag* ou un *balayage de champ*. Dans d'autres contextes, exécution se réfère à un nombre de *macrobloccs*.

3.119 format d'échantillon (SAR, sample aspect ratio): pour aider au processus d'affichage qui n'est pas spécifié dans la présente Recommandation | Norme internationale, il spécifie le rapport entre la distance horizontale voulue entre les colonnes et la distance verticale voulue entre les rangées de la matrice d'échantillons *luma* dans une *trame*. Le format d'échantillon est exprimé par $h:v$, où h est la largeur horizontale et v est la hauteur verticale (en unités arbitraires de distance spatiale).

3.120 normalisation: processus de multiplication des *niveaux de coefficient de transformée* par un facteur, résultant en *coefficients de transformée*.

3.120.1 ensemble de paramètres de séquence: *structure syntaxique* contenant des *éléments syntaxiques* s'appliquant à zéro, une ou plusieurs *séquences vidéo codées* entières, telles que déterminées par le contenu d'un *élément syntaxique seq_parameter_set_id* (identificateur d'ensemble de paramètres de séquence) figurant dans l'*ensemble de paramètres d'image* visé dans l'*élément syntaxique pic_parameter_set_id* (identificateur d'ensemble de paramètres d'image) figurant dans chaque *en-tête de tranche*.

3.120.2 doit: terme utilisé pour exprimer une obligation impérative de conformité à la présente Recommandation | Norme internationale. Lorsqu'il est utilisé pour exprimer une contrainte obligatoire relative aux valeurs d'*éléments syntaxiques* ou aux résultats obtenus par actionnement du *processus de décodage* spécifié, il relève de la responsabilité du *codeur* de faire en sorte que la contrainte soit mise en œuvre. Lorsque ce terme se réfère à des opérations exécutées par le *processus de décodage*, tout *processus de décodage* qui produit des résultats identiques au

processus de décodage décrit ici est conforme aux prescriptions applicables au *processus de décodage* de la présente Recommandation | Norme internationale.

3.120.3 devrait: terme utilisé pour désigner le mode de fonctionnement d'une mise en œuvre qu'il est préconisé d'appliquer dans les conditions habituelles prévues, mais qui n'impose aucune obligation impérative de conformité à la présente Recommandation | Norme internationale.

3.121 tranche SI: *tranche* qui est codée en n'utilisant la *prédiction* qu'à partir d'échantillons décodés au sein de la même *tranche* et en utilisant la quantification des échantillons de la *prédiction*. Une tranche SI peut être codée de telle sorte que ses échantillons décodés puissent être construits de façon identique à une *tranche SP*.

3.122 macrobloc sauté: *macrobloc* pour lequel il n'y a pas de données codées autres qu'une indication que le *macrobloc* doit être décodé comme "sauté". Cette indication peut être commune à plusieurs *macroblocs*.

3.123 tranche: nombre entier de *macroblocs* ou *paires de macroblocs* ordonnés à la suite dans le *balayage de trame* au sein d'un *groupe de tranches* particulier. Pour l'*image codée primaire*, la division de chaque *groupe de tranches* en tranches est une *subdivision*. Bien qu'une tranche contienne des *macroblocs* ou des *paires de macroblocs* qui sont consécutifs dans le balayage de trame au sein d'un groupe de tranches, ces *macroblocs* ou *paires de macroblocs* ne sont pas nécessairement consécutifs dans le balayage de trame au sein de l'*image*. Les adresses des *macroblocs* sont déduites de l'adresse du premier *macrobloc* dans une tranche (comme représenté dans l'*en-tête de tranche*) et dans la correspondance de *macrobloc* à *groupe de tranches*.

3.124 subdivision de données en tranches: méthode de *subdivision d'éléments syntaxiques* choisis en *structures syntaxiques* fondées sur une *catégorie* associée à chaque *élément syntaxique*.

3.125 groupe de tranches: sous-ensemble des *macroblocs* ou *paires de macroblocs* d'une *image*. La division de l'*image* en groupes de tranches est une *subdivision* de l'*image*. La subdivision est spécifiée par la *correspondance de macrobloc* à *groupe de tranche*.

3.126 unités de correspondance de groupe de tranches: unités de la *correspondance d'unité de répartition* à *groupe de tranches*.

3.127 en-tête de tranche: partie d'une *tranche codée* contenant les éléments de données appartenant au premier ou à tous les *macroblocs* représentés dans la tranche.

3.128 source: terme utilisé pour décrire le matériel vidéo ou certains de ses attributs avant le codage.

3.129 tranche SP: *tranche* qui est codée en utilisant l'*interprédiction* à partir d'*images de référence* précédemment décodées, en utilisant au plus un *vecteur cinétique* et un *indice de référence* pour *prédire* les valeurs d'échantillon de chaque *bloc*. Une tranche SP peut être codée de telle sorte que ses échantillons décodés puissent être construits de façon identique à une autre tranche SP ou *tranche SI*.

3.130 préfixe de code de déclenchement: séquence unique de trois *octets* égale à 0x000001 incorporée dans le *flux d'octets* comme préfixe pour chaque *unité NAL*. La localisation d'un préfixe de code de déclenchement peut être utilisée par un *décodeur* pour identifier le début d'une nouvelle *unité NAL* et la fin d'une *unité NAL* antérieure. On peut empêcher l'émulation des préfixes de code de déclenchement au sein des *unités NAL* en incluant des *octets de prévention d'émulation*.

3.131 chaîne de bits de données (SODB, string of data bits): séquence d'un certain nombre de bits représentant des *éléments syntaxiques* présents au sein d'une *charge utile de séquence d'octets bruts* avant le *bit d'arrêt de charge utile de séquence d'octets brut*. Au sein d'une chaîne SODB, le bit le plus à gauche est considéré comme le premier bit et celui de plus fort poids, et le bit le plus à droite est considéré comme le dernier et le bit de plus faible poids.

- 3.132 sous-macrobloc:** un quart des échantillons d'un *macrobloc*, c'est-à-dire un *bloc luma* de 8x8 et deux *blocs chroma* correspondants dont un coin est situé dans un coin du *macrobloc*.
- 3.133 subdivision de sous-macrobloc:** un *bloc* d'échantillons *luma* et les deux *blocs* correspondants d'échantillons *chroma* résultant d'une *subdivision* d'un *sous-macrobloc* pour l'*interprédiction*.
- 3.134 commutation de tranche I:** voir à tranche SI.
- 3.135 commutation de tranche P:** voir à tranche SP.
- 3.136 élément syntaxique:** élément de données représenté dans le *flux binaire*.
- 3.137 structure syntaxique:** zéro ou plusieurs *éléments syntaxiques* présents ensemble dans le *flux binaire* dans un ordre spécifié.
- 3.138 champ supérieur:** un des deux *champs* que comporte une *trame*. Chaque rangée d'un *champ supérieur* est située dans l'espace immédiatement au-dessus de la rangée correspondante du *champ inférieur*.
- 3.139 macrobloc supérieur (d'une paire de macroblocs):** *macrobloc* au sein d'une *paire de macroblocs* qui contient les échantillons de la rangée supérieure d'échantillons pour la *paire de macroblocs*. Pour une *paire de macroblocs de champ*, le *macrobloc supérieur* représente les échantillons provenant de la région du *champ supérieur* de la *trame* qui se tient dans la région de l'espace de la *paire de macroblocs*. Pour une *paire de macroblocs de trame*, le *macrobloc supérieur* représente les échantillons de la *trame* qui est dans la moitié supérieure de la région de l'espace de la *paire de macroblocs*.
- 3.140 coefficient de transformée:** grandeur scalaire, considérée comme faisant partie du domaine de la fréquence, qui est associée à un *indice de fréquence* unidimensionnel ou bidimensionnel particulier dans une partie *transformée inverse* du *processus de décodage*.
- 3.141 niveau de coefficient de transformée:** quantité entière représentant la valeur associée à un indice de fréquence bidimensionnel particulier dans le *processus de décodage* avant de *normaliser* pour le calcul d'une valeur de *coefficient de transformée*.
- 3.142 identificateur unique universel (UUID):** identificateur qui est unique par rapport à l'espace de tous les identificateurs uniques universels.
- 3.143 non spécifié:** le terme non spécifié, lorsqu'il est utilisé dans les paragraphes spécifiant des valeurs d'un *élément syntaxique* particulier, indique que les valeurs n'ont pas de signification spécifiée dans la présente Recommandation | Norme internationale et n'auront pas de signification spécifiée à l'avenir en tant que partie intégrante de la présente Recommandation | Norme internationale.
- 3.144 codage à longueur variable (VLC, *variable length coding*):** procédure réversible pour le codage entropique qui alloue de plus courtes chaînes binaires aux *symboles* dont on s'attend qu'ils soient plus fréquents et des chaînes binaires plus longues aux *symboles* dont on s'attend qu'ils soient moins fréquents.
- 3.145 balayage en zigzag:** ordre séquentiel spécifique des *niveaux de coefficient de transformée* depuis (approximativement) la plus faible fréquence spatiale jusqu'à la plus forte. Le balayage en zigzag est utilisé pour les *niveaux de coefficient de transformée* dans les *macroblocs de trame*.

4 Abréviations

- 4.1 CABAC:** codage arithmétique binaire adaptatif selon le contexte (*context-based adaptive binary arithmetic coding*)

- 4.2 **CAVLC**: codage à longueur variable adaptatif selon le contexte (*context-based adaptive variable length coding*)
- 4.3 **CBR**: débit binaire constant (*constant bit rate*)
- 4.4 **CPB**: mémoire tampon d'images codées (*coded picture buffer*)
- 4.5 **DPB**: mémoire tampon d'images décodées (*decoded picture buffer*)
- 4.6 **DUT**: décodeur soumis aux essais (*decoder under test*)
- 4.7 **FIFO**: premier entré, premier sorti (*first-in, first-out*)
- 4.8 **HRD**: décodeur fictif de référence (*hypothetical reference decoder*)
- 4.9 **HSS**: ordonnanceur fictif de flux binaires (*hypothetical stream scheduler*)
- 4.10 **IDR**: rafraîchissement de décodage instantané (*instantaneous decoding refresh*)
- 4.11 **LSB**: bit de plus faible poids (*least significant bit*)
- 4.12 **MB**: macrobloc
- 4.13 **MBAFF**: codage de trame/champ de macrobloc adaptatif (*macroblock-adaptive frame-field coding*)
- 4.14 **MSB**: bit de plus fort poids (*most significant bit*)
- 4.15 **NAL**: couche d'abstraction du réseau (*network abstraction layer*)
- 4.16 **RBSP**: charge utile de séquence d'octets bruts (*raw byte sequence payload*)
- 4.17 **SEI**: informations d'amélioration supplémentaires (*supplemental enhancement information*)
- 4.18 **SODB**: chaîne de bits de données (*string of data bits*)
- 4.19 **UUID**: identificateur unique universel (*universal unique identifier*)
- 4.20 **VBR**: débit binaire variable (*variable bit rate*)
- 4.21 **VCL**: couche de codage vidéo (*video coding layer*)
- 4.22 **VLC**: codage à longueur variable (*variable length coding*)
- 4.23 **VUI**: informations d'usage de la vidéo (*video usability information*)

5 Conventions

NOTE – Les opérateurs mathématiques utilisés dans la présente Spécification sont similaires à ceux utilisés dans le langage de programmation C. Cependant, les opérations de division entière et de décalage arithmétique sont définies de façon spécifique. Les conventions de numérotage et de comptage commencent généralement à partir de 0.

5.1 Opérateurs arithmétiques

Les opérateurs arithmétiques ci-après sont définis comme suit.

+	Addition
–	Soustraction (dans le cas d'un opérateur binaire) ou négativation (dans le cas d'un opérateur unaire)
*	Multiplication
x^y	Exponentiation. Spécifie x à la puissance y . Dans d'autres contextes, une telle notation est utilisée pour mettre un exposant qui n'est pas destiné à être interprété comme une puissance.

/	Division entière avec troncature du résultat tendant vers zéro. Par exemple, $7/4$ et $-7/-4$ sont tronquées à 1 et $-7/4$ et $7/-4$ sont tronquées à -1 .
÷	Sert à marquer la division dans des équations mathématiques non destinées à faire l'objet d'une troncature ou d'un arrondi.
$\frac{x}{y}$	Sert à marquer la division dans des équations mathématiques non destinées à faire l'objet d'une troncature ou d'un arrondi.
$\sum_{i=x}^y f(i)$	Sommation des $f(i)$ où i prend toutes les valeurs entières de x jusqu'à et y compris y .
$x\%y$	Modulo. Reste de x divisé par y , défini seulement pour des entiers x et y avec $x \geq 0$ et $y > 0$.

Lorsque l'ordre de préséance n'est pas indiqué explicitement par l'utilisation de parenthèses, on appliquera les règles suivantes:

- les opérations de multiplication et division sont censées prendre place avant l'addition et la soustraction;
- les opérations de multiplication et division en séquence sont effectuées à la suite de gauche à droite;
- les opérations d'addition et de soustraction en séquence sont effectuées à la suite de gauche à droite.

5.2 Opérateurs logiques

Les opérateurs logiques ci-après sont définis comme suit:

$x \ \&\& \ y$	est le "ET" de la logique booléenne de x et y
$x \ \ y$	est le "OU" de la logique booléenne de x ou y
!	est le "NON" de la logique booléenne
$x \ ? \ y : \ z$	si x est VRAI ou non égal à 0, il se rapporte à la valeur de y ; autrement, il se rapporte à la valeur de z .

5.3 Opérateurs relationnels

Les opérateurs relationnels ci-après sont définis comme suit:

>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à
=	égal à
!=	différent de

5.4 Opérateurs binaires

Les opérateurs binaires ci-après sont définis comme suit:

&	binaire "et". Lorsqu'il agit sur des arguments entiers, il agit sur une représentation du complément à deux de la valeur entière. Lorsqu'il agit sur un argument binaire qui contient moins de bits qu'un autre argument, l'argument plus court est étendu en ajoutant plus de bits significatifs égaux à 0.
---	--

- | binaire "ou". Lorsqu'il agit sur des arguments entiers, il agit sur une représentation du complément à deux de la valeur entière. Lorsqu'il agit sur un argument binaire qui contient moins de bits qu'un autre argument, l'argument plus court est étendu en ajoutant plus de bits significatifs égaux à 0.
- $x \gg y$ décalage arithmétique à droite de la représentation entière du complément à deux de x par y chiffres binaires. Cette fonction n'est définie que pour des valeurs entières positives de y . Les bits décalés dans les MSB à la suite du décalage à droite doivent avoir une valeur égale au MSB de x avant l'intervention du décalage.
- $x \ll y$ décalage arithmétique à gauche de la représentation entière du complément à deux de x par y chiffres binaires. Cette fonction n'est définie que pour des valeurs entières positives de y . Les bits décalés dans les LSB à la suite du décalage à gauche doivent avoir une valeur égale à 0.

5.5 Opérateurs d'affectation

Les opérateurs arithmétiques ci-après sont définis comme suit:

- = opérateur d'affectation.
- ++ Incrément, c'est-à-dire, $x++$ est équivalent à $x = x + 1$; lorsqu'il est utilisé dans un indice matriciel, il se rapporte à la valeur de la variable avant l'opération d'incrémement.
- Décrément, c'est-à-dire, $x--$ est équivalent à $x = x - 1$; lorsqu'il est utilisé dans un indice matriciel, il se rapporte à la valeur de la variable avant l'opération de décrémement.
- += Incrément d'un montant spécifié, c'est-à-dire, $x += 3$ est équivalent à $x = x + 3$, et $x += (-3)$ est équivalent à $x = x + (-3)$.
- = Décrément d'un montant spécifié, c'est-à-dire, $x -= 3$ est équivalent à $x = x - 3$, et $x -= (-3)$ est équivalent à $x = x - (-3)$.

5.6 Notation de gamme

La notation suivante est utilisée pour spécifier une gamme de valeurs:

$x = y .. z$ x prend des valeurs entières de y à z inclus avec x , y et z entiers.

5.7 Fonctions mathématiques

Les fonctions mathématiques ci-après sont définies comme suit:

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

$$\text{Ceil}(x) \text{ le plus petit entier supérieur ou égal à } x. \quad (5-2)$$

$$\text{Clip1}(x) = \text{Clip3}(0, 255, x) \quad (5-3)$$

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; z < x \\ y & ; z > y \\ z & ; \text{autrement} \end{cases} \quad (5-4)$$

$$\text{Floor}(x) \text{ le plus grand entier inférieur ou égal à } x. \quad (5-5)$$

$$\text{InverseRasterScan}(a, b, c, d, e) = \begin{cases} (a \% (d / b)) * b; & e == 0 \\ (a / (d / b)) * c; & e == 1 \end{cases} \quad (5-6)$$

$$\text{Log2}(x) \text{ renvoie au logarithme à base 2 de } x. \quad (5-7)$$

$$\text{Log10}(x) \text{ renvoie au logarithme à base 10 de } x. \quad (5-8)$$

$$\text{Luma4x4BlkScan}(x, y) = (x / 2) * 4 + (y / 2) * 8 + \text{RasterScan}(x \% 2, y \% 2, 2) \quad (5-9)$$

$$\text{Median}(x, y, z) = x + y + z - \text{Min}(x, \text{Min}(y, z)) - \text{Max}(x, \text{Max}(y, z)) \quad (5-10)$$

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

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

$$\text{RasterScan}(x, y, n_x) = x + y * n_x \quad (5-13)$$

$$\text{Round}(x) = \text{Sign}(x) * \text{Floor}(\text{Abs}(x) + 0,5) \quad (5-14)$$

$$\text{Sign}(x) = \begin{cases} 1 & ; x \geq 0 \\ -1 & ; x < 0 \end{cases} \quad (5-15)$$

$$\text{Sqrt}(x) = \sqrt{x} \quad (5-16)$$

5.8 Variables, éléments syntaxiques et tableaux

Les éléments syntaxiques dans le flux binaire sont représentés en caractères **gras**. Chaque élément syntaxique est décrit par son nom (toujours en minuscules avec des liaisons en caractères de soulignement), ses une ou deux catégories syntaxiques et un ou deux descripteurs pour sa méthode de représentation codée. Le processus de décodage se comporte conformément à la valeur de l'élément syntaxique et aux valeurs des éléments syntaxiques précédemment décodés. Lorsqu'une valeur d'un élément syntaxique est utilisée dans les tableaux syntaxiques ou le texte, il apparaît en caractères normaux (c'est-à-dire, pas en gras).

Dans certains cas, les tableaux syntaxiques peuvent utiliser les valeurs d'autres variables déduites de valeurs d'éléments syntaxiques. De telles variables apparaissent dans les tableaux syntaxiques, ou le texte, et sont désignées par un mélange de majuscules et de minuscules et sans aucun caractère de soulignement. Les variables commençant par une majuscule sont tirées, pour le décodage, de la structure syntaxique courante et de toutes celles qui en dépendent. Les variables commençant par une majuscule peuvent être utilisées dans le processus de décodage pour des structures syntaxiques ultérieures en mentionnant la structure syntaxique d'origine de la variable. Les variables commençant par des minuscules ne sont utilisées qu'à l'intérieur du paragraphe dont elles sont tirées.

Dans certains cas, on utilise de façon interchangeable avec leurs valeurs numériques des noms "mnémoniques" pour des valeurs d'élément syntaxique ou de variable. Parfois, les noms "mnémoniques" sont utilisés sans être associés à aucune valeur numérique. L'association entre valeurs et noms est spécifiée dans le texte. Les noms sont construits à partir d'un ou plusieurs groupes de lettres séparées par un caractère de soulignement. Chaque groupe commence par une majuscule et peut contenir plusieurs majuscules.

NOTE – La syntaxe est décrite d'une façon qui suit de près la construction syntaxique du langage C.

Les fonctions sont décrites par leur nom, qui est construit comme un nom d'élément syntaxique, avec des parenthèses à droite et à gauche incluant zéro ou plusieurs noms de variables (pour les définitions) ou de valeurs (pour l'utilisation), séparés par des virgules (s'il y a plus d'une variable).

Les crochets sont utilisés pour indexer dans des listes ou des matrices. Les listes ou matrices peuvent être des éléments syntaxiques ou des variables. Des matrices bidimensionnelles sont parfois aussi spécifiées à l'aide de la notation matricielle en utilisant des indices pour l'indexation.

NOTE – L'ordre des indices est changé pour les matrices à deux dimensions utilisant des crochets et des indices. Un échantillon à la position horizontale x et la position verticale y dans une matrice d'échantillons en deux dimensions noté $s[x, y]$ serait en notation matricielle désigné s_{yx} .

La notation binaire est indiquée en incluant la chaîne de valeurs de bits dans des guillemets simples. Par exemple, '01000001' représente une chaîne de huit bits ayant seulement ses second et dernier bits égaux à 1.

La notation hexadécimale, indiquée par le préfixe "0x" devant le nombre hexadécimal, peut être utilisée à la place de la notation binaire lorsque le nombre de bits est un multiple entier de 4. Par exemple, 0x41 représente une chaîne binaire de huit bits ayant seulement ses second et dernier bits égaux à 1.

Les valeurs numériques non incluses dans des guillemets simples et sans le préfixe "0x" sont des valeurs décimales.

Une valeur égale à 0 représente une condition FAUX dans une instruction d'essai. La valeur VRAI est représentée par toute autre valeur que zéro.

5.9 Description textuelle des opérations logiques

Dans le texte, une instruction d'opérations logiques qui serait décrite en pseudo-code comme:

```
si( condition 0 )
    instruction 0
autrement si ( condition 1 )
    instruction 1
...
autrement /* remarque informative sur la condition restante */
    instruction n
```

peut être décrite de la manière suivante:

- ... comme suit / ... on applique ce qui suit:
- si condition 0, instruction 0
- Autrement, si condition 1, instruction 1
- ...
- Autrement (remarque informative sur la condition restante), instruction n.

Chaque instruction "si...autrement, si...autrement, ..." dans le texte est introduite par "... comme suit" ou "... on applique ce qui suit" immédiatement suivi de "si ... ". La dernière condition du "si...autrement, si...autrement, ..." est toujours un "autrement, ...". L'entrelacement des instructions "si...autrement, si...autrement, ..." peut se reconnaître en recherchant les "... comme suit" ou "... on applique ce qui suit" avec la terminaison "autrement, ...".

Dans le texte, une instruction d'opérations logiques qui serait décrite en pseudo-code comme:

```
si( condition 0a && condition 0b )
    instruction 0
autrement si ( condition 1a || condition 1b )
    instruction 1
...
autrement
    instruction n
```

peut être décrite de la façon suivante:

- ... comme suit / ... on applique ce qui suit.
- Si toutes les conditions suivantes sont vraies, instruction 0
 - condition 0a
 - condition 0b

- Autrement, si une des conditions suivantes est vraie, instruction 1
 - condition 1a
 - condition 1b
 - ...
- Autrement, instruction n

Dans le texte, une instruction d'opérations logiques qui serait décrite en pseudo-code comme

```

si( condition 0 )
    instruction 0
si( condition 1 )
    instruction 1
  
```

peut être décrite de la façon suivante:

Lorsqu'il y a la condition 0, instruction 0

Lorsqu'il y a la condition 1, instruction 1

5.10 Processus

Les processus sont utilisés pour décrire le décodage des éléments syntaxiques. La spécification et l'invocation d'un processus sont séparées. Tous les éléments syntaxiques et les variables en majuscules qui appartiennent à la structure syntaxique courante et aux structures syntaxiques qui en dépendent sont disponibles dans la spécification et l'invocation d'un processus. La spécification d'un processus peut aussi avoir une variable en minuscule explicitement spécifiée comme entrée. Chaque spécification de processus spécifie explicitement une sortie. La sortie est une variable qui peut aussi bien être en majuscules qu'en minuscules.

L'allocation des variables est spécifiée comme suit:

- si on invoque un processus, les variables sont explicitement affectées en variables d'entrée ou sortie minuscules de la spécification du processus pour le cas où elles n'auraient pas le même nom;
- autrement (lorsque les variables ont le même nom à l'invocation et à la spécification), l'affectation est implicite.

Dans la spécification d'un processus, le nom de la variable peut se référer à un macrobloc spécifique en ayant une valeur égale à l'adresse de ce macrobloc spécifique.

6 Formats de données source, codés, décodés et de sortie, processus de balayage et relations de voisinage

6.1 Formats de flux binaires

Le présent paragraphe spécifie la relation entre le flux d'unités NAL et le flux d'octets, l'un ou l'autre étant appelé le flux binaire.

Le flux binaire peut avoir un des deux formats suivants: le format de flux d'unités NAL ou le format de flux d'octets. Le format de flux d'unités NAL est conceptuellement de type plus "basique". Il consiste en une séquence de structures syntaxiques appelées unités NAL. Cette séquence est ordonnée en ordre de décodage. Il y a des contraintes imposées à l'ordre de décodage (et au contenu) des unités NAL dans le flux d'unités NAL.

Le format de flux d'octets peut être construit à partir du format de flux d'unités NAL en ordonnant les unités NAL dans l'ordre de décodage et en ajoutant à chaque unité NAL un préfixe de code de déclenchement et zéro ou plusieurs octets de valeur zéro pour former un flux d'octets. Le format de flux d'unités NAL peut être extrait du format de flux d'octets en recherchant la localisation du

schéma unique de préfixe de code de déclenchement au sein de ce flux d'octets. Les méthodes de tramage des unités NAL d'une autre façon que par l'utilisation du format de flux d'octet sont en dehors du domaine d'application de la présente Recommandation | Norme internationale. Le format de flux d'octets est spécifié à l'Annexe B.

6.2 Formats d'image source, décodés et de sortie

Le présent paragraphe spécifie la relation entre trames et champs source et décodés qui est donnée via le flux binaire.

La vidéo source qui est représentée par le flux binaire est une séquence de trames et/ou champs (appelés collectivement images) en ordre de décodage.

Les images source et décodées (trames ou champs) sont chacune composées de trois tableaux d'échantillons, un luma et deux chroma.

La variable ChromaFormatFactor est spécifiée au Tableau 6-1, en fonction de la structure d'échantillonnage du format chroma. La valeur de ChromaFormatFactor doit être trouvée égale à 1,5, indiquant un échantillonnage de 4:2:0. En échantillonnage monochrome, il y a seulement une matrice d'échantillon, qui peut être nominalement considérée comme une matrice luma. En échantillonnage 4:2:0, chacune des deux matrices chroma a la moitié de la hauteur et la moitié de la largeur de la matrice luma. En échantillonnage 4:2:2, chacune des deux matrices chroma a la même hauteur et la moitié de la largeur que la matrice luma. En échantillonnage 4:4:4, chacune des deux matrices chroma a la même hauteur et la même largeur que la matrice luma.

NOTE – D'autres valeurs peuvent être valides dans des versions futures de la présente Recommandation | Norme internationale.

Tableau 6-1 – Valeurs de ChromaFormatFactor

Format de chroma	ChromaFormatFactor
monochrome	1
4:2:0	1,5
4:2:2	2
4:4:4	3

La présente Recommandation | Norme internationale représente des séquences de couleur en utilisant l'échantillonnage chroma de 4:2:0. La largeur de la matrice d'échantillons luma de chaque image est un multiple entier de 16. La largeur des matrices d'échantillons chroma de chaque image est un multiple entier de 8. La hauteur de la matrice d'échantillons luma de chaque image codée (qu'elle soit une trame codée ou un champ codé) est un multiple entier de 16 et la hauteur de chaque matrice chroma pour ces images est un multiple entier de 8. Si des trames codées sont présentes au sein d'une séquence vidéo codée qui contient des champs codés ou des trames codées qui utilisent le codage de trame/champ de macrobloc adaptatif, la hauteur de la matrice d'échantillons luma de toutes les trames codées dans la séquence vidéo codée est un multiple entier de 32 et la hauteur de chaque matrice d'échantillons chroma pour ces trames est un multiple entier de 16. La largeur ou hauteur de sortie d'images à partir du processus de décodage n'est pas nécessairement un multiple entier de 16 et peut être spécifiée à l'aide d'une mire (*cropping rectangle*).

La largeur des champs codés se rapportant à un ensemble de paramètres de séquence spécifiques est la même que celle des trames codées se rapportant au même ensemble de paramètres de séquence (voir ci-dessous). La hauteur des champs codés se rapportant à un ensemble de paramètres de séquence spécifiques est la moitié de celle de trames codées se rapportant au même ensemble de paramètres de séquence (voir ci-dessous).

Les localisations verticales et horizontales relatives nominales des échantillons luma et chroma dans les trames sont indiquées à la Figure 6-1. D'autres localisations relatives d'échantillons chroma peuvent être indiquées dans les informations d'utilisation de la vidéo (voir l'Annexe E).

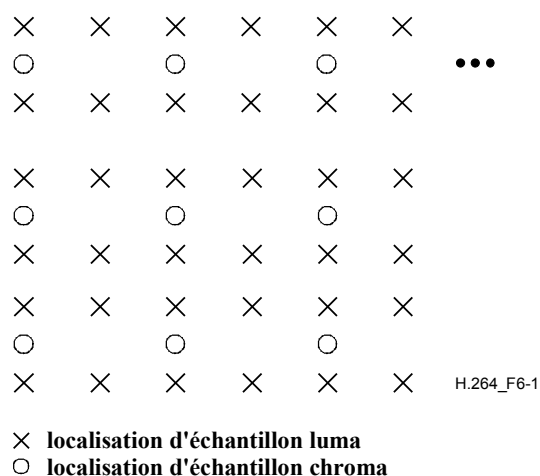


Figure 6-1 – Localisations nominales verticales et horizontales d'échantillons 4:2:0 luma et chroma dans une trame

Une trame se compose de deux champs comme décrit ci-dessous. Une image codée peut représenter une trame codée ou un champ codé individuel. Une séquence vidéo codée se conformant à la présente Recommandation | Norme internationale peut contenir des combinaisons arbitraires de trames codées et de champ codés. Le processus de décodage est aussi spécifié d'une façon qui permet que de plus petites régions d'une trame codée soient codées comme région de trame ou de champ, en utilisant le codage de trame/champ par macrobloc adaptatif.

Il existe deux types de champs de source et décodés: champ supérieur ou champ inférieur. Lorsqu'il y a deux champs en sortie au même moment, ou qu'ils sont combinés pour être utilisés comme trame de référence (voir ci-dessous), les deux champs (qui doivent être de parité opposée) sont entrelacés. La première (c'est-à-dire, celle du haut), la troisième, la cinquième, etc., rangée d'une trame décodée sont les rangées de champ supérieur. La seconde, la quatrième, la sixième, etc., rangée d'une trame décodée sont les rangées de champ inférieur. La première rangée (c'est-à-dire celle du haut) est la rangée numéro 0; la deuxième est la rangée numéro 1; la troisième est la rangée numéro 2, etc. Un champ supérieur ne se compose que des rangées de champ supérieur d'une trame, et un champ inférieur ne se compose que des rangées de champ inférieur d'une trame. Lorsque le champ supérieur ou le champ inférieur d'une trame décodée est utilisé comme champ de référence (voir ci-dessous), seules sont utilisées les rangées numérotées paires (pour un champ supérieur) ou les rangées impaires (pour un champ inférieur) de la trame décodée.

Les localisations relatives nominales verticales et horizontales des échantillons luma et chroma dans les champs inférieurs et supérieurs sont indiquées à la Figure 6-2. Les localisations relatives nominales verticales des échantillons dans un champ supérieur sont spécifiées comme étant décalées vers le haut d'un quart de la hauteur d'un échantillon luma par rapport à la grille d'échantillonnage du champ. Les localisations d'échantillonnage verticales des échantillons chroma dans un champ inférieur sont spécifiées comme décalées vers le bas d'un quart de la hauteur d'un échantillon luma par rapport à la grille d'échantillonnage du champ. D'autres localisations relatives d'échantillon chroma peuvent être indiquées dans les informations d'utilisation de la vidéo (voir l'Annexe E).

NOTE – Le décalage des échantillons chroma est destiné à aligner verticalement ces échantillons sur la localisation habituelle par rapport à la grille d'échantillonnage de trame entière, comme indiqué à la Figure 6-1.

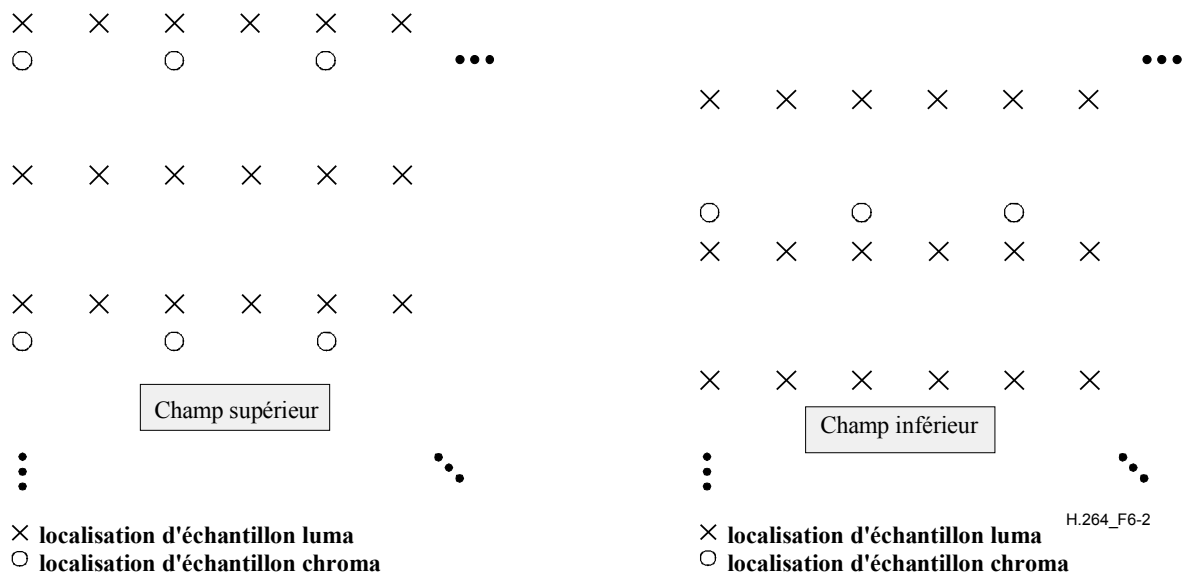


Figure 6-2 – Localisations d'échantillonnage nominales verticales et horizontales des échantillons des champs supérieur et inférieur

6.3 Subdivision spatiale des images et des tranches

Le présent paragraphe spécifie la façon dont une image est divisée en tranches et macroblocs. Les images sont divisées en tranches. Une tranche est une séquence de macroblocs ou, lorsque le décodage de trame/champ de macrobloc adaptatif est utilisé, une séquence de paires de macroblocs.

Chaque macrobloc est composé d'un tableau d'échantillons luma 16x16 et de deux tableaux chroma 8x8. Lorsque le décodage de trame/champ de macrobloc adaptatif n'est pas utilisé, chaque macrobloc représente une région spatiale rectangulaire de l'image. Par exemple, une image peut être divisée en deux tranches, comme indiqué à la Figure 6-3.

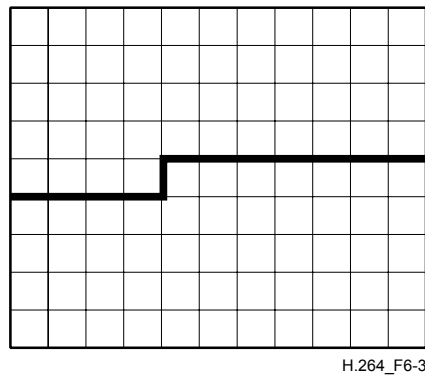


Figure 6-3 – Image avec 11 sur 9 macroblocs qui est subdivisée en deux tranches

Lorsque le décodage de trame/champ de macrobloc adaptatif est utilisé, l'image est subdivisée en tranches contenant un nombre entier de paires de macroblocs, comme indiqué à la Figure 6-4. Chaque paire de macroblocs comporte deux macroblocs.

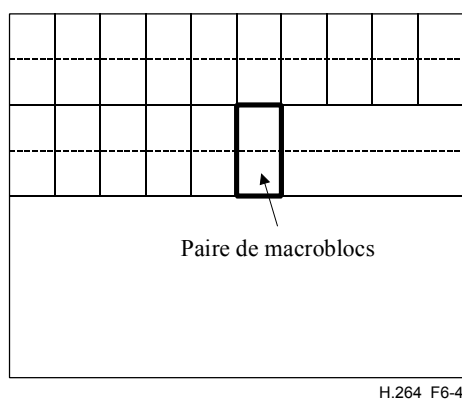


Figure 6-4 – Subdivision de la trame décodée en paires de macroblocs

6.4 Processus de balayage inverse et processus de déduction pour le voisinage

Le présent paragraphe spécifie les processus de balayage inverse; c'est-à-dire, le mappage des indices avec les localisations, et les processus de déduction pour le voisinage.

6.4.1 Processus de balayage inverse de macrobloc

L'entrée dans ce processus est une adresse de macrobloc mbAddr.

La sortie de ce processus est la localisation (x, y) de l'échantillon luma supérieur gauche pour le macrobloc ayant l'adresse mbAddr relative à l'échantillon supérieur gauche de l'image.

Le processus de balayage inverse de macrobloc est spécifié comme suit:

- Si MbaffFrameFlag est égal à 0,

$$x = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 0) \quad (6-1)$$

$$y = \text{InverseRasterScan}(\text{mbAddr}, 16, 16, \text{PicWidthInSamples}_L, 1) \quad (6-2)$$

- Autrement (MbaffFrameFlag est égal à 1), on applique ce qui suit:

$$xO = \text{InverseRasterScan}(\text{mbAddr} / 2, 16, 32, \text{PicWidthInSamples}_L, 0) \quad (6-3)$$

$$yO = \text{InverseRasterScan}(\text{mbAddr} / 2, 16, 32, \text{PicWidthInSamples}_L, 1) \quad (6-4)$$

Selon le macrobloc en cours, on applique ce qui suit.

- Si le macrobloc en cours est un macrobloc de trame

$$x = xO \quad (6-5)$$

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

- Autrement (le macrobloc en cours est un macrobloc de champ)

$$x = xO \quad (6-7)$$

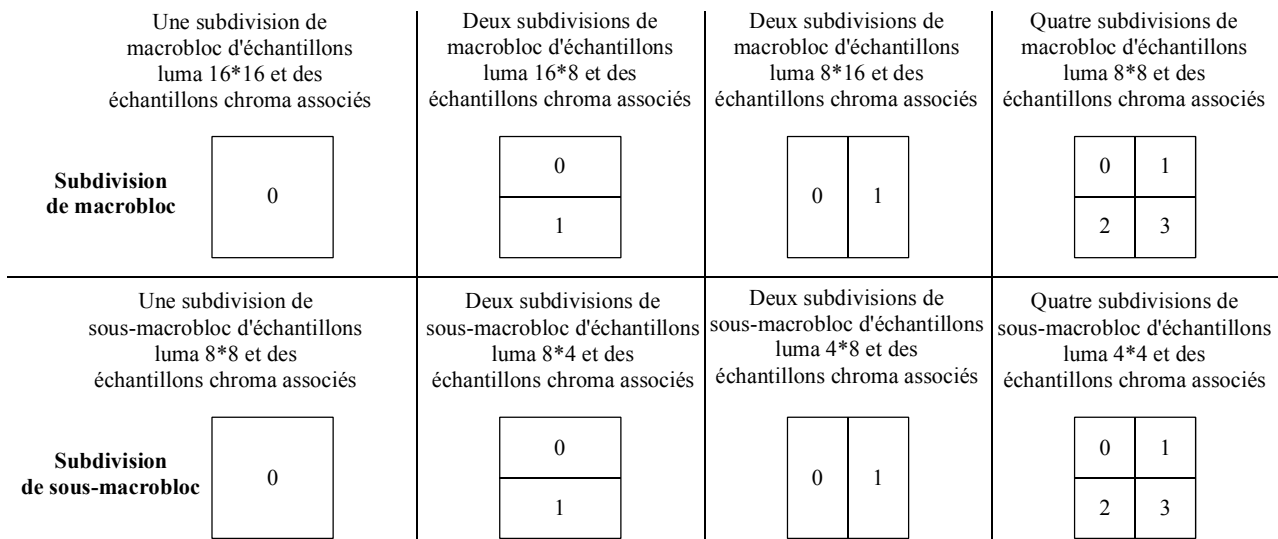
$$y = yO + (\text{mbAddr} \% 2) \quad (6-8)$$

6.4.2 Processus de balayage inverse de subdivision de macrobloc et de sous-macrobloc

Les macroblocs ou sous-macroblocks peuvent être divisés, et les subdivisions sont balayées pour l'interprédiction comme indiqué à la Figure 6-5. Les rectangles extérieurs se rapportent aux échantillons dans un macrobloc ou sous-macrobloc, respectivement. Les rectangles se rapportent aux subdivisions. Le nombre dans chaque rectangle spécifie l'indice du balayage inverse de division de macrobloc ou du balayage inverse de division de sous-macrobloc.

Les fonctions MbPartWidth(), MbPartHeight(), SubMbPartWidth() et SubMbPartHeight() décrivant la largeur et la hauteur des divisions de macrobloc et des divisions de sous-macrobloc sont spécifiées au Tableau 7-10, au Tableau 7-11, au Tableau 7-14 et au Tableau 7-15. MbPartWidth() et MbPartHeight() sont mis aux valeurs appropriées pour chaque macrobloc, en

fonction du type de macrobloc. $\text{SubMbPartWidth}()$ et $\text{SubMbPartHeight}()$ sont mis aux valeurs appropriées pour chaque sous-macrobloc d'un macrobloc avec mb_type égal à P_8x8, P_8x8ref0, ou B_8x8, selon le type du sous-macrobloc.



H.264_F6-5

Figure 6-5 – Divisions de macrobloc, divisions de sous-macrobloc, balayages de division de macrobloc et balayages de division de sous-macrobloc

6.4.2.1 Processus de balayage inverse de division de macrobloc

L'entrée de ce processus est l'indice mbPartIdx d'une division d'un macrobloc.

Le résultat de ce processus est la localisation (x , y) de l'échantillon luma supérieur gauche pour la division de macrobloc mbPartIdx se rapportant à l'échantillon supérieur gauche du macrobloc.

Le processus de balayage inverse de la division de macrobloc est spécifié par:

$$x = \text{InverseRasterScan}(\text{mbPartIdx}, \text{MbPartWidth}(\text{mb_type}), \text{MbPartHeight}(\text{mb_type}), 16, 0) \quad (6-9)$$

$$y = \text{InverseRasterScan}(\text{mbPartIdx}, \text{MbPartWidth}(\text{mb_type}), \text{MbPartHeight}(\text{mb_type}), 16, 1) \quad (6-10)$$

6.4.2.2 Processus de balayage inverse de division de sous-macrobloc

Les entrées à ce processus sont l'indice mbPartIdx d'une division de macrobloc et l'indice subMbPartIdx d'une division de sous-macrobloc.

Le résultat de ce processus est la localisation (x , y) de l'échantillon luma supérieur gauche pour la division de sous-macrobloc subMbPartIdx relative à l'échantillon supérieur gauche du sous-macrobloc.

Le processus de balayage de division de sous-macrobloc inverse est spécifié comme suit.

– Si mb_type est égal à P_8x8, P_8x8ref0, ou B_8x8,

$$x = \text{InverseRasterScan}(\text{subMbPartIdx}, \text{SubMbPartWidth}(\text{sub_mb_type}[\text{mbPartIdx}]), \text{SubMbPartHeight}(\text{sub_mb_type}[\text{mbPartIdx}]), 8, 0) \quad (6-11)$$

$$y = \text{InverseRasterScan}(\text{subMbPartIdx}, \text{SubMbPartWidth}(\text{sub_mb_type}[\text{mbPartIdx}]), \text{SubMbPartHeight}(\text{sub_mb_type}[\text{mbPartIdx}]), 8, 1) \quad (6-12)$$

– Autrement,

$$x = \text{InverseRasterScan}(\text{subMbPartIdx}, 4, 4, 8, 0) \quad (6-13)$$

$$y = \text{InverseRasterScan}(\text{subMbPartIdx}, 4, 4, 8, 1) \quad (6-14)$$

6.4.3 Processus de balayage inverse de bloc luma 4x4

L'entrée à ce processus est l'indice luma4x4BlkIdx d'un bloc luma 4x4.

Le résultat de ce processus est la localisation (x, y) de l'échantillon luma supérieur gauche pour le bloc luma 4x4 avec l'indice luma4x4BlkIdx relatif à l'échantillon luma supérieur gauche du macrobloc.

La Figure 6-6 montre le balayage pour les blocs luma 4x4.

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

Figure 6-6 – Balayage pour les blocs luma 4x4

Le processus de balayage inverse de bloc luma 4x4 est spécifié par

$$x = \text{InverseRasterScan}(\text{luma4x4BlkIdx} / 4, 8, 8, 16, 0) + \text{InverseRasterScan}(\text{luma4x4BlkIdx} \% 4, 4, 4, 8, 0) \quad (6-15)$$

$$y = \text{InverseRasterScan}(\text{luma4x4BlkIdx} / 4, 8, 8, 16, 1) + \text{InverseRasterScan}(\text{luma4x4BlkIdx} \% 4, 4, 4, 8, 1) \quad (6-16)$$

6.4.4 Processus de déduction de la disponibilité pour les adresses de macrobloc

L'entrée pour ce processus est une adresse de macrobloc mbAddr.

Le résultat de ce processus est la disponibilité du macrobloc mbAddr.

NOTE – La signification de la disponibilité est déterminée lorsque ce processus est invoqué.

Le macrobloc est marqué comme disponible, à moins qu'une des conditions suivantes ne soit Vraie, auquel cas le macrobloc doit être marqué comme indisponible:

- mbAddr < 0
- mbAddr > CurrMbAddr
- le macrobloc avec l'adresse mbAddr appartient à une tranche différente de la tranche en cours.

6.4.5 Processus de déduction pour les adresses de macrobloc du voisinage et leur disponibilité

Ce processus ne peut être invoqué que lorsque MbaffFrameFlag est égal à 0.

Les résultats de ce processus sont:

- mbAddrA: l'adresse et l'état de disponibilité du macrobloc à gauche du macrobloc en cours;
- mbAddrB: l'adresse et l'état de disponibilité du macrobloc au-dessus du macrobloc en cours;
- mbAddrC: l'adresse et l'état de disponibilité du macrobloc droit au-dessus du macrobloc en cours;
- mbAddrD: l'adresse et l'état de disponibilité du macrobloc gauche au-dessus du macrobloc en cours.

La Figure 6-7 montre les localisations spatiales relatives des macroblocs avec mbAddrA, mbAddrB, mbAddrC et mbAddrD par rapport au macrobloc en cours avec CurrMbAddr.

mbAddrD	mbAddrB	mbAddrC
mbAddrA	CurrMbAddr	

Figure 6-7 – Macroblocs voisins pour un macrobloc donné

L'entrée du processus au § 6.4.4 est $mbAddrA = CurrMbAddr - 1$ et le résultat dépend de la disponibilité du macrobloc mbAddrA. De plus, mbAddrA est marqué comme non disponible lorsque $CurrMbAddr \% PicWidthInMbs$ est égal à 0.

L'entrée du processus au § 6.4.4 est $mbAddrB = CurrMbAddr - PicWidthInMbs$ et le résultat dépend de la disponibilité du macrobloc mbAddrB.

L'entrée du processus dans le § 6.4.4 est $mbAddrC = CurrMbAddr - PicWidthInMbs + 1$ et le résultat dépend de la disponibilité du macrobloc mbAddrC. De plus, mbAddrC est marqué comme non disponible lorsque $(CurrMbAddr + 1) \% PicWidthInMbs$ est égal à 0.

L'entrée du processus au § 6.4.4 est $mbAddrD = CurrMbAddr - PicWidthInMbs - 1$ et le résultat dépend de la disponibilité du macrobloc mbAddrD. De plus, mbAddrD est marqué comme non disponible lorsque $CurrMbAddr \% PicWidthInMbs$ est égal à 0.

6.4.6 Processus de déduction des adresses de macrobloc du voisinage et de leur disponibilité dans les trames MBAFF

Ce processus ne peut être invoqué que lorsque MbaffFrameFlag est égal à 1.

Les résultats de ce processus sont:

- mbAddrA: l'adresse et l'état de disponibilité du macrobloc supérieur de la paire de macroblocs à gauche de la paire de macroblocs en cours;
- mbAddrB: l'adresse et l'état de disponibilité du macrobloc supérieur de la paire de macroblocs au-dessus de la paire de macroblocs en cours;
- mbAddrC: l'adresse et l'état de disponibilité du macrobloc supérieur de la paire de macroblocs à droite au-dessus de la paire de macroblocs en cours;
- mbAddrD: l'adresse et l'état de disponibilité du macrobloc supérieur de la paire de macroblocs à gauche au-dessus de la paire de macroblocs en cours.

La Figure 6-8 montre les localisations spatiales relatives des macroblocs avec mbAddrA, mbAddrB, mbAddrC et mbAddrD par rapport au macrobloc en cours avec CurrMbAddr.

mbAddrA, mbAddrB, mbAddrC et mbAddrD ont des valeurs identiques sans considération de savoir si le macrobloc en cours est le macrobloc supérieur ou inférieur d'une paire de macroblocs.

mbAddrD	mbAddrB	mbAddrC
mbAddrA	CurrMbAddr or	
	CurrMbAddr	

Figure 6-8 – Macroblocs du voisinage pour un macrobloc donné dans des trames MBAFF

L'entrée du processus au § 6.4.4 est $mbAddrA = 2 * (CurrMbAddr / 2 - 1)$ et le résultat dépend de la disponibilité du macrobloc mbAddrA. De plus, mbAddrA est marqué comme non disponible lorsque $(CurrMbAddr / 2) \% PicWidthInMbs$ est égal à 0.

L'entrée du processus au § 6.4.4 est $mbAddrB = 2 * (CurrMbAddr / 2 - PicWidthInMbs)$ et le résultat dépend de la disponibilité du macrobloc mbAddrB.

L'entrée du processus au § 6.4.4 est $mbAddrC = 2 * (CurrMbAddr / 2 - PicWidthInMbs + 1)$ et le résultat dépend de la disponibilité du macrobloc mbAddrC. De plus, mbAddrC est marqué comme non disponible lorsque $(CurrMbAddr / 2 + 1) \% PicWidthInMbs$ est égal à 0.

L'entrée du processus au § 6.4.4 est $mbAddrD = 2 * (CurrMbAddr / 2 - PicWidthInMbs - 1)$ et le résultat dépend de la disponibilité du macrobloc mbAddrD. De plus, mbAddrD est marqué comme non disponible lorsque $(CurrMbAddr / 2) \% PicWidthInMbs$ est égal à 0.

6.4.7 Processus de déduction pour les macroblocs, blocs et subdivisions du voisinage

Le paragraphe 6.4.7.1 spécifie le processus de déduction pour les macroblocs du voisinage.

Le paragraphe 6.4.7.2 spécifie le processus de déduction pour les blocs luma 8x8 du voisinage.

Le paragraphe 6.4.7.3 spécifie le processus de déduction pour les blocs luma 4x4 du voisinage.

Le paragraphe 6.4.7.4 spécifie le processus de déduction pour les blocs chroma 4x4 du voisinage.

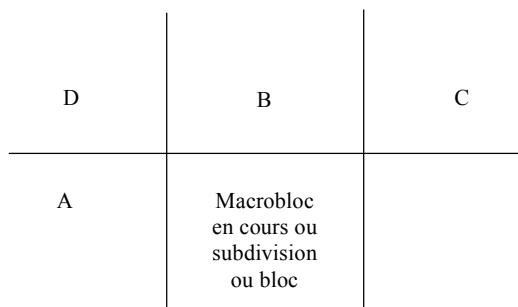
Le paragraphe 6.4.7.5 spécifie le processus de déduction pour les subdivisions du voisinage.

Le Tableau 6-2 spécifie les valeurs pour la différence de localisation luma (xD, yD) pour l'entrée et le remplacement pour N dans mbAddrN, mbPartIdxN, subMbPartIdxN, luma8x8BlkIdxN, luma4x4BlkIdxN et chroma4x4BlkIdxN pour la sortie. Ces allocations d'entrée et de sortie sont utilisées aux § 6.4.7.1 à 6.4.7.5. La variable predPartWidth est spécifiée lorsqu'on se rapporte au Tableau 6-2.

Tableau 6-2 – Spécification des allocations d'entrée et de sortie pour les § 6.4.7.1 à 6.4.7.5

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

La Figure 6-9 illustre la localisation relative des macroblocs, blocs ou subdivisions A, B, C et D du voisinage par rapport au macrobloc, subdivision ou bloc en cours lorsque le macrobloc, subdivision, ou bloc en cours est en mode de codage de trame.



H.264_F6-9

Figure 6-9 – Détermination des macroblocs, blocs et subdivisions du voisinage (pour information)

6.4.7.1 Processus de déduction pour les macroblocs du voisinage

Les résultats de ce processus sont:

- mbAddrA: l'adresse des macroblocs à la gauche du macrobloc en cours et son état de disponibilité; et
- mbAddrB: l'adresse du macroblocs au-dessus du macrobloc en cours et son état de disponibilité.

mbAddrN (N étant A ou B) est déduit comme suit.

- La différence de localisation luma (xD, yD) est établie conformément au Tableau 6-2.
- Le processus de déduction pour les localisations du voisinage, comme spécifié au § 6.4.8, est invoqué pour les localisations luma avec (xN, yN) égal à (xD, yD), et le résultat est alloué à mbAddrN.

6.4.7.2 Processus de déduction pour les blocs luma 8x8 du voisinage

L'entrée pour ce processus est un indice luma8x8BlkIdx de bloc luma 8x8.

L'indice luma8x8BlkIdx spécifie les blocs luma 8x8 d'un macrobloc dans un balayage de trame.

Les résultats de ce processus sont:

- mbAddrA: égal à CurrMbAddr ou à l'adresse des macroblocs à gauche du macrobloc en cours et son état de disponibilité;
- luma8x8BlkIdxA: indice du bloc luma 8x8 à gauche du bloc 8x8 avec l'indice luma8x8BlkIdx et son état de disponibilité;
- mbAddrB: égal à CurrMbAddr ou à l'adresse des macroblocs au-dessus du macrobloc en cours et son état de disponibilité;
- luma8x8BlkIdxB: indice du bloc luma 8x8 au-dessus du bloc 8x8 avec l'indice luma8x8BlkIdx et son état de disponibilité.

mbAddrN et luma8x8BlkIdxN (N étant A ou B) sont déduits comme suit.

- La différence de localisation luma (xD, yD) est établie conformément au Tableau 6-2.

- La localisation luma (xN, yN) est spécifiée par:

$$xN = (\text{luma8x8BlkIdx} \% 2) * 8 + xD \quad (6-17)$$

$$yN = (\text{luma8x8BlkIdx} / 2) * 8 + yD \quad (6-18)$$

- Le processus de déduction pour les localisations du voisinage, comme spécifié au § 6.4.8, est invoqué pour les localisations luma avec (x_N , y_N) comme entrée et le résultat est alloué à $mbAddrN$ et (x_W , y_W).
- La variable $luma8x8BlkIdxN$ est déduite comme suit.
 - Si $mbAddrN$ n'est pas disponible, $luma8x8BlkIdxN$ est marqué comme non disponible.
 - Autrement ($mbAddrN$ est disponible), le bloc luma 8x8 dans le macrobloc $mbAddrN$ couvrant la localisation luma (x_W , y_W) doit être alloué à $luma8x8BlkIdxN$.

6.4.7.3 Processus de déduction pour les blocs luma 4x4 du voisinage

L'entrée pour ce processus est un indice $luma4x4BlkIdx$ de bloc luma.

Les résultats de ce processus sont:

- $mbAddrA$: égal à $CurrMbAddr$ ou à l'adresse du macrobloc à gauche du macrobloc en cours et son état de disponibilité;
- $luma4x4BlkIdxA$: indice du bloc luma 4x4 à gauche du bloc 4x4 avec l'indice $luma4x4BlkIdx$ et son état de disponibilité;
- $mbAddrB$: égal à $CurrMbAddr$ ou à l'adresse du macrobloc au-dessus du macrobloc en cours et son état de disponibilité;
- $luma4x4BlkIdxB$: indice du bloc luma 4x4 au-dessus du bloc 4x4 avec l'indice $luma4x4BlkIdx$ et son état de disponibilité.

$mbAddrN$ et $luma4x4BlkIdxN$ (N étant A ou B) sont déduits comme suit.

- La différence de localisation luma (x_D , y_D) est établie conformément au Tableau 6-2.
- Le processus de balayage inverse de bloc luma 4x4, comme spécifié au § 6.4.3, est invoqué avec $luma4x4BlkIdx$ comme étant l'entrée et (x , y) comme résultat.
- La localisation luma (x_N , y_N) est spécifiée par:

$$x_N = x + x_D \quad (6-19)$$

$$y_N = y + y_D \quad (6-20)$$

- Le processus de déduction pour les localisations du voisinage, comme spécifié au § 6.4.8, est invoqué pour les localisations luma avec (x_N , y_N) comme entrée et le résultat est alloué à $mbAddrN$ et (x_W , y_W).
- La variable $luma4x4BlkIdxN$ est déduite comme suit.
 - Si $mbAddrN$ n'est pas disponible, $luma4x4BlkIdxN$ est marqué comme non disponible.
 - Autrement ($mbAddrN$ est disponible), le bloc luma 4x4 dans le macrobloc $mbAddrN$ couvrant la localisation luma (x_W , y_W) doit être alloué à $luma4x4BlkIdxN$.

6.4.7.4 Processus de déduction pour les blocs chroma 4x4 du voisinage

L'entrée est ici un bloc chroma 4x4 en cours $chroma4x4BlkIdx$.

Les résultats de ce processus sont:

- $mbAddrA$: égal à $CurrMbAddr$ ou à l'adresse du macrobloc à gauche du macrobloc en cours et son état de disponibilité;
- $chroma4x4BlkIdxA$: indice du bloc chroma 4x4 à gauche du bloc chroma 4x4 avec l'indice $chroma4x4BlkIdx$ et son état de disponibilité;
- $mbAddrB$: égal à $CurrMbAddr$ ou à l'adresse du macrobloc au-dessus du macrobloc en cours et son état de disponibilité;
- $chroma4x4BlkIdxB$: indice du bloc chroma 4x4 au-dessus de l'indice $chroma4x4BlkIdx$ de bloc chroma 4x4 et son état de disponibilité.

Le processus de déduction pour le bloc luma 8x8 voisin est invoqué avec $\text{luma8x8BlkIdx} = \text{chroma4x4BlkIdx}$ comme entrée et avec mbAddrA , $\text{chroma4x4BlkIdxA} = \text{luma8x8BlkIdxA}$, mbAddrB , et $\text{chroma4x4BlkIdxB} = \text{luma8x8BlkIdxB}$ comme résultat.

6.4.7.5 Processus de déduction pour les subdivisions du voisinage

Les entrées à ce processus sont:

- un indice de subdivision de macrobloc mbPartIdx ;
- un type de sous-macrobloc en cours currSubMbType ;
- un indice de subdivision de sous-macrobloc subMbPartIdx .

Les résultats de ce processus sont:

- $\text{mbAddrA}\backslash\text{mbPartIdxA}\backslash\text{subMbPartIdxA}$: spécifiant la subdivision de macrobloc ou sous-macrobloc à gauche du macrobloc en cours et son état de disponibilité, ou la subdivision de sous-macrobloc $\text{CurrMbAddr}\backslash\text{mbPartIdx}\backslash\text{subMbPartIdx}$ et son état de disponibilité;
- $\text{mbAddrB}\backslash\text{mbPartIdxB}\backslash\text{subMbPartIdxB}$: spécifiant la subdivision de macrobloc ou sous-macrobloc au-dessus du macrobloc en cours et son état de disponibilité, ou la subdivision de sous-macrobloc $\text{CurrMbAddr}\backslash\text{mbPartIdx}\backslash\text{subMbPartIdx}$ et son état de disponibilité;
- $\text{mbAddrC}\backslash\text{mbPartIdxC}\backslash\text{subMbPartIdxC}$: spécifiant la subdivision du macrobloc ou sous-macrobloc au-dessus à droite du macrobloc en cours et son état de disponibilité, ou la subdivision de sous-macrobloc $\text{CurrMbAddr}\backslash\text{mbPartIdx}\backslash\text{subMbPartIdx}$ et son état de disponibilité;
- $\text{mbAddrD}\backslash\text{mbPartIdxD}\backslash\text{subMbPartIdxD}$: spécifiant la subdivision de macrobloc ou sous-macrobloc au-dessus à gauche du macrobloc en cours et son état de disponibilité, ou la subdivision de sous-macrobloc $\text{CurrMbAddr}\backslash\text{mbPartIdx}\backslash\text{subMbPartIdx}$ et son état de disponibilité.

mbAddrN , mbPartIdxN et subMbPartIdx (N étant A, B, C ou D) sont déduits comme suit.

- Le processus de balayage inverse de subdivision de macrobloc, comme décrit au § 6.4.2.1, est invoqué avec mbPartIdx comme entrée et (x, y) comme résultat.
- La localisation de l'échantillon luma supérieur gauche à l'intérieur d'une subdivision de macrobloc (x_S, y_S) est déduite comme suit.
 - Si mb_type est égal à P_8x8 , P_8x8ref0 ou B_8x8 , le processus de balayage inverse de subdivision de sous-macrobloc, comme décrit au § 6.4.2.2, est invoqué avec subMbPartIdx comme entrée et (x_S, y_S) comme résultat.
 - Autrement, (x_S, y_S) sont établis à $(0, 0)$.
- La variable predPartWidth du Tableau 6-2 est spécifiée comme suit.
 - Si mb_type est égal à P_Skip , B_Skip , ou B_Direct_16x16 , $\text{predPartWidth} = 16$.
 - Autrement, si mb_type est égal à B_8x8 , on applique ce qui suit.
 - Si currSubMbType est égal à B_Direct_8x8 , $\text{predPartWidth} = 16$.

NOTE – Lorsque currSubMbType est égal à B_Direct_8x8 et que $\text{direct_spatial_mv_pred_flag}$ est égal à 1, le vecteur cinétique prédit est le vecteur cinétique prédit pour le macrobloc tout entier.
 - Autrement, $\text{predPartWidth} = \text{SubMbPartWidth}(\text{sub_mb_type}[\text{mbPartIdx}])$.
 - Autrement, si mb_type est égal à P_8x8 ou P_8x8ref0 , $\text{predPartWidth} = \text{SubMbPartWidth}(\text{sub_mb_type}[\text{mbPartIdx}])$.
 - Autrement, $\text{predPartWidth} = \text{MbPartWidth}(\text{mb_type})$.

- La différence de localisation luma (x_D, y_D) est établie conformément au Tableau 6-2.
- La localisation luma voisine (x_N, y_N) est spécifiée par:

$$x_N = x + x_S + x_D \quad (6-21)$$

$$y_N = y + y_S + y_D \quad (6-22)$$
- Le processus de déduction pour les localisations du voisinage, comme spécifié au § 6.4.8, est invoqué pour les localisations luma avec (x_N, y_N) comme entrée et le résultat est alloué à $mbAddrN$ et (x_W, y_W).
- Selon $mbAddrN$, on applique ce qui suit.
 - Si $mbAddrN$ n'est pas disponible, la subdivision de macrobloc ou sous-macrobloc $mbAddrN \backslash mbPartIdxN \backslash subMbPartIdxN$ est marquée comme non disponible.
 - Autrement ($mbAddrN$ est disponible), on applique ce qui suit.
 - La subdivision de macrobloc dans le macrobloc $mbAddrN$ couvrant la localisation luma (x_W, y_W) doit être allouée à $mbPartIdxN$ et la sous-subdivision de macrobloc à l'intérieur de la subdivision de macrobloc $mbPartIdxN$ couvrant l'échantillon (x_W, y_W) dans le macrobloc $mbAddrN$ doit être allouée à $subMbPartIdxN$.
 - Lorsque la subdivision donnée par $mbPartIdxN$ et $subMbPartIdxN$ n'est pas encore décodée, la subdivision de macrobloc $mbPartIdxN$ et la sous-subdivision de macrobloc $subMbPartIdxN$ sont marquées comme non disponibles.

NOTE – C'est le cas de cette dernière condition lorsque, par exemple, $mbPartIdx = 2$, $subMbPartIdx = 3$, $x_D = 4$, $y_D = -1$, c'est-à-dire, lorsqu'est exigé C voisin du dernier bloc luma 4×4 du troisième sous-macrobloc.

6.4.8 Processus de déduction pour les localisations du voisinage

L'entrée pour ce processus est une localisation luma ou chroma (x_N, y_N) exprimée par rapport au coin supérieur gauche du macrobloc en cours.

Les résultats de ce processus sont:

- $mbAddrN$: égal à $CurrMbAddr$ ou à l'adresse du macrobloc voisin qui contient (x_N, y_N) et son état de disponibilité;
- (x_W, y_W): la localisation (x_N, y_N) exprimée par rapport au coin supérieur gauche du macrobloc $mbAddrN$ (plutôt que par rapport au coin supérieur gauche du macrobloc en cours).

Soit $maxWH$ une variable spécifiant une valeur maximale des composants de la localisation x_N, y_N, x_W et y_W . $maxWH$ est déduite comme suit.

- Si ce processus est invoqué pour les localisations luma du voisinage,

$$maxWH = 16 \quad (6-23)$$
- Autrement (ce processus est invoqué pour les localisations chroma du voisinage),

$$maxWH = 8 \quad (6-24)$$

Selon la variable $MbaffFrameFlag$, les localisations luma du voisinage sont déduites comme suit.

- Si $MbaffFrameFlag$ est égal à 0, on applique la spécification pour les localisations luma du voisinage dans les champs et dans les trames non MBAFF, comme décrit au § 6.4.8.1.
- Autrement ($MbaffFrameFlag$ est égal à 1), on applique la spécification pour les localisations luma du voisinage dans les trames MBAFF comme décrit au § 6.4.8.2.

6.4.8.1 Spécification pour les localisations luma du voisinage dans les champs et les trames non MBAFF

Les spécifications du présent paragraphe sont appliquées lorsque MbaffFrameFlag est égal à 0.

Le processus de déduction pour les adresses de macroblocs de voisinage, et leur disponibilité dans le § 6.4.5 est invoqué avec mbAddrA, mbAddrB, mbAddrC, et mbAddrD ainsi que leur état de disponibilité comme résultat.

Le Tableau 6-3 spécifie mbAddrN en fonction de (xN, yN).

Tableau 6-3 – Spécification de mbAddrN

xN	yN	mbAddrN
< 0	< 0	mbAddrD
< 0	0 .. maxWH – 1	mbAddrA
0 .. maxWH – 1	< 0	mbAddrB
0 .. maxWH – 1	0 .. maxWH – 1	CurrMbAddr
> maxWH – 1	< 0	mbAddrC
> maxWH – 1	0 .. maxWH – 1	Indisponible
	> maxWH – 1	Indisponible

La localisation luma de voisinage (xW, yW) se rapportant au coin supérieur gauche du macrobloc mbAddrN est déduite comme:

$$xW = (xN + \text{maxWH}) \% \text{maxWH} \quad (6-25)$$

$$yW = (yN + \text{maxWH}) \% \text{maxWH} \quad (6-26)$$

6.4.8.2 Spécification pour les localisations luma du voisinage dans les trames MBAFF

Les spécifications du présent paragraphe s'appliquent lorsque MbaffFrameFlag est égal à 1.

Le processus de déduction pour les adresses de macroblocs du voisinage et leur disponibilité dans le § 6.4.6 est invoqué avec mbAddrA, mbAddrB, mbAddrC, et mbAddrD ainsi que leur état de disponibilité comme résultat.

Le Tableau 6-4 spécifie les adresses de macroblocs mbAddrN et yM en deux étapes ordonnées:

- Spécification d'une adresse de macrobloc mbAddrX en fonction de (xN, yN) et des variables suivantes:
 - La variable currMbFrameFlag est déduite comme suit:
 - Si le macrobloc avec l'adresse CurrMbAddr est un macrobloc de trame, currMbFrameFlag est mis égal à 1.
 - Autrement (le macrobloc avec l'adresse CurrMbAddr est un macrobloc de champ), currMbFrameFlag est mis égal à 0.
 - La variable mbIsTopMbFlag est déduite comme suit.
 - Si le macrobloc avec l'adresse CurrMbAddr est un macrobloc supérieur (CurrMbAddr % 2 est égal à 0), mbIsTopMbFlag est mis égal à 1;
 - Autrement (le macrobloc avec l'adresse CurrMbAddr est un macrobloc inférieur, CurrMbAddr % 2 est égal à 1), mbIsTopMbFlag est mis égal à 0.

2. Selon la disponibilité de mbAddrX, on applique ce qui suit.
- Si mbAddrX n'est pas disponible, mbAddrN est marqué comme indisponible.
 - Autrement (mbAddrX est disponible), mbAddrN est marqué comme disponible et le Tableau 6-4 spécifie mbAddrN et yM en fonction de (xN, yN), currMbFrameFlag, mbIsTopMbFlag, et la variable mbAddrXFrameFlag, qui est déduite comme suit:
 - si le macrobloc mbAddrX est un macrobloc de trame, mbAddrXFrameFlag est mis égal à 1;
 - autrement (le macrobloc mbAddrX est un macrobloc de champ), mbAddrXFrameFlag est mis égal à 0.

Les valeurs non spécifiées (na) des fanions ci-dessus dans le Tableau 6-4 indiquent que la valeur du fanion correspondant n'est pas pertinente pour les rangées en cours du tableau.

Tableau 6-4 – Spécification de mbAddrN et yM

xN	yN	currMbFrameFlag	mbIsTopMbFlag	mbAddrX	mbAddrXFrameFlag	condition supplémentaire	mbAddrN	yM	
< 0	< 0	1	1	mbAddrD			mbAddrD + 1	yN	
			0	mbAddrA	1		mbAddrA	yN	
				0		0		mbAddrA + 1	$(yN + \text{maxWH}) \gg 1$
		0	mbAddrD	1		1		mbAddrD + 1	$2 * yN$
			0	mbAddrD	0		mbAddrD	yN	
			0	mbAddrD			mbAddrD + 1	yN	
< 0	$0 \dots \text{maxWH} - 1$	1	1	mbAddrA	1		mbAddrA	yN	
					0	$yN \% 2 == 0$	mbAddrA	$yN \gg 1$	
						$yN \% 2 != 0$	mbAddrA + 1	$yN \gg 1$	
			0	mbAddrA	1		mbAddrA + 1	yN	
					0	$yN \% 2 == 0$	mbAddrA	$(yN + \text{maxWH}) \gg 1$	
						$yN \% 2 != 0$	mbAddrA + 1	$(yN + \text{maxWH}) \gg 1$	
		0	mbAddrA	1	1	$yN < (\text{maxWH} / 2)$	mbAddrA	$yN \ll 1$	
						$yN \geq (\text{maxWH} / 2)$	mbAddrA + 1	$(yN \ll 1) - \text{maxWH}$	
				0	mbAddrA	1	$yN < (\text{maxWH} / 2)$	mbAddrA	$(yN \ll 1) + 1$
							$yN \geq (\text{maxWH} / 2)$	mbAddrA + 1	$(yN \ll 1) + 1 - \text{maxWH}$
			0		mbAddrA + 1	yN			
$0 \dots \text{maxWH} - 1$	< 0	1	1	mbAddrB			mbAddrB + 1	yN	
			0	CurrMbAddr			CurrMbAddr - 1	yN	
		0	1	mbAddrB	1		mbAddrB + 1	$2 * yN$	
				0	mbAddrB	0		mbAddrB	yN
			0	mbAddrB			mbAddrB + 1	yN	
$0 \dots \text{maxWH} - 1$	$0 \dots \text{maxWH} - 1$			CurrMbAddr			CurrMbAddr	yN	
> maxWH - 1	< 0	1	1	mbAddrC			mbAddrC + 1	yN	
			0	Indisponible			Indisponible	na	
		0	1	mbAddrC	1		mbAddrC + 1	$2 * yN$	
				0	mbAddrC	0		mbAddrC	yN
			0	mbAddrC			mbAddrC + 1	yN	
> maxWH - 1	$0 \dots \text{maxWH} - 1$			Indisponible			Indisponible	na	
	> maxWH - 1			Indisponible			Indisponible	na	

La localisation luma voisine (x_W, y_W) se rapportant au coin supérieur gauche du macrobloc $mbAddrN$ est déduite par:

$$x_W = (x_N + \max_{WH}) \% \max_{WH} \quad (6-27)$$

$$y_W = (y_M + \max_{WH}) \% \max_{WH} \quad (6-28)$$

7 Syntaxe et sémantique

7.1 Méthode de description de la syntaxe en forme tabulaire

Les tableaux de syntaxe décrivent un super ensemble de la syntaxe de tous les flux binaires admis en entrée. Des contraintes syntaxiques supplémentaires peuvent être spécifiées dans d'autres paragraphes.

NOTE – Un décodeur réel devrait implémenter des moyens d'identification des points d'entrée dans le flux binaire pour identifier et traiter les flux binaires non conformes. Les méthodes d'identification et de traitement des erreurs et autres situations similaires ne sont pas décrites ici.

Le tableau suivant donne une liste d'exemples de pseudo-code utilisé pour décrire la syntaxe. Lorsque `syntax_element` apparaît, il spécifie qu'un élément de données est lu (extrait) à partir du flux binaire et du pointeur de flux binaire.

	C	Descripteur
/* Une instruction peut être un élément syntaxique avec une catégorie de syntaxe et un descripteur associés ou peut être une expression utilisée pour spécifier les conditions pour l'existence, le type, et la quantité des éléments syntaxiques, comme dans les deux exemples suivants */		
syntax_element	3	ue(v)
instruction posant une condition		
/* Un groupe d'instructions entre des accolades est une instruction composée et elle est traitée fonctionnellement comme une instruction unique */		
{		
instruction		
instruction		
...		
}		
/* Une structure "tandis que" spécifie un essai pour savoir si une condition est Vraie, et si elle est Vraie, elle spécifie l'évaluation répétitive d'une instruction (ou instruction composée) jusqu'à ce que la condition ne soit plus Vraie */		
tandis que(condition)		
instruction		
/* Une structure "faire ... tandis que" spécifie l'évaluation d'une instruction une seule fois, suivie par un essai pour savoir si une condition est Vraie, et si elle est Vraie, elle spécifie une évaluation répétée de l'instruction jusqu'à ce que la condition ne soit plus Vraie */		
faire		
instruction		
tandis que(condition)		
/* Une structure "si ... autrement" spécifie un essai de savoir si une condition est Vraie, et si la condition est Vraie, elle spécifie l'évaluation d'une instruction primaire, et autrement, elle spécifie l'évaluation d'une instruction de remplacement. La partie "autrement" de la structure et l'instruction de remplacement associée est omise s'il n'est pas besoin d'une évaluation d'instruction de remplacement */		
si(condition)		
instruction primaire		
autrement		
instruction de remplacement		
/* Une structure "pour" spécifie l'évaluation d'une instruction initiale, suivie par l'essai d'une condition, et si la condition est Vraie, elle spécifie des évaluations répétées d'une instruction primaire suivie d'une instruction ultérieure jusqu'à ce que la condition ne soit plus Vraie */		
pour(instruction initiale; condition; instruction ultérieure)		
instruction primaire		

7.2 Spécification de fonctions, catégories et descripteurs syntaxiques

Les fonctions présentées ici sont utilisées dans la description syntaxique. Ces fonctions supposent l'existence d'un pointeur de flux binaire avec une indication de la position du prochain bit qui sera lu par le processus de décodage à partir du flux binaire.

`byte_aligned()` (*aligné octet*) spécifié comme suit.

- Si la position actuelle dans le flux binaire est à la frontière d'un octet, c'est-à-dire si le prochain bit dans le flux binaire est le premier bit d'un octet, la valeur résultante de `byte_aligned()` est égale à VRAI.
- Autrement, la valeur résultante de `byte_aligned()` est égale à FAUX.

`more_data_in_byte_stream()` (*plus de données dans le flux d'octet*), qui n'est utilisé que dans la structure syntaxique de flux d'octets d'unité NAL spécifiée à l'Annexe B, est spécifié comme suit.

- Si plus de données suivent dans le flux d'octets, la valeur résultante de `more_data_in_byte_stream()` est égale à VRAI.
- Autrement, la valeur résultante de `more_data_in_byte_stream()` est égale à FAUX.

`more_rbsp_data()` (*plus de données de charge utile RBSP*) est spécifié comme suit.

- S'il y a plus de données dans une charge utile RBSP avant `rbsp_trailing_bits()`, la valeur résultante de `more_rbsp_data()` est égale à VRAI.
- Autrement, la valeur résultante de `more_rbsp_data()` est égale à FAUX.

La méthode permettant de déterminer s'il y a plus de données dans la charge utile RBSP est spécifiée par l'application (ou à l'Annexe B pour les applications qui utilisent le format de flux d'octets).

`more_rbsp_trailing_data()` (*plus de données de traîne de charge utile RBSP*) est spécifié comme suit.

- S'il y a plus de données dans une charge utile RBSP, la valeur résultante de `more_rbsp_trailing_data()` est égale à VRAI.
- Autrement, la valeur résultante de `more_rbsp_trailing_data()` est égale à FAUX.

`next_bits(n)` fournit les bits suivants dans le flux binaire pour des besoins de comparaison, sans faire avancer le pointeur de flux binaire. Il permet de jeter un coup d'œil aux *n* prochains bits dans le flux binaire avec *n* comme argument. Lorsqu'il est utilisé au sein du flux d'octets, comme spécifié à l'Annexe B, `next_bits(n)` retourne une valeur de 0 si moins de *n* bits restent dans le flux d'octets.

`read_bits(n)` lit les *n* prochains bits du flux binaire et avance le pointeur de flux binaire de *n* positions de bit. Lorsque *n* est égal à 0, `read_bits(n)` est spécifié pour retourner une valeur égale à 0 et ne pas avancer le pointeur de flux binaire.

Catégories (indiqué dans le tableau par **C**) spécifie la subdivision de données de tranche en au plus trois subdivisions de données de tranche. La subdivision de données de tranche A contient tous les éléments syntaxiques de catégorie 2. La subdivision de données de tranche B contient tous les éléments syntaxiques de catégorie 3. La subdivision de données de tranche C contient tous les éléments syntaxiques de catégorie 4. La signification des autres valeurs de catégorie n'est pas spécifiée. Pour certains éléments syntaxiques, deux valeurs de catégorie, séparées par une barre verticale, sont utilisées. Dans ces cas, la valeur de catégorie à appliquer est spécifiée plus loin dans le texte. Pour les structures syntaxiques utilisées au sein d'autres structures syntaxiques, la liste des catégories de tous les éléments syntaxiques trouvés au sein de la structure syntaxique incluse est séparée par une barre verticale. Un élément syntaxique ou structure syntaxique avec la catégorie marquée "Toutes" est présent au sein de toutes les structures syntaxiques qui incluent cet élément ou structure syntaxique. Pour les structures syntaxiques utilisées au sein d'autres structures syntaxiques, on considère qu'une valeur numérique de catégorie fournie dans un tableau syntaxique

à l'endroit de l'inclusion d'une structure syntaxique contenant un élément syntaxique dont la catégorie est marquée "Toutes" s'applique aux éléments syntaxiques de catégorie "Toutes".

Les descripteurs suivants spécifient le processus d'analyse grammaticale de chaque élément syntaxique. Pour certains éléments syntaxiques, deux descripteurs, séparés par une barre verticale, sont utilisés. Dans ces cas, le descripteur de gauche s'applique lorsque `entropy_coding_mode_flag` (*fanion de mode de codage entropique*) est égal à 0 et le descripteur de droite s'applique lorsque `entropy_coding_mode_flag` est égal à 1.

- `ae(v)`: élément syntaxique arithmétique à codage entropique adaptable au contexte. Le processus d'analyse grammaticale pour ce descripteur est spécifié au § 9.3.
- `b(8)`: octet ayant un schéma de chaîne binaire (8 bits). Le processus d'analyse grammaticale pour ce descripteur est spécifié par la valeur résultante de la fonction `read_bits(8)`.
- `ce(v)`: élément syntaxique de longueur variable à codage entropique adaptable au contexte avec le bit de gauche en premier. Le processus d'analyse grammaticale pour ce descripteur est spécifié au § 9.2.
- `f(n)`: chaîne binaire à schéma fixe utilisant `n` bits écrits (de gauche à droite) avec le bit de gauche en premier. Le processus d'analyse grammaticale pour ce descripteur est spécifié par la valeur résultante de la fonction `read_bits(n)`.
- `i(n)`: entier signé utilisant `n` bits. Lorsque `n` est "v" dans le tableau syntaxique, le nombre de bits varie en fonction de la valeur des autres éléments syntaxiques. Le processus d'analyse grammaticale pour ce descripteur est spécifié par la valeur résultante de la fonction `read_bits(n)` interprétée comme la représentation entière d'un complément à deux avec le bit de plus fort poids écrit en premier.
- `me(v)`: élément syntaxique mappé codé en Exp-Golomb avec le bit de gauche en premier. Le processus d'analyse grammaticale pour ce descripteur est spécifié au § 9.1.
- `se(v)`: élément syntaxique entier signé codé en Exp-Golomb avec le bit de gauche en premier. Le processus d'analyse grammaticale pour ce descripteur est spécifié au § 9.1.
- `te(v)`: élément syntaxique tronqué codé en Exp-Golomb avec le bit de gauche en premier. Le processus d'analyse grammaticale pour ce descripteur est spécifié au § 9.1.
- `u(n)`: entier non signé utilisant `n` bits. Lorsque `n` est "v" dans le tableau syntaxique, le nombre de bits varie en fonction de la valeur des autres éléments syntaxiques. Le processus d'analyse grammaticale pour ce descripteur est spécifié par la valeur résultante de la fonction `read_bits(n)` interprétée comme une représentation binaire d'un entier non signé avec le bit de plus fort poids écrit en premier.
- `ue(v)`: élément syntaxique entier non signé codé en Exp-Golomb avec le bit de gauche en premier. Le processus d'analyse grammaticale pour ce descripteur est spécifié au § 9.1.

7.3 Syntaxe en forme tabulaire

7.3.1 Syntaxe d'unité NAL

nal_unit(NumBytesInNALunit) {	C	Descripteur
forbidden_zero_bit	Toutes	f(1)
nal_ref_idc	Toutes	u(2)
nal_unit_type	Toutes	u(5)
NumBytesInRBSP = 0		
pour(i = 1; i < NumBytesInNALunit; i++) {		
si(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {		
rbsp_byte [NumBytesInRBSP++]	Toutes	b(8)
rbsp_byte [NumBytesInRBSP++]	Toutes	b(8)
i += 2		
emulation_prevention_three_byte /* égal à 0x03 */	Toutes	f(8)
} autrement		
rbsp_byte [NumBytesInRBSP++]	Toutes	b(8)
}		
}		

7.3.2 Syntaxe de charges utiles de séquence d'octets bruts et de bits de traîne de charge utile RBSP

7.3.2.1 Syntaxe de charge utile RBSP d'ensemble de paramètres de séquence

	C	Descripteur
seq_parameter_set_rbsp() {		
profile_idc	0	u(8)
constraint_set0_flag	0	u(1)
constraint_set1_flag	0	u(1)
constraint_set2_flag	0	u(1)
reserved_zero_5bits /* equal to 0 */	0	u(5)
level_idc	0	u(8)
seq_parameter_set_id	0	ue(v)
log2_max_frame_num_minus4	0	ue(v)
pic_order_cnt_type	0	ue(v)
si(pic_order_cnt_type == 0)		
log2_max_pic_order_cnt_lsb_minus4	0	ue(v)
autrement si(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)
pour(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)
si(!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)
si(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)
si(vui_parameters_present_flag)		
vui_parameters()	0	
rbsp_trailing_bits()	0	
}		

7.3.2.2 Syntaxe de charge utile RBSP d'ensemble de paramètres d'image

	C	Descripteur
pic parameter set rbsp() {		
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)
si(num slice groups minus1 > 0) {		
slice group map type	1	ue(v)
si(slice group map type == 0)		
pour(iGroup = 0; iGroup <= num slice groups minus1; iGroup++)		
run length minus1[iGroup]	1	ue(v)
autrement si(slice group map type == 2)		
pour(iGroup = 0; iGroup < num slice groups minus1; iGroup++) {		
top left[iGroup]	1	ue(v)
bottom right[iGroup]	1	ue(v)
}		
autrement si(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)
} autrement si(slice_group_map_type == 6) {		
pic size in map units minus1	1	ue(v)
pour(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 /* par rapport à 26 */	1	se(v)
pic_init_qs_minus26 /* par rapport à 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)
rbsp_trailing_bits()	1	
}		

7.3.2.3 Syntaxe de charge utile RBSP d'informations d'amélioration supplémentaires

	C	Descripteur
sei_rbsp() {		
faire		
sei_message()	5	
tandis que (more_rbsp_data())		
rbsp_trailing_bits()	5	
}		

7.3.2.3.1 Syntaxe de message d'informations d'amélioration supplémentaires

	C	Descripteur
sei_message() {		
payloadType = 0		
tandis que(next_bits(8) == 0xFF) {		
ff_byte /* égal à 0xFF */	5	f(8)
payloadType += 255		
}		
last_payload_type byte	5	u(8)
payloadType += last_payload_type byte		
payloadSize = 0		
tandis que(next_bits(8) == 0xFF) {		
ff_byte /* égal à 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 Syntaxe de charge utile RBSP de délimiteur d'unité d'accès

	C	Descripteur
access_unit_delimiter_rbsp() {		
primary_pic type	6	u(3)
rbsp_trailing_bits()	6	
}		

7.3.2.5 Syntaxe de charge utile RBSP de fin de séquence

	C	Descripteur
end_of_seq_rbsp() {		
}		

7.3.2.6 Syntaxe de charge utile RBSP de fin de flux

	C	Descripteur
end_of_stream_rbsp() {		
}		

7.3.2.7 Syntaxe de charge utile RBSP de données de remplissage

	C	Descripteur
filler_data_rbsp(NumBytesInRBSP) {		
tandis que(next_bits(8) == 0xFF)		
ff_byte /* égal à 0xFF */	9	f(8)
rbsp_trailing_bits()	9	
}		

7.3.2.8 Syntaxe de charge utile RBSP de couche de tranche sans subdivision

	C	Descripteur
slice_layer_without_partitioning_rbsp() {		
slice_header()	2	
slice_data() /* toutes les catégories de syntaxe de slice_data() */	2 3 4	
rbsp_slice_trailing_bits()	2	
}		

7.3.2.9 Syntaxe de charge utile RBSP de subdivision de données de tranche

7.3.2.9.1 Syntaxe de charge utile RBSP de subdivision de données de tranche A

	C	Descripteur
slice_data_partition_a_layer_rbsp() {		
slice_header()	2	
slice_id	2	ue(v)
slice_data() /* seulement les parties de catégorie 2 de la syntaxe de slice_data() */	2	
rbsp_slice_trailing_bits()	2	
}		

7.3.2.9.2 Syntaxe de charge utile RBSP de subdivision de données de tranche B

	C	Descripteur
slice_data_partition_b_layer_rbsp() {		
slice_id	3	ue(v)
si(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	3	ue(v)
slice_data() /* seulement les parties de catégorie 3 de la syntaxe de slice_data() */	3	
rbsp_slice_trailing_bits()	3	
}		

7.3.2.9.3 Syntaxe de charge utile RBSP de subdivision de données de tranche C

	C	Descripteur
slice_data_partition_c_layer_rbsp() {		
slice_id	4	ue(v)
si(redundant_pic_cnt_present_flag)		
redundant_pic_cnt	4	ue(v)
slice_data() /* seulement les parties de catégorie 4 de la syntaxe de slice_data() */	4	
rbsp_slice_trailing_bits()	4	
}		

7.3.2.10 Syntaxe des bits de traîne de tranche de charge utile RBSP

	C	Descripteur
rbsp_slice_trailing_bits() {		
rbsp_trailing_bits()	Toutes	
si(entropy_coding_mode_flag)		
tandis que(more_rbsp_trailing_data())		
cabac_zero_word /* égal à 0x0000 */	Toutes	f(16)
}		

7.3.2.11 Syntaxe des bits de traîne de charge utile RBSP

	C	Descripteur
rbsp_trailing_bits() {		
rbsp_stop_one_bit /* égal à 1 */	Toutes	f(1)
tandis que(!byte_aligned())		
rbsp_alignment_zero_bit /* égal à 0 */	Toutes	f(1)
}		

7.3.3 Syntaxe d'en-tête de tranche

	C	Descripteur
slice header() {		
first mb in slice	2	ue(v)
slice type	2	ue(v)
pic parameter set id	2	ue(v)
frame num	2	u(v)
si(!frame mbs only flag) {		
field pic flag	2	u(1)
si(field pic flag)		
bottom field flag	2	u(1)
}		
si(nal unit type == 5)		
idr pic id	2	ue(v)
si(pic order cnt type == 0) {		
pic order cnt lsb	2	u(v)
si(pic order present flag && !field pic flag)		
delta pic order cnt bottom	2	se(v)
}		
si(pic order cnt type == 1 && !delta pic order always zero flag) {		
delta pic order cnt[0]	2	se(v)
si(pic order present flag && !field pic flag)		
delta pic order cnt[1]	2	se(v)
}		
si(redundant pic cnt present flag)		
redundant pic cnt	2	ue(v)
si(slice type == B)		
direct spatial mv pred flag	2	u(1)
si(slice type == P slice type == SP slice type == B) {		
num ref idx active override flag	2	u(1)
si(num ref idx active override flag) {		
num ref idx I0 active minus1	2	ue(v)
si(slice type == B)		
num ref idx I1 active minus1	2	ue(v)
}		
}		
ref pic list reordering()	2	
si((weighted_pred_flag && (slice_type == P slice_type == SP)) (weighted_bipred_idc == 1 && slice_type == B))		
pred_weight_table()	2	
si(nal_ref_idc != 0)		
dec_ref_pic_marking()	2	
si(entropy_coding_mode_flag && slice_type != I && slice_type != SI)		
cabac_init_idc	2	ue(v)
slice_qp_delta	2	se(v)
si(slice_type == SP slice_type == SI) {		
si(slice_type == SP)		
sp_for_switch_flag	2	u(1)
slice_qs_delta	2	se(v)
}		
si(deblocking_filter_control_present_flag) {		
disable_deblocking_filter_idc	2	ue(v)
si(disable_deblocking_filter_idc != 1) {		
slice_alpha_c0_offset_div2	2	se(v)
slice_beta_offset_div2	2	se(v)
}		
}		
si(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 Syntaxe de réorganisation de la liste d'images de référence

	C	Descripteur
ref pic list reordering() {		
si(slice type != I && slice type != SI) {		
ref pic list reordering flag I0	2	u(1)
si(ref pic list reordering flag I0)		
faire {		
reordering of pic nums idc	2	ue(v)
si(reordering_of_pic_nums_idc == 0 reordering_of_pic_nums_idc == 1)		
abs diff pic num minus1	2	ue(v)
autrement si(reordering_of_pic_nums_idc == 2)		
long term pic num	2	ue(v)
} tandis que(reordering_of_pic_nums_idc != 3)		
}		
si(slice_type == B) {		
ref pic list reordering flag I1	2	u(1)
si(ref pic list reordering flag I1)		
faire{		
reordering of pic nums idc	2	ue(v)
si(reordering_of_pic_nums_idc == 0 reordering_of_pic_nums_idc == 1)		
abs diff pic num minus1	2	ue(v)
autrement si(reordering_of_pic_nums_idc == 2)		
long term pic num	2	ue(v)
} tandis que(reordering_of_pic_nums_idc != 3)		
}		
}		
}		

7.3.3.2 Syntaxe de tableau de pondération de prédiction

	C	Descripteur
pred_weight_table() {		
luma_log2_weight_denom	2	ue(v)
chroma_log2_weight_denom	2	ue(v)
pour(i=0; i<= num_ref_idx_l0_active_minus1; i++) {		
luma_weight_l0_flag	2	u(1)
si(luma_weight_l0_flag) {		
luma_weight_l0[i]	2	se(v)
luma_offset_l0[i]	2	se(v)
}		
chroma_weight_l0_flag	2	u(1)
si(chroma_weight_l0_flag)		
pour(j=0; j<2; j++) {		
Chroma_weight_l0[i][j]	2	se(v)
Chroma_offset_l0[i][j]	2	se(v)
}		
}		
si(slice_type == B)		
pour(i=0; i<= num_ref_idx_l1_active_minus1; i++) {		
luma_weight_l1_flag	2	u(1)
si(luma_weight_l1_flag) {		
luma_weight_l1[i]	2	se(v)
luma_offset_l1[i]	2	se(v)
}		
chroma_weight_l1_flag	2	u(1)
si(chroma_weight_l1_flag)		
pour(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 Syntaxe de marquage d'image de référence décodée

	C	Descripteur
dec_ref_pic_marking() {		
si(nal_unit_type == 5) {		
no_output_of_prior_pics_flag	2 5	u(1)
long_term_reference_flag	2 5	u(1)
} autrement {		
adaptive_ref_pic_marking_mode_flag	2 5	u(1)
si(adaptive_ref_pic_marking_mode_flag)		
faire {		
memory_management_control_operation	2 5	ue(v)
si(memory_management_control_operation == 1 memory_management_control_operation == 3)		
difference_of_pic_nums_minus1	2 5	ue(v)
si(memory_management_control_operation == 2)		
Long term pic num	2 5	ue(v)
si(memory_management_control_operation == 3 memory_management_control_operation == 6)		
Long term frame idx	2 5	ue(v)
si(memory_management_control_operation == 4)		
Max long term frame idx plus1	2 5	ue(v)
} tandis que(memory_management_control_operation != 0)		
}		
}		

7.3.4 Syntaxe de données de tranche

	C	Descripteur
slice_data() {		
si(entropy_coding_mode_flag)		
tandis que(!byte_aligned())		
cabac_alignment_one_bit	2	f(1)
CurrMbAddr = first_mb_in_slice * (1 + MbaffFrameFlag)		
moreDataFlag = 1		
prevMbSkipped = 0		
faire {		
si(slice_type != I && slice_type != SI)		
si(!entropy_coding_mode_flag) {		
mb_skip_run	2	ue(v)
prevMbSkipped = (mb_skip_run > 0)		
pour(i=0; i<mb_skip_run; i++)		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
moreDataFlag = more_rbsp_data()		
} autrement {		
mb_skip_flag	2	ae(v)
moreDataFlag = !mb_skip_flag		
}		
} si(moreDataFlag) {		
si(MbaffFrameFlag && (CurrMbAddr % 2 == 0		
(CurrMbAddr % 2 == 1 && prevMbSkipped))		
mb_field_decoding_flag	2	u(1) ae(v)
Macroblock_layer()	2 3 4	
}		
} si(!entropy_coding_mode_flag)		
moreDataFlag = more_rbsp_data()		
autrement {		
si(slice_type != I && slice_type != SI)		
prevMbSkipped = mb_skip_flag		
si(MbaffFrameFlag && CurrMbAddr % 2 == 0)		
moreDataFlag = 1		
autrement {		
end_of_slice_flag	2	ae(v)
moreDataFlag = !end_of_slice_flag		
}		
}		
CurrMbAddr = NextMbAddress(CurrMbAddr)		
} tandis que(moreDataFlag)		
}		

7.3.5 Syntaxe de couche de macrobloc

	C	Descripteur
macroblock layer() {		
mb_type	2	ue(v) ae(v)
si(mb_type == I_PCM) {		
tandis que(!byte_aligned())		
pcm_alignment_zero_bit	2	f(1)
pour(i = 0; i < 256 * ChromaFormatFactor; i++)		
pcm_byte[i]	2	u(8)
} autrement {		
si(MbPartPredMode(mb_type, 0) != Intra_4x4 && MbPartPredMode(mb_type, 0) != Intra_16x16 && NumMbPart(mb_type) == 4)		
sub_mb_pred(mb_type)	2	
autrement		
mb_pred(mb_type)	2	
si(MbPartPredMode(mb_type, 0) != Intra_16x16)		
coded_block_pattern	2	me(v) ae(v)
si(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 Syntaxe de la prédiction de macrobloc

	C	Descripteur
mb_pred(mb_type) {		
si(MbPartPredMode(mb_type, 0) == Intra_4x4 MbPartPredMode(mb_type, 0) == Intra_16x16) {		
si(MbPartPredMode(mb_type, 0) == Intra_4x4)		
pour(luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++) {		
prev_intra4x4_pred_mode_flag[luma4x4BlkIdx]	2	u(1) ae(v)
si(!prev_intra4x4_pred_mode_flag[luma4x4BlkIdx])		
rem_intra4x4_pred_mode[luma4x4BlkIdx]	2	u(3) ae(v)
}		
intra_chroma_pred_mode	2	ue(v) ae(v)
} autrement si(MbPartPredMode(mb_type, 0) != Direct) {		
pour(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
si((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)
pour(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
si((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)
pour(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
si(MbPartPredMode(mb_type, mbPartIdx) != Pred_L1)		
pour(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0[mbPartIdx][0][compIdx]	2	se(v) ae(v)
pour(mbPartIdx = 0; mbPartIdx < NumMbPart(mb_type); mbPartIdx++)		
si(MbPartPredMode(mb_type, mbPartIdx) != Pred_L0)		
pour(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1[mbPartIdx][0][compIdx]	2	se(v) ae(v)
}		
}		
}		
}		

7.3.5.2 Syntaxe de la prédiction de sous-macrobloc

	C	Descripteur
sub_mb_pred(mb_type) {		
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
sub_mb_type [mbPartIdx]	2	ue(v) ae(v)
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
si((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)
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
si((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)
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
si(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L1)		
pour(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
pour(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l0 [mbPartIdx][subMbPartIdx][compIdx]	2	se(v) ae(v)
pour(mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++)		
si(sub_mb_type[mbPartIdx] != B_Direct_8x8 && SubMbPredMode(sub_mb_type[mbPartIdx]) != Pred_L0)		
pour(subMbPartIdx = 0; subMbPartIdx < NumSubMbPart(sub_mb_type[mbPartIdx]); subMbPartIdx++)		
pour(compIdx = 0; compIdx < 2; compIdx++)		
mvd_l1 [mbPartIdx][subMbPartIdx][compIdx]	2	se(v) ae(v)
}		

7.3.5.3 Syntaxe des données résiduelles

residual() {	C	Descripteur
si(!entropy coding mode flag)		
residual_block = residual_block_cavlc		
autrement		
residual_block = residual_block_cabac		
si(MbPartPredMode(mb_type, 0) == Intra_16x16)		
residual_block(Intra16x16DCLevel, 16)	3	
pour(i8x8 = 0; i8x8 < 4; i8x8++) /* chaque bloc luma 8x8 */		
pour(i4x4 = 0; i4x4 < 4; i4x4++) /* chaque sous-bloc 4x4 de bloc */		
si(CodedBlockPatternLuma & (1 << i8x8)) {		
si(MbPartPredMode(mb_type, 0) == Intra_16x16)		
residual_block(Intra16x16ACLevel[i8x8 * 4 + i4x4], 15)	3	
autrement		
residual_block(LumaLevel[i8x8 * 4 + i4x4], 16)	3 4	
} autrement {		
si(MbPartPredMode(mb_type, 0) == Intra_16x16)		
pour(i = 0; i < 15; i++)		
Intra16x16ACLevel[i8x8 * 4 + i4x4][i] = 0		
autrement		
pour(i = 0; i < 16; i++)		
LumaLevel[i8x8 * 4 + i4x4][i] = 0		
}		
pour(iCbCr = 0; iCbCr < 2; iCbCr++)		
si(CodedBlockPatternChroma & 3) /* constante chromatique résiduelle présente */		
residual_block(ChromaDCLevel[iCbCr], 4)	3 4	
autrement		
pour(i = 0; i < 4; i++)		
ChromaDCLevel[iCbCr][i] = 0		
pour(iCbCr = 0; iCbCr < 2; iCbCr++)		
pour(i4x4 = 0; i4x4 < 4; i4x4++)		
si(CodedBlockPatternChroma & 2)		
/* variable chromatique résiduelle présente. */		
residual_block(ChromaACLevel[iCbCr][i4x4], 15)	3 4	
autrement		
pour(i = 0; i < 15; i++)		
ChromaACLevel[iCbCr][i4x4][i] = 0		
}		

7.3.5.3.1 Syntaxe du codage CAVLC de bloc résiduel

	C	Descripteur
residual_block_cavlc(coeffLevel, maxNumCoeff) {		
pour(i = 0; i < maxNumCoeff; i++)		
coeffLevel[i] = 0		
coeff_token	3 4	ce(v)
si(TotalCoeff(coeff_token) > 0) {		
si(TotalCoeff(coeff_token) > 10 && TrailingOnes(coeff_token) < 3)		
suffixLength = 1		
autrement		
suffixLength = 0		
pour(i = 0; i < TotalCoeff(coeff_token); i++)		
si(i < TrailingOnes(coeff_token)) {		
trailing_ones_sign_flag	3 4	u(1)
level[i] = 1 - 2 * trailing_ones_sign_flag		
} autrement {		
level_prefix	3 4	ce(v)
levelCode = (level_prefix << suffixLength)		
si(suffixLength > 0 level_prefix >= 14) {		
level_suffix	3 4	u(v)
levelCode += level_suffix		
}		
si(level_prefix == 15 && suffixLength == 0)		
levelCode += 15		
si(i == TrailingOnes(coeff_token) &&		
TrailingOnes(coeff_token) < 3)		
levelCode += 2		
si(levelCode % 2 == 0)		
level[i] = (levelCode + 2) >> 1		
autrement		
level[i] = (-levelCode - 1) >> 1		
si(suffixLength == 0)		
suffixLength = 1		
si(Abs(level[i]) > (3 << (suffixLength - 1)) &&		
suffixLength < 6)		
suffixLength++		
}		
si(TotalCoeff(coeff_token) < maxNumCoeff) {		
total_zeros	3 4	ce(v)
zerosLeft = total_zeros		
} autrement		
zerosLeft = 0		
pour(i = 0; i < TotalCoeff(coeff_token) - 1; i++) {		
si(zerosLeft > 0) {		
run_before	3 4	ce(v)
run[i] = run_before		
} autrement		
run[i] = 0		
zerosLeft = zerosLeft - run[i]		
}		
run[TotalCoeff(coeff_token) - 1] = zerosLeft		
coeffNum = - 1		
pour (i = TotalCoeff(coeff_token) - 1; i >= 0; i--) {		
coeffNum += run[i] + 1		
coeffLevel[coeffNum] = level[i]		
}		
}		
}		

7.3.5.3.2 Syntaxe du codage CABAC de bloc résiduel

	C	Descripteur
<code>residual_block_cabac(coeffLevel, maxNumCoeff) {</code>		
<code>coded_block_flag</code>	3 4	ae(v)
<code>si(coded_block_flag) {</code>		
<code>numCoeff = maxNumCoeff</code>		
<code>i = 0</code>		
<code>faire {</code>		
<code>significant_coeff_flag[i]</code>	3 4	ae(v)
<code>si(significant_coeff_flag[i]) {</code>		
<code>last_significant_coeff_flag[i]</code>	3 4	ae(v)
<code>si(last_significant_coeff_flag[i]) {</code>		
<code>numCoeff = i + 1</code>		
<code>pour(j = numCoeff; j < maxNumCoeff; j++)</code>		
<code>coeffLevel[j] = 0</code>		
<code>}</code>		
<code>}</code>		
<code>i++</code>		
<code>} tandis que(i < numCoeff-1)</code>		
<code>coeff_abs_level_minus1[numCoeff-1]</code>	3 4	ae(v)
<code>coeff_sign_flag[numCoeff-1]</code>	3 4	ae(v)
<code>coeffLevel[numCoeff-1] =</code> <code>(coeff_abs_level_minus1[numCoeff - 1] + 1) *</code> <code>(1 - 2 * coeff_sign_flag[numCoeff - 1])</code>		
<code>pour(i = numCoeff-2; i >= 0; i--) {</code>		
<code>si(significant_coeff_flag[i]) {</code>		
<code>coeff_abs_level_minus1[i]</code>	3 4	ae(v)
<code>coeff_sign_flag[i]</code>	3 4	ae(v)
<code>coeffLevel[i] = (coeff_abs_level_minus1[i] + 1) *</code> <code>(1 - 2 * coeff_sign_flag[i])</code>		
<code>} autrement</code>		
<code>coeffLevel[i] = 0</code>		
<code>}</code>		
<code>} autrement</code>		
<code>pour(i = 0; i < maxNumCoeff; i++)</code>		
<code>coeffLevel[i] = 0</code>		
<code>}</code>		

7.4 Sémantique

7.4.1 Sémantique d'unité NAL

NOTE – La couche VCL est spécifiée pour représenter efficacement le contenu des données vidéo. La couche NAL est spécifiée pour formater ces données et fournir des informations d'en-tête d'une façon appropriée pour le transport sur des canaux de communication ou des moyens de stockage variés. Toutes les données sont contenues dans les unités NAL, chacune d'elles contient un nombre entier d'octets. Une unité NAL spécifie un format générique à utiliser à la fois pour les systèmes orientés paquet et à flux binaire. Le format des unités NAL est identique aussi bien pour le transport orienté paquet que pour le flux d'octets avec l'exception que chaque unité NAL peut être précédée d'un préfixe de code de déclenchement et d'octets de bourrage dans le format de flux d'octets.

NumBytesInNALunit spécifie la taille de l'unité NAL en octets. Cette valeur est nécessaire pour le décodage de l'unité NAL. Une certaine forme de démarcation des frontières de l'unité NAL est nécessaire pour permettre le calcul de NumBytesInNALunit. Une telle méthode de démarcation est spécifiée à l'Annexe B pour le format de flux d'octets. D'autres méthodes de démarcation peuvent être spécifiées en dehors de la présente Recommandation | Norme internationale.

forbidden_zero_bit doit être égal à 0.

nal_ref_idc non égal à 0 spécifie que le contenu de l'unité NAL comporte un ensemble de paramètres de séquence ou un ensemble de paramètres d'image ou une tranche d'une image de référence ou subdivision de données de tranche d'une image de référence.

nal_ref_idc égal à 0 pour une unité NAL contenant une tranche ou subdivision de données de tranche indique que la tranche ou subdivision de données de tranche fait partie d'une image de non-référence.

nal_ref_idc ne doit pas être égal à 0 pour les unités NAL d'ensemble de paramètres de séquence ou d'ensemble de paramètres d'image. Lorsque nal_ref_idc est égal à 0 pour une unité NAL de tranche ou subdivision de données de tranche d'une image particulière, il doit être égal à 0 pour toute unité NAL de tranche et subdivision de données de tranche de l'image.

nal_ref_idc doit être non égal à 0 pour les unités NAL IDR, c'est-à-dire, les unités NAL avec nal_unit_type égal à 5.

nal_ref_idc doit être égal à 0 pour toutes les unités NAL ayant nal_unit_type égal à 6, 9, 10, 11 ou 12.

nal_unit_type spécifie le type de structure de données de charge utile RBSP contenues dans l'unité NAL comme spécifié au Tableau 7-1. Les unités NAL VCL sont spécifiées comme étant celles des unités NAL qui ont nal_unit_type égal à 1 à 5, inclus. Toutes les unités NAL restantes sont appelées unités NAL non VCL.

La colonne marquée "C" dans le Tableau 7-1 donne la liste des catégories d'éléments syntaxiques qui peuvent être présents dans l'unité NAL. De plus, des éléments syntaxiques avec la catégorie de syntaxe "Toutes" peuvent être présents, selon la détermination de la syntaxe et la sémantique de la structure des données de la charge utile RBSP. La présence ou l'absence de tous éléments syntaxiques d'une catégorie particulière de la liste est déterminée à partir de la syntaxe et la sémantique de la structure des données de la charge utile RBSP. nal_unit_type ne doit pas être égal à 3 ou 4 à moins qu'au moins un élément syntaxique ne soit présent dans la structure des données de la charge utile RBSP ayant une valeur de catégorie d'élément syntaxique égale à la valeur de nal_unit_type et n'appartenant pas à la catégorie "Toutes".

Tableau 7-1 – Codes de type d'unité NAL

nal_unit_type	Contenu de la structure syntaxique de l'unité NAL et de la charge RBSP	C
0	Non spécifié	
1	Tranche codée d'une image non IDR slice_layer_without_subdivision_rbsp()	2, 3, 4
2	Subdivision A de données de tranche codée slice_data_partition_a_layer_rbsp()	2
3	Subdivision B de données de tranche codée slice_data_partition_b_layer_rbsp()	3
4	Subdivision C de données de tranche codée slice_data_partition_c_layer_rbsp()	4
5	Tranche codée d'une image IDR slice_layer_without_subdivision_rbsp()	2, 3
6	Informations d'amélioration supplémentaires (SEI, <i>supplemental enhancement information</i>) sei_rbsp()	5
7	Ensemble de paramètres de séquence seq_parameter_set_rbsp()	0
8	Ensemble de paramètres d'image pic_parameter_set_rbsp()	1
9	Délimiteur d'unité d'accès access_unit_delimiter_rbsp()	6
10	Fin de séquence end_of_seq_rbsp()	7
11	Fin de flux end_of_stream_rbsp()	8
12	Données de remplissage filler_data_rbsp()	9
13..23	Réservé	
24..31	Non spécifié	

Les unités NAL qui utilisent nal_unit_type égal à 0 ou dans la gamme de 24..31, inclusivement, ne doivent pas affecter le processus de décodage spécifié dans la présente Recommandation | Norme internationale.

NOTE – Les types d'unité NAL 0 et 24..31 peuvent être utilisés comme déterminé par l'application. Il n'est pas spécifié de décodage pour ces valeurs de nal_unit_type dans la présente Recommandation | Norme internationale.

Les décodeurs doivent ignorer (retirer du flux binaire et mettre à l'écart) le contenu de toutes les unités NAL qui utilisent les valeurs réservées de nal_unit_type.

NOTE – Cette exigence permet une définition d'extensions futures compatibles avec la présente Recommandation | Norme internationale.

Dans le texte, unité NAL de tranche codée se réfère collectivement à une tranche codée d'unité NAL d'une image non IDR ou à une tranche codée d'unité NAL d'image IDR.

Lorsque la valeur de nal_unit_type est égale à 5 pour une unité NAL contenant une tranche d'image codée, la valeur de nal_unit_type doit être 5 dans toutes les autres unités NAL VCL de la même image codée. Une telle image est appelée une image IDR.

NOTE – Une subdivision de données de tranche ne peut pas être utilisée pour des images IDR.

rbsp_byte[i] est le ième octet d'une charge utile RBSP. Une charge utile RBSP est spécifiée comme une séquence ordonnée d'octets comme suit.

La charge utile contient une chaîne SODB (*chaîne de bits de données*) comme suit.

- Si la chaîne SODB est vide (c'est-à-dire, zéro bits en longueur), la charge utile RBSP est vide elle aussi.
- Autrement, la charge utile RBSP contient la chaîne SODB comme suit.
 - 1) Le premier octet de la charge utile RBSP contient les huit bits (celui de plus fort poids le plus à gauche) de la chaîne SODB; le prochain octet de la charge utile RBSP doit contenir les huit bits suivants de la chaîne SODB, etc., jusqu'à ce que restent moins de huit bits de la chaîne SODB.
 - 2) **rbsp_trailing_bits()** sont présents après la chaîne SODB comme suit:
 - i) les premiers bits (celui de plus fort poids le plus à gauche) du dernier octet de la charge utile RBSP contient les bits restants de la chaîne SODB, (s'il y en a);
 - ii) le bit suivant consiste en un seul **rbsp_stop_one_bit** égal à 1; et
 - iii) lorsque le **rbsp_stop_one_bit** n'est pas le dernier bit d'un octet aligné sur les octets, un ou plusieurs **rbsp_alignment_zero_bit** sont présents pour assurer l'alignement d'octet.
 - 3) Un ou plusieurs éléments syntaxiques à 16 bits **cabac_zero_word** égaux à 0x0000 peuvent être présents dans certaines charges utiles RBSP après les **rbsp_trailing_bits()** à la fin de la charge utile RBSP.

Les structures syntaxiques ayant ces propriétés de charge utile RBSP sont notées dans les tableaux syntaxiques en utilisant un suffixe "_rbsp". Ces structures doivent être transportées au sein des unités NAL en tant que contenu des octets de données de **rbsp_byte[i]**. L'association des structures syntaxiques de la charge utile RBSP aux unités NAL doit être conforme à ce qui est spécifié au Tableau 7-1.

NOTE – Lorsque les frontières de la charge utile RBSP sont connues, le décodeur peut extraire la chaîne SODB de la charge utile RBSP par concaténation des bits des octets de la RBSP et en mettant à l'écart le **rbsp_stop_one_bit**, qui est le dernier bit (celui de plus faible poids, le plus à droite) égal à 1, et en mettant à l'écart tous les bits suivants (de moindre poids, les plus à droite) qui le suivent, qui sont égaux à 0. Les données nécessaires pour le processus de décodage sont contenues dans la partie SODB de la charge utile RBSP.

emulation_prevention_three_byte est un octet égal à 0x03. Lorsqu'un octet **emulation_prevention_three_byte** est présent dans l'unité NAL, il doit être mis à l'écart par le processus de décodage.

Le dernier octet de l'unité NAL ne doit pas être égal à 0x00.

Au sein de l'unité NAL, les séquences de trois octets suivantes ne doivent pas survenir dans une position d'alignement d'octet:

- 0x000000
- 0x000001
- 0x000002

Au sein de l'unité NAL, toute séquence de quatre octets qui débute par 0x000003, autre que les séquences suivantes, ne doit survenir à aucune des positions d'alignement d'octet:

- 0x00000300
- 0x00000301

- 0x00000302
- 0x00000303

7.4.1.1 Encapsulation d'une chaîne SODB au sein d'une charge utile RBSP (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

La forme de l'encapsulation d'une chaîne SODB au sein d'une charge utile RBSP et l'utilisation de `emulation_prevention_three_byte` pour l'encapsulation d'une charge utile RBSP au sein d'une unité NAL sont spécifiées pour les raisons suivantes:

- pour empêcher l'émulation de codes de démarrage au sein des unités NAL alors qu'on permet à toute chaîne SODB arbitraire d'être représentée au sein d'une unité NAL;
- pour permettre l'identification de la fin de la chaîne SODB au sein de l'unité NAL en recherchant la charge utile RBSP pour le `rbsp_stop_one_bit` commençant à la fin de la charge utile RBSP;
- pour permettre à une unité NAL d'avoir une taille plus grande que celle de la chaîne SODB dans certaines circonstances (en utilisant un ou plusieurs `cabac_zero_word`).

Le codeur peut produire une unité NAL à partir d'une charge utile RBSP par la procédure suivante:

les données de la charge utile RBSP sont recherchées dans les bits d'alignement d'octet des schémas binaires suivants:

'00000000 00000000 000000xx' (où xx représente tout schéma à 2 bits: 00, 01, 10 ou 11),

et un octet égal à 0x03 est inséré pour remplacer ces schémas binaires par les schémas

'00000000 00000000 00000011 000000xx',

et finalement, lorsque le dernier octet des données de la charge utile RBSP est égal à 0x00 (ce qui ne peut survenir que lorsque la charge utile RBSP se termine par un `cabac_zero_word`), un octet final égal à 0x03 est ajouté à la fin des données.

La séquence d'octets résultante est alors mise en préfixe avec le premier octet de l'unité NAL contenant l'indication du type de structure de données de charge utile RBSP qu'il contient. Il en résulte la construction de l'unité NAL entière.

Ce processus peut permettre à toute chaîne SODB d'être représentée dans une unité NAL tout en s'assurant que:

- aucun préfixe de code de déclenchement aligné sur les octets n'est émulé au sein de l'unité NAL; et
- aucune séquence de 8 bits de valeur zéro suivie d'un préfixe de code de déclenchement, sans considération de l'alignement d'octet, n'est émulée au sein de l'unité NAL.

7.4.1.2 Ordre des unités NAL et association aux images codées, unités d'accès et séquences vidéo

Le présent paragraphe spécifie les contraintes sur l'ordre des unités NAL dans le flux binaire. Tout ordre des unités NAL dans le flux binaire obéissant à ces contraintes est désigné dans le texte comme l'ordre de décodage des unités NAL. Au sein d'une unité NAL, la syntaxe des § 7.3, D.1 et E.1 spécifie l'ordre de décodage des éléments syntaxiques. Les décodeurs conformes à la présente Recommandation | Norme internationale doivent être capables de recevoir des unités NAL et leurs éléments syntaxiques dans l'ordre de décodage.

7.4.1.2.1 Ordre de séquence et charges utiles RBSP d'ensemble de paramètres d'image et leur activation

NOTE – Le mécanisme de séquence et d'ensemble de paramètres d'image découple la transmission des informations peu changeantes de la transmission des données de macroblocs codés. Les ensembles de séquence et de paramètres d'image peuvent, dans certaines applications, être convoyés "hors bande" en utilisant un mécanisme de transport fiable.

Une charge utile RBSP d'ensemble de paramètres d'image inclut des paramètres auxquels peuvent se référer les unités NAL de tranche codée ou les unités NAL de subdivision A de données de tranche codée d'une ou plusieurs images codées. Chaque charge utile RBSP d'ensemble de paramètres d'image, au départ, est considérée comme n'étant pas active au début de l'opération du processus de décodage. Au plus, une charge utile RBSP d'ensemble de paramètres d'image est considérée comme étant active à un moment donné quelconque durant l'opération du processus de décodage, et l'activation de telle ou telle charge utile RBSP d'ensemble de paramètres d'image déclenche la désactivation de la charge utile RBSP d'ensemble de paramètres d'image précédemment active (le cas échéant).

Lorsqu'une charge utile RBSP d'ensemble de paramètres d'image (avec une valeur particulière de `pic_parameter_set_id`) n'est pas active et qu'elle est prise en charge en référence par une unité NAL de tranche codée ou une unité NAL de subdivision A de données de tranche codée (utilisant cette valeur de `pic_parameter_set_id`), elle est activée. Cette charge utile RBSP d'ensemble de paramètres d'image est appelée la charge utile active d'ensemble de paramètres d'image jusqu'à ce qu'elle soit désactivée par l'activation d'une autre charge utile RBSP d'ensemble de paramètres d'image. Une charge utile RBSP d'ensemble de paramètres d'image, avec cette valeur particulière de `pic_parameter_set_id`, doit être disponible pour le processus de décodage avant son activation.

Toute unité NAL d'ensemble de paramètres d'image contenant la valeur de `pic_parameter_set_id` pour la charge utile active d'ensemble de paramètres d'image doit avoir le même contenu que celui de la charge utile RBSP active de l'ensemble de paramètres d'image à moins qu'elle ne suive la dernière unité NAL VCL d'une image codée et précède la première unité NAL VCL d'une autre image codée.

Une charge utile RBSP d'ensemble de paramètres de séquence inclut des paramètres auxquels peuvent se référer une ou plusieurs charges utiles RBSP d'ensemble de paramètres d'image ou une ou plusieurs unités NAL SEI contenant un message SEI de période de temporisation. Chaque charge utile RBSP d'ensemble de paramètres de séquence, au départ, est considérée comme n'étant pas active au début de l'opération du processus de décodage. Au plus, une charge utile RBSP d'ensemble de paramètres de séquence est considérée comme étant active à un moment donné quelconque durant l'opération du processus de décodage, et l'activation de telle ou telle charge utile RBSP d'ensemble de paramètres de séquence déclenche la désactivation de la charge utile RBSP d'ensemble de paramètres de séquence précédemment active (le cas échéant).

Lorsqu'une charge utile RBSP d'ensemble de paramètres de séquence (avec une valeur particulière de `seq_parameter_set_id`) n'est pas déjà active et qu'elle est prise en référence par l'activation d'une charge utile RBSP d'ensemble de paramètres d'image (utilisant cette valeur de `seq_parameter_set_id`) ou est prise en référence par une unité NAL SEI contenant un message SEI de période de temporisation (utilisant cette valeur de `seq_parameter_set_id`), elle est activée. Cette charge utile RBSP d'ensemble de paramètres de séquence est appelée la charge utile RBSP active d'ensemble de paramètres de séquence jusqu'à ce qu'elle soit désactivée par l'activation d'une autre charge utile RBSP d'ensemble de paramètres de séquence. Une charge utile RBSP d'ensemble de paramètres de séquence, avec cette valeur particulière de `seq_parameter_set_id`, doit être disponible pour le processus de décodage avant son activation. Une charge utile RBSP activée d'ensemble de paramètres de séquence doit rester active pour la séquence vidéo codée entière.

NOTE – Etant donné qu'une unité d'accès IDR ouvre une nouvelle séquence vidéo codée et qu'une charge utile RBSP activée d'ensemble de paramètres de séquence doit rester active pour la séquence vidéo codée entière, une charge utile RBSP d'ensemble de paramètres de séquence ne peut être activée que par un message SEI de période de temporisation faisant partie d'une unité d'accès IDR.

Toute unité NAL d'ensemble de paramètres de séquence contenant la valeur de `seq_parameter_set_id` pour la charge utile RBSP active d'ensemble de paramètres de séquence doit avoir le même contenu que celui de la charge utile RBSP active d'ensemble de paramètres de séquence à moins qu'elle ne suive la dernière unité d'accès d'une séquence vidéo codée et ne précède la première unité NAL VCL et la première unité NAL SEI contenant un message SEI de période de temporisation (le cas échéant) d'une autre séquence vidéo codée.

NOTE – Si une charge utile RBSP d'ensemble de paramètres d'image ou une charge utile RBSP d'ensemble de paramètres de séquence est transportée au sein du flux binaire, ces contraintes imposent un ordre aux unités NAL qui contiennent la charge utile RBSP d'ensemble de paramètres d'image ou respectivement, la charge utile RBSP d'ensemble de paramètres de séquence. Autrement (la charge utile RBSP d'ensemble de paramètres d'image ou la charge utile RBSP d'ensemble de paramètres de séquence est convoyée par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale), elle doit être disponible pour le processus de décodage à temps pour que ces contraintes puissent être respectées.

Toutes les contraintes qui sont organisées sur la relation entre les valeurs des éléments syntaxiques (et les valeurs des variables extraites de ces éléments syntaxiques) des ensembles de paramètres de séquence et des ensembles de paramètres d'image et d'autres éléments syntaxiques sont des expressions de contraintes qui s'appliquent uniquement à l'ensemble de paramètres de séquence actif et à l'ensemble de paramètres d'image actif. Si une charge utile RBSP d'ensemble de paramètres de séquence qui n'est pas activée dans le flux binaire est présente, ses éléments syntaxiques doivent avoir des valeurs qui seraient conformes aux contraintes spécifiées si elle était activée par référence dans un autre flux binaire conforme. Si une charge utile RBSP d'ensemble de paramètres d'image qui n'est pas toujours activée dans le flux binaire est présente, ses éléments syntaxiques doivent avoir des valeurs qui seraient conformes aux contraintes spécifiées si elle était activée par référence dans un autre flux binaire conforme.

Durant l'opération du processus de décodage (voir l'article 8), les valeurs des paramètres de l'ensemble actif de paramètres d'image et de l'ensemble actif de paramètres de séquence doivent être considérées comme effectives. Pour l'interprétation des messages SEI, les valeurs des paramètres de l'ensemble de paramètres d'image et de l'ensemble de paramètres de séquence qui sont actifs pour l'opération du processus de décodage pour les unités NAL VCL de l'image codée primaire dans la même unité d'accès, doivent être considérées comme effectives à moins qu'il n'en soit spécifié autrement dans la sémantique de message SEI.

7.4.1.2.2 Ordre des unités d'accès et association aux séquences vidéo codées

Un flux binaire conforme à la présente Recommandation | Norme internationale consiste en une ou plusieurs séquences vidéo codées.

Une séquence vidéo codée consiste en une ou plusieurs unités d'accès. L'ordre des unités NAL et des images codées et leur association aux unités d'accès sont décrits au § 7.4.1.2.3.

La première unité d'accès de chaque séquence vidéo codée est une unité d'accès IDR. Toutes les unités d'accès suivantes dans la séquence vidéo codée sont des unités d'accès non IDR.

Les valeurs de compte d'ordre d'image pour les images codées en unités d'accès consécutives dans l'ordre de décodage contenant des images qui ne sont pas de référence doivent être non décroissantes.

Lorsqu'elle est présente, une unité d'accès suivant une unité d'accès qui contient une fin de séquence d'unité NAL doit être une unité d'accès IDR.

Lorsqu'une unité NAL SEI contient des données qui appartiennent à plus d'une unité d'accès (par exemple, lorsqu'une unité NAL SEI a une séquence vidéo codée comme domaine d'application), elle doit être contenue dans la première unité d'accès à laquelle elle s'applique.

Lorsqu'une fin de flux d'unité NAL est présente dans une unité d'accès, cette unité d'accès doit être la dernière unité d'accès dans le flux binaire et la fin de flux de l'unité NAL doit être la dernière unité NAL dans cette unité d'accès.

7.4.1.2.3 Ordre des unités NAL et images codées et association aux unités d'accès

Une unité d'accès consiste en une image codée primaire, zéro ou plusieurs images codées redondantes correspondantes, et zéro ou plusieurs unités NAL non VCL. L'association des unités NAL VCL aux images codées redondantes ou primaires est décrite au § 7.4.1.2.5.

La première de toutes les unités NAL suivantes après la dernière unité NAL VCL d'une image codée primaire spécifie le début d'une nouvelle unité d'accès.

- unité NAL de délimiteur d'unité d'accès (le cas échéant);
- unité NAL d'ensemble de paramètres de séquence (le cas échéant);
- unité NAL d'ensemble de paramètres d'image (le cas échéant);
- unité NAL SEI (le cas échéant);
- des unités NAL avec `nal_unit_type` dans la gamme de 13 à 18 inclus;
- première unité NAL VCL d'une image codée primaire (toujours présente).

Les contraintes pour la détection de la première unité NAL VCL d'une image codée primaire sont spécifiées au § 7.4.1.2.4.

Les contraintes suivantes doivent être respectées par l'ordre des images codées et des unités NAL non VCL au sein d'une unité d'accès.

- Lorsqu'une unité NAL de délimiteur d'unité d'accès est présente, elle doit être la première unité NAL. Il doit y avoir au plus une unité NAL de délimiteur d'unité d'accès dans chaque unité d'accès.
- Lorsqu'une unité NAL SEI est présente, elle doit précéder l'image codée primaire.
- Lorsqu'une unité NAL SEI contenant un message SEI de période de temporisation est présente, le message SEI de période de temporisation doit être la première charge utile de message SEI de la première unité NAL SEI dans l'unité d'accès.
- L'image codée primaire doit précéder les images codées redondantes correspondantes.
- Lorsque les images codées redondantes sont présentes, elles doivent être ordonnées en ordre ascendant de la valeur de `redundant_pic_cnt`.
- Lorsqu'une unité NAL de fin de séquence est présente, elle doit suivre l'image codée primaire et toutes les images codées redondantes (s'il en est).
- Lorsqu'une unité NAL de fin de flux est présente, elle doit être la dernière unité NAL.
- Des unités NAL ayant `nal_unit_type` égal à 0, 12 ou dans la gamme de 19 à 31 inclus, ne doivent pas précéder la première unité NAL VCL de l'image codée primaire.

NOTE – Les unités NAL d'ensemble de paramètres de séquence ou d'ensemble de paramètres d'image peuvent être présentes dans une unité d'accès, mais ne peuvent pas suivre la dernière unité NAL VCL de l'image codée primaire au sein de l'unité d'accès, car cette condition spécifierait le début d'une nouvelle unité d'accès.

NOTE – Lorsqu'une unité NAL ayant `nal_unit_type` égal à 7 ou 8 est présente dans une unité d'accès, elle peut ne pas être référencée dans les images codées de l'unité d'accès dans laquelle elle est présente, et peut être référencée dans des images codées d'unités d'accès ultérieures.

La structure des unités d'accès ne contenant aucune unité NAL avec nal_unit_type égal à 0, 7, 8 ou dans la gamme de 12 à 31 inclus, est indiquée à la Figure 7-1.

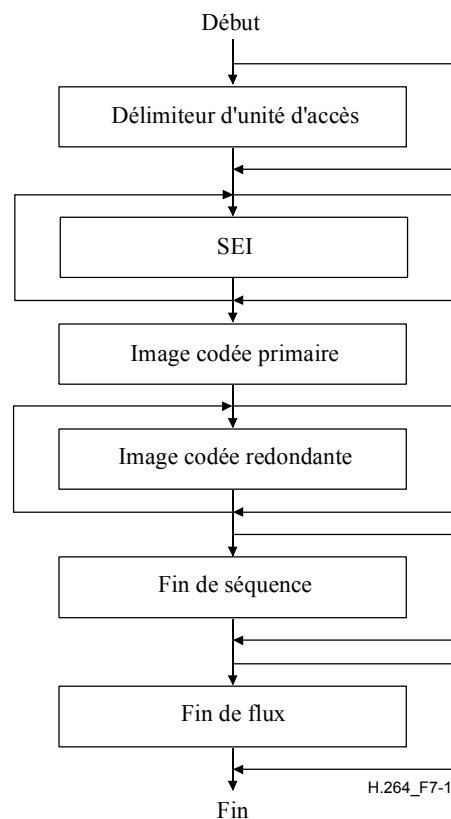


Figure 7-1 – Structure d'une unité d'accès ne contenant aucune unité NAL avec nal_unit_type égal à 0, 7, 8 ou dans la gamme de 12 à 31 inclus

7.4.1.2.4 Détection de la première unité NAL VCL d'une image codée primaire

Le présent paragraphe spécifie les contraintes applicables à la syntaxe d'une unité NAL VCL qui sont suffisantes pour permettre la détection de la première unité NAL VCL de chaque image codée primaire.

Toute unité NAL de tranche codée ou unité NAL de subdivision A de données de tranche codée de l'image codée primaire de l'unité d'accès en cours doit être différente de toute unité NAL de tranche codée ou unité NAL de subdivision A de données de tranche codée de l'image codée primaire de l'unité d'accès précédente d'une ou plusieurs des façons suivantes.

- frame_num diffère en valeur. La valeur de frame_num utilisée pour tester cette condition est la valeur de frame_num qui apparaît dans la syntaxe de l'en-tête de tranche, indépendamment de savoir s'il ressort que cette valeur a été égale à 0 pour utilisation ultérieure dans le processus de décodage en raison de la présence de l'opération de commande de gestion de mémoire égale à 5.

NOTE – Une des conséquences de l'énoncé ci-dessus est qu'une image codée primaire dont frame_num est égal à 1 ne peut pas contenir une opération memory_management_control_operation égale à 5 à moins qu'une des autres conditions énumérées ci-dessous ne soit remplie pour la prochaine image codée primaire qui la suit (le cas échéant).

- pic_parameter_set_id diffère en valeur.
- field_pic_flag diffère en valeur.
- bottom_field_flag est présent dans les deux et diffère en valeur.

- nal_ref_idc diffère en valeur avec une des valeurs nal_ref_idc égale à 0.
- pic_order_cnt_type est égal à 0 pour les deux et soit pic_order_cnt_lsb diffère en valeur, soit delta_pic_order_cnt_bottom diffère en valeur.
- pic_order_cnt_type est égal à 1 pour les deux et soit delta_pic_order_cnt[0] diffère en valeur, soit delta_pic_order_cnt[1] diffère en valeur.
- nal_unit_type diffère en valeur avec une des valeur nal_unit_type égale à 5.
- nal_unit_type est égal à 5 pour les deux et idr_pic_id diffère en valeur.

NOTE – Certaines des unités NAL VCL dans les images codées redondantes ou certaines unités NAL non VCL (par exemple, une unité NAL de délimiteur d'unité d'accès) peuvent aussi être utilisées pour la détection des frontières entre les unités d'accès, et peuvent donc aider à la détection du début d'une nouvelle image codée primaire.

7.4.1.2.5 Ordre des unités NAL VCL et association aux images codées

Chaque unité NAL VCL fait partie d'une image codée.

L'ordre des unités NAL VCL au sein d'une image IDR codée est obligatoirement le suivant.

- Si un ordre de tranche arbitraire est admis comme spécifié à l'Annexe A, la tranche codée des unités NAL d'une image IDR peut avoir n'importe quel ordre par rapport à chacune des autres.
- Autrement (l'ordre arbitraire de tranche n'est pas admis), l'ordre de tranche codée des unités NAL d'une image IDR doit être dans l'ordre croissant des adresses de macrobloc pour le premier macrobloc de chaque tranche codée d'unité NAL d'une image IDR.

L'ordre des unités NAL VCL au sein d'une image codée non IDR est obligatoirement le suivant.

- Si l'ordre arbitraire de tranche est admis comme spécifié à l'Annexe A, la tranche codée des unités NAL d'image non IDR ou des unités NAL de subdivision A de données de tranche codée peut avoir n'importe quel ordre par rapport à chacune des autres. Une unité NAL de subdivision A de données de tranche codée avec une valeur particulière de slice_id doit précéder chaque unité NAL de subdivision B de données de tranche codée présente avec la même valeur de slice_id. Une unité NAL de subdivision A de données de tranche codée avec une valeur particulière de slice_id doit précéder toute unité NAL de subdivision C de données de tranche codée présente avec la même valeur de slice_id. Lorsqu'une unité NAL de subdivision B de données de tranche codée avec une valeur particulière de slice_id est présente, elle doit précéder toute unité NAL de subdivision C de données de tranche codée avec la même valeur de slice_id.
- Autrement (l'ordre arbitraire de tranche n'est pas admis), l'ordre de tranche codée des unités NAL d'image non IDR ou des unités NAL de subdivision A de données de tranche codée doit être l'ordre croissant des adresses de macrobloc pour le premier macrobloc de chaque tranche codée d'une unité NAL d'image non IDR ou d'une unité NAL de subdivision A de données de tranche codée. Une unité NAL avec une valeur particulière de slice_id doit immédiatement précéder toute unité NAL de subdivision B de données de tranche codée avec la même valeur de slice_id. Une unité NAL de subdivision A de données de tranche codée avec une valeur particulière de slice_id doit immédiatement précéder toute unité NAL de subdivision A de données de tranche codée présente avec la même valeur de slice_id, lorsque n'est pas présente une unité NAL de subdivision B de données de tranche codée avec la même valeur de slice_id. Lorsqu'est présente une unité NAL de subdivision B de données de tranche codée avec une valeur particulière de slice_id, elle doit immédiatement précéder toute unité NAL de subdivision C de données de tranche codée avec la même valeur de slice_id.

Les unités NAL ayant `nal_unit_type` égal à 12 peuvent être présentes dans l'unité d'accès mais elles ne doivent pas précéder la première unité NAL VCL de l'image codée primaire au sein de l'unité d'accès.

Les unités NAL ayant `nal_unit_type` égal à 0 ou dans la gamme de 24 à 31 inclus, qui ne sont pas spécifiés, peuvent être présentes dans une unité d'accès mais ne doivent pas précéder la première unité NAL VCL de l'image codée primaire au sein de l'unité d'accès.

Les unités NAL ayant `nal_unit_type` dans la gamme de 19 à 23 inclus, qui sont réservés, ne doivent pas précéder la première unité NAL VCL de l'image codée primaire au sein de l'unité d'accès (lorsqu'elle sera spécifiée dans l'avenir par l'UIT-T | ISO/CEI).

7.4.2 Charges utiles de séquence d'octets bruts et sémantique des bits de traîne RBSP

7.4.2.1 Sémantique de la charge utile RBSP d'ensemble de paramètres de séquence

`profile_idc` et `level_idc` indiquent le profil et le niveau auquel se conforme le flux binaire, comme spécifié à l'Annexe A.

`constraint_set0_flag` égal à 1 indique que le flux binaire obéit à toutes les contraintes spécifiées au § A.2.1. `constraint_set0_flag` égal à 0 indique que le flux binaire peut obéir ou non à toutes les contraintes spécifiées au § A.2.1.

`constraint_set1_flag` égal à 1 indique que le flux binaire obéit à toutes les contraintes spécifiées au § A.2.2. `constraint_set1_flag` égal à 0 indique que le flux binaire peut ou non obéir à toutes les contraintes spécifiées au § A.2.2.

`constraint_set2_flag` égal à 1 indique que le flux binaire obéit à toutes les contraintes spécifiées au § A.2.3. `constraint_set2_flag` égal à 0 indique que le flux binaire peut ou non obéir à toutes les contraintes spécifiées au § A.2.3.

NOTE – Lorsque plus d'une instruction `constraint_set0_flag`, `constraint_set1_flag` ou `constraint_set2_flag` est égale à 1, le flux binaire obéit aux contraintes de tous les paragraphes indiqués du § A.2.

`reserved_zero_5bits` doit être égal à 0 dans les flux binaires se conformant à la présente Recommandation | Norme internationale. Les autres valeurs de `reserved_zero_5bits` pourront à l'avenir être spécifiées par l'UIT-T | ISO/CEI. Les décodeurs doivent ignorer la valeur de `reserved_zero_5bits`.

`seq_parameter_set_id` identifie les ensembles de paramètres de séquence auxquels se réfère l'ensemble de paramètres d'image. La valeur de `seq_parameter_set_id` doit être dans la gamme de 0 à 31 inclus.

NOTE – Lorsque c'est faisable, les codeurs devraient utiliser des valeurs distinctes de `seq_parameter_set_id` lorsque les valeurs des autres éléments syntaxiques d'ensembles de paramètres de séquence diffèrent plutôt que de changer les valeurs des éléments syntaxiques associés à une valeur spécifique de `seq_parameter_set_id`.

`log2_max_frame_num_minus4` spécifie la valeur de la variable `MaxFrameNum` qui est utilisée comme suit dans les calculs de `frame_num` qui s'y rapportent:

$$\text{MaxFrameNum} = 2^{(\text{log2_max_frame_num_minus4} + 4)} \quad (7-1)$$

La valeur de `log2_max_frame_num_minus4` doit être dans la gamme de 0 à 12 inclus.

`pic_order_cnt_type` spécifie la méthode de décodage du compte d'ordre d'image (comme spécifié au § 8.2.1). La valeur de `pic_order_cnt_type` doit être dans la gamme de 0 à 2 inclus.

`pic_order_cnt_type` ne doit pas être égal à 2 dans une séquence vidéo codée qui contient un des éléments suivants:

- une unité d'accès contenant une trame de non-référence suivie immédiatement par une unité d'accès contenant une image de non-référence;

- deux unités d'accès contenant chacune un champ avec les deux champs formant ensemble une paire complémentaire de champs de non-référence suivie immédiatement par une unité d'accès contenant une image de non-référence;
- une unité d'accès contenant un champ de non-référence suivi immédiatement par une unité d'accès contenant une autre image de non-référence qui ne forme pas une paire complémentaire de champs de non-référence avec la première des deux unités d'accès.

log2_max_pic_order_cnt_lsb_minus4 spécifie la valeur de la variable `MaxPicOrderCntLsb` qui est utilisée comme suit dans le processus de décodage pour le compte d'ordre d'image comme spécifié au § 8.2.1:

$$\text{MaxPicOrderCntLsb} = 2^{(\text{log2_max_pic_order_cnt_lsb_minus4} + 4)} \quad (7-2)$$

La valeur de `log2_max_pic_order_cnt_lsb_minus4` doit être dans la gamme de 0 à 12 inclus.

delta_pic_order_always_zero_flag égal à 1 spécifie que `delta_pic_order_cnt[0]` et `delta_pic_order_cnt[1]` ne sont pas présents dans les en-têtes de tranche de la séquence et doivent être supposés égaux à 0. `delta_pic_order_always_zero_flag` égal à 0 spécifie que `delta_pic_order_cnt[0]` est présent dans les en-têtes de tranche de la séquence et `delta_pic_order_cnt[1]` peut être présent dans les en-têtes de tranche de la séquence.

offset_for_non_ref_pic est utilisé pour calculer le compte d'ordre d'image d'une image de non-référence comme spécifié au § 8.2.1. La valeur de `offset_for_non_ref_pic` doit être dans la gamme de -2^{31} à $2^{31} - 1$ inclus.

offset_for_top_to_bottom_field est utilisé pour calculer le compte d'ordre d'image du champ inférieur dans une trame comme spécifié au § 8.2.1. La valeur de `offset_for_top_to_bottom_field` doit être dans la gamme de -2^{31} à $2^{31} - 1$ inclus.

num_ref_frames_in_pic_order_cnt_cycle est utilisé dans le processus de décodage pour le compte d'ordre d'image comme spécifié au § 8.2.1. La valeur de `num_ref_frames_in_pic_order_cnt_cycle` doit être dans la gamme de 0 à 255 inclus.

offset_for_ref_frame[i] est un élément d'une liste de valeurs de `num_ref_frames_in_pic_order_cnt_cycle` utilisées dans le processus de décodage pour le compte d'ordre d'image comme spécifié au § 8.2.1. La valeur de `offset_for_ref_frame[i]` doit être dans la gamme de -2^{31} à $2^{31} - 1$ inclus.

num_ref_frames spécifie le nombre maximal de trames de référence de court terme et de long terme, de paires de champs de référence complémentaires et de champs de référence non appariés qui peuvent être utilisés par le processus de décodage pour l'interprédiction de toute image dans la séquence. `num_ref_frames` détermine aussi la taille de l'opération de fenêtre glissante, comme spécifié au § 8.2.5.3. La valeur de `num_ref_frames` doit être dans la gamme de 0 à `MaxDpbSize` (comme spécifié au § A.3.1) inclus.

gaps_in_frame_num_value_allowed_flag spécifie les valeurs admises de `frame_num` comme spécifié au § 7.4.3 et le processus de décodage dans le cas de la déduction d'un trou dans les valeurs de `frame_num` comme spécifié au § 8.2.5.2.

pic_width_in_mbs_minus1 plus 1 spécifie la largeur de chaque image décodée en unités de macroblocs.

La variable pour la largeur d'image en unités de macroblocs est déduite comme suit:

$$\text{PicWidthInMbs} = \text{pic_width_in_mbs_minus1} + 1 \quad (7-3)$$

La variable pour la largeur d'image pour le composant luma est déduite comme suit:

$$\text{PicWidthInSamples}_L = \text{PicWidthInMbs} * 16 \quad (7-4)$$

La variable pour la largeur d'image pour les composants chroma est déduite comme suit:

$$\text{PicWidthInSamples}_C = \text{PicWidthInMbs} * 8 \quad (7-5)$$

pic_height_in_map_units_minus1 plus 1 spécifie la hauteur en unités de mapage de groupe de tranches d'une trame ou champ décodé.

Les variables **PicHeightInMapUnits** et **PicSizeInMapUnits** sont déduites comme suit:

$$\text{PicHeightInMapUnits} = \text{pic_height_in_map_units_minus1} + 1 \quad (7-6)$$

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

frame_mbs_only_flag égal à 0 spécifie que les images codées de la séquence vidéo codée peuvent être soit des champs codés soit des trames codées. **frame_mbs_only_flag** égal à 1 spécifie que chaque image codée de la séquence vidéo codée est une trame codée ne contenant que des macroblocs de trame.

La gamme admise de valeurs pour **pic_width_in_mbs_minus1**, **pic_height_in_map_units_minus1**, et **frame_mbs_only_flag** est spécifiée par des contraintes à l'Annexe A.

En fonction de **frame_mbs_only_flag**, la sémantique est attribuée comme suit à **pic_height_in_map_units_minus1**.

- Si **frame_mbs_only_flag** est égal à 0, **pic_height_in_map_units_minus1** plus 1 est la hauteur d'un champ en unités de macroblocs.
- Autrement (**frame_mbs_only_flag** est égal à 1), **pic_height_in_map_units_minus1** plus 1 est la hauteur d'une trame en unités de macroblocs.

La variable **FrameHeightInMbs** est déduite comme suit:

$$\text{FrameHeightInMbs} = (2 - \text{frame_mbs_only_flag}) * \text{PicHeightInMapUnits} \quad (7-8)$$

mb_adaptive_frame_field_flag égal à 0 spécifie qu'il n'y a pas de commutation entre macroblocs de trame et de champ au sein d'une image. **mb_adaptive_frame_field_flag** égal à 1 spécifie l'utilisation possible d'une commutation entre macroblocs de trame et de champ au sein des trames. Lorsque **mb_adaptive_frame_field_flag** n'est pas présent, on doit déduire qu'il est égal à 0.

direct_8x8_inference_flag spécifie la méthode utilisée dans le processus de déduction pour les vecteurs cinétiques luma pour **B_Skip**, **B_Direct_16x16** et **B_Direct_8x8** comme spécifié au § 8.4.1.2. Lorsque **frame_mbs_only_flag** est égal à 0, **direct_8x8_inference_flag** doit être égal à 1.

frame_cropping_flag égal à 1 spécifie que les paramètres de décalage de cadrage de trame viennent à la suite dans l'ensemble de paramètres de séquence. **frame_cropping_flag** égal à 0 spécifie que les paramètres de décalage de cadrage de trame ne sont pas présents.

frame_crop_left_offset, **frame_crop_right_offset**, **frame_crop_top_offset**, **frame_crop_bottom_offset** spécifient comme suit les échantillons d'une trame au sein d'un rectangle:

- Si **frame_mbs_only_flag** est égal à 1, la mire contient des échantillons luma avec des coordonnées horizontales de $2 * \text{frame_crop_left_offset}$ jusqu'à $\text{PicWidthInSamples}_L - (2 * \text{frame_crop_right_offset} + 1)$ et des coordonnées verticales de $2 * \text{frame_crop_top_offset}$ jusqu'à $(\text{FrameHeightInMbs} * 16) - (2 * \text{frame_crop_bottom_offset} + 1)$ inclus. Dans ce cas, la valeur de **frame_crop_left_offset** doit être dans la gamme de 0 à $8 * \text{PicWidthInMbs} - (\text{frame_crop_right_offset} + 1)$ inclus; et la valeur de **frame_crop_top_offset** doit être dans la gamme de 0 à $8 * \text{FrameHeightInMbs} - (\text{frame_crop_bottom_offset} + 1)$ inclus.
- Autrement (**frame_mbs_only_flag** est égal à 0), la mire contient des échantillons luma avec des coordonnées horizontales de $2 * \text{frame_crop_left_offset}$ jusqu'à $\text{PicWidthInSamples}_L - (2 * \text{frame_crop_right_offset} + 1)$ et des coordonnées verticales de $4 * \text{frame_crop_top_offset}$ jusqu'à $(\text{FrameHeightInMbs} * 16) - (4 * \text{frame_crop_bottom_offset} + 1)$ inclus. Dans ce cas la valeur de

`frame_crop_left_offset` doit être dans la gamme de 0 à $8 * \text{PicWidthInMbs} - (\text{frame_crop_right_offset} + 1)$ inclus; et la valeur de `frame_crop_top_offset` doit être dans la gamme de 0 à $4 * \text{FrameHeightInMbs} - (\text{frame_crop_bottom_offset} + 1)$ inclus.

Lorsque `frame_cropping_flag` est égal à 0, on doit déduire les valeurs suivantes: `frame_crop_left_offset` = 0, `frame_crop_right_offset` = 0, `frame_crop_top_offset` = 0 et `frame_crop_bottom_offset` = 0.

Les échantillons spécifiés des deux matrices chroma sont les échantillons qui ont les coordonnées de trame $(x/2, y/2)$, où (x, y) sont les coordonnées de trame des échantillons luma spécifiés.

Pour les champs décodés, les échantillons spécifiés des champs décodés sont les échantillons qui tombent dans les rectangles spécifiés dans les coordonnées de trame.

`vui_parameters_present_flag` égal à 1 spécifie que la structure syntaxique `vui_parameters()` spécifiée à l'Annexe E est présente à la suite dans le flux binaire. `vui_parameters_present_flag` égal à 0 spécifie que la structure syntaxique `vui_parameters()` spécifiée à l'Annexe E n'est pas présente à la suite dans le flux binaire.

7.4.2.2 Sémantique de la charge utile RBSP d'ensemble de paramètres d'image

`pic_parameter_set_id` identifie l'ensemble de paramètres d'image qui est référencé dans l'en-tête de tranche. La valeur de `pic_parameter_set_id` doit être dans la gamme de 0 à 255 inclus.

`seq_parameter_set_id` se rapporte à l'ensemble actif de paramètres de séquence. La valeur de `seq_parameter_set_id` doit être dans la gamme de 0 à 31 inclus.

`entropy_coding_mode_flag` choisit la méthode de décodage entropique à appliquer pour les éléments syntaxiques pour lesquels deux descripteurs apparaissent comme suit dans les tableaux syntaxiques:

- Si `entropy_coding_mode_flag` est égal à 0, on applique la méthode spécifiée par le descripteur de gauche dans le tableau syntaxique (Exp-Golomb codé, voir le § 9.1 ou CAVLC, voir le § 9.2).
- Autrement (`entropy_coding_mode_flag` est égal à 1), on applique la méthode spécifiée par le descripteur de droite dans le tableau syntaxique (CABAC, voir le § 9.3).

`pic_order_present_flag` égal à 1 spécifie que les éléments syntaxiques se rapportant au compte d'ordre d'image sont présents dans les en-têtes de tranche comme spécifié au § 7.3.3. `pic_order_present_flag` égal à 0 spécifie que les éléments syntaxiques se rapportant au compte d'ordre d'image ne sont pas présents dans les en-têtes de tranche.

`num_slice_groups_minus1` plus 1 spécifie le nombre des groupes de tranches pour une image. Lorsque `num_slice_groups_minus1` est égal à 0, toutes les tranches de l'image appartiennent au même groupe de tranches. La gamme admise de `num_slice_groups_minus1` est spécifiée à l'Annexe A.

`slice_group_map_type` spécifie comment est codée le mappage d'unités de mappage de groupe de tranches avec les groupes de tranches. La valeur de `slice_group_map_type` doit être dans la gamme de 0 à 6 inclus.

`slice_group_map_type` égal à 0 spécifie des groupes de tranches entrelacés.

`slice_group_map_type` égal à 1 spécifie un mappage dispersé de groupe de tranches.

`slice_group_map_type` égal à 2 spécifie un ou plusieurs groupes de tranches "de premier plan" et un groupe de tranches "de reste".

Les valeurs de `slice_group_map_type` égales à 3, 4 et 5 spécifient un changement de groupes de tranches. Lorsque `num_slice_groups_minus1` n'est pas égal à 1, `slice_group_map_type` ne doit pas être égal à 3, 4 ou 5.

slice_group_map_type égal à 6 spécifie une allocation explicite d'un groupe de tranches à chaque unité de mappage de groupe de tranches.

Les unités de mappage de groupe de tranches sont spécifiées comme suit.

- Si frame_mbs_only_flag est égal à 0 et mb_adaptive_frame_field_flag est égal à 1 et si l'image codée est une trame, les unités de mappage de groupe de tranches sont des unités de paires de macroblocs.
- Autrement, si frame_mbs_only_flag est égal à 1 ou si une image codée est un champ, les unités de mappage de groupe de tranches sont des unités de macroblocs.
- Autrement (frame_mbs_only_flag est égal à 0 et mb_adaptive_frame_field_flag est égal à 0 et l'image codée est une trame), les unités de mappage de groupe de tranches sont des unités de deux macroblocs qui sont verticalement contigus comme dans une paire de macroblocs de trame d'une trame MBAFF.

run_length_minus1[i] est utilisé pour spécifier le nombre d'unités de mappage de groupe de tranches consécutives à allouer au ième groupe de tranches dans l'ordre de balayage de trame d'unités de mappage de groupe de tranches. La valeur de run_length_minus1[i] doit être dans la gamme de 0 à PicSizeInMapUnits – 1 inclus.

top_left[i] et **bottom_right[i]** spécifient respectivement les coins supérieur gauche et inférieur droit d'un rectangle. top_left[i] et bottom_right[i] sont des positions d'unité de mappage de groupe de tranches dans un balayage de trame de l'image pour les unités de mappage de groupe de tranches. Pour chaque rectangle i, toutes les contraintes suivantes doivent être respectées par les valeurs des éléments syntaxiques top_left[i] et bottom_right[i]:

- top_left[i] doit être inférieur ou égal à bottom_right[i] et bottom_right[i] doit être inférieur à PicSizeInMapUnits;
- (top_left[i] % PicWidthInMbs) doit être inférieur ou égal à la valeur de (bottom_right[i] % PicWidthInMbs).

slice_group_change_direction_flag est utilisé avec slice_group_map_type pour spécifier le type de mappage précis lorsque slice_group_map_type est 3, 4 ou 5.

slice_group_change_rate_minus1 est utilisé pour spécifier la variable SliceGroupChangeRate. SliceGroupChangeRate spécifie le multiple en nombre d'unités de mappage de groupe de tranches par lequel peut changer la taille d'un groupe de tranches d'une image à la suivante. La valeur de slice_group_change_rate_minus1 doit être dans la gamme de 0 à PicSizeInMapUnits – 1 inclus. La variable SliceGroupChangeRate est spécifiée comme suit:

$$\text{SliceGroupChangeRate} = \text{slice_group_change_rate_minus1} + 1 \quad (7-9)$$

pic_size_in_map_units_minus1 est utilisé pour spécifier le nombre d'unités de mappage de groupe de tranches dans l'image. pic_size_in_map_units_minus1 doit être égal à PicSizeInMapUnits – 1.

slice_group_id[i] identifie un groupe de tranches de la ième unité de mappage de groupe de tranches dans l'ordre de balayage de trame. La taille de l'élément syntaxique slice_group_id[i] est de $\text{Ceil}(\text{Log}_2(\text{num_slice_groups_minus1} + 1))$ bits. La valeur de slice_group_id[i] doit être dans la gamme de 0 à num_slice_groups_minus1 inclus.

num_ref_idx_l0_active_minus1 spécifie l'indice de référence maximal pour la liste 0 d'image de référence qui doit être utilisée pour décoder chaque tranche de l'image dans laquelle la liste 0 est utilisée lorsque num_ref_idx_active_override_flag est égal à 0 pour la tranche. Lorsque MbaffFrameFlag est égal à 1, num_ref_idx_l0_active_minus1 est la valeur d'indice maximale pour le décodage de macroblocs de trame et $2 * \text{num_ref_idx_l0_active_minus1} + 1$ est la valeur d'indice maximale pour le décodage de macroblocs de champ. La valeur de num_ref_idx_l0_active_minus1 doit être dans la gamme de 0 à 31 inclus.

num_ref_idx_l1_active_minus1 a la même sémantique que **num_ref_idx_l0_active_minus1** avec l0 et liste 0 remplacés respectivement par l1 et liste 1.

weighted_pred_flag égal à 0 spécifie que la prédiction pondérée ne doit pas être appliquée aux tranches P et SP. **weighted_pred_flag** égal à 1 spécifie que la prédiction pondérée doit être appliquée aux tranches P et SP.

weighted_bipred_idc égal à 0 spécifie que la prédiction pondérée par défaut doit s'appliquer aux tranches B. **weighted_bipred_idc** égal à 1 spécifie que la prédiction pondérée explicite doit s'appliquer aux tranches B. **weighted_bipred_idc** égal à 2 spécifie que la prédiction pondérée implicite doit s'appliquer aux tranches B. La valeur de **weighted_bipred_idc** doit être dans la gamme de 0 à 2 inclus.

pic_init_qp_minus26 spécifie la valeur initiale moins 26 de SliceQP_Y pour chaque tranche. La valeur initiale est modifiée à la couche de tranche lorsqu'une valeur différente de zéro de **slice_qp_delta** est décodée, et est modifiée ultérieurement lorsqu'une valeur différente de zéro de **mb_qp_delta** est décodée à la couche de macrobloc. La valeur de **pic_init_qp_minus26** doit être dans la gamme de -26 à +25 inclus.

pic_init_qs_minus26 spécifie la valeur initiale moins 26 de SliceQS_Y pour tous les macroblocs dans les tranches SP ou SI. La valeur initiale est modifiée à la couche de tranche lorsqu'une valeur différente de zéro de **slice_qs_delta** est décodée. La valeur de **pic_init_qs_minus26** doit être dans la gamme de -26 à +25 inclus.

chroma_qp_index_offset spécifie le décalage qui doit être ajouté à QP_Y et QS_Y pour s'appliquer au tableau des valeurs de QP_C . La valeur de **chroma_qp_index_offset** doit être dans la gamme de -12 à +12 inclus.

deblocking_filter_control_present_flag égal à 1 spécifie qu'un ensemble d'éléments syntaxiques contrôlant les caractéristiques du filtre de déblocage est présent dans l'en-tête de tranche. **deblocking_filter_control_present_flag** égal à 0 spécifie que l'ensemble d'éléments syntaxiques contrôlant les caractéristiques du filtre de déblocage n'est pas présent dans l'en-tête de tranches et que ce sont leurs valeurs déduites qui sont actives.

constrained_intra_pred_flag égal à 0 spécifie que l'intraprédiction permet l'utilisation des données résiduelles et des échantillons décodés des macroblocs du voisinage codés en utilisant les modes de prédiction intermacrobloc pour la prédiction des macroblocs codés en utilisant les modes de prédiction intramacrobloc. **constrained_intra_pred_flag** égal à 1 spécifie une intraprediction soumise à des contraintes, auquel cas la prédiction des macroblocs codés en utilisant les modes de prédiction intramacrobloc n'utilise que les données résiduelles et les échantillons décodés provenant des types de macrobloc I ou SI.

redundant_pic_cnt_present_flag égal à 0 spécifie que l'élément syntaxique **redundant_pic_cnt** n'est pas présent dans l'en-tête de tranches, les subdivisions B de données et les subdivisions C de données qui se réfèrent (directement ou par association avec une subdivision A de données correspondante) à l'ensemble de paramètres d'image. **redundant_pic_cnt_present_flag** égal à 1 spécifie que l'élément syntaxique **redundant_pic_cnt** est présent dans tout en-tête de tranches, subdivision B de données et subdivision C de données qui se réfère (directement ou par association avec une subdivision A de données correspondante) à l'ensemble de paramètres d'image.

7.4.2.3 Sémantique de charge utile RBSP d'informations d'amélioration supplémentaires

Les informations d'amélioration supplémentaires (SEI) contiennent des informations qui ne sont pas nécessaires pour décoder les échantillons d'images codées à partir des unités NAL VCL.

7.4.2.3.1 Sémantique de message d'informations d'amélioration supplémentaires

Une unité NAL SEI contient un ou plusieurs messages SEI. Chaque message SEI se compose des variables spécifiant le type **payloadType** et la taille **payloadSize** de la charge utile SEI. Les charges

utiles SEI sont spécifiées à l'Annexe D. La taille de charge utile de SEI déduite `payloadSize` est spécifiée en octets et doit être égale au nombre d'octets de la charge utile SEI.

ff_byte est un octet égal à 0xFF identifiant le besoin d'une plus longue représentation de la structure syntaxique qui est utilisée en son sein.

last_payload_type_byte est le dernier octet du type de charge utile d'un message SEI.

last_payload_size_byte est le dernier octet de la taille d'un message SEI.

7.4.2.4 Sémantique de la charge utile RBSP de délimiteur d'unité d'accès

Le délimiteur d'unité d'accès peut être utilisé pour indiquer le type de tranche présente dans une image codée primaire et pour simplifier la détection de la frontière entre les unités d'accès. Il n'y a pas de processus normatif de décodage associé au délimiteur d'unité d'accès.

primary_pic_type indique que les valeurs de `slice_type` pour toutes les tranches de l'image codée primaire font partie de l'ensemble dont la liste figure au Tableau 7-2 pour les valeurs données de `primary_pic_type`.

Tableau 7-2 – Signification de `primary_pic_type`

<code>primary_pic_type</code>	Valeurs de <code>slice_type</code> qui peuvent être présentes dans l'image codée primaire
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

7.4.2.5 Sémantique de charge utile RBSP de fin de séquence

La charge utile RBSP de fin de séquence spécifie que l'unité d'accès suivante dans le flux binaire dans l'ordre de décodage (s'il en est) doit être une unité d'accès IDR. Le contenu syntaxique de la chaîne SODB et de la charge utile RBSP pour la fin de la séquence RBSP est vide. Aucun processus de décodage normalisé n'est spécifié pour une RBSP de fin de séquence.

7.4.2.6 Sémantique de charge utile RBSP de fin de flux

La charge utile RBSP de fin de flux indique qu'aucune unité NAL supplémentaire ne doit être présente dans le flux binaire qui suit la charge utile RBSP de fin de flux dans l'ordre de décodage. Le contenu syntaxique de la chaîne SODB et de la charge utile RBSP pour la RBSP de fin de flux est vide. Aucun processus de décodage normalisé n'est spécifié pour une RBSP de fin de flux.

7.4.2.7 Sémantique de charge utile RBSP de données de remplissage

La charge utile RBSP de données de remplissage contient des octets dont la valeur doit être égale à 0xFF. Aucun processus de décodage normalisé n'est spécifié pour une RBSP de données de remplissage.

ff_byte est un octet égal à 0xFF.

7.4.2.8 Sémantique de charge utile RBSP de couche de tranche sans subdivision

La charge utile RBSP de couche de tranche sans subdivision consiste en un en-tête de tranche et des données de tranche.

7.4.2.9 Sémantique de charge utile RBSP de subdivision de données de tranche

7.4.2.9.1 Sémantique de charge utile RBSP de subdivision A de données de tranche

Lorsqu'on utilise la subdivision de données de tranche, les données codées pour une tranche seule sont partagées en trois subdivisions séparées. La subdivision A contient tous les éléments syntaxiques de catégorie 2.

Les éléments syntaxiques de catégorie 2 incluent tous les éléments syntaxiques des structures syntaxiques d'en-tête de tranche et de données de tranche autres que les éléments syntaxiques dans les structures syntaxiques residual().

slice_id identifie la tranche associée à la subdivision de données. Chaque tranche doit avoir une valeur de **slice_id** unique au sein de l'image codée qui contient la tranche. Lorsqu'un ordre de tranche arbitraire n'est pas admis, comme spécifié à l'Annexe A, la première tranche d'une image codée, dans l'ordre de décodage, doit avoir **slice_id** égal à 0 et la valeur de **slice_id** doit être incrémentée de un pour chaque tranche suivante de l'image codée dans l'ordre de décodage.

La gamme de **slice_id** est spécifiée comme suit.

- Si **MbaffFrameFlag** est égal à 0, **slice_id** doit être dans la gamme de 0 à $\text{PicSizeInMbs} - 1$ inclus.
- Autrement (**MbaffFrameFlag** est égal à 1), **slice_id** doit être dans la gamme de 0 à $\text{PicSizeInMbs} / 2 - 1$ inclus.

7.4.2.9.2 Sémantique de charge utile RBSP de subdivision B de données de tranche

Lorsqu'on utilise la subdivision de données de tranche, les données codées pour une tranche seule sont partagées en trois subdivisions séparées. La subdivision B contient tous les éléments syntaxiques de catégorie 3.

Les éléments syntaxiques de catégorie 3 incluent tous les éléments syntaxiques des structures syntaxiques residual() et des structures syntaxiques utilisées au sein de cette structure syntaxique pour les types collectifs de macrobloc I et SI comme spécifié au Tableau 7-7.

slice_id a la même sémantique que spécifié au § 7.4.2.9.1.

redundant_pic_cnt doit être égal à 0 pour les tranches et les subdivisions de données de tranche appartenant à l'image codée primaire. Le compte **redundant_pic_cnt** doit être supérieur à 0 pour les tranches codées et les subdivisions de données de tranche codée dans les images codées redondantes. Lorsque **redundant_pic_cnt** n'est pas présent, sa valeur doit être supposée égale à 0. La valeur de **redundant_pic_cnt** doit être dans la gamme de 0 à 127 inclus.

La présence d'une charge utile RBSP de subdivision B de données de tranche est spécifiée comme suit.

- Si les éléments syntaxiques d'une charge utile RBSP de subdivision B de données de tranche indiquent la présence de tout élément syntaxique de catégorie 3 dans les données de tranche pour une tranche, une charge utile RBSP de subdivision B de données de tranche ayant la même valeur de **slice_id** et de **redundant_pic_cnt** que dans la RBSP de subdivision A de données de tranche doit être présente.
- Autrement (les éléments syntaxiques d'une charge utile RBSP de subdivision A de données de tranche n'indiquent pas la présence d'éléments syntaxiques de catégorie 3 dans les données de tranche pour une tranche), aucune charge utile RBSP de subdivision B de données de tranche ayant la même valeur de **slice_id** et de **redundant_pic_cnt** que dans la RBSP de subdivision A de données de tranche ne doit être présente.

7.4.2.9.3 Sémantique de charge utile RBSP de subdivision C de données de tranche

Lorsqu'on utilise la subdivision de données de tranche, les données codées pour une tranche seule sont partagées en trois subdivisions séparées. La subdivision C contient tous les éléments syntaxiques de catégorie 4.

Les éléments syntaxiques de catégorie 4 incluent tous les éléments syntaxiques des structures syntaxiques residual() et des structures syntaxiques utilisées au sein de cette structure syntaxique pour les types collectifs de macrobloc P et B comme spécifié au Tableau 7-7.

slice_id a la même sémantique que spécifié au § 7.4.2.9.1.

redundant_pic_cnt a la même sémantique que spécifié au § 7.4.2.9.2.

La présence d'une charge utile RBSP de subdivision C de données de tranche est spécifiée comme suit.

- Si les éléments syntaxiques d'une charge utile RBSP de subdivision A de données de tranche indiquent la présence de tout élément syntaxique de catégorie 4 dans les données de tranche pour une tranche, une charge utile RBSP de subdivision C de données de tranche ayant la même valeur de **slice_id** et de **redundant_pic_cnt** que dans la RBSP de subdivision A de données de tranche doit être présente.
- Autrement (les éléments syntaxiques d'une charge utile RBSP de subdivision A de données de tranche n'indiquent pas la présence d'éléments syntaxiques de catégorie 4 dans les données de tranche pour une tranche), aucune charge utile RBSP de subdivision C de données de tranche ayant la même valeur de **slice_id** et de **redundant_pic_cnt** que dans la RBSP de subdivision A de données de tranche ne doit être présente.

7.4.2.10 Sémantique des bits de traîne de tranche de charge utile RBSP

cabac_zero_word est une séquence alignée sur les octets de deux octets égale à 0x0000.

Soit **NumBytesInVclNALunits** la somme des valeurs de **NumBytesInNALunit** pour toutes les unités NAL VCL d'une image codée.

Soit **BinCountsInNALunits** le nombre de fois que la fonction du processus d'analyse grammaticale **DecodeBin()**, spécifiée au § 9.3.3.2, est invoquée pour décoder le contenu de toutes les unités NAL VCL d'une image codée. Lorsque **entropy_coding_mode_flag** est égal à 1, **BinCountsInNALunits** ne doit pas excéder $(32 \div 3) * \text{NumBytesInVclNALunits} + 96 * \text{PicSizeInMbs}$.

NOTE – La contrainte sur le nombre maximum de segments résultant du décodage du contenu de la couche de tranche des unités NAL peut être satisfaite en insérant un nombre d'éléments syntaxiques **cabac_zero_word** pour augmenter la valeur de **NumBytesInVclNALunits**. Chaque **cabac_zero_word** est représenté dans une unité NAL par la séquence de trois octets 0x000003 (résultant de la contrainte sur le contenu de l'unité NAL qui requiert l'inclusion d'un **emulation_prevention_three_byte** pour chaque **cabac_zero_word**).

7.4.2.11 Sémantique des bits de traîne de charge utile RBSP

rbsp_stop_one_bit est un bit unique égal à 1.

rbsp_alignment_zero_bit est un bit unique égal à 0.

7.4.3 Sémantique d'en-tête de tranche

Lorsqu'elle est présente, la valeur des éléments syntaxiques d'en-tête de tranche **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** et **slice_group_change_cycle** doit être la même dans tous les en-têtes de tranche d'une image codée.

first_mb_in_slice spécifie l'adresse du premier macrobloc dans la tranche. Lorsque l'ordre arbitraire de tranche n'est pas admis comme spécifié à l'Annexe A, la valeur de **first_mb_in_slice** ne

doit pas être inférieure à la valeur de `first_mb_in_slice` pour toute autre tranche de l'image en cours qui précède la tranche en cours dans l'ordre de décodage.

La première adresse de macrobloc de la tranche est déduite comme suit.

- Si `MbaffFrameFlag` est égal à 0, `first_mb_in_slice` est l'adresse de macrobloc du premier macrobloc dans la tranche et `first_mb_in_slice` doit être dans la gamme de 0 à `PicSizeInMbs - 1` inclus.
- Autrement (`MbaffFrameFlag` est égal à 1), `first_mb_in_slice * 2` est l'adresse de macrobloc du premier macrobloc dans la tranche, qui est le macrobloc supérieur de la première paire de macroblocs dans la tranche, et `first_mb_in_slice` doit être dans la gamme de 0 à `PicSizeInMbs / 2 - 1` inclus.

slice_type spécifie le type de codage de la tranche conformément au Tableau 7-3.

Tableau 7-3 – Association du nom au slice_type

slice_type	Nom de slice_type
0	P (tranche P)
1	B (tranche B)
2	I (tranche I)
3	SP (tranche SP)
4	SI (tranche SI)
5	P (tranche P)
6	B (tranche B)
7	I (tranche I)
8	SP (tranche SP)
9	SI (tranche SI)

Les valeurs de `slice_type` dans la gamme de 5..9 spécifient, en plus du type de codage de la tranche en cours, que toutes les autres tranches de l'image codée en cours doivent avoir une valeur de `slice_type` égale à la valeur courante de `slice_type` ou égale à la valeur courante de `slice_type - 5`.

Lorsque `nal_unit_type` est égal à 5 (image IDR), `slice_type` doit être égal à 2, 4, 7 ou 9.

Lorsque `num_ref_frames` est égal à 0, `slice_type` doit être égal à 2, 4, 7 ou 9.

pic_parameter_set_id spécifie l'ensemble de paramètres d'image utilisé. La valeur de `pic_parameter_set_id` doit être dans la gamme de 0 à 255 inclus.

frame_num est utilisé comme un identificateur pour les images et doit être représenté par `log2_max_frame_num_minus4 + 4` bits dans le flux binaire. `frame_num` est soumis aux contraintes suivantes.

La variable `PrevRefFrameNum` est déduite comme suit:

- Si l'image en cours est une image IDR, `PrevRefFrameNum` est mis égal à 0.
- Autrement (l'image en cours n'est pas une image IDR), `PrevRefFrameNum` est mis égal à la valeur de `frame_num` pour l'unité d'accès précédente dans l'ordre de décodage qui contient une image de référence.

La valeur de `frame_num` est soumise aux contraintes suivantes:

- Si l'image en cours est une image IDR, `frame_num` doit être égal à 0.
- Autrement (l'image en cours n'est pas une image IDR), par rapport à l'image codée primaire dans la précédente unité d'accès dans l'ordre de décodage qui contient une image de référence en tant que précédente image de référence, la valeur de `frame_num` pour l'image en cours ne doit pas être égale à `PrevRefFrameNum` à moins que toutes les trois conditions suivantes ne soient Vraies:
 - l'image en cours et la précédente image de référence appartiennent à des unités d'accès consécutives dans l'ordre de décodage;
 - l'image en cours et la précédente image de référence sont des champs de référence ayant une parité opposée;
 - une ou plusieurs des conditions suivantes sont Vraies:
 - l'image de référence précédente est une image IDR;
 - l'image de référence précédente inclut un élément syntaxique `memory_management_control_operation` égal à 5.
 - NOTE – Lorsque l'image de référence précédente inclut un élément syntaxique `memory_management_control_operation` égal à 5, `PrevRefFrameNum` est égal à 0;
 - il y a une image codée primaire qui précède la précédente image de référence et l'image codée primaire qui précède la précédente image de référence n'a pas de `frame_num` égal à `PrevRefFrameNum`;
 - il y a une image codée primaire qui précède la précédente image de référence et l'image codée primaire qui précède la précédente image de référence n'est pas une image de référence.

Lorsque la valeur de `frame_num` n'est pas égale à `PrevRefFrameNum`, on applique ce qui suit.

- Il ne doit pas y avoir de champ ou trame précédant dans l'ordre de décodage qui soit actuellement marqué comme "utilisé pour référence à court terme" qui ait une valeur de `frame_num` égale à toute valeur prise par la variable `UnusedShortTermFrameNum` dans ce qui suit:

```
UnusedShortTermFrameNum = ( PrevRefFrameNum + 1 ) % MaxFrameNum
while( UnusedShortTermFrameNum != frame_num )
    UnusedShortTermFrameNum = ( UnusedShortTermFrameNum + 1 ) % MaxFrameNum
```

(7-10)

- La valeur de `frame_num` est soumise aux contraintes suivantes:
 - Si `gaps_in_frame_num_value_allowed_flag` est égal à 0, la valeur de `frame_num` pour l'image en cours doit être égale à $(PrevRefFrameNum + 1) \% MaxFrameNum$.
 - Autrement (`gaps_in_frame_num_value_allowed_flag` est égal à 1), on applique ce qui suit:
 - Si `frame_num` est supérieur à `PrevRefFrameNum`, il ne doit y avoir aucune image de non-référence dans le flux binaire qui suit l'image de référence précédente et précède l'image en cours dans l'ordre de décodage pour lequel l'une des deux conditions suivantes est Vraie.
 - La valeur de `frame_num` pour l'image de non-référence est inférieure à `PrevRefFrameNum`.
 - La valeur de `frame_num` pour l'image de non-référence est supérieure à la valeur de `frame_num` pour l'image en cours.
 - Autrement (`frame_num` est inférieur à `PrevRefFrameNum`), il ne doit y avoir aucune image de non-référence dans le flux binaire qui suit l'image de référence

précédente et précède l'image en cours dans l'ordre de décodage pour lequel les deux conditions suivantes sont vraies.

- La valeur de `frame_num` pour l'image de non-référence est inférieure à `PrevRefFrameNum`.
- La valeur de `frame_num` pour l'image de non-référence est supérieure à la valeur de `frame_num` pour l'image en cours.

Une image incluant un `memory_management_control_operation` égal à 5 doit avoir un `frame_num` soumis aux contraintes décrites ci-dessus et après le décodage de l'image en cours et le traitement des opérations de commande de gestion de mémoire, l'image doit être supposée avoir eu `frame_num` égal à 0 pour toutes les utilisations suivantes dans le processus de décodage, sauf comme indiqué au § 7.4.1.2.4.

NOTE – Lorsque l'image codée primaire n'est pas une image IDR et ne contient pas d'élément syntaxique `memory_management_control_operation` égal à 5, la valeur de `frame_num` d'une image codée redondante est la même que la valeur de `frame_num` dans l'image codée primaire. Autrement, l'image codée redondante inclut un élément syntaxique `memory_management_control_operation` égal à 5 et l'image codée primaire correspondante est une image IDR.

field_pic_flag égal à 1 spécifie que la tranche est une tranche d'un champ codé. `field_pic_flag` égal à 0 spécifie que la tranche est une tranche d'une trame codée. Lorsque `field_pic_flag` n'est pas présent il doit être supposé égal à 0.

La variable `MbaffFrameFlag` est déduite comme suit:

$$\text{MbaffFrameFlag} = (\text{mb_adaptive_frame_field_flag} \ \&\& \ !\text{field_pic_flag}) \quad (7-11)$$

La variable pour la hauteur de l'image en unités de macroblocs est déduite comme suit:

$$\text{PicHeightInMbs} = \text{FrameHeightInMbs} / (1 + \text{field_pic_flag}) \quad (7-12)$$

La variable pour la hauteur de l'image pour le composant luma est déduite comme suit:

$$\text{PicHeightInSamples}_L = \text{PicHeightInMbs} * 16 \quad (7-13)$$

La variable pour la hauteur de l'image pour le composant chroma est déduite comme suit:

$$\text{PicHeightInSamples}_C = \text{PicHeightInMbs} * 8 \quad (7-14)$$

La variable `PicSizeInMbs` pour l'image en cours est déduite conformément à:

$$\text{PicSizeInMbs} = \text{PicWidthInMbs} * \text{PicHeightInMbs} \quad (7-15)$$

La variable `MaxPicNum` est déduite comme suit:

- Si `field_pic_flag` est égal à 0, `MaxPicNum` est mis égal à `MaxFrameNum`.
- Autrement (`field_pic_flag` est égal à 1), `MaxPicNum` est mis égal à $2 * \text{MaxFrameNum}$.

La variable `CurrPicNum` est déduite comme suit:

- Si `field_pic_flag` est égal à 0, `CurrPicNum` est mis égal à `frame_num`.
- Autrement (`field_pic_flag` est égal à 1), `CurrPicNum` est mis égal à $2 * \text{frame_num} + 1$.

bottom_field_flag égal à 1 spécifie que la tranche fait partie d'un champ inférieur codé. `bottom_field_flag` égal à 0 spécifie que l'image est un champ supérieur codé. Lorsque cet élément syntaxique n'est pas présent pour la tranche en cours, il doit être supposé égal à 0.

idr_pic_id identifie une image IDR. Les valeurs de `idr_pic_id` dans toutes les tranches d'une image IDR doivent rester inchangées. Lorsque deux unités d'accès consécutives dans l'ordre de décodage sont toutes deux des unités d'accès IDR, la valeur de `idr_pic_id` dans les tranches de la première d'une telle unité d'accès IDR doit différer de celle de `idr_pic_id` dans la seconde unité d'accès IDR. La valeur de `idr_pic_id` doit être dans la gamme de 0 à 65535 inclus.

pic_order_cnt_lsb spécifie le compte d'ordre d'image modulo `MaxPicOrderCntLsb` pour le champ supérieur d'une trame codée ou pour un champ codé. La taille de l'élément syntaxique `pic_order_cnt_lsb` est $\log_2 \text{max_pic_order_cnt_lsb_minus4} + 4$ bits. La valeur de `pic_order_cnt_lsb` doit être dans la gamme de 0 à `MaxPicOrderCntLsb - 1`, inclus.

delta_pic_order_cnt_bottom spécifie comme suit la différence de compte d'ordre d'image entre le champ inférieur et le champ supérieur d'une trame codée.

- Si l'image en cours inclut une `memory_management_control_operation` égale à 5, la valeur de `delta_pic_order_cnt_bottom` doit être dans la gamme de $(1 - \text{MaxPicOrderCntLsb})$ à $2^{31} - 1$ inclus.
- Autrement (l'image en cours n'inclut pas une `memory_management_control_operation` égale à 5), la valeur de `delta_pic_order_cnt_bottom` doit être dans la gamme de -2^{31} à $2^{31} - 1$ inclus.

Lorsque cet élément syntaxique n'est pas présent dans le flux binaire pour la tranche en cours, il doit être supposé égal à 0.

delta_pic_order_cnt[0] spécifie la différence de compte d'ordre d'image par rapport au compte d'ordre d'image attendu pour le champ supérieur d'une trame codée ou pour un champ codé comme spécifié au § 8.2.1. La valeur de `delta_pic_order_cnt[0]` doit être dans la gamme de -2^{31} à $2^{31} - 1$ inclus. Lorsque cet élément syntaxique n'est pas présent dans le flux binaire pour la tranche en cours, il doit être supposé égal à 0.

delta_pic_order_cnt[1] spécifie la différence de compte d'ordre d'image par rapport au compte d'ordre d'image attendu pour le champ inférieur d'une trame codée comme spécifié au § 8.2.1. La valeur de `delta_pic_order_cnt[1]` doit être dans la gamme de -2^{31} à $2^{31} - 1$ inclus. Lorsque cet élément syntaxique n'est pas présent dans le flux binaire pour la tranche en cours, il doit être supposé égal à 0.

redundant_pic_cnt doit être égal à 0 pour les tranches et subdivisions de données de tranche appartenant à l'image codée primaire. La valeur de `redundant_pic_cnt` doit être supérieure à 0 pour les tranches codées ou subdivisions de données de tranche codée d'une image codée redondante. Lorsque `redundant_pic_cnt` n'est pas présent dans le flux binaire, sa valeur doit être supposée égale à 0. La valeur de `redundant_pic_cnt` doit être dans la gamme de 0 à 127 inclus.

NOTE – Il ne devrait pas y avoir de différence notable entre toute zone de l'image primaire décodée et une zone correspondante résultant de l'application du processus de décodage spécifié à l'article 8 pour toute image redondante dans la même unité d'accès.

La valeur de `pic_parameter_set_id` dans une tranche codée ou subdivision de données de tranche codée d'une image codée redondante doit être telle que la valeur de `pic_order_present_flag` dans l'ensemble de paramètres d'image utilisés dans une image codée redondante soit égale à la valeur de `pic_order_present_flag` dans l'ensemble de paramètres d'image utilisé dans l'image codée primaire correspondante.

Lorsqu'ils sont présents dans l'image codée primaire et toute image codée redondante, les éléments syntaxiques suivants doivent avoir la même valeur: `field_pic_flag`, `bottom_field_flag` et `idr_pic_id`.

Lorsque la valeur de `nal_ref_idc` dans une unité NAL VCL d'une unité d'accès est égale à 0, la valeur de `nal_ref_idc` dans toutes les autres unités NAL VCL de la même unité d'accès doit être égale à 0.

NOTE – La contrainte ci-dessus a les implications suivantes. Si la valeur de `nal_ref_idc` pour les unités NAL VCL de l'image codée primaire est égale à 0, la valeur de `nal_ref_idc` pour les unités NAL VCL de toute image codée redondante correspondante est égale à 0; autrement (la valeur de `nal_ref_idc` pour les unités NAL VCL de l'image codée primaire est supérieure à 0), la valeur de `nal_ref_idc` pour les unités NAL VCL de toute image codée redondante correspondante est aussi supérieure à 0.

L'état de marquage des images de référence et la valeur de `frame_num` après que le processus de marquage d'image de référence décodée comme spécifié au § 8.2.5 est invoqué pour l'image codée primaire ou pour toute image codée redondante de la même unité d'accès, doivent être identiques, sans considérer si l'image codée primaire ou toute image codée redondante (à la place de l'image codée primaire) de l'unité d'accès serait décodée.

NOTE – La contrainte ci-dessus a aussi les implications suivantes.

Si une image codée primaire n'est pas une image IDR, le contenu de la structure syntaxique `dec_ref_pic_marking()` doit être identique dans tous les en-têtes de tranche de l'image codée primaire et toutes les images codées redondantes correspondantes à l'image codée primaire.

Autrement (une image codée primaire est une image IDR), on applique ce qui suit.

Si une image codée redondante correspondant à l'image codée primaire est une image IDR, le contenu de la structure syntaxique `dec_ref_pic_marking()` doit être identique dans tous les en-têtes de tranche de l'image codée primaire et les images codées redondantes correspondantes à l'image codée primaire.

Autrement (une image redondante correspondant à l'image codée primaire n'est pas une image IDR), tous les en-têtes de tranche de l'image redondante doivent contenir une structure syntaxique `dec_ref_pic_marking()` incluant un élément syntaxique `memory_management_control_operation` égal à 5 et on applique ce qui suit.

Si la valeur de `long_term_reference_flag` dans l'image codée primaire est égale à 0, la structure syntaxique `dec_ref_pic_marking` de l'image codée redondante ne doit pas inclure d'élément syntaxique `memory_management_control_operation` égal à 6.

Autrement (la valeur de `long_term_reference_flag` dans l'image codée primaire est égale à 1), la structure syntaxique `dec_ref_pic_marking` de l'image codée redondante doit inclure des éléments syntaxiques `memory_management_control_operation` égaux à 5, 4 et 6 en ordre de décodage et la valeur de `max_long_term_trame_idx_plus1` doit être égale à 1 et la valeur de `long_term_trame_idx` doit être égale à 0.

Les valeurs de `TopFieldOrderCnt` et `BottomFieldOrderCnt` (s'ils sont applicables) qui résultent après achèvement du processus de décodage pour toute image codée redondante ou l'image codée primaire de la même unité d'accès doivent être identiques sans considérer si l'image codée primaire ou toute image codée redondante (à la place de l'image codée primaire) de l'unité d'accès serait décodée.

Aucun processus de décodage n'est requis pour une tranche codée ou subdivision de données de tranche codée d'une image codée redondante. Lorsque le `redundant_pic_cnt` dans l'en-tête de tranche d'une tranche codée est supérieur à 0, le décodeur peut mettre à l'écart la tranche codée. Cependant, une tranche codée ou subdivision de données de tranche codée de toute image codée redondante doit respecter les mêmes contraintes qu'une tranche codée ou subdivision de données de tranche codée d'une image primaire.

NOTE – Lorsque quelques-uns des échantillons dans l'image primaire décodée ne peuvent pas être correctement décodés à cause d'erreurs ou pertes dans la transmission de la séquence et qu'une tranche redondante codée peut être correctement décodée, le décodeur devrait remplacer les échantillons de l'image primaire décodée par les échantillons correspondants de la tranche redondante décodée. Lorsque plus d'une tranche redondante couvre la région considérée de l'image primaire, la tranche redondante ayant la plus faible valeur de `redundant_pic_cnt` devrait être utilisée.

Les tranches redondantes et les subdivisions de données de tranche ayant la même valeur de `redundant_pic_cnt` appartiennent à la même image redondante. Les tranches décodées au sein de la même image redondante ne doivent pas nécessairement couvrir l'image entière et ne doivent pas se chevaucher.

`direct_spatial_mv_pred_flag` spécifie comme suit la méthode utilisée dans le processus de décodage pour déduire les vecteurs cinétiques et les indices de référence pour l'interprédiction.

- Si `direct_spatial_mv_pred_flag` est égal à 1, le processus de déduction pour les vecteurs cinétiques luma pour `B_Skip`, `B_Direct_16x16` et `B_Direct_8x8` décrit au § 8.4.1.2 doit utiliser la prédiction spatiale de mode direct comme spécifié au § 8.4.1.2.2.
- Autrement (`direct_spatial_mv_pred_flag` est égal à 0), le processus de déduction pour les vecteurs cinétiques luma pour `B_Skip`, `B_Direct_16x16` et `B_Direct_8x8` décrit au § 8.4.1.2 doit utiliser la prédiction temporelle de mode direct comme spécifié au § 8.4.1.2.3.

num_ref_idx_active_override_flag égal à 0 spécifie que les valeurs des éléments syntaxiques `num_ref_idx_l0_active_minus1` et `num_ref_idx_l1_active_minus1` spécifiés dans l'ensemble de paramètres d'image référencé sont effectifs. `num_ref_idx_active_override_flag` égal à 1 spécifie que `num_ref_idx_l0_active_minus1` et `num_ref_idx_l1_active_minus1` spécifiés dans l'ensemble de paramètres d'image pris en référence sont supplantés pour la tranche en cours (et seulement pour la tranche en cours) par les valeurs suivantes dans l'en-tête de tranche.

Lorsque la tranche en cours est une tranche P, SP ou B et que `field_pic_flag` est égal à 0 et que la valeur de `num_ref_idx_l0_active_minus1` dans l'ensemble de paramètres d'image excède 15, `num_ref_idx_active_override_flag` doit être égal à 1.

Lorsque la tranche en cours est une tranche B et que `field_pic_flag` est égal à 0 et que la valeur de `num_ref_idx_l1_active_minus1` dans l'ensemble de paramètres d'image excède 15, `num_ref_idx_active_override_flag` doit être égal à 1.

num_ref_idx_l0_active_minus1 spécifie l'indice de référence maximal pour la liste 0 d'images de référence qui doit être utilisée pour décoder la tranche.

La gamme de `num_ref_idx_l0_active_minus1` est spécifiée comme suit.

- Si `field_pic_flag` est égal à 0, `num_ref_idx_l0_active_minus1` doit être dans la gamme de 0 à 15 inclus. Lorsque `MbaffFrameFlag` est égal à 1, `num_ref_idx_l0_active_minus1` est la valeur maximale d'indice pour le décodage de macroblocs de trame et $2 * \text{num_ref_idx_l0_active_minus1} + 1$ est la valeur maximale d'indice pour le décodage de macroblocs de champ.
- Autrement (`field_pic_flag` est égal à 1), `num_ref_idx_l0_active_minus1` doit être dans la gamme de 0 à 31 inclus.

num_ref_idx_l1_active_minus1 a la même sémantique que `num_ref_idx_l0_active_minus1` avec l0 et liste 0 remplacés respectivement par l1 et liste 1.

cabac_init_idc spécifie l'indice pour déterminer le tableau d'initialisation utilisé dans le processus d'initialisation pour les variables de contexte. La valeur de `cabac_init_idc` doit être dans la gamme de 0 à 2 inclus.

slice_qp_delta spécifie la valeur initiale de QP_Y à utiliser pour tous les macroblocs dans la tranche jusqu'à sa modification par la valeur de `mb_qp_delta` dans la couche de macrobloc. Le paramètre initial de quantification QP_Y pour la tranche est calculé comme suit:

$$\text{SliceQP}_Y = 26 + \text{pic_init_qp_minus26} + \text{slice_qp_delta} \quad (7-16)$$

La valeur de `slice_qp_delta` doit être limitée de telle sorte que QP_Y soit dans la gamme de 0 à 51 inclus.

sp_for_switch_flag spécifie le processus de décodage à utiliser pour décoder comme suit des macroblocs P dans une tranche SP.

- Si `sp_for_switch_flag` est égal à 0, les macroblocs P dans la tranche SP doivent être décodés en utilisant le processus de décodage SP pour les images non commutantes, comme spécifié au § 8.6.1.

- Autrement (`sp_for_switch_flag` est égal à 1), les macroblocs P dans la tranche SP doivent être décodés en utilisant les processus de décodage SP et SI pour les images commutantes, comme spécifié au § 8.6.2.

slice_qs_delta spécifie la valeur de QS_Y pour tous les macroblocs dans les tranches SP and SI. Le paramètre de quantification QS_Y pour la tranche est calculé comme suit:

$$QS_Y = 26 + pic_init_qs_minus26 + slice_qs_delta \quad (7-17)$$

La valeur de `slice_qs_delta` doit être limitée de telle sorte que QS_Y soit dans la gamme de 0 à 51 inclus. Cette valeur de QS_Y est utilisée pour le décodage de tous les macroblocs dans les tranches SI avec `mb_type` égal à SI et tous les macroblocs dans les tranches SP avec le mode de prédiction égal à inter.

disable_deblocking_filter_idc spécifie si le fonctionnement du filtre de démontage de bloc peut être désactivé sur certaines bordures de blocs de la tranche et spécifie pour quelles bordures le filtrage est désactivé. Lorsque `disable_deblocking_filter_idc` n'est pas présent dans l'en-tête de tranche, la valeur de `disable_deblocking_filter_idc` doit être supposée égale à 0.

La valeur de `disable_deblocking_filter_idc` doit être dans la gamme de 0 à 2 inclus.

slice_alpha_c0_offset_div2 spécifie le décalage utilisé en accédant aux tableaux α et t_{C0} de filtre de démontage de bloc pour les opérations de filtrage contrôlées par les macroblocs au sein de la tranche. A partir de cette valeur, le décalage qui doit être appliqué lors de l'utilisation de ces tableaux doit être calculé comme suit:

$$\text{FilterOffsetA} = slice_alpha_c0_offset_div2 \ll 1 \quad (7-18)$$

La valeur de `slice_alpha_c0_offset_div2` doit être dans la gamme de -6 à +6 inclus. Lorsque `slice_alpha_c0_offset_div2` n'est pas présent dans l'en-tête de tranche, la valeur de `slice_alpha_c0_offset_div2` doit être supposée égale à 0.

slice_beta_offset_div2 spécifie le décalage utilisé en accédant au tableau β de filtre de démontage de bloc pour les opérations de filtrage contrôlées par les macroblocs au sein de la tranche. A partir de cette valeur, le décalage qui est appliqué lors de l'utilisation du tableau β du filtre de démontage de bloc doit être calculé comme suit:

$$\text{FilterOffsetB} = slice_beta_offset_div2 \ll 1 \quad (7-19)$$

La valeur de `slice_beta_offset_div2` doit être dans la gamme de -6 à +6 inclus. Lorsque `slice_beta_offset_div2` n'est pas présent dans l'en-tête de tranche, la valeur de `slice_beta_offset_div2` doit être supposée égale à 0.

slice_group_change_cycle est utilisé pour déduire le nombre d'unités de mappage de groupes de tranche dans le groupe de tranches 0 lorsque `slice_group_map_type` est égal à 3, 4 ou 5, comme spécifié par:

$$\text{MapUnitsInSliceGroup0} = \text{Min}(slice_group_change_cycle * \text{SliceGroupChangeRate}, \text{PicSizeInMapUnits}) \quad (7-20)$$

La valeur de `slice_group_change_cycle` est représentée dans le flux binaire par le nombre de bits suivant:

$$\text{Ceil}(\text{Log2}(\text{PicSizeInMapUnits} \div \text{SliceGroupChangeRate} + 1)) \quad (7-21)$$

La valeur de `slice_group_change_cycle` doit être dans la gamme de 0 à $\text{Ceil}(\text{PicSizeInMapUnits} \div \text{SliceGroupChangeRate})$, inclus.

7.4.3.1 Sémantique de remise en ordre de liste d'images de référence

Les éléments syntaxiques `reordering_of_pic_nums_idc`, `abs_diff_pic_num_minus1` et `long_term_pic_num` spécifient le changement à utiliser pour décoder la tranche en passant des listes d'images de référence initiales aux listes d'images de référence.

ref_pic_list_reordering_flag_10 égal à 1 spécifie que la remise en ordre des éléments syntaxiques **reordering_of_pic_nums_idc** est présente pour spécifier la liste 0 d'images de référence. **ref_pic_list_reordering_flag_10** égal à 0 spécifie que cet élément syntaxique n'est pas présent.

Lorsque **ref_pic_list_reordering_flag_10** est égal à 1, le nombre de fois où **reordering_of_pic_nums_idc** n'est pas égal à 3 à la suite de **ref_pic_list_reordering_flag_10** ne doit pas excéder **num_ref_idx_10_active_minus1 + 1**.

Lorsque dans la liste initiale d'images de référence produite comme spécifié au § 8.2.4.2 **RefPicList0[num_ref_idx_10_active_minus1]** est égal à "pas d'image de référence", **ref_pic_list_reordering_flag_10** doit être égal à 1 et **reordering_of_pic_nums_idc** ne doit pas être égal à 3 jusqu'à ce que dans la liste remise en ordre produite comme spécifié au § 8.2.4.3, **RefPicList0[num_ref_idx_10_active_minus1]** ne soit pas égal à "pas d'image de référence".

ref_pic_list_reordering_flag_11 égal à 1 spécifie que l'élément syntaxique **reordering_of_pic_nums_idc** est présent pour spécifier la liste 1 d'images de référence. **ref_pic_list_reordering_flag_11** égal à 0 spécifie que cet élément syntaxique n'est pas présent.

Lorsque **ref_pic_list_reordering_flag_11** est égal à 1, le nombre de fois où **reordering_of_pic_nums_idc** n'est pas égal à 3 à la suite de **ref_pic_list_reordering_flag_11** ne doit pas excéder **num_ref_idx_11_active_minus1 + 1**.

Lorsqu'on décode une tranche B et que **RefPicList1[num_ref_idx_11_active_minus1]** dans la liste initiale d'images de référence produite comme spécifié au § 8.2.4.2 est égal à "pas d'image de référence", **ref_pic_list_reordering_flag_11** doit être égal à 1 et **reordering_of_pic_nums_idc** ne doit pas être égal à 3 jusqu'à ce que dans la liste remise en ordre produite comme spécifié au § 8.2.4.3, **RefPicList1[num_ref_idx_11_active_minus1]** ne soit pas égal à "pas d'image de référence".

reordering_of_pic_nums_idc, conjointement avec **abs_diff_pic_num_minus1** ou **long_term_pic_num**, spécifie laquelle des images de référence est remise en mappage. Les valeurs de **reordering_of_pic_nums_idc** sont spécifiées au Tableau 7-4. La valeur du premier **reordering_of_pic_nums_idc** qui suit immédiatement après **ref_pic_list_reordering_flag_10** ou **ref_pic_list_reordering_flag_11** ne doit pas être égale à 3.

Tableau 7-4 – Opérations de reordering_of_pic_nums_idc pour la remise en ordre des listes d'images de référence

reordering_of_pic_nums_idc	Remise en ordre spécifiée
0	abs_diff_pic_num_minus1 est présent et correspond à une différence à soustraire d'une valeur de prédiction du numéro d'image
1	abs_diff_pic_num_minus1 est présent et correspond à une différence à ajouter à une valeur de prédiction du numéro d'image
2	long_term_pic_num est présent et spécifie le numéro d'image à long terme pour une image de référence
3	Fin de boucle pour la remise en ordre de la liste initiale d'images de référence

abs_diff_pic_num_minus1 plus 1 spécifie la différence absolue entre le numéro d'image de l'image en cours de déplacement jusqu'à l'indice actuel dans la liste et la valeur de prédiction du numéro d'image. **abs_diff_pic_num_minus1** doit être compris entre 0 et **MaxPicNum - 1**. Les valeurs admises de **abs_diff_pic_num_minus1** sont de plus restreintes comme spécifié au § 8.2.4.3.1.

long_term_pic_num spécifie le numéro d'image à long terme de l'image en cours déplacée vers l'indice actuel dans la liste. Lors du décodage d'une trame codée, **long_term_pic_num** doit être égal

à un LongTermPicNum alloué à une des trames de référence ou paire de champs de référence complémentaires marquée "utilisé pour référence à long terme". Lors du décodage d'un champ codé, long_term_pic_num doit être égal à un LongTermPicNum alloué à un des champs de référence marqués comme "utilisé pour référence à long terme".

7.4.3.2 Sémantique de tableau de prédiction de pondération

luma_log2_weight_denom est le logarithme de base 2 du dénominateur pour tous les facteurs de pondération luma. La valeur de luma_log2_weight_denom doit être dans la gamme de 0 à 7 inclus.

chroma_log2_weight_denom est le logarithme de base 2 du dénominateur pour tous les facteurs de pondération chroma. La valeur de chroma_log2_weight_denom doit être dans la gamme de 0 à 7 inclus.

luma_weight_10_flag égal à 1 spécifie que les facteurs de pondération pour la prédiction de la liste 0 de composants luma sont présents. luma_weight_10_flag égal à 0 spécifie que ces facteurs de pondération ne sont pas présents.

luma_weight_10[i] est le facteur de pondération appliqué à la valeur de prédiction luma pour la prédiction de liste 0 en utilisant RefPicList0[i]. Lorsque luma_weight_10_flag est égal à 1, la valeur de luma_weight_10[i] doit être dans la gamme de -128 à 127 inclus. Lorsque luma_weight_10_flag est égal à 0, luma_weight_10[i] doit être supposé égal à $2^{\text{luma_log2_weight_denom}}$ pour RefPicList0[i].

luma_offset_10[i] est le décalage supplémentaire appliqué à la valeur de prédiction luma pour la prédiction de liste 0 en utilisant RefPicList0[i]. La valeur de luma_offset_10[i] doit être dans la gamme de -128 à 127 inclus. Lorsque luma_weight_10_flag est égal à 0, luma_offset_10[i] doit être supposé égal à 0 pour RefPicList0[i].

chroma_weight_10_flag égal à 1 spécifie que les facteurs de pondération pour les valeurs de prédiction chroma de prédiction de liste 0 sont présents. chroma_weight_10_flag égal à 0 spécifie que ces facteurs de pondération ne sont pas présents.

chroma_weight_10[i][j] est le facteur de pondération appliqué aux valeurs de prédiction chroma pour la prédiction de liste 0 en utilisant RefPicList0[i] avec j égal à 0 pour Cb et j égal à 1 pour Cr. Lorsque chroma_weight_10_flag est égal à 1, la valeur de chroma_weight_10[i][j] doit être dans la gamme de -128 à 127 inclus. Lorsque chroma_weight_10_flag est égal à 0, chroma_weight_10[i][j] doit être supposé égal à $2^{\text{chroma_log2_weight_denom}}$ pour RefPicList0[i].

chroma_offset_10[i][j] est le décalage supplémentaire appliqué aux valeurs de prédiction chroma pour la prédiction de liste 0 en utilisant RefPicList0[i] avec j égal à 0 pour Cb et j égal à 1 pour Cr. La valeur de chroma_offset_10[i][j] doit être dans la gamme de -128 à 127 inclus. Lorsque chroma_weight_10_flag est égal à 0, chroma_offset_10[i][j] doit être supposé égal à 0 pour RefPicList0[i].

luma_weight_11_flag, **luma_weight_11**, **luma_offset_11**, **chroma_weight_11_flag**, **chroma_weight_11**, **chroma_offset_11** ont la même sémantique que respectivement luma_weight_10_flag, luma_weight_10, luma_offset_10, chroma_weight_10_flag, chroma_weight_10, chroma_offset_10, avec l0, list 0 et List0 remplacés respectivement par 11, list 1 et List1.

7.4.3.3 Sémantique de marquage d'image de référence décodée

Les éléments syntaxiques 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 et max_long_term_frame_idx_plus1 spécifient le marquage des images de référence.

Le marquage d'une image de référence peut être "non utilisé comme référence", "utilisé pour référence de court terme" ou "utilisé pour référence de long terme", mais seulement un parmi ces trois. Lorsqu'une image de référence est référencée comme ayant le marquage "utilisé pour référence", cela se réfère collectivement à une image marquée comme "utilisé pour référence de court terme" ou "utilisé pour référence de long terme", mais pas les deux à la fois.

L'élément syntaxique `adaptive_ref_pic_marking_mode_flag` et le contenu de la structure syntaxique de marquage d'image de référence décodée doivent être identiques pour toutes les tranches codées d'une image codée.

La catégorie syntaxique de la structure syntaxique de marquage d'image de référence décodée doit être déduite comme suit:

- si la structure syntaxique de marquage d'image de référence décodée est dans un en-tête de tranche, la catégorie syntaxique de la structure syntaxique de marquage d'image de référence décodée doit être supposée égale à 2;
- autrement (la structure syntaxique de marquage d'image de référence décodée est dans un message SEI de répétition de marquage d'image de référence décodée comme spécifié à l'Annexe D), la catégorie syntaxique de la structure syntaxique de marquage d'image de référence décodée doit être supposée égale à 5.

`no_output_of_prior_pics_flag` spécifie comment sont traitées les images précédemment décodées dans la mémoire tampon d'image décodée après le décodage d'une image IDR. Voir l'Annexe C. Lorsque l'image IDR est la première image IDR dans le flux binaire, la valeur de `no_output_of_prior_pics_flag` n'a pas d'effet sur le processus de décodage. Lorsque l'image IDR n'est pas la première image IDR dans le flux binaire et que la valeur de `PicWidthInMbs`, `FrameHeightInMbs` ou `max_dec_frame_buffering`, déduite de l'ensemble actif de paramètres de séquence, est différente de la valeur de `PicWidthInMbs`, `FrameHeightInMbs` ou `max_dec_frame_buffering`, déduite de l'ensemble actif de paramètres de séquence pour la séquence précédente, `no_output_of_prior_pics_flag` peut être supposé égal à 1 par le décodeur, sans considération de la valeur réelle de `no_output_of_prior_pics_flag`.

`long_term_reference_flag` égal à 0 spécifie que la variable `MaxLongTermFrameIdx` est mise égale à "pas d'indices de trame à long terme" et que l'image IDR est marquée comme "utilisé pour référence à court terme". `long_term_reference_flag` égal à 1 spécifie que la variable `MaxLongTermFrameIdx` est mise égale à 0 et que l'image IDR en cours est marquée "utilisé pour référence à long terme" et que `LongTermFrameIdx` est alloué égal à 0. Lorsque `num_ref_frames` est égal à 0, `long_term_reference_flag` doit être égal à 0.

`adaptive_ref_pic_marking_mode_flag` choisit le mode de marquage d'image de référence de l'image décodée en cours comme spécifié au Tableau 7-5. `adaptive_ref_pic_marking_mode_flag` doit être égal à 1 lorsque le nombre des trames, des paires de champs complémentaires et des champs non appariés qui sont actuellement marqués comme "utilisé pour référence à long terme" est égal à $\text{Max}(\text{num_ref_frames}, 1)$.

Tableau 7-5 – Interprétation de `adaptive_ref_pic_marking_mode_flag`

<code>adaptive_ref_pic_marking_mode_flag</code>	Mode de marquage d'image de référence spécifié
0	Mode de marquage d'image de référence par fenêtre glissante: mode de marquage donnant un mécanisme de premier entré, premier sorti pour les images de référence de court terme
1	Mode de marquage d'image de référence adaptatif: mode de marquage d'image de référence fournissant des éléments syntaxiques pour spécifier le marquage des images de référence comme "non utilisé comme référence" et pour allouer des indices de trame à long terme

memory_management_control_operation spécifie une opération de contrôle qui doit être appliquée pour gérer le marquage d'image de référence. L'élément syntaxique `memory_management_control_operation` est suivi des données nécessaires pour l'opération spécifiée par la valeur de `memory_management_control_operation`. Les valeurs et opérations de contrôle associées à `memory_management_control_operation` sont spécifiées au Tableau 7-6. Les éléments syntaxiques `memory_management_control_operation` sont traités par le processus de décodage dans l'ordre dans lequel ils figurent dans l'en-tête de tranche, et les contraintes sémantiques exprimées pour chaque `memory_management_control_operation` s'appliquent à l'emplacement précis, dans cet ordre, où ce `memory_management_control_operation` est traité.

`memory_management_control_operation` ne doit pas être égal à 1 dans un en-tête de tranche à moins que l'image de court terme spécifiée ne soit actuellement marquée comme "utilisé pour référence" lorsque le `memory_management_control_operation` est traité par le processus de décodage et que l'image de court terme spécifiée n'ait pas été allouée à un indice de trame de long terme et ne soit pas allouée à un indice de trame de long terme dans la même structure syntaxique de marquage d'image de référence décodée.

`memory_management_control_operation` ne doit pas être égal à 2 dans un en-tête de tranche à moins que le numéro de l'image de long terme spécifiée ne se réfère à une trame ou champ actuellement marqué comme "utilisé pour référence" lorsque le `memory_management_control_operation` est traité par le processus de décodage.

`memory_management_control_operation` ne doit pas être égal à 3 dans un en-tête de tranche à moins que l'image de référence de court terme spécifiée ne soit actuellement marquée comme "utilisé pour référence" lorsque `memory_management_control_operation` est traité par le processus de décodage et que l'image de référence de court terme spécifiée n'ait pas eu précédemment d'indice de trame de long terme alloué et n'ait pas eu d'autre indice de trame de long terme alloué dans la même structure syntaxique de marquage d'image de référence décodée.

`memory_management_control_operation` ne doit pas être égal à 3 ou 6 lorsque la valeur de la variable `MaxLongTermFrameIdx` est égale à "pas d'indices de trames de long terme" lorsque le `memory_management_control_operation` est traité par le processus de décodage.

Il ne doit pas y avoir plus d'un `memory_management_control_operation` égal à 4 présent dans un en-tête de tranche.

`memory_management_control_operation` ne doit pas être égal à 5 dans un en-tête de tranche à moins qu'aucun `memory_management_control_operation` dans la gamme de 1 à 3 ne soit présent dans la même structure syntaxique de marquage d'image de référence décodée.

Il ne doit pas y avoir plus d'un `memory_management_control_operation` égal à 6 présent dans un en-tête de tranche.

Lorsqu'un `memory_management_control_operation` égal à 6 est présent, tout `memory_management_control_operation` égal à 2, 3 ou 4 qui suit le `memory_management_control_operation` égal à 6 dans le même en-tête de tranche ne doit pas spécifier pour l'image en cours la valeur de marquage "non utilisé comme référence".

Un `memory_management_control_operation` égal à 6 ne doit pas précéder un `memory_management_control_operation` égal à 5 dans le même en-tête de tranche.

NOTE – Ces contraintes interdisent toute combinaison de plusieurs éléments syntaxiques `memory_management_control_operation` qui spécifieraient pour l'image en cours la valeur de marquage "non utilisé comme référence". Toutefois, d'autres combinaisons d'éléments syntaxiques `memory_management_control_operation` sont autorisées, qui peuvent affecter l'état de marquage d'autres images de référence plusieurs fois dans le même en-tête de tranche. En particulier, un `memory_management_control_operation` égal à 3 qui spécifie qu'un indice de trame de long terme soit alloué à une image de référence de court terme donnée peut être suivi dans le même en-tête de tranche d'un `memory_management_control_operation` égal à 2, 3 ou 4 qui spécifie que la même image de référence soit ultérieurement marquée comme "non utilisé comme référence".

Il ne doit pas y avoir présent dans un en-tête de tranche plus d'un `memory_management_control_operation` qui spécifie la même action à entreprendre.

**Tableau 7-6 – Valeurs d'opération de commande de gestion de mémoire
(`memory_management_control_operation`)**

<code>memory_management_control_operation</code>	Opération de commande de gestion de mémoire
0	Fin de boucle de <code>memory_management_control_operation</code>
1	Marque une image à court terme comme "non utilisé comme référence"
2	Marque une trame ou champ ayant un numéro d'image à long terme comme "non utilisé comme référence"
3	Alloue un indice de trame de long terme à une image de court terme
4	Spécifie l'indice maximal de trame de long terme
5	Marque toutes les images de référence comme "non utilisé comme référence" et met la variable <code>MaxLongTermFrameIdx</code> à "pas d'indices de trame de long terme"
6	Alloue un indice de trame de long terme à l'image décodée en cours

Lorsqu'on décode un champ et qu'une commande `memory_management_control_operation` égale à 3 alloue un indice de trame de long terme à un champ qui fait partie d'une trame de référence de court terme ou d'une paire de champs de référence de court terme complémentaires, une autre commande `memory_management_control_operation` doit être présente dans la même structure syntaxique de marquage d'image de référence décodée pour allouer le même indice de trame de long terme à l'autre champ de la même trame ou paire de champs de référence complémentaires.

Lorsque le premier champ (dans l'ordre de décodage) d'une paire de champs de référence complémentaires inclut un `long_term_reference_flag` égal à 1 ou une commande `memory_management_control_operation` égale à 6, la structure syntaxique de marquage d'image de référence décodée pour l'autre champ de la paire de champs de référence complémentaires doit contenir une commande `memory_management_control_operation` égale à 6 qui alloue le même indice de trame de long terme à l'autre champ.

difference_of_pic_nums_minus1 est utilisé (avec `memory_management_control_operation` égal à 3 ou 1) pour allouer un indice de trame de long terme à une image de référence de court terme ou pour marquer une image de référence de court terme comme "non utilisé comme référence". Lorsque le `memory_management_control_operation` associé est traité par le processus de décodage, le numéro d'image résultant déduit de `difference_of_pic_nums_minus1` doit être un numéro d'image alloué à l'une des images de référence marquées comme "utilisé pour référence" et non alloué précédemment à un indice de trame de long terme.

Le numéro d'image résultant est soumis aux contraintes suivantes.

- Si `field_pic_flag` est égal à 0, le numéro d'image résultant doit être inclus dans l'ensemble des numéros d'image alloués aux trames de référence ou paires de champs de référence complémentaires.

NOTE – Lorsque `field_pic_flag` est égal à 0, le numéro d'image résultant doit être un numéro d'image alloué à une paire de champs de référence complémentaires dont les deux champs sont marqués comme "utilisé pour référence" ou à une trame dont les deux champs sont marqués comme "utilisé pour référence". En particulier, lorsque `field_pic_flag` est égal à 0, le marquage d'un champ ou d'une trame non apparié(e) dont un seul champ est marqué comme "utilisé pour référence" ne peut être affecté par un `memory_management_control_operation` égal à 1.

- Autrement (`field_pic_flag` est égal à 1), le numéro d'image résultant doit être dans l'ensemble des numéros d'image alloués aux champs de référence.

long_term_pic_num est utilisé (avec `memory_management_control_operation` égal à 2) pour marquer une image de référence à long terme comme "non utilisé comme référence". Lorsque le `memory_management_control_operation` est traité par le processus de décodage, `long_term_pic_num` doit être égal à un numéro d'image à long terme alloué à une des images de référence qui est actuellement marquée comme "utilisé pour référence à long terme".

Le numéro d'image à long terme résultant est soumis aux contraintes suivantes.

- Si `field_pic_flag` est égal à 0, le numéro d'image à long terme résultant doit être inclus dans l'ensemble de numéros d'image à long terme alloués aux trames de références ou paires de champs de référence complémentaires.

NOTE – Lorsque `field_pic_flag` est égal à 0, le numéro d'image à long terme résultant doit être un numéro d'image à long terme alloué à une paire de champs de référence complémentaires dont les deux champs sont marqués comme "utilisé pour référence" ou à une trame dont les deux champs sont marqués comme "utilisé pour référence". En particulier, lorsque `field_pic_flag` est égal à 0, le marquage d'un champ ou d'une trame non apparié(e) dont un seul champ est marqué comme "utilisé pour référence" ne peut être affecté par un `memory_management_control_operation` égal à 2.

- Autrement (`field_pic_flag` est égal à 1), le numéro d'image à long terme résultant doit être inclus dans l'ensemble de numéros d'image à long terme alloués aux champs de référence.

long_term_frame_idx est utilisé (avec `memory_management_control_operation` égal à 3 ou 6) pour allouer un indice de trame de long terme à une image. Lorsque le `memory_management_control_operation` est traité par le processus de décodage, la valeur de `long_term_frame_idx` doit être dans la gamme de 0 à `MaxLongTermFrameIdx`, inclus.

max_long_term_frame_idx_plus1 moins 1 spécifie la valeur maximal de l'indice de trame de long terme admise pour les images de références de long terme (jusqu'à réception d'une autre valeur de `max_long_term_frame_idx_plus1`). La valeur de `max_long_term_frame_idx_plus1` doit être dans la gamme de 0 à `num_ref_frames` inclus.

7.4.4 Sémantique de données de tranche

cabac_alignment_one_bit est un bit égal à 1.

mb_skip_run spécifie le nombre de macroblocs consécutifs sautés pour lesquels, lorsqu'on décode une tranche P ou SP, on doit supposer que le **mb_type** est **P_Skip** et le type de macrobloc est désigné collectivement comme type de macrobloc P ou pour lesquels, lorsqu'on décode une tranche B, on doit supposer que le **mb_type** est **B_Skip** et le type de macrobloc est désigné collectivement comme type de macrobloc B. La valeur de **mb_skip_run** doit être dans la gamme de 0 à **PicSizeInMbs – CurrMbAddr** inclus.

mb_skip_flag égal à 1 spécifie que pour le macrobloc en cours, lorsqu'on décode une tranche P ou SP, on doit supposer que le **mb_type** est **P_Skip** et que le type de macrobloc est désigné collectivement comme type de macrobloc P ou que pour lequel, lorsqu'on décode une tranche B, on doit supposer que le **mb_type** est **B_Skip** et que le type de macrobloc est désigné collectivement comme type de macrobloc B. **mb_skip_flag** égal à 0 spécifie que le macrobloc en cours n'est pas sauté.

mb_field_decoding_flag égal à 0 spécifie que la paire de macroblocs en cours est une paire de macroblocs de trame. **mb_field_decoding_flag** égal à 1 spécifie que la paire de macroblocs est une paire de macroblocs de champ. Les deux macroblocs d'une paire de macroblocs de trame sont désignés dans le texte comme macroblocs de trame, tandis que les deux macroblocs d'une paire de macroblocs de champ sont désignés dans le texte comme macroblocs de champ.

Lorsque **mb_field_decoding_flag** n'est pas présent pour l'un ou l'autre macrobloc d'une paire de macroblocs, la valeur de **mb_field_decoding_flag** est déduite comme suit.

- S'il y a une paire de macroblocs dans le voisinage immédiatement à gauche de la paire de macroblocs en cours dans la même tranche, la valeur de **mb_field_decoding_flag** doit être supposée égale à la valeur de **mb_field_decoding_flag** pour la paire de macroblocs du voisinage immédiatement à gauche de la paire de macroblocs en cours,
- Autrement, s'il n'y a pas de paire de macroblocs dans le voisinage immédiatement à gauche de la paire de macroblocs en cours dans la même tranche et s'il y a une paire de macroblocs dans le voisinage immédiatement au-dessus de la paire de macroblocs en cours dans la même tranche, la valeur de **mb_field_decoding_flag** doit être supposée égale à la valeur de **mb_field_decoding_flag** pour la paire de macroblocs dans le voisinage immédiatement au-dessus de la paire de macroblocs en cours,
- Autrement (il n'y a pas de paire de macroblocs dans le voisinage ni immédiatement à gauche ni immédiatement au-dessus de la paire de macroblocs en cours dans la même tranche), la valeur de **mb_field_decoding_flag** doit être supposée égale à 0.

end_of_slice_flag égal à 0 spécifie qu'un autre macrobloc suit dans la tranche. **end_of_slice_flag** égal à 1 spécifie la fin de la tranche et qu'aucun autre macrobloc ne suit.

La fonction **NextMbAddress()** utilisée dans le tableau de syntaxe des données de tranche est spécifiée au § 8.2.2.

7.4.5 Sémantique de couche de macrobloc

mb_type spécifie le type de macrobloc. La sémantique de **mb_type** dépend du type de tranche.

Les tableaux et la sémantique sont spécifiés pour les divers types de macrobloc pour les tranches I, SI, P, SP et B. Chaque tableau présente la valeur de **mb_type**, le nom de **mb_type**, le nombre de subdivisions de macrobloc utilisées (donné par la fonction **NumMbPart(mb_type)**), le mode de prédiction du macrobloc (lorsqu'il n'est pas subdivisé) ou la première subdivision (donnée par la fonction **MbPartPredMode(mb_type, 0)**) et le mode de prédiction de la seconde subdivision (donnée par la fonction **MbPartPredMode(mb_type, 1)**). Lorsqu'une valeur n'est pas applicable, elle est mentionnée par "na". Dans le texte, la valeur de **mb_type** peut être mentionnée comme le type de macrobloc et une valeur X de **MbPartPredMode()** peut être mentionnée dans le texte par "mode de prévision de macrobloc (subdivision) X" ou comme "macroblocs de prédiction X".

Le Tableau 7-7 montre les types collectifs de macrobloc admis pour slice_type.

NOTE – Quelques types de macrobloc avec le mode de prédiction Pred_L0 sont classés dans le type de macrobloc B.

Tableau 7-7 – Types collectifs de macrobloc admis pour slice_type

slice_type	Types collectifs de macrobloc admis
I (tranche)	I (voir le Tableau 7-8) (types de macrobloc)
P (tranche)	P (voir le Tableau 7-10) et I (voir le Tableau 7-8) (types de macrobloc)
B (tranche)	B (voir le Tableau 7-11) et I (voir le Tableau 7-8) (types de macrobloc)
SI (tranche)	SI (voir le Tableau 7-9) et I (voir le Tableau 7-8) (types de macrobloc)
SP (tranche)	P (voir le Tableau 7-10) et I (voir le Tableau 7-8) (types de macrobloc)

Les types de macrobloc auxquels on peut se référer collectivement sous le nom de types de macrobloc I sont spécifiés au Tableau 7-8.

Les types de macrobloc pour les tranches I sont tous des types de macrobloc I.

Tableau 7-8 – Types de macrobloc pour tranches I

mb_type	Nom de mb_type	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	I_4x4	Intra_4x4	na	Equation 7-22	Equation 7-22
1	I_16x16_0_0_0	Intra_16x16	0	0	0
2	I_16x16_1_0_0	Intra_16x16	1	0	0
3	I_16x16_2_0_0	Intra_16x16	2	0	0
4	I_16x16_3_0_0	Intra_16x16	3	0	0
5	I_16x16_0_1_0	Intra_16x16	0	1	0
6	I_16x16_1_1_0	Intra_16x16	1	1	0
7	I_16x16_2_1_0	Intra_16x16	2	1	0
8	I_16x16_3_1_0	Intra_16x16	3	1	0
9	I_16x16_0_2_0	Intra_16x16	0	2	0
10	I_16x16_1_2_0	Intra_16x16	1	2	0
11	I_16x16_2_2_0	Intra_16x16	2	2	0
12	I_16x16_3_2_0	Intra_16x16	3	2	0
13	I_16x16_0_0_1	Intra_16x16	0	0	15
14	I_16x16_1_0_1	Intra_16x16	1	0	15
15	I_16x16_2_0_1	Intra_16x16	2	0	15
16	I_16x16_3_0_1	Intra_16x16	3	0	15
17	I_16x16_0_1_1	Intra_16x16	0	1	15
18	I_16x16_1_1_1	Intra_16x16	1	1	15
19	I_16x16_2_1_1	Intra_16x16	2	1	15
20	I_16x16_3_1_1	Intra_16x16	3	1	15
21	I_16x16_0_2_1	Intra_16x16	0	2	15
22	I_16x16_1_2_1	Intra_16x16	1	2	15
23	I_16x16_2_2_1	Intra_16x16	2	2	15
24	I_16x16_3_2_1	Intra_16x16	3	2	15
25	I_PCM	na	na	na	na

La sémantique suivante est allouée aux types de macrobloc dans le Tableau 7-8:

I_4x4: le macrobloc est codé comme un macrobloc de prédiction Intra_4x4.

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: le macrobloc est codé comme un macrobloc de mode de prédiction Intra_16x16.

A chaque macrobloc de prédiction Intra_16x16 est alloué un Intra16x16PredMode, qui spécifie le mode de prédiction Intra_16x16. CodedBlockPatternChroma contient la valeur du schéma de bloc codé pour chroma comme spécifié au Tableau 7-12. CodedBlockPatternLuma spécifie si des niveaux de coefficient de transformée AC différents de zéro sont présents pour le composant luma. CodedBlockPatternLuma égal à 0 spécifie que tous les niveaux de coefficient de transformée AC dans le composant luma du macrobloc sont égaux à 0. CodedBlockPatternLuma égal à 15 spécifie qu'au moins un des niveaux de coefficient de transformée AC figurant dans le composant luma du macrobloc est différent de 0, requérant un balayage des niveaux de coefficient de transformée AC pour tous les 16 des blocs 4x4 dans le bloc 16x16.

Intra_4x4 spécifie le mode de prédiction de macrobloc et spécifie que le processus de prédiction Intra_4x4 est invoqué comme spécifié au § 8.3.1. Intra_4x4 est un mode de prédiction intramacrobloc.

Intra_16x16 spécifie le mode de prédiction de macrobloc et spécifie que le processus de prédiction Intra_16x16 est invoqué comme spécifié au § 8.3.2. Intra_16x16 est un mode de prédiction intramacrobloc.

Pour un macrobloc codé avec mb_type égal à I_PCM, on doit supposer le mode de prédiction intramacrobloc.

Un type de macrobloc auquel on peut se référer sous le nom de type de macrobloc SI est spécifié au Tableau 7-9.

Les types de macrobloc pour les tranches SI sont spécifiés au Tableau 7-9 et au Tableau 7-8. La valeur 0 de mb_type est spécifiée au Tableau 7-9 et les valeurs 1 à 26 de mb_type sont spécifiées au Tableau 7-8, indexées en soustrayant 1 de la valeur de mb_type.

Tableau 7-9 – Type de macrobloc avec la valeur 0 pour les tranches SI

mb_type	Nom de mb_type	MbPartPredMode (mb_type, 0)	Intra16x16PredMode	CodedBlockPatternChroma	CodedBlockPatternLuma
0	SI	Intra_4x4	na	Equation 7-22	Equation 7-22

La sémantique suivante est allouée aux types de macrobloc dans le Tableau 7-9. Le macrobloc SI est codé comme macrobloc de prédiction Intra_4x4.

Les types de macrobloc qu'on peut mentionner collectivement comme types de macrobloc P sont spécifiés au Tableau 7-10.

Les types de macrobloc pour les tranches P et SP sont spécifiés au Tableau 7-10 et au Tableau 7-8. Les valeurs de mb_type 0 à 4 sont spécifiées au Tableau 7-10 et les valeurs de mb_type de 5 à 30 sont spécifiées au Tableau 7-8, indexées en soustrayant 5 de la valeur de mb_type.

Tableau 7-10 – Valeurs de 0 à 4 de type de macrobloc pour les tranches P et SP

mb_type	Nom de mb_type	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
Supposée	P_Skip	1	Pred_L0	na	16	16

La sémantique suivante est allouée aux types de macrobloc dans le Tableau 7-10.

- P_L0_16x16: les échantillons du macrobloc sont prédits avec une subdivision de macrobloc luma d'une taille de 16x16 échantillons luma et les échantillons chroma associés.
- P_L0_L0_MxN, avec MxN remplacé par 16x8 ou 8x16: les échantillons du macrobloc sont prédits en utilisant deux subdivisions luma d'une taille de MxN égale à 16x8 ou deux subdivisions luma d'une taille de MxN égale à 8x16 et les échantillons chroma associés, respectivement.
- P_8x8: pour chaque sous-macrobloc est présent un élément syntaxique (sub_mb_type) additionnel dans le flux binaire qui spécifie le type du sous-macrobloc correspondant (voir le § 7.4.5.2).
- P_8x8ref0: a la même sémantique que P_8x8 mais il n'y a pas d'élément syntaxique présent pour l'indice de référence (ref_idx_10) dans le flux binaire et ref_idx_10[mbPartIdx] doit être supposé égal à 0 pour tous les sous-macroblobs du macrobloc (avec des indices mbPartIdx égaux à 0..3).
- P_Skip: pas d'autres données présentes pour le macrobloc dans le flux binaire.

La sémantique suivante est allouée au mode de prédiction de macroblobs (MbPartPredMode()) dans le Tableau 7-10.

- Pred_L0: spécifie que le processus d'interprédiction est invoqué en utilisant la prédiction de liste 0. Pred_L0 est un mode de prédiction intermacrobloc.

Les types de macrobloc qu'on peut mentionner collectivement comme types de macrobloc B sont spécifiés au Tableau 7-11.

Les types de macrobloc pour les tranches B sont spécifiés au Tableau 7-11 et au Tableau 7-8. Les valeurs de mb_type de 0 à 22 sont spécifiées au Tableau 7-11 et les valeurs de mb_type de 23 à 48 sont spécifiées au Tableau 7-8, indexées en soustrayant 23 de la valeur de mb_type.

Tableau 7-11 – Valeurs de type de macrobloc de 0 à 22 pour tranches B

mb_type	Nom de mb_type	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
Supposée	B_Skip	na	Direct	na	8	8

La sémantique suivante est allouée aux types de macrobloc dans le Tableau 7-11:

- B_Direct_16x16: il n'y a pas de différence de vecteur cinétique ou d'indice de référence présent pour le macrobloc dans le flux binaire. Les fonctions MbPartWidth(B_Direct_16x16) et MbPartHeight(B_Direct_16x16) sont utilisées dans le processus de déduction pour les indices des vecteurs cinétiques et de trame de référence au § 8.4.1 pour la prédiction de mode direct.
- B_X_16x16 avec X remplacé par L0, L1 ou Bi: les échantillons du macrobloc sont prédits avec une subdivision de macrobloc luma d'une taille d'échantillons luma 16x16 et les échantillons chroma associés. Pour un macrobloc du type B_X_16x16 avec X remplacé par L0 ou L1, une différence de vecteur cinétique et un indice de référence sont présents dans le flux binaire pour le macrobloc. Pour un macrobloc de type B_X_16x16 avec X remplacé

par B_i , deux différences de vecteur cinétique et deux indices de référence sont présents dans le flux binaire pour le macrobloc.

- $B_{X0_X1_MxN}$, avec $X0$, $X1$ se rapportant à la première et la seconde subdivision de macrobloc et remplacés par $L0$, $L1$ ou B_i et MxN remplacé par $16x8$ ou $8x16$: les échantillons du macrobloc sont prédits en utilisant respectivement deux subdivisions luma d'une taille MxN égale à $16x8$ ou deux subdivisions luma d'une taille de MxN égale à $8x16$ et les échantillons chroma associés. Pour une subdivision de macrobloc $X0$ ou $X1$ avec $X0$ ou $X1$ remplacés par $L0$ ou $L1$, une différence de vecteur cinétique et un indice de référence sont présents dans le flux binaire. Pour une subdivision de macrobloc $X0$ ou $X1$ avec $X0$ ou $X1$ remplacé par B_i , deux différences de vecteur cinétique et deux indices de référence sont présents dans le flux binaire pour la subdivision de macrobloc.
- B_{8x8} : pour chaque sous-macrobloc est présent un élément syntaxique (`sub_mb_type`) additionnel envoyé dans le flux binaire, qui spécifie le type du sous-macrobloc correspondant (voir le § 7.4.5.2).
- B_{Skip} : il n'y a pas d'autres données présentes pour le macrobloc dans le flux binaire. Les fonctions `MbPartWidth(B_Skip)` et `MbPartHeight(B_Skip)` sont utilisées dans le processus de déduction pour les vecteurs cinétiques et les indices de trame de référence au § 8.4.1 pour la prédiction de mode direct.

La sémantique suivante est allouée aux modes de prédiction de macrobloc (`MbPartPredMode()`) au Tableau 7-11.

- Direct: il n'y a pas de différence de vecteur cinétique ou d'indice de référence présent pour le macrobloc (dans le cas de B_{Skip} ou B_{Direct_16x16}) dans le flux binaire. Direct est un mode d'interprédiction de macrobloc.
- $Pred_L0$: voir la sémantique pour le Tableau 7-10.
- $Pred_L1$: spécifie que le processus d'interprédiction est invoqué en utilisant la prédiction de liste 1. $Pred_L1$ est un mode d'interprédiction de macrobloc.
- $BiPred$: spécifie que le processus d'interprédiction est invoqué en utilisant la prédiction de liste 0 et de liste 1. $BiPred$ est un mode d'interprédiction de macrobloc.

pcm_alignment_zero_bit est un bit égal à 0.

pcm_byte[i] est une valeur d'échantillon. `pcm_byte[i]` ne doit pas être égal à 0. Les 256 premières valeurs de `pcm_byte[i]` représentent les valeurs d'échantillon luma dans le balayage de trame au sein du macrobloc. Les $(256 * (ChromaFormatFactor - 1)) / 2$ `pcm_byte[i]` valeurs suivantes représentent les valeurs d'échantillon Cb dans le balayage de trame au sein du macrobloc. Les $(256 * (ChromaFormatFactor - 1)) / 2$ `pcm_byte[i]` dernières valeurs représentent les valeurs d'échantillon Cr dans le balayage de trame au sein du macrobloc.

coded_block_pattern spécifie lequel des six blocs $8x8$ – luma et chroma – peut contenir des niveaux de coefficient de transformée différents de zéro. Pour les macroblocs dont le mode de prédiction n'est pas égal à `Intra_16x16`, **coded_block_pattern** est présent dans le flux binaire et les variables `CodedBlockPatternLuma` et `CodedBlockPatternChroma` sont déduites comme suit.

$$\begin{aligned} \text{CodedBlockPatternLuma} &= \text{coded_block_pattern} \% 16 \\ \text{CodedBlockPatternChroma} &= \text{coded_block_pattern} / 16 \end{aligned} \quad (7-22)$$

Lorsque `coded_block_pattern` est présent, `CodedBlockPatternLuma` spécifie, pour chacun des quatre blocs luma $8x8$ du macrobloc, un des cas suivants:

- Tous les niveaux du coefficient de transformée des quatre blocs luma $4x4$ dans le bloc luma $8x8$ sont égaux à zéro;
- Un ou plusieurs niveaux de coefficient de transformée d'un ou de plusieurs des blocs luma $4x4$ dans le bloc luma $8x8$ doivent avoir une valeur différente de zéro.

La signification de CodedBlockPatternChroma est spécifiée au Tableau 7-12.

Tableau 7-12 – Spécification des valeurs de CodedBlockPatternChroma

CodedBlockPatternChroma	Description
0	Tous les niveaux de coefficient de transformée chroma sont égaux à 0
1	Un ou plusieurs niveaux de coefficient de transformée chroma DC doivent avoir une valeur différente de zéro. Tous les niveaux de coefficient de transformée chroma AC sont égaux à 0.
2	Zéro ou plusieurs niveaux de coefficient de transformée chroma DC ont une valeur différente de zéro. Un ou plusieurs niveaux de coefficient de transformée chroma AC doivent avoir une valeur différente de zéro.

mb_qp_delta peut changer la valeur de QP_Y dans la couche de macrobloc. La valeur décodée de **mb_qp_delta** doit être dans la gamme de -26 à $+25$ inclus. **mb_qp_delta** doit être supposé égal à 0 lorsqu'il n'est pas présent pour tout macrobloc (y compris les types de macroblocs **P_Skip** et **B_Skip**).

La valeur de QP_Y est calculée comme suit.

$$QP_Y = (QP_{Y,PREV} + mb_qp_delta + 52) \% 52 \quad (7-23)$$

où $QP_{Y,PREV}$ est le paramètre de quantification luma, QP_Y , du précédent macrobloc dans l'ordre de décodage dans la tranche en cours. Pour le premier macrobloc dans la tranche, $QP_{Y,PREV}$ est mis initialement égal à $SliceQP_Y$ calculé dans l'équation 7-16 au début de chaque tranche.

7.4.5.1 Sémantique de prédiction de macrobloc

Tous les échantillons du macrobloc sont prédits. Les modes de prédiction sont déduits en utilisant les éléments syntaxiques suivants.

prev_intra4x4_pred_mode_flag[luma4x4BlkIdx] et **rem_intra4x4_pred_mode**[luma4x4BlkIdx] spécifient la prédiction Intra_4x4 des blocs luma 4x4 avec l'indice luma4x4BlkIdx = 0..15.

intra_chroma_pred_mode spécifie le type de prédiction spatiale utilisé pour chroma dans les macroblocs utilisant la prédiction Intra_4x4 ou Intra_16x16, comme indiqué au Tableau 7-13.

Tableau 7-13 – Relations entre intra_chroma_pred_mode et modes de prédiction spatiale

intra_chroma_pred_mode	Mode d'intraprédiction chroma
0	DC
1	Horizontal
2	Vertical
3	Plan

ref_idx_10[mbPartIdx] lorsqu'il est présent spécifie l'indice dans la liste 0 de l'image de référence à utiliser pour la prédiction.

La gamme de `ref_idx_10[mbPartIdx]`, l'indice dans la liste 0 de l'image de référence et, si applicable, la parité du champ au sein de l'image de référence utilisée pour la prédiction sont spécifiés comme suit.

- Si `MbaffFrameFlag` est égal à 0 ou si `mb_field_decoding_flag` est égal à 0, la valeur de `ref_idx_10[mbPartIdx]` doit être dans la gamme de 0 à `num_ref_idx_10_active_minus1` inclus.
- Autrement (`MbaffFrameFlag` est égal à 1 et `mb_field_decoding_flag` est égal à 1), la valeur de `ref_idx_10[mbPartIdx]` doit être dans la gamme de 0 à $2 * \text{num_ref_idx_10_active_minus1} + 1$, inclus.

Lorsque seule une image de référence est utilisée pour l'interprédiction, les valeurs de `ref_idx_10[mbPartIdx]` doivent être supposées égales à 0.

`ref_idx_11[mbPartIdx]` a la même sémantique que `ref_idx_10`, avec 10 et liste 0 remplacés respectivement par 11 et liste 1.

`mvd_10[mbPartIdx][0][compIdx]` spécifie la différence entre une composante vectorielle à utiliser et sa prédiction. L'indice `mbPartIdx` spécifie à quelle subdivision de macrobloc `mvd_10` est alloué. La subdivision du macrobloc est spécifiée par `mb_type`. La différence de composante horizontale de vecteur cinétique est décodée la première dans l'ordre de décodage et est allouée à `CompIdx = 0`. Le composant vertical de vecteur cinétique est décodé le deuxième dans l'ordre de décodage et est alloué à `CompIdx = 1`. La gamme des composants de `mvd_10[mbPartIdx][0][compIdx]` est spécifiée par des contraintes sur les valeurs des variables de vecteur cinétique qui en sont déduites comme spécifié à l'Annexe A.

`mvd_11[mbPartIdx][0][compIdx]` a la même sémantique que `mvd_10`, avec 10 et L0 remplacés par 11 et L1, respectivement.

7.4.5.2 Sémantique de prédiction de sous-macrobloc

`sub_mb_type[mbPartIdx]` spécifie les types de sous-macrobloc.

Les tableaux et la sémantique sont spécifiés pour les divers types de sous-macrobloc pour les tranches P, SP et B. Chaque tableau présente la valeur de `sub_mb_type`, le nom de `sub_mb_type`, le nombre de subdivisions de sous-macrobloc utilisées (donné par la fonction `NumSubMbPart(sub_mb_type)`) et le mode de prédiction du sous-macrobloc (donné par la fonction `SubMbPredMode(sub_mb_type)`). Dans le texte, la valeur de `sub_mb_type` peut être référencée "type de sous-macrobloc". Dans le texte, la valeur de `SubMbPredMode()` peut être référencée "mode de prédiction de sous-macrobloc".

Les types de sous-macrobloc pour les types de macrobloc P sont spécifiés au Tableau 7-14.

Tableau 7-14 – Types de sous-macrobloc dans les macroblocs P

<code>sub_mb_type[mbPartIdx]</code>	Nom de <code>sub_mb_type[mbPartIdx]</code>	<code>NumSubMbPart</code> (<code>sub_mb_type[mbPartIdx]</code>)	<code>SubMbPredMode</code> (<code>sub_mb_type[mbPartIdx]</code>)	<code>SubMbPartWidth</code> (<code>sub_mb_type[mbPartIdx]</code>)	<code>SubMbPartHeight</code> (<code>sub_mb_type[mbPartIdx]</code>)
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

La sémantique suivante est allouée aux types de sous-macrobloc dans le Tableau 7-14.

- P_L0_MxN, avec MxN remplacé par 8x8, 8x4, 4x8 ou 4x4: les échantillons du sous-macrobloc sont prédits en utilisant une subdivision luma d'une taille MxN égale à 8x8, deux subdivisions luma d'une taille MxN égale à 8x4 ou deux subdivisions luma d'une taille MxN égale à 4x8 ou quatre subdivisions luma d'une taille MxN égale à 4x4 et les échantillons chroma associés, respectivement.

La sémantique suivante est allouée aux modes de prédiction de sous-macrobloc (SubMbPredMode()) au Tableau 7-14.

- Pred_L0: voir la sémantique au Tableau 7-10.

Les types de sous-macrobloc pour les types de macrobloc B sont spécifiés au Tableau 7-15.

Tableau 7-15 – Types de sous-macrobloc dans les macroblocs B

sub_mb_type mbPartIdx	Nom de sub_mb_type mbPartIdx	NumSubMbPart (sub_mb_type mbPartIdx)	SubMbPredMode (sub_mb_type mbPartIdx)	SubMbPartWidth (sub_mb_type mbPartIdx)	SubMbPartHeight (sub_mb_type mbPartIdx)
na	B_Skip	na	Direct	4	4
na	B_Direct_16x16	na	Direct	4	4
0	B_Direct_8x8	na	Direct	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

La sémantique suivante est allouée aux types de macrobloc dans le Tableau 7-15:

- B_Skip et B_Direct_16x16: aucune différence de vecteur cinétique ou d'indice de référence n'est présente pour le sous-macrobloc dans le flux binaire. Les fonctions SubMbPartWidth() et SubMbPartHeight() sont utilisées dans le processus de déduction pour les vecteurs cinétiques et les indices de trame de référence au § 8.4.1 pour la prédiction en mode direct.
- B_Direct_8x8: aucune différence de vecteur cinétique ou d'indice de référence n'est présente pour le sous-macrobloc dans le flux binaire. Les fonctions SubMbPartWidth(B_Direct_8x8) et SubMbPartHeight(B_Direct_8x8) sont utilisées dans le processus de déduction pour les vecteurs cinétiques et les indices de trame de référence au § 8.4.1 pour la prédiction en mode direct.

- B_X_MxN, avec X remplacé par L0, L1 ou Bi et MxN remplacé par 8x8, 8x4, 4x8 ou 4x4: les échantillons du sous-macrobloc sont prédits en utilisant une subdivision luma de taille MxN égale à 8x8 ou les échantillons du sous-macrobloc sont prédits en utilisant deux subdivisions luma d'une taille MxN égale à 8x4 ou les échantillons du sous-macrobloc sont prédits en utilisant deux subdivisions luma d'une taille MxN égale à 4x8 ou les échantillons du sous-macrobloc sont prédits en utilisant quatre subdivisions luma de taille MxN égale à 4x4 et les échantillons chroma associés, respectivement. Toutes les subdivisions de sous-macrobloc partagent le même indice de référence. Pour une subdivision MxN de sous-macrobloc dans un sous-macrobloc dont le sub_mb_type est B_X_MxN avec X remplacé par L0 ou L1, une différence de vecteur cinétique est présente dans le flux binaire. Pour une subdivision MxN de sous-macrobloc dans un sous-macrobloc dont le sub_mb_type est B_Bi_MxN, deux différences de vecteur cinétique sont présentes dans le flux binaire.

La sémantique suivante est allouée aux modes de prédiction de sous-macrobloc (SubMbPredMode()) au Tableau 7-15.

- Direct: voir la sémantique pour le Tableau 7-11.
- Pred_L0: voir la sémantique pour le Tableau 7-10.
- Pred_L1: voir la sémantique pour le Tableau 7-11.
- BiPred: voir la sémantique pour le Tableau 7-11.

ref_idx_10[mbPartIdx] a la même sémantique que ref_idx_10 au § 7.4.5.1.

ref_idx_11[mbPartIdx] a la même sémantique que ref_idx_11 au § 7.4.5.1.

mvd_10[mbPartIdx][subMbPartIdx][compIdx] a la même sémantique que mvd_10 au § 7.4.5.1, excepté qu'il s'applique à l'indice de subdivision de sous-macrobloc avec subMbPartIdx. Les indices mbPartIdx et subMbPartIdx spécifient à quelle subdivision de macrobloc et subdivision de sous-macrobloc mvd_10 est alloué.

mvd_11[mbPartIdx][subMbPartIdx][compIdx] a la même sémantique que mvd_11 au § 7.4.5.1.

7.4.5.3 Sémantique de données résiduelles

La structure syntaxique residual_block(), qui est utilisée pour l'analyse grammaticale des niveaux des coefficients de transformée, est allouée comme suit.

- Si entropy_coding_mode_flag est égal à 0, residual_block est mis égal à residual_block_cavlc, qui est utilisé pour l'analyse grammaticale des éléments syntaxiques pour les niveaux de coefficients de transformée.
- Autrement (entropy_coding_mode_flag est égal à 1), residual_block est mis égal à residual_block_cabac, qui est utilisé pour l'analyse grammaticale des éléments syntaxiques pour les niveaux de coefficients de transformée.

En fonction de mb_type, luma ou chroma, la structure syntaxique residual_block(coeffLevel, maxNumCoeff) est utilisée avec les arguments coeffLevel, qui est une liste contenant les niveaux de coefficient de transformée maxNumCoeff dont l'analyse grammaticale est effectuée dans residual_block() et maxNumCoeff comme suit.

- En fonction de MbPartPredMode(mb_type, 0), on applique ce qui suit.
 - Si MbPartPredMode(mb_type, 0) est égal à Intra_16x16, les niveaux de coefficient de transformée sont analysés dans la liste Intra16x16DCLevel et dans les 16 listes Intra16x16ACLevel[i]. Intra16x16DCLevel contient les 16 niveaux de coefficient de transformée des niveaux de coefficient de transformée DC pour chaque bloc luma 4x4. Pour chacun des 16 blocs luma 4x4 indexés par i = 0..15, les 15 niveaux de coefficients de transformée AC du i^{ème} bloc sont analysés dans la i^{ème} liste Intra16x16ACLevel[i].

- Autrement ($\text{MbPartPredMode}(\text{mb_type}, 0)$ n'est pas égal à Intra_16x16), pour chacun des 16 blocs luma $4x4$ indexés par $i = 0..15$, les 16 niveaux de coefficient de transformée AC du $i^{\text{ème}}$ bloc sont analysés dans la $i^{\text{ème}}$ liste $\text{LumaLevel}[i]$.
- Pour chaque composant chroma, indexé par $i\text{CbCr} = 0..1$, les 4 niveaux de coefficient de transformée DC des blocs chroma $4x4$ sont analysés grammaticalement dans la $i\text{CbCr}^{\text{ème}}$ liste $\text{ChromaDCLevel}[i\text{CbCr}]$.
- Pour chacun des blocs chroma $4x4$, indexés par $i4x4 = 0..3$, de chaque composant chroma, indexé par $i\text{CbCr} = 0..1$, les 15 niveaux de coefficient de transformée AC sont analysés grammaticalement dans la $i4x4^{\text{ème}}$ liste du $i\text{CbCr}^{\text{ème}}$ composant chroma $\text{ChromaACLevel}[i\text{CbCr}][i4x4]$.

7.4.5.3.1 Sémantique du codage CAVLC de bloc résiduel

La fonction $\text{TotalCoeff}(\text{coeff_token})$ qui est utilisée au § 7.3.5.3.1 renvoie le nombre de niveaux de coefficient de transformée différents de zéro déduits de coeff_token .

La fonction $\text{TrailingOnes}(\text{coeff_token})$ qui est utilisée au § 7.3.5.3.1 renvoie ceux de traîne qui sont déduits de coeff_token .

coeff_token spécifie le nombre total de niveaux de coefficient de transformée différents de zéro et le nombre de niveaux de coefficient de transformée de traîne dans un balayage de niveau de coefficient de transformée. Un niveau de coefficient de transformée de traîne est un des jusqu'à trois niveaux de coefficient de transformée consécutifs différents de zéro ayant une valeur absolue égale à 1 à la fin d'un balayage des niveaux de coefficient de transformée différents de zéro. La gamme de coeff_token est spécifiée au § 9.2.1.

trailing_ones_sign_flag spécifie le signe du niveau de coefficient de transformée de traîne comme suit.

- Si $\text{trailing_ones_sign_flag}$ est égal à 0, le niveau de coefficient de transformée correspondant est décodé comme +1.
- Autrement ($\text{trailing_ones_sign_flag}$ égal à 1), le niveau de coefficient de transformée correspondant est décodé comme -1.

level_prefix et **level_suffix** spécifient la valeur d'un niveau de coefficient de transformée différent de zéro. La gamme de level_prefix et level_suffix est spécifiée au § 9.2.2.

total_zeros spécifie le nombre total de niveaux de coefficient de transformée de valeur zéro qui sont localisés avant la position du dernier niveau de coefficient de transformée différent de zéro dans le balayage de niveaux de coefficient de transformée. La gamme de total_zeros est spécifiée au § 9.2.3.

run_before spécifie le nombre de niveaux de coefficient de transformée consécutifs dans le balayage avec la valeur zéro avant un niveau de coefficient de transformée de valeur différente de zéro. La gamme de run_before est spécifiée au § 9.2.3.

coeffLevel contient maxNumCoeff niveaux de coefficient de transformée pour la liste en cours de niveaux de coefficient de transformée.

7.4.5.3.2 Sémantique du codage CABAC de bloc résiduel

coded_block_flag spécifie si le bloc contient des niveaux de coefficient de transformée différents de zéro comme suit.

- Si coded_block_flag est égal à 0, le bloc ne contient aucun niveau de coefficient de transformée différent de zéro.
- Autrement (coded_block_flag est égal à 1), le bloc contient au moins un niveau de coefficient de transformée différent de zéro.

significant_coeff_flag[i] spécifie si le niveau de coefficient de transformée à la position de balayage *i* est différent de zéro comme suit.

- Si **significant_coeff_flag[i]** est égal à 0, le niveau de coefficient de transformée à la position de balayage *i* est mis égal à 0.
- Autrement (**significant_coeff_flag[i]** est égal à 1), le niveau de coefficient de transformée à la position de balayage *i* a une valeur différente de zéro.

last_significant_coeff_flag[i] spécifie comme suit, pour la position de balayage *i*, s'il y a des niveaux de coefficient de transformée différents de zéro pour les *i + 1* jusqu'à **maxNumCoeff – 1** positions de balayage suivantes.

- Si **last_significant_coeff_flag[i]** est égal à 1, tous les niveaux de coefficient de transformée suivants (dans l'ordre de balayage) du bloc ont leur valeur égale à 0.
- Autrement (**last_significant_coeff_flag[i]** est égal à 0), il y a d'autres niveaux de coefficient de transformée différents de zéro sur le chemin de balayage.

coeff_abs_level_minus1[i] est la valeur absolue d'un niveau de coefficient de transformée moins 1. La valeur de **coeff_abs_level_minus1** est contrainte par les limites fixées au § 8.5.

coeff_sign_flag[i] spécifie le signe d'un niveau de coefficient de transformée comme suit.

- Si **coeff_sign_flag** est égal à 0, le niveau de coefficient de transformée correspondant a une valeur positive.
- Autrement (**coeff_sign_flag** est égal à 1), le niveau de coefficient de transformée correspondant a une valeur négative.

coeffLevel contient **maxNumCoeff** niveaux de coefficient de transformée pour la liste en cours des niveaux de coefficient de transformée.

8 Processus de décodage

Les résultats de ce processus sont des échantillons décodés de l'image en cours (parfois désignée par la variable **CurrPic**).

Le présent article décrit le processus de décodage, étant donnés les éléments syntaxiques et les variables en majuscules de l'article 7.

Le processus de décodage est spécifié de telle sorte que tous les décodeurs doivent produire des résultats numériquement identiques. Tout processus de décodage qui produit des résultats identiques à la fin du processus décrit ici se conforme aux exigences du processus de décodage de la présente Recommandation | Norme internationale.

Chaque image mentionnée dans le présent article est une image primaire. Chaque tranche mentionnée dans le présent article est une tranche d'une image primaire. Chaque subdivision de données de tranche mentionnée dans le présent article est une subdivision de données de tranche d'une image primaire.

Un aperçu général du processus de décodage peut être donnée comme suit.

- Le décodage des unités NAL est spécifié au § 8.1.
- Les processus du § 8.2 spécifient les processus de décodage utilisant des éléments syntaxiques dans la couche de tranche et au-dessus.
 - Les variables et fonctions se rapportant au compte d'ordre d'image sont décrites au § 8.2.1 (elles ne doivent être invoquées que pour une seule tranche d'une image).
 - Les variables et fonctions se rapportant au mappage de macrobloc à groupe de tranches sont décrites au § 8.2.2 (elles ne doivent être invoquées que pour une seule tranche d'une image).

- La méthode de combinaison des diverses subdivisions, lorsque la subdivision de données de tranche est utilisée, est décrite au § 8.2.3.
- Avant le décodage de chaque tranche, la déduction des listes d'images de référence, comme décrite au § 8.2.4, est nécessaire pour l'interprédiction.
- Lorsque l'image en cours est une image de référence et après que toutes les tranches de l'image en cours ont été décodées, le processus de marquage d'image de référence décodée du § 8.2.5 spécifie comment l'image en cours est utilisée dans le processus de décodage de l'interprédiction dans les images décodées ultérieurement.
- Aux § 8.3, 8.4, 8.5, 8.6 et 8.7 sont spécifiés les processus de décodage utilisant des éléments syntaxiques dans la couche de macrobloc et au-dessus.
 - Le processus d'intraprédiction pour les macroblocs I et SI, excepté pour les macroblocs I_PCM comme spécifié au § 8.3, a pour résultat les échantillons d'intraprédiction. Pour les macroblocs I_PCM, le § 8.3 spécifie directement un processus de construction d'image. Les résultats sont les échantillons construits avant le processus de filtrage de démontage de blocs.
 - Le processus d'interprédiction pour les macroblocs P et B est spécifié au § 8.4 et les échantillons d'interprédiction en sont le résultat.
 - Le processus de décodage de coefficient de transformée et de construction d'image avant le processus de filtrage de démontage de blocs est spécifié au § 8.5. Ce processus déduit les échantillons pour les macroblocs I et B ainsi que pour les macroblocs P dans les tranches P. Les résultats sont les échantillons construits avant le processus de filtrage de démontage de blocs.
 - Le processus de décodage pour les macroblocs P dans les tranches SP ou les macroblocs SI est spécifié au § 8.6. Ce processus déduit les échantillons pour les macroblocs P dans les tranches SP et pour les macroblocs SI. Les résultats sont les échantillons construits avant le processus de filtrage de démontage de blocs.
 - Les échantillons construits avant le processus de filtrage de démontage de blocs qui sont contre les bordures des blocs et macroblocs sont traités par un filtre de démontage de blocs comme spécifié au § 8.7, dont les résultats sont les échantillons décodés.

8.1 Processus de décodage d'unité NAL

Les entrées pour ce processus sont des unités NAL.

Les résultats de ce processus sont des structures syntaxiques de charge utile RBSP encapsulées au sein des unités NAL.

Le processus de décodage extrait pour chaque unité NAL la structure syntaxique de charge utile RBSP de l'unité NAL et effectue alors les processus de décodage spécifiés pour la structure syntaxique de charge utile RBSP dans l'unité NAL comme suit.

Le paragraphe 8.2 décrit le processus de décodage pour des unités NAL avec `nal_unit_type` égal à 1 jusqu'à 5.

Le paragraphe 8.3 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec `nal_unit_type` égal à 1, 2 et 5.

Le paragraphe 8.4 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec `nal_unit_type` égal à 1 et 2.

Le paragraphe 8.5 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec `nal_unit_type` égal à 1 et 3 jusqu'à 5.

Le paragraphe 8.6 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec `nal_unit_type` égal à 1 et 3 jusqu'à 5.

Le paragraphe 8.7 décrit le processus de décodage pour un macrobloc ou partie d'un macrobloc codé dans des unités NAL avec `nal_unit_type` égal à 1 jusqu'à 5.

Les unités NAL avec `nal_unit_type` égal à 7 et 8 contiennent respectivement des ensembles de paramètres de séquence et des ensembles de paramètres d'image. Les ensembles de paramètres d'image sont utilisés dans les processus de décodage des autres unités NAL comme déterminé par référence à un ensemble de paramètres d'image au sein des en-têtes de tranche de chaque image. Les ensembles de paramètres de séquence sont utilisés dans les processus de décodage des autres unités NAL comme déterminé par référence à un ensemble de paramètres de séquence au sein des ensembles de paramètres d'image de chaque séquence.

Il n'y a pas de processus de décodage normalisé spécifié pour des unités NAL avec `nal_unit_type` égal à 6, 9, 10, 11 et 12.

8.2 Processus de décodage de tranche

8.2.1 Processus de décodage pour le compte d'ordre d'image

Les résultats de ce processus sont `TopFieldOrderCnt` (si applicable) et `BottomFieldOrderCnt` (si applicable).

Les comptes d'ordre d'image sont utilisés pour déterminer les arrangements d'image initiaux pour les images de référence dans le décodage de tranches B (voir les § 8.2.4.2.3 et 8.2.4.2.4), pour représenter les différences d'ordre d'image entre les trames ou champs pour le calcul du vecteur cinétique dans le mode temporel direct (voir le § 8.4.1.2.3), pour la prédiction pondérée en mode implicite dans les tranches B (voir le § 8.4.2.3.2) et pour la vérification de conformité du décodeur (voir le § C.4).

Les informations de compte d'ordre d'image sont déduites comme suit pour chaque trame, champ (qu'il soit décodé à partir d'un champ codé ou comme partie d'une trame décodée) ou paire de champs complémentaires:

- chaque trame codée est associée à deux comptes d'ordre d'image, appelés respectivement `TopFieldOrderCnt` et `BottomFieldOrderCnt` pour son champ supérieur et son champ inférieur;
- chaque champ codé est associé à un compte d'ordre d'image, appelé `TopFieldOrderCnt` pour un champ supérieur codé et `BottomFieldOrderCnt` pour un champ inférieur;
- chaque paire de champs complémentaires est associée à deux comptes d'ordre d'image, qui sont respectivement le `TopFieldOrderCnt` pour son champ supérieur codé et le `BottomFieldOrderCnt` pour son champ inférieur codé.

`TopFieldOrderCnt` et `BottomFieldOrderCnt` indiquent l'ordre d'image du champ supérieur ou champ inférieur correspondant par rapport au premier champ résultant de la précédente image IDR ou de la précédente image de référence incluant un `memory_management_control_operation` égal à 5 dans l'ordre de décodage.

`TopFieldOrderCnt` et `BottomFieldOrderCnt` sont déduits en invoquant un des processus de décodage pour le type de compte d'ordre d'image 0, 1 et 2, respectivement aux § 8.2.1.1, 8.2.1.2 et 8.2.1.3. Lorsque l'image en cours inclut une opération de commande de gestion de mémoire égale à 5, après le décodage de l'image en cours, `tempPicOrderCnt` est mis égal à `PicOrderCnt(CurrPic)`, `TopFieldOrderCnt` de l'image en cours (s'il en est une) est mis égal à `TopFieldOrderCnt – tempPicOrderCnt` et `BottomFieldOrderCnt` de l'image en cours (s'il en est une) est mis égal à `BottomFieldOrderCnt – tempPicOrderCnt`.

Le flux binaire ne doit pas contenir de données qui auraient pour résultat un $\text{Min}(\text{TopFieldOrderCnt}, \text{BottomFieldOrderCnt})$ qui ne soit pas égal à 0 pour une trame IDR codée, un `TopFieldOrderCnt` qui ne soit pas égal à 0 pour un champ supérieur IDR codé ou un

BottomFieldOrderCnt qui ne soit pas égal à 0 pour un champ inférieur IDR codé. Ainsi, au moins un de TopFieldOrderCnt et BottomFieldOrderCnt doit être égal à 0 pour les champs d'une trame IDR codée.

Lorsque l'image en cours n'est pas une image IDR, on applique ce qui suit.

- Considérer la liste de variables listD contenant comme éléments les valeurs TopFieldOrderCnt et BottomFieldOrderCnt associées à la liste des images et incluant tout ce qui suit:
 - la première image dans la liste est l'image précédente de n'importe lequel des types suivants:
 - une image IDR;
 - une image contenant un memory_management_control_operation égal à 5;
 - les images supplémentaires suivantes:
 - Si pic_order_cnt_type est égal à 0, toutes les autres images qui suivent dans l'ordre de décodage après la première image dans la liste et qui ne sont pas des trames "non existant" déduites par le processus de décodage pour les intervalles dans frame_num spécifiés au § 8.2.5.2 et qui précèdent l'image en cours dans l'ordre de décodage ou constituent l'image en cours. Lorsque pic_order_cnt_type est égal à 0 et que l'image en cours n'est pas une trame "non existant" déduite par le processus de décodage pour les intervalles dans frame_num spécifiés au § 8.2.5.2, l'image en cours est incluse dans listD avant l'invocation du processus de marquage d'image de référence décodée.
 - Autrement (pic_order_cnt_type est différent de 0), toutes les autres images qui suivent dans l'ordre de décodage après la première image dans la liste et qui précèdent l'image en cours dans l'ordre de décodage ou constituent l'image en cours. Lorsque pic_order_cnt_type est différent de 0, l'image en cours est incluse dans listD avant l'invocation du processus de marquage d'image de référence décodée.
- Considérer la liste de variables listO qui contient les éléments de listD triés en ordre ascendant. listO ne doit contenir aucun des éléments suivants:
 - une paire de TopFieldOrderCnt et BottomFieldOrderCnt pour une trame ou paire de champs complémentaires qui ne soient pas à des positions consécutives dans listO;
 - un TopFieldOrderCnt qui ait une valeur égale à un autre TopFieldOrderCnt;
 - un BottomFieldOrderCnt qui ait une valeur égale à un autre BottomFieldOrderCnt;
 - un BottomFieldOrderCnt qui ait une valeur égale à un TopFieldOrderCnt à moins que le BottomFieldOrderCnt et le TopFieldOrderCnt n'appartiennent à la même trame codée ou paire de champs complémentaires.

Le flux binaire ne doit pas contenir de données qui résulteraient en des valeurs de TopFieldOrderCnt, BottomFieldOrderCnt, PicOrderCntMsb ou FrameNumOffset utilisées dans le processus de décodage, comme spécifié aux § 8.2.1.1 à 8.2.1.3, qui excèdent la gamme de valeurs de -2^{31} à $2^{31} - 1$ inclus.

La fonction PicOrderCnt(picX) est spécifiée comme suit:

$$\begin{aligned} & \text{si(picX est une trame ou une paire de champs complémentaires)} \\ & \quad \text{PicOrderCnt(picX)} = \text{Min(TopFieldOrderCnt, BottomFieldOrderCnt) de la trame ou paire} \\ & \quad \text{de champs complémentaires picX} \\ & \text{ou si(picX est un champ supérieur)} \\ & \quad \text{PicOrderCnt(picX)} = \text{TopFieldOrderCnt de champ picX} \\ & \text{ou si(picX est un champ inférieur)} \\ & \quad \text{PicOrderCnt(picX)} = \text{BottomFieldOrderCnt de champ picX} \end{aligned} \tag{8-1}$$

Alors DiffPicOrderCnt(picA, picB) est spécifié comme suit:

$$\text{DiffPicOrderCnt}(\text{picA}, \text{picB}) = \text{PicOrderCnt}(\text{picA}) - \text{PicOrderCnt}(\text{picB}) \quad (8-2)$$

Le flux binaire doit contenir des données ayant pour résultat des valeurs de DiffPicOrderCnt(picA, picB) utilisées dans le processus de décodage qui soient dans la gamme de -2^{15} à $2^{15} - 1$ inclus.

NOTE – Soit X l'image en cours et Y et Z deux autres images dans la même séquence, Y et Z sont considérées comme étant dans la même direction d'ordre de sortie que X lorsque DiffPicOrderCnt(X, Y) et DiffPicOrderCnt(X, Z) sont tous deux positifs ou tous deux négatifs.

NOTE – De nombreuses applications allouent PicOrderCnt(X) proportionnel au temps d'échantillonnage de l'image X par rapport au temps d'échantillonnage d'une image IDR.

Lorsque l'image en cours inclut un memory_management_control_operation égal à 5, PicOrderCnt(CurrPic) doit être supérieur à PicOrderCnt(toute autre image de listD).

8.2.1.1 Processus de décodage pour le type 0 de compte d'ordre d'image

Ce processus est invoqué lorsque pic_order_cnt_type est égal à 0.

L'entrée pour ce processus est le PicOrderCntMsb de la précédente image de référence dans l'ordre de décodage comme spécifié au présent paragraphe.

Les résultats de ce processus sont TopFieldOrderCnt ou BottomFieldOrderCnt ou les deux.

Les variables prevPicOrderCntMsb et prevPicOrderCntLsb sont déduites comme suit.

- Si l'image en cours est une image IDR, prevPicOrderCntMsb et prevPicOrderCntLsb sont mises égales à 0.
- Autrement (si l'image en cours n'est pas une image IDR), on applique ce qui suit.
 - Si la précédente image de référence dans l'ordre de décodage inclut un memory_management_control_operation égal à 5, on applique ce qui suit.
 - Si la précédente image de référence dans l'ordre de décodage n'est pas un champ inférieur, prevPicOrderCntMsb est mis égal à 0 et prevPicOrderCntLsb est mis égal à la valeur de TopFieldOrderCnt pour la précédente image de référence dans l'ordre de décodage.
 - Autrement (la précédente image de référence dans l'ordre de décodage est un champ inférieur de référence), prevPicOrderCntMsb et prevPicOrderCntLsb sont mises égales à 0.
 - Autrement (la précédente image de référence dans l'ordre de décodage n'incluait pas un memory_management_control_operation égal à 5), prevPicOrderCntMsb est mise égale à PicOrderCntMsb de l'image précédente de référence dans l'ordre de décodage et prevPicOrderCntLsb est mise égale à la valeur de pic_order_cnt_lsb de l'image de référence précédente dans l'ordre de décodage.

PicOrderCntMsb de l'image en cours est déduit comme suit:

$$\begin{aligned} & \text{si}((\text{pic_order_cnt_lsb} < \text{prevPicOrderCntLsb}) \ \&\& \\ & \quad ((\text{prevPicOrderCntLsb} - \text{pic_order_cnt_lsb}) \geq (\text{MaxPicOrderCntLsb} / 2))) \\ & \quad \text{PicOrderCntMsb} = \text{prevPicOrderCntMsb} + \text{MaxPicOrderCntLsb} \\ \text{ou si}((\text{pic_order_cnt_lsb} > \text{prevPicOrderCntLsb}) \ \&\& \\ & \quad ((\text{pic_order_cnt_lsb} - \text{prevPicOrderCntLsb}) > (\text{MaxPicOrderCntLsb} / 2))) \\ & \quad \text{PicOrderCntMsb} = \text{prevPicOrderCntMsb} - \text{MaxPicOrderCntLsb} \\ \text{autrement} \\ & \quad \text{PicOrderCntMsb} = \text{prevPicOrderCntMsb} \end{aligned} \quad (8-3)$$

Lorsque l'image en cours n'est pas un champ inférieur, TopFieldOrderCnt est déduit comme suit:

$$\begin{aligned} & \text{si}(!\text{field_pic_flag} \ || \ !\text{bottom_field_flag}) \\ & \quad \text{TopFieldOrderCnt} = \text{PicOrderCntMsb} + \text{pic_order_cnt_lsb} \end{aligned} \quad (8-4)$$

Lorsque l'image en cours n'est pas un champ supérieur, BottomFieldOrderCnt est déduit comme suit:

```

si( !field_pic_flag )
    BottomFieldOrderCnt = TopFieldOrderCnt + delta_pic_order_cnt_bottom
ou si( bottom_field_flag )
    BottomFieldOrderCnt = PicOrderCntMsb + pic_order_cnt_lsb

```

(8-5)

8.2.1.2 Processus de décodage pour le type 1 de compte d'ordre d'image

Ce processus est invoqué lorsque pic_order_cnt_type est égal à 1.

L'entrée pour ce processus est FrameNumOffset de l'image précédente dans l'ordre de décodage comme spécifié au présent paragraphe.

Les résultats de ce processus sont TopFieldOrderCnt ou BottomFieldOrderCnt ou les deux.

Les valeurs de TopFieldOrderCnt et BottomFieldOrderCnt sont déduites comme spécifié au présent paragraphe. Soit prevFrameNum égal au frame_num de l'image précédente dans l'ordre de décodage.

Lorsque l'image en cours n'est pas une image IDR, la variable prevFrameNumOffset est déduite comme suit.

- Si l'image précédente dans l'ordre de décodage inclut un memory_management_control_operation égal à 5, prevFrameNumOffset est mis égal à 0.
- Autrement (l'image précédente dans l'ordre de décodage n'inclut pas un memory_management_control_operation égal à 5), prevFrameNumOffset est mis égal à la valeur de FrameNumOffset de l'image précédente dans l'ordre de décodage.

NOTE – Lorsque gaps_in_frame_num_value_allowed_flag est égal à 1, l'image précédente dans l'ordre de décodage peut être une frame "non existant" déduite par le processus de décodage pour les intervalles dans frame_num spécifiés au § 8.2.5.2.

La déduction se fait selon les étapes suivantes.

1. La variable FrameNumOffset est déduite comme suit:

```

si( nal_unit_type == 5 )
    FrameNumOffset = 0
ou si( prevFrameNum > frame_num )
    FrameNumOffset = prevFrameNumOffset + MaxFrameNum
ou
    FrameNumOffset = prevFrameNumOffset

```

(8-6)

2. La variable absFrameNum est déduite comme suit:

```

si( num_ref_frames_in_pic_order_cnt_cycle != 0 )
    absFrameNum = FrameNumOffset + frame_num
ou
    absFrameNum = 0
si( nal_ref_idc == 0 && absFrameNum > 0 )
    absFrameNum = absFrameNum - 1

```

(8-7)

3. Lorsque absFrameNum > 0, picOrderCntCycleCnt et frameNumInPicOrderCntCycle sont déduits comme suit:

```

si( 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. La variable expectedDeltaPerPicOrderCntCycle est déduite comme suit:

```

expectedDeltaPerPicOrderCntCycle = 0
pour( i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++ )
    expectedDeltaPerPicOrderCntCycle += offset_for_ref_frame[ i ]

```

(8-9)

5. La variable `expectedPicOrderCnt` est déduite comme suit:

```
si( absFrameNum > 0 ){
    expectedPicOrderCnt = picOrderCntCycleCnt * expectedDeltaPerPicOrderCntCycle
    pour( i = 0; i <= frameNumInPicOrderCntCycle; i++ )
        expectedPicOrderCnt = expectedPicOrderCnt + offset_for_ref_frame[ i ]
} ou
    expectedPicOrderCnt = 0
si( nal_ref_idc == 0 )
    expectedPicOrderCnt = expectedPicOrderCnt + offset_for_non_ref_pic
```

 (8-10)

6. Les variables `TopFieldOrderCnt` ou `BottomFieldOrderCnt` sont déduites comme suit:

```
si( !field_pic_flag ) {
    TopFieldOrderCnt = expectedPicOrderCnt + delta_pic_order_cnt[ 0 ]
    BottomFieldOrderCnt = TopFieldOrderCnt +
        offset_for_top_to_bottom_field + delta_pic_order_cnt[ 1 ]
} ou si( !bottom_field_flag )
    TopFieldOrderCnt = expectedPicOrderCnt + delta_pic_order_cnt[ 0 ]
ou
    BottomFieldOrderCnt = expectedPicOrderCnt + offset_for_top_to_bottom_field + delta_pic_order_cnt[ 0 ]
```

 (8-11)

8.2.1.3 Processus de décodage pour le type 2 de compte d'ordre d'image

Ce processus est invoqué lorsque `pic_order_cnt_type` est égal à 2.

Les résultats de ce processus sont `TopFieldOrderCnt` ou `BottomFieldOrderCnt` ou les deux.

Soit `prevFrameNum` égal au `frame_num` de l'image précédente dans l'ordre de décodage.

Lorsque l'image en cours n'est pas une image IDR, la variable `prevFrameNumOffset` est déduite comme suit.

- Si l'image précédente dans l'ordre de décodage inclut un `memory_management_control_operation` égal à 5, `prevFrameNumOffset` est mis égal à 0.
- Autrement (l'image précédente dans l'ordre de décodage n'inclut pas un `memory_management_control_operation` égal à 5), `prevFrameNumOffset` est mis égal à la valeur de `FrameNumOffset` de l'image précédente dans l'ordre de décodage.

NOTE – Lorsque `gaps_in_frame_num_value_allowed_flag` est égal à 1, l'image précédente dans l'ordre de décodage peut être une frame "non existant" déduite par le processus de décodage pour les intervalles dans `frame_num` spécifiés au § 8.2.5.2.

La variable `FrameNumOffset` est déduite comme suit.

```
si( nal_unit_type == 5 )
    FrameNumOffset = 0
ou si( prevFrameNum > frame_num )
    FrameNumOffset = prevFrameNumOffset + MaxFrameNum
ou
    FrameNumOffset = prevFrameNumOffset
```

 (8-12)

La variable `tempPicOrderCnt` est déduite comme suit:

```
si( nal_unit_type == 5 )
    tempPicOrderCnt = 0
ou si( nal_ref_idc == 0 )
    tempPicOrderCnt = 2 * ( FrameNumOffset + frame_num ) - 1
ou
    tempPicOrderCnt = 2 * ( FrameNumOffset + frame_num )
```

 (8-13)

Les variables `TopFieldOrderCnt` ou `BottomFieldOrderCnt` sont déduites comme suit:

```
si( !field_pic_flag ) {
    TopFieldOrderCnt = tempPicOrderCnt
    BottomFieldOrderCnt = tempPicOrderCnt
} ou si( bottom_field_flag )
```

 (8-14)

BottomFieldOrderCnt = tempPicOrderCnt

ou

TopFieldOrderCnt = tempPicOrderCnt

NOTE – Le type 2 de compte d'ordre d'image ne peut pas être utilisé dans une séquence vidéo codée qui contient des images de non-référence consécutives qui auraient pour résultat plus d'une de ces images ayant la même valeur de TopFieldOrderCnt ou plus d'une de ces images ayant la même valeur de BottomFieldOrderCnt.

NOTE – Le type 2 de compte d'ordre d'image a pour résultat un ordre de sortie qui est le même que l'ordre de décodage.

8.2.2 Mappage de processus de décodage pour macrobloc avec groupe de tranches

Les entrées pour ce processus sont l'ensemble actif de paramètres d'image et l'en-tête de tranche de la tranche à décoder.

Le résultat de ce processus est un mappage de macrobloc à groupe de tranche MbToSliceGroupMap.

Ce processus est invoqué au début de chaque tranche.

NOTE – Le résultat de ce processus est égal pour toutes les tranches d'une image.

Lorsque num_slice_groups_minus1 est égal à 1 et slice_group_map_type est égal à 3, 4 ou 5, les groupes de tranches 0 et 1 ont une taille et une forme déterminées par slice_group_change_direction_flag comme indiqué au Tableau 8-1 et spécifié aux § 8.2.2.4-8.2.2.6.

Tableau 8-1 – Type précis de mappage de groupe de tranches

slice_group_map_type	slice_group_change_direction_flag	Type précis de mappage de groupe de tranches
3	0	Sortie de boîte dans le sens des aiguilles d'une montre
3	1	Sortie de boîte dans le sens contraire des aiguilles d'une montre
4	0	Balayage de trame
4	1	Balayage de trame inverse
5	0	Balayage à droite
5	1	Balayage à gauche

Dans un tel cas, les unités de mappage de groupe de tranches MapUnitsInSliceGroup0, dans l'ordre croissant spécifié, sont allouées pour le groupe de tranches 0 et les unités de mappage de groupe de tranches PicSizeInMapUnits – MapUnitsInSliceGroup0 restantes de l'image sont allouées pour le groupe de tranches 1.

Lorsque num_slice_groups_minus1 est égal à 1 et slice_group_map_type est égal à 4 ou 5, la variable sizeOfUpperLeftGroup est définie comme suit:

$$\text{sizeOfUpperLeftGroup} = \begin{cases} \text{slice_group_change_direction_flag} & \text{?} \\ (\text{PicSizeInMapUnits} - \text{MapUnitsInSliceGroup0}) : \text{MapUnitsInSliceGroup0} & \end{cases} \quad (8-15)$$

La variable mapUnitToSliceGroupMap est déduite comme suit.

- Si num_slice_groups_minus1 est égal à 0, la correspondance d'unité de correspondance au groupe de tranches est générée pour tous les i de 0 à PicSizeInMapUnits – 1 inclus, comme spécifié par:

$$\text{mapUnitToSliceGroupMap}[i] = 0 \quad (8-16)$$

- Autrement (`num_slice_groups_minus1` n'est pas égal à 0), `mapUnitToSliceGroupMap` est déduit comme suit.
 - Si `slice_group_map_type` est égal à 0, la déduction de `mapUnitToSliceGroupMap` comme spécifié au § 8.2.2.1 s'applique.
 - Autrement, si `slice_group_map_type` est égal à 1, la déduction de `mapUnitToSliceGroupMap` comme spécifié au § 8.2.2.2 s'applique.
 - Autrement, si `slice_group_map_type` est égal à 2, la déduction de `mapUnitToSliceGroupMap` comme spécifié au § 8.2.2.3 s'applique.
 - Autrement, si `slice_group_map_type` est égal à 3, la déduction de `mapUnitToSliceGroupMap` comme spécifié au § 8.2.2.4 s'applique.
 - Autrement, si `slice_group_map_type` est égal à 4, la déduction de `mapUnitToSliceGroupMap` comme spécifié au § 8.2.2.5 s'applique.
 - Autrement, si `slice_group_map_type` est égal à 5, la déduction de `mapUnitToSliceGroupMap` comme spécifié au § 8.2.2.6 s'applique.
 - Autrement (`slice_group_map_type` est égal à 6), la déduction de `mapUnitToSliceGroupMap` comme spécifié au § 8.2.2.7 s'applique.

Après déduction de `mapUnitToSliceGroupMap`, le processus spécifié au § 8.2.2.8 est invoqué pour convertir l'unité de mappage en mappage de groupe de tranches `mapUnitToSliceGroupMap` avec le mappage de macrobloc à groupe de tranches `MbToSliceGroupMap`. Après déduction du mappage de macrobloc à groupe de tranches comme spécifié au § 8.2.2.8, la fonction `NextMbAddress(n)` est définie comme la valeur de la variable `nextMbAddress` déduite comme spécifié par:

```
i = n + 1
tandis que( i < PicSizeInMbs && MbToSliceGroupMap[ i ] != MbToSliceGroupMap[ n ] )
    i++;
nextMbAddress = i
```

(8-17)

8.2.2.1 Spécification pour le type de mappage de groupe de tranches entrelacées

Les spécifications du présent paragraphe s'appliquent lorsque `slice_group_map_type` est égal à 0.

Le mappage d'unité de mappage à groupe de tranches est généré comme spécifié par:

```
i = 0
faire
    pour( iGroup = 0; iGroup <= num_slice_groups_minus1 && i < PicSizeInMapUnits;
        i += run_length_minus1[ iGroup++ ] + 1 )
        pour( j = 0; j <= run_length_minus1[ iGroup ] && i + j < PicSizeInMapUnits; j++ )
            mapUnitToSliceGroupMap[ i + j ] = iGroup
    tandis que( i < PicSizeInMapUnits )
```

(8-18)

8.2.2.2 Spécification pour le type de mappage de groupe de tranches dispersées

Les spécifications du présent paragraphe s'appliquent lorsque `slice_group_map_type` est égal à 1.

Le mappage d'unité de mappage à groupe de tranches est généré comme spécifié par:

```
pour( 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 Spécification pour le type de mappage de groupe de tranches de premier plan avec surplus

Les spécifications du présent paragraphe s'appliquent lorsque `slice_group_map_type` est égal à 2.

Le mappage d'unité de mappage à groupe de tranches est généré comme spécifié par:

```

pour( i = 0; i < PicSizeInMapUnits; i++ )
    mapUnitToSliceGroupMap[ i ] = num_slice_groups_minus1
pour( 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
    pour( y = yTopLeft; y <= yBottomRight; y++ )
        pour( x = xTopLeft; x <= xBottomRight; x++ )
            mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] = iGroup
}

```

(8-20)

Après application du processus spécifié dans l'équation 8-20, il doit y avoir au moins une valeur de i de 0 à $\text{PicSizeInMapUnits} - 1$ inclus, pour laquelle $\text{mapUnitToSliceGroupMap}[i]$ est égal à $iGroup$ pour chaque valeur de $iGroup$ de 0 à $\text{num_slice_groups_minus1}$ inclus (c'est-à-dire, chaque groupe de tranches doit contenir au moins une unité de mappage de groupe de tranches).

NOTE – Les rectangles peuvent se chevaucher. Le groupe de tranches 0 contient les macroblocs qui sont au sein du rectangle spécifié par $\text{top_left}[0]$ et $\text{bottom_right}[0]$. Un groupe de tranches ayant un ID de groupe de tranches supérieur à 0 et inférieur à $\text{num_slice_groups_minus1}$ contient les macroblocs qui sont au sein du rectangle spécifié pour ce groupe de tranches qui ne sont pas au sein du rectangle spécifié pour tout groupe de tranches ayant un ID de groupe de tranches plus petit. Le groupe de tranches avec un ID de groupe de tranches égal à $\text{num_slice_groups_minus1}$ contient les macroblocs qui ne sont pas dans les autres groupes de tranches.

8.2.2.4 Spécification pour les types de mappage de groupe de tranches en sortie de boîte

Les spécifications du présent paragraphe s'appliquent lorsque $\text{slice_group_map_type}$ est égal à 3.

Le mappage d'unité de mappage à groupe de tranches est généré comme spécifié par:

```

pour( 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 )
pour( k = 0; k < MapUnitsInSliceGroup0; k += mapUnitVacant ) {
    mapUnitVacant = ( mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] == 1 )
    si( mapUnitVacant )
        mapUnitToSliceGroupMap[ y * PicWidthInMbs + x ] = 0
    si( xDir == -1 && x == leftBound ) {
        leftBound = Max( leftBound - 1, 0 )
        x = leftBound
        ( xDir, yDir ) = ( 0, 2 * slice_group_change_direction_flag - 1 )
    } ou si( xDir == 1 && x == rightBound ) {
        rightBound = Min( rightBound + 1, PicWidthInMbs - 1 )
        x = rightBound
        ( xDir, yDir ) = ( 0, 1 - 2 * slice_group_change_direction_flag )
    } ou si( yDir == -1 && y == topBound ) {
        topBound = Max( topBound - 1, 0 )
        y = topBound
        ( xDir, yDir ) = ( 1 - 2 * slice_group_change_direction_flag, 0 )
    } ou si( yDir == 1 && y == bottomBound ) {
        bottomBound = Min( bottomBound + 1, PicHeightInMapUnits - 1 )
        y = bottomBound
        ( xDir, yDir ) = ( 2 * slice_group_change_direction_flag - 1, 0 )
    } ou
        ( x, y ) = ( x + xDir, y + yDir )
}
}

```

(8-21)

8.2.2.5 Spécification pour les types de mappage de groupe de tranches à balayage de trame

Les spécifications du présent paragraphe s'appliquent lorsque `slice_group_map_type` est égal à 4.

Le mappage d'unité de mappage à groupe de tranches est généré comme spécifié par:

```
pour( i = 0; i < PicSizeInMapUnits; i++ )
  si( i < sizeOfUpperLeftGroup )
    mapUnitToSliceGroupMap[ i ] = slice_group_change_direction_flag
  ou
    mapUnitToSliceGroupMap[ i ] = 1 - slice_group_change_direction_flag
```

 (8-22)

8.2.2.6 Spécification pour les types de mappage de groupe de tranches de balayage

Les spécifications du présent paragraphe s'appliquent lorsque `slice_group_map_type` est égal à 5.

Le mappage d'unité de mappage à groupe de tranches est généré comme spécifié par:

```
k = 0;
pour( j = 0; j < PicWidthInMbs; j++ )
  pour( i = 0; i < PicHeightInMapUnits; i++ )
    si( k++ < sizeOfUpperLeftGroup )
      mapUnitToSliceGroupMap[ i * PicWidthInMbs + j ] = slice_group_change_direction_flag
    ou
      mapUnitToSliceGroupMap[ i * PicWidthInMbs + j ] = 1 - slice_group_change_direction_flag
```

 (8-23)

8.2.2.7 Spécification pour type de mappage explicite de groupe de tranches

Les spécifications du présent paragraphe s'appliquent lorsque `slice_group_map_type` est égal à 6.

Le mappage d'unité de mappage à groupe de tranches est généré comme spécifié par:

```
mapUnitToSliceGroupMap[ i ] = slice_group_id[ i ]
```

 (8-24)

pour tout `i` dans la gamme de 0 à `PicSizeInMapUnits - 1` inclus.

8.2.2.8 Spécification pour conversion d'unité de mappage de groupe de tranches à macrobloc

Pour chaque valeur de `i` dans la gamme de 0 à `PicSizeInMbs - 1` inclus, le mappage de macrobloc à groupe de tranches est spécifié comme suit.

- Si `frame_mbs_only_flag` est égal à 1 ou `field_pic_flag` est égal à 1, le mappage de macrobloc à groupe de tranches est spécifié par:

```
MbToSliceGroupMap[ i ] = mapUnitToSliceGroupMap[ i ]
```

 (8-25)

- Autrement, si `MbaffFrameFlag` est égal à 1, le mappage de macrobloc à groupe de tranches est spécifié par:

```
MbToSliceGroupMap[ i ] = mapUnitToSliceGroupMap[ i / 2 ]
```

 (8-26)

- Autrement (`frame_mbs_only_flag` est égal à 0 et `mb_adaptive_frame_field_flag` est égal à 0 et `field_pic_flag` est égal à 0), le mappage de macrobloc à groupe de tranches est spécifié par:

```
MbToSliceGroupMap[ i ] = mapUnitToSliceGroupMap[ ( i / ( 2 * PicWidthInMbs ) ) * PicWidthInMbs
+ ( i % PicWidthInMbs ) ]
```

 (8-27)

8.2.3 Processus de décodage pour subdivision de données de tranche

Les entrées pour ce processus sont:

- une charge utile RBSP de couche de subdivision A de données de tranche;
- lorsque des éléments syntaxiques de catégorie 3 sont présents dans les données de tranche, une charge utile RBSP de couche de subdivision A de données de tranche ayant le même `slice_id` que dans la charge utile RBSP de couche de subdivision A de données de tranche; et

- lorsque des éléments syntaxiques de catégorie 4 sont présents dans les données de tranche, une charge utile RBSP de couche de subdivision C de données de tranche ayant le même `slice_id` que dans la charge utile RBSP de couche de subdivision A de données de tranche.

NOTE – La charge utile RBSP de couche de subdivision B de données de tranche et la charge utile RBSP de couche de subdivision C de données de tranche ne sont pas nécessairement présentes.

Le résultat de ce processus est une tranche codée.

Lorsque la subdivision de données de tranche n'est pas utilisée, les tranches codées sont représentées par une couche de tranche sans subdivision de charge utile RBSP contenant un en-tête de tranche suivi par une structure syntaxique de données de tranche qui contient tous les éléments syntaxiques des catégories 2, 3 et 4 (voir la colonne catégorie au § 7.3) des données de macrobloc pour les macroblocs de la tranche.

Lorsque la subdivision de données de tranche est utilisée, les données de macrobloc d'une tranche sont subdivisées en une, deux ou trois subdivisions contenues dans des unités NAL séparées. La subdivision A contient un en-tête de subdivision A de données de tranche et tous les éléments syntaxiques de catégorie 2. La subdivision B, lorsqu'elle est présente, contient un en-tête de subdivision B de données de tranche et tous les éléments syntaxiques de catégorie 3. La subdivision C, lorsqu'elle est présente, contient un en-tête de subdivision C de données de tranche et tous les éléments syntaxiques de catégorie 4.

Lorsque la subdivision de données de tranche est utilisée, les éléments syntaxiques de chaque catégorie sont analysés grammaticalement à partir d'une unité NAL séparée, qui ne doit pas nécessairement être présente lorsque n'existe aucun symbole de leurs catégories respectives. Le processus de décodage doit traiter les subdivisions de données de tranche d'une tranche codée d'une manière équivalente au traitement d'une couche de tranche correspondante sans subdivision de charge utile RBSP en extrayant chaque élément syntaxique de la subdivision de données de tranche dans lequel apparaît l'élément syntaxique en fonction de l'allocation de subdivision de données de tranche dans les tableaux syntaxiques du § 7.3.

NOTE – Les éléments syntaxiques de catégorie 3 sont pertinents pour le décodage des données résiduelles des types de macrobloc I et SI. Les éléments syntaxiques de catégorie 4 sont pertinents pour le décodage des données résiduelles des types de macrobloc P et B. La catégorie 2 englobe tous les autres éléments syntaxiques relatifs au décodage de macroblocs et leurs informations sont souvent mentionnées comme informations d'en-tête. L'en-tête de subdivision A de données de tranche contient tous les éléments syntaxiques de l'en-tête de tranche et en plus les `slice_id` qui sont utilisés pour associer les subdivisions B et C de données de tranche avec la subdivision A de données de tranche. Les en-têtes de subdivision B et C de données de tranche contiennent l'élément syntaxique `slice_id` qui établit leur association avec la subdivision de données de tranche A de la tranche.

8.2.4 Processus de décodage pour la construction des listes d'images de référence

Ce processus est invoqué au début du décodage de chaque tranche P, SP ou B.

Les résultats de ce processus sont une liste d'images de référence `RefPicList0` et, lors du décodage d'une tranche B, une seconde liste d'images de référence `RefPicList1`.

Les images décodées de référence sont marquées comme "utilisé pour référence à court terme" ou "utilisé pour référence à long terme" comme spécifié par le flux binaire et décrit au § 8.2.5. Les images décodées de référence à court terme sont identifiées par la valeur de `frame_num`. Un indice de trame de long terme est alloué aux images décodées de référence à long terme comme spécifié par le flux binaire et indiqué au § 8.2.5.

Le paragraphe 8.2.4.1 est invoqué pour spécifier:

- l'allocation des variables `FrameNum`, `FrameNumWrap`, et `PicNum` à chacune des images de référence de court terme;

- l'allocation de la variable LongTermPicNum à chacune des images de référence de long terme.

Les images de référence sont indexées par des indices de référence comme spécifié au § 8.4.2.1. Un indice de référence est un indice dans une liste de variables PicNum et LongTermPicNum, qui s'appelle une liste d'images de référence. Lors du décodage d'une tranche P ou SP, il y a une seule liste d'images de référence RefPicList0. Lors du décodage d'une tranche B, il y a une seconde liste indépendante d'image de référence RefPicList1 en plus de RefPicList0.

Au début du décodage de chaque tranche, la liste d'images de référence RefPicList0 et RefPicList1 pour les tranches B, est déduite comme suit.

- Une liste initiale d'image de référence RefPicList0 et RefPicList1 pour les tranches B, est déduite comme spécifié au § 8.2.4.2.
- La liste initiale d'image de référence RefPicList0 et RefPicList1 pour les tranches B, est modifiée comme spécifié au § 8.2.4.3.

Le nombre des entrées dans la liste modifiée d'image de référence RefPicList0 est $\text{num_ref_idx_l0_active_minus1} + 1$ et pour les tranches B le nombre des entrées dans la liste modifiée d'image de référence RefPicList1 est $\text{num_ref_idx_l1_active_minus1} + 1$. Une image de référence peut apparaître à plus d'un indice dans les listes modifiées d'image de référence RefPicList0 ou RefPicList1.

8.2.4.1 Processus de décodage pour les numéros d'image

Ce processus est invoqué lorsque le processus de décodage pour la construction des listes d'images de référence spécifié au § 8.2.4 ou le processus de marquage d'image de référence décodée spécifié au § 8.2.5 est invoqué.

Les variables FrameNum, FrameNumWrap, PicNum, LongTermFrameIdx et LongTermPicNum sont utilisées pour le processus d'initialisation pour les listes d'image de référence du § 8.2.4.2, le processus de modification pour les listes d'image de référence du § 8.2.4.3 et pour le processus de marquage d'image de référence décodée du § 8.2.5.

Les variables FrameNum et FrameNumWrap de chaque image de référence à court terme sont allouées comme suit. D'abord, FrameNum est mis égal à l'élément syntaxique frame_num qui a été décodé dans l'en-tête de tranche(s) de l'image de référence à court terme correspondante. Puis la variable FrameNumWrap est déduite comme

$$\begin{aligned} &\text{si}(\text{FrameNum} > \text{frame_num}) \\ &\quad \text{FrameNumWrap} = \text{FrameNum} - \text{MaxFrameNum} \\ &\text{ou} \\ &\quad \text{FrameNumWrap} = \text{FrameNum} \end{aligned} \tag{8-28}$$

où la valeur de frame_num utilisée dans l'équation 8-28 est le frame_num de l'en-tête de tranche(s) pour l'image en cours.

A chaque image de référence à long terme, la variable LongTermFrameIdx est allouée comme spécifié au § 8.2.5.

Une variable PicNum est allouée à chaque image de référence à court terme et une variable LongTermPicNum est allouée à chaque image de référence à long terme. Les valeurs de ces variables dépendent de la valeur de field_pic_flag et bottom_field_flag pour l'image en cours et elles sont établies comme suit.

- Si field_pic_flag est égal à 0, on applique ce qui suit.
 - Pour chaque trame de référence à court terme ou paire de champs de référence complémentaires:

$$\text{PicNum} = \text{FrameNumWrap} \tag{8-29}$$

- Pour chaque trame de référence à long terme ou paire de champs de référence complémentaires à long terme:

$$\text{LongTermPicNum} = \text{LongTermFrameIdx} \quad (8-30)$$

NOTE – Lors du décodage d'une trame, la valeur de MbaffFrameFlag n'a pas d'influence sur les déductions des § 8.2.4.2, 8.2.4.3 et 8.2.5.

- Autrement (`field_pic_flag` est égal à 1), on applique ce qui suit.

- Pour chaque champ de référence à court terme, on applique ce qui suit.

- Si le champ de référence a la même parité que le champ en cours

$$\text{PicNum} = 2 * \text{FrameNumWrap} + 1 \quad (8-31)$$

- Autrement (le champ de référence a la parité opposée à celle du champ en cours)

$$\text{PicNum} = 2 * \text{FrameNumWrap} \quad (8-32)$$

- Pour chaque champ de référence à long terme, on applique ce qui suit.

- Si le champ de référence a la même parité que le champ en cours

$$\text{LongTermPicNum} = 2 * \text{LongTermFrameIdx} + 1 \quad (8-33)$$

- Autrement (le champ de référence a la parité opposée à celle du champ en cours)

$$\text{LongTermPicNum} = 2 * \text{LongTermFrameIdx} \quad (8-34)$$

8.2.4.2 Processus d'initialisation pour les listes d'image de référence

Ce processus d'initialisation est invoqué lors du décodage d'un en-tête de tranche P, SP ou B.

Les résultats de ce processus sont la liste initiale d'image de référence `RefPicList0` et lors du décodage d'une tranche B, la liste initiale d'image de référence `RefPicList1`.

`RefPicList0` et `RefPicList1` ont en entrées initiales les variables `PicNum` et `LongTermPicNum` comme spécifié aux § 8.2.4.2.1 à 8.2.4.2.5.

Lorsque le nombre d'entrées dans les `RefPicList0` ou `RefPicList1` initiales produites comme spécifié aux § 8.2.4.2.1 à 8.2.4.2.5 est supérieur à respectivement, `num_ref_idx_l0_active_minus1 + 1` ou `num_ref_idx_l1_active_minus1 + 1`, les anciennes positions d'entrées `num_ref_idx_l0_active_minus1` ou `num_ref_idx_l1_active_minus1` sont détruites de la liste initiale d'image de référence.

Lorsque le nombre d'entrées dans la `RefPicList0` ou `RefPicList1` initiale produites comme spécifié aux § 8.2.4.2.1 à 8.2.4.2.5 est inférieur respectivement à `num_ref_idx_l0_active_minus1 + 1` ou `num_ref_idx_l1_active_minus1 + 1`, les entrées restantes dans la liste initiale d'image de référence sont mises égales à "pas d'image de référence".

8.2.4.2.1 Processus d'initialisation pour la liste d'images de référence pour les tranches P et SP en trames

Ce processus d'initialisation est invoqué lors du décodage d'une tranche P ou SP dans une trame codée.

Le résultat de ce processus est la liste initiale d'image de référence `RefPicList0`.

Lorsque ce processus est invoqué, il doit y avoir au moins une trame de référence ou une paire de champs de référence complémentaires qui soit actuellement marquée comme "utilisé pour référence à court terme" ou "utilisé pour référence à long terme".

La liste d'images de référence `RefPicList0` est ordonnée de telle sorte que les trames de référence de court terme et les paires de champs de référence complémentaires de court terme aient des indices inférieurs à ceux des trames de référence de long terme et des paires de champs de référence complémentaires de long terme.

Les trames de référence et les paires de champs de référence complémentaires de court terme sont ordonnées en commençant par la trame ou paire de champs complémentaires avec la plus forte valeur de PicNum et en procédant en ordre décroissant jusqu'à la trame ou paire de champs complémentaires avec la plus faible valeur de PicNum.

Les trames de référence et paires de champs de référence complémentaires de long terme sont ordonnées en commençant par la trame ou paire de champs complémentaires avec la plus faible valeur de LongTermPicNum value et en procédant en ordre croissant jusqu'à la trame ou paire de champs complémentaires avec la plus forte valeur de LongTermPicNum.

NOTE – Un champ de référence non apparié n'est pas utilisé pour l'interprétation pour le décodage d'une trame, indépendamment de la valeur de MbaffFrameFlag.

Par exemple, lorsque trois trames de références sont marquées comme "utilisé pour référence à court terme" avec PicNum égal à 300, 302 et 303 et que deux trames de référence sont marquées comme "utilisé pour référence à long terme" avec LongTermPicNum égal à 0 et 3, l'ordre initial des indices est:

- RefPicList0[0] est mis égal à l'image de référence à court terme pour PicNum = 303;
- RefPicList0[1] est mis égal à l'image de référence à court terme pour PicNum = 302;
- RefPicList0[2] est mis égal à l'image de référence à court terme pour PicNum = 300;
- RefPicList0[3] est mis égal à l'image de référence à long terme pour LongTermPicNum = 0;
- RefPicList0[4] est mis égal à l'image de référence à long terme pour LongTermPicNum = 3.

8.2.4.2.2 Processus d'initialisation pour la liste d'images de référence pour des tranches P et SP en champs

Ce processus d'initialisation est invoqué lors du décodage d'une tranche P ou SP dans un champ codé.

Le résultat de ce processus est la liste initiale d'images de référence RefPicList0.

Lors du décodage d'un champ, chaque champ inclus dans la liste d'images de référence a un indice séparé dans la liste.

NOTE – Lors du décodage d'un champ, il y a effectivement au moins deux fois autant d'images disponibles à référencer qu'il y en aurait lors du décodage d'une trame à la même position dans l'ordre de décodage.

Deux listes ordonnées de trames de référence, refFrameList0ShortTerm et refFrameList0LongTerm, sont déduites comme suit. Pour les besoins de la formation de cette liste de trames, les trames décodées, les paires de champs de référence complémentaires, les champs de référence non appariés et les trames de référence dans lesquels un seul champ est marqué "utilisé pour référence à court terme" ou "utilisé pour référence à long terme" sont tous considérés comme des trames de référence.

- Le FrameNumWrap de toutes les trames ayant un ou plusieurs champs marqués "utilisé pour référence à court terme" est inclus dans la liste de trames de référence à court terme refFrameList0ShortTerm. Lorsque le champ en cours est le second champ (dans l'ordre de décodage) d'une paire de champs de référence complémentaires et que le premier champ est marqué comme "utilisé pour référence à court terme", le FrameNumWrap du premier champ est inclus dans la liste refFrameList0ShortTerm. refFrameList0ShortTerm est ordonné en commençant par la trame avec la valeur FrameNumWrap la plus élevée et en procédant en ordre décroissant jusqu'à la trame avec la valeur FrameNumWrap la plus faible.

- Le LongTermFrameIdx de toutes les trames ayant un ou plusieurs champs marqués "utilisé pour référence à long terme" est inclus dans la liste des trames de référence à long terme refFrameList0LongTerm. Lorsque le champ en cours est le second champ (dans l'ordre de décodage) d'une paire de champs de référence complémentaires et que le premier champ est marqué comme "utilisé pour référence à long terme", le LongTermFrameIdx du premier champ est inclus dans la liste refFrameList0LongTerm. refFrameList0LongTerm est ordonnée en commençant par la trame avec la valeur de LongTermFrameIdx la plus faible et en procédant en ordre croissant jusqu'à la trame avec la valeur de LongTermFrameIdx la plus forte.

Le processus spécifié au § 8.2.4.2.5 est invoqué avec refFrameList0ShortTerm et refFrameList0LongTerm donnés en entrée et le résultat est alloué à RefPicList0.

8.2.4.2.3 Processus d'initialisation des listes d'images de référence pour les tranches B en trames

Ce processus d'initialisation est invoqué lors du décodage d'une tranche B dans une trame codée.

Les résultats de ce processus sont les listes initiales d'images de référence RefPicList0 et RefPicList1.

Lorsque ce processus est invoqué, il doit y avoir au moins une trame de référence ou une paire de champs de référence complémentaires qui soit actuellement marquée comme "utilisé pour référence à court terme" ou "utilisé pour référence à long terme".

Pour les tranches B, l'ordre des images de référence à court terme dans les listes d'images de référence RefPicList0 et RefPicList1 dépend de l'ordre de sortie, tel qu'il est donné par PicOrderCnt(). Lorsque pic_order_cnt_type est égal à 0, les images de référence qui sont marquées comme "non existant" comme spécifié au § 8.2.5.2 ne sont incluses ni dans RefPicList0 ni dans RefPicList1.

NOTE – Lorsque gaps_in_frame_num_value_allowed_flag est égal à 1, les codeurs devraient utiliser la réorganisation de la liste d'images de référence pour assurer le bon fonctionnement du processus de décodage (notamment lorsque pic_order_cnt_type est égal à 0, auquel cas PicOrderCnt() n'est pas déduit pour les trames "non existant").

La liste d'images de référence RefPicList0 est ordonnée de telle sorte que les trames de référence à court terme et les paires de champs de référence complémentaires à court terme aient des indices inférieurs à ceux des trames de référence à long terme et des paires de champs de référence complémentaires à long terme. Elle est ordonnée comme suit.

- Les trames de référence à court terme et les paires de champs de référence complémentaires à court terme sont ordonnées en commençant par la trame de référence à court terme ou paire de champs de référence complémentaire frm0 avec la valeur de PicOrderCnt(frm0) la plus forte qui soit inférieure à la valeur de PicOrderCnt(CurrPic) et en procédant en ordre décroissant jusqu'à la trame de référence à court terme ou paire de champs de référence complémentaire frm1 qui ait la plus faible valeur de PicOrderCnt(frm1) et ensuite en continuant avec la trame de référence à court terme ou paire de champs de référence complémentaires frm2 avec la plus faible valeur de PicOrderCnt(frm2) supérieure à la valeur de PicOrderCnt(CurrPic) de la trame en cours et en procédant en ordre croissant jusqu'à la trame de référence à court terme ou paire de champs de référence complémentaires frm3 qui a la plus forte valeur de PicOrderCnt(frm3).
- Les trames de référence à long terme et les paires de champs de référence complémentaires à long terme sont ordonnées en commençant par la trame de référence à long terme ou la paire de champs de référence complémentaires qui a la plus faible valeur de LongTermPicNum et en procédant en ordre croissant jusqu'à la trame de référence à long terme ou la paire de champs de référence complémentaires qui a la plus forte valeur de LongTermPicNum.

La liste d'images de la référence RefPicList1 est ordonnée de telle sorte que les trames de référence à court terme et les paires de champs de référence complémentaires à court terme aient des indices inférieurs à ceux des trames de référence à long terme et des paires de champs de référence complémentaires à long terme. Elle est ordonnée comme suit.

- Les trames de référence à court terme et les paires de champs de référence complémentaires à court terme sont ordonnées en commençant par la trame de référence à court terme ou paire de champs de référence complémentaires frm4 avec la valeur de PicOrderCnt(frm4) la plus faible supérieure à la valeur de PicOrderCnt(CurrPic) de la trame en cours et en procédant en ordre croissant jusqu'à la trame de référence à court terme ou paire de champs de référence complémentaires frm5 qui ait la plus forte valeur de PicOrderCnt(frm5) et ensuite en continuant avec la trame de référence à court terme ou paire de champs de référence complémentaire frm6 ayant la plus forte valeur de PicOrderCnt(frm6) inférieure à la valeur de PicOrderCnt(CurrPic) de la trame en cours et en procédant en ordre décroissant jusqu'à la trame de référence à court terme ou paire de champs de référence complémentaires frm7 qui ait la plus faible valeur de PicOrderCnt(frm7).
- Les trames de référence à long terme et les paires de champs de référence complémentaires à long terme sont ordonnées en commençant par la trame de référence ou paire de champs de référence complémentaires à long terme qui a la plus faible valeur de LongTermPicNum et en procédant en ordre croissant jusqu'à la trame de référence ou paire de champs de référence complémentaires à long terme qui a la plus forte valeur de LongTermPicNum.
- Lorsque la liste d'images de référence RefPicList1 a plus d'une entrée et que RefPicList1 est identique à la liste d'images de référence list RefPicList0, les deux premières entrées RefPicList1[0] et RefPicList1[1] sont interchangeables.

NOTE – Un champ de référence non apparié ne sera pas utilisé pour l'interprétation de trames indépendamment de la valeur de MbaffFrameFlag.

8.2.4.2.4 Processus d'initialisation des listes d'images de référence pour les tranches B en champs

Ce processus d'initialisation est invoqué lors du décodage d'une tranche B dans un champ codé.

Les résultats de ce processus sont les listes initiales d'images de référence RefPicList0 et RefPicList1.

Lors du décodage d'un champ, chaque champ d'une trame de référence mémorisée est identifié comme une image de référence séparée avec un indice unique. L'ordre des images de référence à court terme dans les listes d'image de référence RefPicList0 et RefPicList1 dépend de l'ordre de sortie, comme donné par PicOrderCnt(). Lorsque pic_order_cnt_type est égal à 0, les images de référence qui sont marquées comme "non existant" comme spécifié au § 8.2.5.2 ne sont incluses ni dans RefPicList0 ni dans RefPicList1.

NOTE – Lorsque gaps_in_frame_num_value_allowed_flag est égal à 1, les codeurs devraient utiliser la réorganisation de la liste d'images de référence pour assurer le bon fonctionnement du processus de décodage (notamment lorsque pic_order_cnt_type est égal à 0, auquel cas PicOrderCnt() n'est pas déduit pour les trames "non existant").

NOTE – Lors du décodage d'un champ, il y a effectivement au moins deux fois autant d'images disponibles pour référence qu'il en aurait lors du décodage d'une trame à la même position dans l'ordre de décodage.

Trois listes ordonnées de trames de référence, refFrameList0ShortTerm, refFrameList1ShortTerm et refFrameListLongTerm, sont déduites comme suit. Pour les besoins de la formation de ces listes de trames, le terme entrée de référence désigne dans ce qui suit les trames de référence décodées, les paires de champs de référence complémentaires ou les champs de référence non appariés. Lorsque pic_order_cnt_type est égal à 0, le terme entrée de référence ne désigne pas les trames qui sont marquées comme "non existant" comme spécifié au § 8.2.5.2.

- refFrameList0ShortTerm est ordonnée en commençant par l'entrée de référence f0 avec la plus forte valeur de PicOrderCnt(f0) inférieure ou égale à la valeur de PicOrderCnt(CurrPic) du champ en cours et en procédant en ordre décroissant jusqu'à l'entrée de référence à court terme f1 qui ait la plus faible valeur de PicOrderCnt(f1) et ensuite en continuant avec l'entrée de référence f2 ayant la plus faible valeur de PicOrderCnt(f2) supérieure à la valeur de PicOrderCnt(CurrPic) du champ en cours et en procédant en ordre croissant jusqu'à l'entrée de référence à court terme f3 qui ait la plus forte valeur de PicOrderCnt(f3).

NOTE – Lorsque le champ en cours suit dans l'ordre de décodage un champ codé fldPrev avec lequel ils forment ensemble une paire de champs de référence complémentaires, fldPrev est inclus dans la liste refFrameList0ShortTerm en utilisant PicOrderCnt(fldPrev) et on applique la méthode de classement décrite à la phrase précédente.
- refFrameList1ShortTerm est ordonnée en commençant par l'entrée de référence f4 ayant la plus faible valeur de PicOrderCnt(f4) supérieure à la valeur de PicOrderCnt(CurrPic) du champ en cours et en procédant en ordre croissant jusqu'à l'entrée de référence à court terme f5 qui ait la plus forte valeur de PicOrderCnt(f5) et ensuite en continuant avec l'entrée de référence f6 ayant la plus forte valeur de PicOrderCnt(f6) inférieure ou égale à la valeur de PicOrderCnt(CurrPic) du champ en cours et en procédant en ordre décroissant jusqu'à l'entrée de référence à court terme f7 qui ait la plus faible valeur de PicOrderCnt(f7).

NOTE – Lorsque le champ en cours suit dans l'ordre de décodage un champ codé fldPrev avec lequel ils forment ensemble une paire de champs de référence complémentaires, fldPrev est inclus dans la liste refFrameList1ShortTerm en utilisant PicOrderCnt(fldPrev) et on applique la méthode de classement décrite à la phrase précédente.
- refFrameListLongTerm est ordonnée en commençant par l'entrée de référence ayant la plus faible valeur de LongTermFrameIdx et en procédant en ordre croissant jusqu'à l'entrée de référence ayant la plus forte valeur de LongTermFrameIdx.

NOTE – Lorsque le champ complémentaire de l'image en cours est marqué "utilisé pour référence à long terme", il est inclus dans la liste refFrameListLongTerm. Une entrée de référence dans laquelle seul un champ est marqué comme "utilisé pour référence à long terme" est incluse dans la liste refFrameListLongTerm.

Le processus spécifié au § 8.2.4.2.5 est invoqué avec refFrameList0ShortTerm et refFrameListLongTerm est donné en entrée et le résultat est alloué à RefPicList0.

Le processus spécifié au § 8.2.4.2.5 est invoqué avec refFrameList1ShortTerm et refFrameListLongTerm est donné en entrée et le résultat est alloué à RefPicList1.

Lorsque la liste d'images de référence RefPicList1 a plus d'une entrée et que RefPicList1 est identique à la liste d'images de référence RefPicList0, les deux premières entrées RefPicList1[0] et RefPicList1[1] sont interchangées.

8.2.4.2.5 Processus d'initialisation pour les listes d'images de référence en champs

Les entrées de ce processus sont les listes de trame de référence refFrameListXShortTerm (X pouvant être 0 ou 1) et refFrameListLongTerm.

Le résultat de ce processus est la liste d'images de référence RefPicListX (qui peut être RefPicList0 ou RefPicList1).

La liste d'images de référence RefPicListX est une liste ordonnée de telle sorte que les champs de référence de court terme aient de plus faibles indices que les champs de référence à long terme. Etant donné les listes de trames de référence refFrameListXShortTerm et refFrameListLongTerm, elle est déduite comme suit.

- Les champs de références à court terme sont ordonnés en choisissant des champs de référence à partir de la liste ordonnée de trames refFrameListXShortTerm en alternant entre champs de parité différente, en commençant avec un champ qui a la même parité que le champ en cours lorsqu'il est présent. Lorsqu'un champ d'une trame de référence n'a pas été décodé ou n'est pas marqué comme "utilisé pour référence à court terme", le champ manquant est ignoré et à la place est inséré dans RefPicListX le prochain champ de référence enregistré disponible de la parité choisie tiré de la liste ordonnée de trames refFrameListXShortTerm. Lorsqu'il n'y a plus de champ de référence à court terme de l'autre parité dans la liste ordonnée de trames refFrameListXShortTerm, les prochains champs non encore indexés de la parité disponible sont insérés dans RefPicListX dans l'ordre dans lequel ils surviennent dans la liste ordonnée de trames refFrameListXShortTerm.
- Les champs de référence à long terme sont ordonnés en choisissant des champs de référence à partir de la liste ordonnée de trames refFrameListLongTerm en alternant entre champs de parité différente, en commençant par un champ qui a la même parité que le champ en cours lorsqu'il est présent. Lorsqu'un champ d'une trame de référence n'a pas été décodé ou n'est pas marqué comme "utilisé pour référence à long terme", le champ manquant est ignoré et à la place est inséré dans RefPicListX le prochain champ de référence enregistré disponible de la parité choisie tiré de la liste ordonnée de trames refFrameListLongTerm. Lorsqu'il n'y a plus de champ de référence à long terme de l'autre parité dans la liste ordonnée de trames refFrameListLongTerm, les prochains champs non encore indexés de la parité disponible sont insérés dans RefPicListX dans l'ordre dans lequel ils surviennent dans la liste ordonnée de trames refFrameListLongTerm.

8.2.4.3 Processus de remise en ordre des listes d'images de référence

L'entrée pour ce processus est la liste d'images de référence RefPicList0 et, lors du décodage d'une tranche B, aussi la liste d'images de référence RefPicList1.

Les résultats de ce processus sont une liste d'images de référence list RefPicList0 qui peut être modifiée et aussi, lors du décodage d'une tranche B, une possibilité de liste d'images de référence RefPicList1 modifiée.

Lorsque ref_pic_list_reordering_flag_l0 est égal à 1, on applique ce qui suit.

- Soit refIdxL0 un indice dans la liste d'images de référence RefPicList0. Il est au départ mis égal à 0.
- Les éléments syntaxiques reordering_of_pic_nums_idc correspondants sont traités dans l'ordre dans lequel ils surviennent dans le flux binaire. Pour chacun de ces éléments syntaxiques, on applique ce qui suit.
 - Si reordering_of_pic_nums_idc est égal à 0 ou à 1, le processus spécifié au § 8.2.4.3.1 est invoqué avec RefPicList0 et refIdxL0 est donné en entrée et le résultat est alloué à RefPicList0 et refIdxL0.
 - Autrement, si reordering_of_pic_nums_idc est égal à 2, le processus spécifié au § 8.2.4.3.2 est invoqué avec RefPicList0 et refIdxL0 est donné en entrée et le résultat est alloué à RefPicList0 et refIdxL0.
 - Autrement (reordering_of_pic_nums_idc est égal à 3), le processus de remise en ordre de la liste d'images de référence RefPicList0 est terminé.

Lorsque ref_pic_list_reordering_flag_l1 est égal à 1, on applique ce qui suit.

- Soit refIdxL1 un indice dans la liste d'images de référence RefPicList1. Il est au départ mis égal à 0.

- Les éléments syntaxiques `reordering_of_pic_nums_idc` correspondants sont traités dans l'ordre dans lequel ils surviennent dans le flux binaire. Pour chacun de ces éléments syntaxiques, on applique ce qui suit.
 - Si `reordering_of_pic_nums_idc` est égal à 0 ou à 1, le processus spécifié au § 8.2.4.3.1 est invoqué avec `RefPicList1` et `refIdxL1` est donné en entrée et le résultat est alloué à `RefPicList1` et `refIdxL1`.
 - Autrement, si `reordering_of_pic_nums_idc` est égal à 2, le processus spécifié au § 8.2.4.3.2 est invoqué avec `RefPicList1` et `refIdxL1` est donné en entrée et le résultat est alloué à `RefPicList1` et `refIdxL1`.
 - Autrement (`reordering_of_pic_nums_idc` est égal à 3), le processus de remise en ordre de la liste d'images de référence `RefPicList1` est terminé.

8.2.4.3.1 Processus de remise en ordre des listes d'images de référence pour images à court terme

Les entrées pour ce processus sont la liste d'images de référence `RefPicListX` (X étant 0 ou 1) et un indice `refIdxLX` dans cette liste.

Les résultats de ce processus sont une liste d'images de référence qui peut être modifiée `RefPicListX` (X étant 0 ou 1) et l'indice `refIdxLX` incrémenté.

La variable `picNumLXNoWrap` est déduite comme suit:

- Si `reordering_of_pic_nums_idc` est égal à 0
 - si($\text{picNumLXPred} - (\text{abs_diff_pic_num_minus1} + 1) < 0$)
 $\text{picNumLXNoWrap} = \text{picNumLXPred} - (\text{abs_diff_pic_num_minus1} + 1) + \text{MaxPicNum}$ (8-35)
 - ou
 $\text{picNumLXNoWrap} = \text{picNumLXPred} - (\text{abs_diff_pic_num_minus1} + 1)$

- Autrement (`reordering_of_pic_nums_idc` est égal à 1)
 - si($\text{picNumLXPred} + (\text{abs_diff_pic_num_minus1} + 1) \geq \text{MaxPicNum}$)
 $\text{picNumLXNoWrap} = \text{picNumLXPred} + (\text{abs_diff_pic_num_minus1} + 1) - \text{MaxPicNum}$ (8-36)
 - ou
 $\text{picNumLXNoWrap} = \text{picNumLXPred} + (\text{abs_diff_pic_num_minus1} + 1)$

`picNumLXPred` est la valeur de prédiction pour la variable `picNumLXNoWrap`. Lorsque le processus spécifié au présent paragraphe est invoqué la première fois pour une tranche (c'est-à-dire, pour la première occurrence de `reordering_of_pic_nums_idc` égal à 0 ou 1 dans la syntaxe `ref_pic_list_reordering()`), `picNumL0Pred` et `picNumL1Pred` sont au départ mis égaux à `CurrPicNum`. Après chaque allocation de `picNumLXNoWrap`, la valeur de `picNumLXNoWrap` est allouée à `picNumLXPred`.

La variable `picNumLX` est déduite comme suit

$$\begin{aligned} &\text{si}(\text{picNumLXNoWrap} > \text{CurrPicNum}) \\ &\quad \text{picNumLX} = \text{picNumLXNoWrap} - \text{MaxPicNum} \quad (8-37) \\ &\text{ou} \\ &\quad \text{picNumLX} = \text{picNumLXNoWrap} \end{aligned}$$

`picNumLX` doit être égal au `PicNum` d'une image de référence qui est marquée comme "utilisé pour référence à court terme" et doit être différent du `PicNum` d'une image de référence à court terme qui est marquée comme "non existant".

La procédure suivante doit être respectée pour placer l'image avec le numéro d'image à court terme `picNumLX` dans la position d'indice `refIdxLX`, déplacer la position de toutes autres images restantes plus loin dans la liste et incrémenter la valeur de `refIdxLX`.

```
pour( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx-- )
  RefPicListX[ cIdx ] = RefPicListX[ cIdx - 1 ]
RefPicListX[ refIdxLX++ ] = image de référence à court terme avec PicNum égal à picNumLX
```

```

nIdx = refIdxLX
pour( cIdx = refIdxLX; cIdx <= num_ref_idx_IX_active_minus1 + 1; cIdx++ )
    si( PicNumF( RefPicListX[ cIdx ] ) != picNumLX )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]

```

(8-38)

Où la fonction PicNumF(RefPicListX[cIdx]) est déduite comme suit:

- si l'image RefPicListX[cIdx] est marquée comme "utilisé pour référence à court terme", PicNumF(RefPicListX[cIdx]) est le PicNum de l'image RefPicListX[cIdx];
- autrement (l'image RefPicListX[cIdx] n'est pas marquée comme "utilisé pour référence à court terme"), PicNumF(RefPicListX[cIdx]) est égal à MaxPicNum.

NOTE – Une valeur de MaxPicNum ne peut jamais être égale à picNumLX.

NOTE – Au sein de cette procédure de pseudocode, la longueur de la liste RefPicListX est temporairement allongée d'un élément par rapport à la longueur nécessaire pour la liste finale. Après l'exécution de cette procédure, seuls les éléments de 0 à num_ref_idx_IX_active_minus1 de la liste doivent être retenus.

8.2.4.3.2 Processus de remise en ordre des listes d'images de référence pour les images à long terme

Les entrées pour ce processus sont la liste d'images de référence RefPicListX (X étant 0 ou 1) et un indice refIdxLX dans cette liste.

Les résultats de ce processus sont une liste d'images de référence RefPicListX (X étant 0 ou 1) qui peut être modifiée et l'indice incrémenté refIdxLX.

La procédure suivante doit être suivie pour placer l'image avec le numéro d'image à long terme long_term_pic_num dans la position d'indice refIdxLX, déplacer la position de toutes autres images restantes plus loin dans la liste et incrémenter la valeur de refIdxLX.

```

pour( cIdx = num_ref_idx_IX_active_minus1 + 1; cIdx > refIdxLX; cIdx-- )
    RefPicListX[ cIdx ] = RefPicListX[ cIdx - 1 ]
RefPicListX[ refIdxLX++ ] = image de référence à court terme avec LongTermPicNum égal à
long_term_pic_num
nIdx = refIdxLX
pour( cIdx = refIdxLX; cIdx <= num_ref_idx_IX_active_minus1 + 1; cIdx++ )
    si( LongTermPicNumF( RefPicListX[ cIdx ] ) != long_term_pic_num )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]

```

(8-39)

Où la fonction LongTermPicNumF(RefPicListX[cIdx]) est déduite comme suit:

- Si l'image RefPicListX[cIdx] est marquée comme "utilisé pour référence à long terme", LongTermPicNumF(RefPicListX[cIdx]) est le LongTermPicNum de l'image RefPicListX[cIdx].
- Autrement (l'image RefPicListX[cIdx] n'est pas marquée comme "utilisé pour référence à long terme"), LongTermPicNumF(RefPicListX[cIdx]) est égal à $2 * (\text{MaxLongTermFrameIdx} + 1)$.

NOTE – Une valeur de $2 * (\text{MaxLongTermFrameIdx} + 1)$ ne peut jamais être égale à long_term_pic_num.

NOTE – Au sein de cette procédure de pseudo-code, la longueur de la liste RefPicListX est temporairement allongée d'un élément par rapport à la longueur nécessaire pour la liste finale. Après l'exécution de cette procédure, seuls les éléments de 0 à num_ref_idx_IX_active_minus1 de la liste doivent être retenus.

8.2.5 Processus de marquage d'image de référence décodée

Ce processus est invoqué pour des images décodées lorsque nal_ref_idc n'est pas égal à 0.

NOTE – Un processus (le processus de décodage pour les intervalles dans frame_num spécifié au § 8.2.5.2) défini dans le présent paragraphe peut aussi être invoqué lorsque nal_ref_idc est égal à 0, comme spécifié au § 8.

Une image décodée avec `nal_ref_idc` différent de 0, mentionnée comme image de référence, est marquée comme "utilisé pour référence à court terme" ou "utilisé pour référence à long terme". Pour une trame de référence décodée, ses deux champs sont marqués de la même façon qu'une trame. Pour une paire de champs de référence complémentaires, la paire est marquée de la même façon que ses deux champs. Une image qui est marquée comme "utilisé pour référence à court terme" est identifiée par son `FrameNum` et, lorsque c'est un champ, par sa parité. Une image qui est marquée comme "utilisé pour référence à long terme" est identifiée par son `LongTermFrameIdx` et, lorsque c'est un champ, par sa parité.

Les trames ou paires de champs complémentaires marquées comme "utilisé pour référence à court terme" ou comme "utilisé pour référence à long terme" peuvent être utilisées comme référence pour l'interprédiction lors du décodage d'une trame jusqu'à ce que la trame, la paire de champs complémentaires ou un de ses champs constituants soit marqué comme "non utilisé comme référence". Un champ marqué comme "utilisé pour référence à court terme" ou comme "utilisé pour référence à long terme" peut être utilisé comme référence pour l'interprédiction lors du décodage d'un champ jusqu'à ce qu'il soit marqué comme "non utilisé comme référence".

Une image peut être marquée comme "non utilisé comme référence" par le processus de marquage d'image de référence en fenêtre glissante, mécanisme de "premier entré, premier sorti" spécifié au § 8.2.5.3 ou par le processus de marquage d'image de référence à commande adaptative de mémoire, opération de marquage adaptatif personnalisée spécifiée au § 8.2.5.4.

Une image de référence à court terme est identifiée pour être utilisée dans le processus de décodage par ses variables `FramNum` et `FrameNumWrap` et par son numéro d'image `PicNum` et une image de référence à long terme est identifiée pour être utilisée dans le processus de décodage par son numéro d'image à long terme `LongTermPicNum`. Lorsque l'image en cours n'est pas une image IDR, le § 8.2.4.1 est invoqué pour spécifier l'allocation des variables `FrameNum`, `FrameNumWrap`, `PicNum` et `LongTermPicNum`.

8.2.5.1 Séquence des opérations pour le processus de marquage d'image de référence décodée

Le marquage d'image de référence décodée s'effectue suivant les étapes ordonnées ci-après.

1. Lorsque `frame_num` de l'image en cours n'est pas égal à `PrevRefFrameNum` et n'est pas égal à $(\text{PrevRefFrameNum} + 1) \% \text{MaxFrameNum}$, le processus de décodage pour les intervalles dans `frame_num` est effectué conformément au § 8.2.5.2.
2. Toutes les tranches de l'image en cours sont décodées.
3. Selon que l'image en cours est ou non une image IDR, on applique ce qui suit.
 - Si l'image en cours est une image IDR, on applique ce qui suit.
 - Toutes les images de référence doivent être marquées comme "non utilisé comme référence".
 - En fonction de `long_term_reference_flag`, on applique ce qui suit.
 - Si `long_term_reference_flag` est égal à 0, l'image IDR doit être marquée comme "utilisé pour référence à court terme" et `MaxLongTermFrameIdx` doit être mis égal à "pas d'indice de trame à long terme".
 - Autrement (`long_term_reference_flag` est égal à 1), l'image IDR doit être marquée comme "utilisé pour référence à long terme", l'indice `LongTermFrameIdx` pour l'image IDR doit être mis égal à 0 et `MaxLongTermFrameIdx` doit être mis égal à 0.
 - Autrement (l'image en cours n'est pas une image IDR), on applique ce qui suit.
 - Si `adaptive_ref_pic_marking_mode_flag` est égal à 0, le processus spécifié au § 8.2.5.3 est invoqué.

- Autrement (`adaptive_ref_pic_marking_mode_flag` est égal à 1), le processus spécifié au § 8.2.5.4 est invoqué.

4. Lorsque l'image en cours n'est pas une image IDR et qu'elle n'était pas marquée comme "utilisé pour référence à long terme" par `memory_management_control_operation` égal à 6, elle est marquée comme "utilisé pour référence à court terme".

Après le marquage de l'image de référence décodée en cours, le nombre total de trames avec au moins un champ marqué comme "utilisé pour référence", plus le nombre de paires de champs complémentaires avec au moins un champ marqué comme "utilisé pour référence", plus le nombre de champs non appariés marqués comme "utilisé pour référence", ne doit pas être supérieur à `num_ref_frames`.

8.2.5.2 Processus de décodage pour les intervalles dans `frame_num`

Ce processus est invoqué lorsque `frame_num` n'est pas égal à `PrevRefFrameNum` et n'est pas égal à $(\text{PrevRefFrameNum} + 1) \% \text{MaxFrameNum}$.

NOTE – Bien que ce processus soit spécifié dans un paragraphe du § 8.2.5 (qui définit un processus invoqué uniquement lorsque `nal_ref_idc` est différent de 0), ce processus peut aussi être invoqué lorsque `nal_ref_idc` est égal à 0 (comme spécifié à l'article 8). Les raisons de la présence de ce paragraphe dans la structure de la présente Recommandation | Norme internationale sont historiques.

NOTE – Ce processus ne peut être invoqué que pour un flux binaire conforme lorsque `gaps_in_frame_num_value_allowed_flag` est égal à 1. Lorsque `gaps_in_frame_num_value_allowed_flag` est égal à 0 et `frame_num` n'est pas égal à `PrevRefFrameNum` et n'est pas égal à $(\text{PrevRefFrameNum} + 1) \% \text{MaxFrameNum}$, le processus de décodage devrait en déduire une perte d'images non intentionnelle.

Lorsque ce processus est invoqué, un ensemble de valeurs de `frame_num` appartenant à des images "non existant" est déduit comme étant toutes les valeurs prises par `UnusedShortTermFrameNum` dans l'équation 7-10 excepté la valeur de `frame_num` pour l'image en cours.

Le processus de décodage doit générer et marquer une trame pour chacune des valeurs de `frame_num` appartenant aux images "non existant", dans l'ordre selon lequel les valeurs de `UnusedShortTermFrameNum` sont générées par l'équation 7-10, en utilisant le processus de marquage d'image à "fenêtre glissante" comme spécifié au § 8.2.5.3. Les trames générées doivent aussi être marquées comme "non existant" et "utilisé pour référence à court terme". Les valeurs d'échantillon des trames générées peuvent être mises à n'importe quelle valeur. Ces trames générées qui sont marquées comme "non existant" ne doivent pas être mentionnées dans le processus d'interprédiction, dans les commandes de remise en ordre pour les listes d'images de référence pour les images à court terme (§ 8.2.4.3.1) et dans le processus d'allocation d'un indice `LongTermFrameIdx` à une image à court terme (§ 8.2.5.4.3).

Lorsque `pic_order_cnt_type` est différent de 0, `TopFieldOrderCnt` et `BottomFieldOrderCnt` sont calculés pour chacune des trames "non existant" en invoquant le processus de décodage pour le compte d'ordre d'image visé au § 8.2.1. En cas d'invocation du processus visé au § 8.2.1 pour une trame "non existant" donnée, l'image en cours est considérée comme étant une image dont `frame_num` est supposé être égal à `UnusedShortTermFrameNum`, `nal_ref_idc` est supposé être égal à 0, `nal_unit_type` est supposé être égal à 5, `field_pic_flag` est supposé être égal à 0, `adaptive_ref_pic_marking_mode_flag` est supposé être égal à 0, `delta_pic_order_cnt[0]` (si nécessaire) est supposé être égal à 0 et `delta_pic_order_cnt[1]` (si nécessaire) est supposé être égal à 0.

NOTE – Le processus de décodage devrait conclure à une perte d'image non intentionnelle lorsque toute valeur de `frame_num` appartenant à une image "non existant" est mentionnée dans le processus d'interprédiction, dans les commandes de remise en ordre des listes d'images de référence pour les images à court terme (§ 8.2.4.3.1) ou dans le processus d'allocation d'un indice `LongTermFrameIdx` à une image à court terme (§ 8.2.5.4.3). Le processus de décodage ne devrait pas conclure à une perte d'image non intentionnelle lorsqu'une opération de commande de gestion de mémoire non égale à 3 est appliquée à une trame marquée comme "non existant".

8.2.5.3 Processus de marquage d'image de référence décodée en fenêtre glissante

Ce processus est invoqué lorsque `adaptive_ref_pic_marking_mode_flag` est égal à 0.

En fonction des propriétés de l'image en cours spécifiée ci-dessous, on applique ce qui suit.

- Si l'image en cours est un champ codé qui est le second champ dans l'ordre de décodage d'une paire de champs de référence complémentaires et que le premier champ a été marqué comme "utilisé pour référence à court terme", l'image en cours est aussi marquée comme "utilisé pour référence à court terme".
- Autrement, on applique ce qui suit.
 - Soit `numShortTerm` le nombre total de trames de référence, de paires de champs de référence complémentaires et de champs de référence non appariés pour lesquels au moins un champ est marqué comme "utilisé pour référence à court terme". Soit `numLongTerm` le nombre total de trames de référence, de paires de champs de référence complémentaires et de champs de référence non appariés pour lesquels au moins un champ est marqué comme "utilisé pour référence à long terme".
 - Lorsque `numShortTerm + numLongTerm` est égal à `Max(num_ref_frames, 1)`, la condition que `numShortTerm` soit supérieur à 0 doit être remplie et la trame de référence à court terme, la paire de champs de référence complémentaires ou le champ de référence non apparié qui a la plus faible valeur de `FrameNumWrap` est marqué comme "non utilisé comme référence". Lorsque c'est une trame ou une paire de champs complémentaires, ses deux champs sont aussi marqués comme "non utilisé comme référence".

8.2.5.4 Processus de marquage d'image de référence décodée en commande de mémoire adaptative

Ce processus est invoqué lorsque `adaptive_ref_pic_marking_mode_flag` est égal à 1.

Les commandes `memory_management_control_operation` avec des valeurs de 1 à 6 sont traitées dans l'ordre dans lequel elles surviennent dans le flux binaire après que l'image en cours a été décodée. Pour chaque commande de ces `memory_management_control_operation`, un des processus spécifiés aux § 8.2.5.4.1 à 8.2.5.4.6 est invoqué en fonction de la valeur de `memory_management_control_operation`. La commande `memory_management_control_operation` avec la valeur 0 spécifie la fin des commandes `memory_management_control_operation`.

Les opérations de commande de gestion de mémoire sont appliquées comme suit aux images.

- Si `field_pic_flag` est égal à 0, les commandes `memory_management_control_operation` sont appliquées aux trames ou paires de champs de référence complémentaires spécifiées.
- Autrement (`field_pic_flag` est égal à 1), les commandes `memory_management_control_operation` sont appliquées aux champs de référence individuels spécifiés.

8.2.5.4.1 Processus de marquage d'image à court terme comme "non utilisé comme référence"

Ce processus est invoqué lorsque `memory_management_control_operation` est égal à 1.

Soit `picNumX` spécifié par

$$\text{picNumX} = \text{CurrPicNum} - (\text{difference_of_pic_nums_minus1} + 1) \quad (8-40)$$

En fonction de `field_pic_flag` la valeur de `picNumX` est utilisée pour marquer une image à court terme comme "utilisé comme référence" comme suit.

- Si `field_pic_flag` est égal à 0, la trame de référence à court terme ou la paire de champs de référence complémentaires à court terme spécifiée par `picNumX` et ses deux champs sont marqués comme "non utilisé comme référence".
- Autrement (`field_pic_flag` est égal à 1), le champ de référence à court terme spécifié par `picNumX` est marqué comme "non utilisé comme référence". Lorsque ce champ de référence fait partie d'une trame de référence ou paire de champs de référence complémentaires, la trame ou paire de champs complémentaires est aussi marquée comme "non utilisé comme référence", mais le marquage de l'autre champ n'est pas changé.

8.2.5.4.2 Processus de marquage d'une image à long terme comme "non utilisé comme référence"

Ce processus est invoqué lorsque `memory_management_control_operation` est égal à 2.

En fonction de `field_pic_flag` la valeur de `LongTermPicNum` est utilisée pour marquer une image à long terme comme "non utilisé comme référence" comme suit.

- Si `field_pic_flag` est égal à 0, la trame de référence à long terme ou la paire de champs de référence complémentaires à long terme ayant `LongTermPicNum` égal à `long_term_pic_num` et ses deux champs sont marqués comme "non utilisé comme référence".
- Autrement (`field_pic_flag` est égal à 1), le champ de référence à long terme spécifié par `LongTermPicNum` égal à `long_term_pic_num` est marqué comme "non utilisé comme référence". Lorsque ce champ de référence fait partie d'une trame de référence ou d'une paire de champs de référence complémentaires, la trame ou paire de champs complémentaires est aussi marquée comme "non utilisé comme référence", mais le marquage de l'autre champ n'est pas changé.

8.2.5.4.3 Processus d'allocation d'un `LongTermFrameIdx` à une image de référence à court terme

Ce processus est invoqué lorsque `memory_management_control_operation` est égal à 3.

Etant donné l'élément syntaxique `difference_of_pic_nums_minus1`, la variable `picNumX` est obtenue comme spécifié au § 8.2.5.4.1. `picNumX` doit se référer à une trame ou paire de champs de référence complémentaires ou champ de référence non apparié marqué comme "utilisé pour référence à court terme" et non marqué comme "non existant".

Lorsque `LongTermFrameIdx` égal à `long_term_frame_idx` est déjà alloué à une trame de référence à long terme ou une paire de champs de référence complémentaires à long terme, cette trame ou paire de champs complémentaires et ses deux champs sont marqués comme "non utilisé comme référence". Lorsque `LongTermFrameIdx` est déjà alloué à un champ de référence non apparié et que le champ n'est pas le champ complémentaire de l'image spécifiée par `picNumX`, ce champ est marqué comme "non utilisé comme référence".

En fonction de `field_pic_flag` la valeur de `LongTermFrameIdx` est utilisée pour marquer une image de "utilisé pour référence à court terme" à "utilisé pour référence à long terme" comme suit.

- Si `field_pic_flag` est égal à 0, le marquage de la trame de référence à court terme ou paire de champs de référence complémentaires à court terme spécifiée par `picNumX` et ses deux champs est changé de "utilisé pour référence à court terme" à "utilisé pour référence à long terme" et le `LongTermFrameIdx` est alloué égal à `long_term_frame_idx`.
- Autrement (`field_pic_flag` est égal à 1), le marquage du champ de référence à court terme spécifié par `picNumX` est changé de "utilisé pour référence à court terme" en "utilisé pour référence à long terme" et le `LongTermFrameIdx` est alloué égal à `long_term_frame_idx`.

8.2.5.4.4 Processus de décodage pour MaxLongTermFrameIdx

Ce processus est invoqué lorsque `memory_management_control_operation` est égal à 4.

Toutes images pour lesquelles `LongTermFrameIdx` est supérieur à `max_long_term_frame_idx_plus1 - 1` et qui sont marquées comme "utilisé pour référence à long terme" doivent être marquées comme "non utilisé comme référence".

La variable `MaxLongTermFrameIdx` est déduite comme suit.

- Si `max_long_term_frame_idx_plus1` est égal à 0, `MaxLongTermFrameIdx` doit être mis égal à "pas d'indice de trame à long terme".
- Autrement (`max_long_term_frame_idx_plus1` est supérieur à 0), `MaxLongTermFrameIdx` doit être mis égal à `max_long_term_frame_idx_plus1 - 1`.

NOTE – La commande `memory_management_control_operation` égale à 4 peut être utilisée pour marquer des images de référence à long terme comme "non utilisé comme référence". La fréquence de transmission `max_long_term_frame_idx_plus1` n'est pas spécifiée par la présente Recommandation | Norme internationale. Cependant, le codeur ne devrait pas envoyer de commande `memory_management_control_operation` égale à 4 à réception d'un message d'erreur, tel qu'un message de demande de rafraîchissement interne.

8.2.5.4.5 Processus de marquage de toutes les images de référence comme "non utilisé comme référence" et mise de MaxLongTermFrameIdx à "pas d'indice de trame à long terme"

Ce processus est invoqué lorsque `memory_management_control_operation` est égal à 5.

Toutes images de référence sont marquées comme "non utilisé comme référence" et la variable `MaxLongTermFrameIdx` est mise égale à "pas d'indice de trame à long terme".

8.2.5.4.6 Processus pour allouer un indice de trame à long terme à l'image en cours

Ce processus est invoqué lorsque `memory_management_control_operation` est égal à 6.

Lorsqu'une variable `LongTermFrameIdx` égale à `long_term_frame_idx` est déjà allouée à une trame de référence à long terme ou à une paire de champs de référence complémentaires à long terme, cette trame ou paire de champs complémentaires et ses deux champs sont marqués comme "non utilisé comme référence". Lorsque `LongTermFrameIdx` est déjà alloué à un champ de référence non apparié et que le champ n'est pas le champ complémentaire de l'image en cours, ce champ est marqué comme "non utilisé comme référence".

L'image en cours est marquée comme "utilisé pour référence à long terme" et `LongTermFrameIdx` est alloué égal à `long_term_frame_idx`.

Lorsque `field_pic_flag` est égal à 0, ses deux champs sont aussi marqués comme "utilisé pour référence à long terme" et `LongTermFrameIdx` est alloué égal à `long_term_frame_idx`.

Lorsque `field_pic_flag` est égal à 1 et que l'image en cours est un second (dans l'ordre de décodage) champ d'une paire de champs de référence complémentaires, la paire est aussi marquée comme "utilisé pour référence à long terme" et `LongTermFrameIdx` est alloué égal à `long_term_frame_idx`.

Après le marquage de l'image de référence décodée en cours, le nombre total de trames avec au moins un champ marqué comme "utilisé pour référence", plus le nombre de paires de champs complémentaires avec au moins un champ marqué comme "utilisé pour référence", plus le nombre de champs non appariés marqués comme "utilisé pour référence", ne doit pas être supérieur à `Max(num_ref_frames, 1)`.

NOTE – Dans certaines circonstances, l'énoncé ci-dessus peut imposer une contrainte relative à l'ordre dans lequel un élément syntaxique `memory_management_control_operation` égal à 6 peut figurer dans la syntaxe de marquage d'image de référence décodée par rapport à un élément syntaxique `memory_management_control_operation` égal à 1, 2 ou 4.

8.3 Processus d'intraprédiction

Ce processus est invoqué pour les types de macrobloc I et SI.

Les entrées pour ce processus sont des échantillons construits antérieurement au processus de filtrage de démontage de blocs à partir des macroblocs du voisinage et pour le mode de prédiction Intra_4x4, les valeurs associées de Intra4x4PredMode à partir des macroblocs du voisinage.

Les résultats de ce processus sont spécifiés comme suit.

- Si mb_type n'est pas égal à I_PCM, les échantillons d'intraprédiction des composants du macrobloc ou dans le cas du processus de prédiction Intra_4x4 pour des échantillons luma, les résultats sont des matrices d'échantillons luma 4x4 en tant que parties de la matrice 16x16 d'échantillons de prédiction du macrobloc.
- Autrement (mb_type est égal à I_PCM), les échantillons de macrobloc construits antérieurement au processus de filtrage de démontage de blocs.

La variable MvCnt est mise égale à 0.

En fonction de la valeur de mb_type on applique ce qui suit.

- Si mb_type est égal à I_PCM, le processus spécifié au § 8.3.4 est invoqué.
- Autrement (mb_type n'est pas égal à I_PCM), on applique ce qui suit.
 - Les processus de décodage pour les modes d'intraprédiction sont décrits comme suit pour le composant luma.
 - Si le mode de prédiction de macrobloc est égal à Intra_4x4, la spécification du § 8.3.1 s'applique.
 - Autrement (le mode de prédiction de macrobloc est égal à Intra_16x16), la spécification du § 8.3.2 s'applique.
 - Les processus de décodage pour les modes d'intraprédiction pour les composants chroma sont décrits au § 8.3.3.

Les échantillons utilisés dans le processus d'intraprédiction doivent être des valeurs d'échantillon antérieures à l'altération par toute opération de filtrage de démontage de blocs.

8.3.1 Processus de prédiction Intra_4x4 pour les échantillons luma

Ce processus est invoqué lorsque le mode de prédiction de macrobloc est égal à Intra_4x4.

Les entrées pour ce processus sont des échantillons luma construits antérieurement au processus de filtrage de démontage de blocs à partir des macroblocs du voisinage et des valeurs associées de Intra4x4PredMode à partir des macroblocs du voisinage ou des paires de macroblocs.

Les résultats de ce processus sont les matrices d'échantillons luma 4x4 en tant que parties de la matrice luma 16x16 d'échantillons de prédiction du macrobloc $pred_L$.

Le composant luma d'un macrobloc consiste en 16 blocs d'échantillons luma 4x4. Ces blocs subissent un balayage inverse en utilisant le processus de balayage inverse de bloc luma 4x4 comme spécifié au § 6.4.3.

Pour tous les blocs luma 4x4 du composant luma d'un macrobloc avec luma4x4BlkIdx = 0..15, la variable Intra4x4PredMode[luma4x4BlkIdx] est déduite comme spécifié au § 8.3.1.1.

Pour chaque bloc luma d'échantillons 4x4 indexé en utilisant luma4x4BlkIdx = 0..15,

1. Le processus de prédiction d'échantillon Intra_4x4 du § 8.3.1.2 est invoqué avec luma4x4BlkIdx et les échantillons construits antérieurement (dans l'ordre de décodage) au processus de filtrage de démontage de blocs à partir des blocs luma adjacents en entrée, le résultat étant les échantillons de prédiction luma Intra_4x4 $pred_{4x4L}[x, y]$ avec $x, y = 0..3$.

2. La position de l'échantillon supérieur gauche d'un bloc luma 4x4 avec l'indice luma4x4BlkIdx à l'intérieur du macrobloc en cours est déduite en invoquant le processus de balayage de bloc luma 4x4 inverse décrit au § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (xO, yO) et x, y = 0..3.

$$\text{pred}_L[xO + x, yO + y] = \text{pred4x4}_L[x, y] \quad (8-41)$$

3. Le processus de décodage du coefficient de transformée et le processus de construction d'image avant le processus de filtrage de démontage de blocs du § 8.5 sont invoqués avec pred_L et luma4x4BlkIdx comme entrée et les échantillons S'_L construits pour le bloc luma 4x4 en cours comme résultat.

8.3.1.1 Processus de déduction pour Intra4x4PredMode

Les entrées pour ce processus sont l'indice du bloc luma 4x4 luma4x4BlkIdx et les matrices de variables Intra4x4PredMode qui ont été déduites précédemment (dans l'ordre de décodage) pour les macroblocs adjacents.

Le résultat de ce processus est la variable Intra4x4PredMode[luma4x4BlkIdx].

Le Tableau 8-2 spécifie les valeurs pour Intra4x4PredMode[luma4x4BlkIdx] et les noms associés.

Tableau 8-2 – Spécification de Intra4x4PredMode[luma4x4BlkIdx] et des noms associés

Intra4x4PredMode[luma4x4BlkIdx]	Nom de Intra4x4PredMode[luma4x4BlkIdx]
0	Intra_4x4_Vertical (mode de prédiction)
1	Intra_4x4_Horizontal (mode de prédiction)
2	Intra_4x4_DC (mode de prédiction)
3	Intra_4x4_Diagonal_Down_Left (mode de prédiction)
4	Intra_4x4_Diagonal_Down_Right (mode de prédiction)
5	Intra_4x4_Vertical_Right (mode de prédiction)
6	Intra_4x4_Horizontal_Down (mode de prédiction)
7	Intra_4x4_Vertical_Left (mode de prédiction)
8	Intra_4x4_Horizontal_Up (mode de prédiction)

Intra4x4PredMode[luma4x4BlkIdx] étiqueté 0, 1, 3, 4, 5, 6, 7 et 8 représente les directions de prédiction comme illustré à la Figure 8-1.

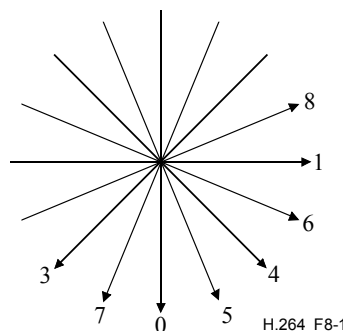


Figure 8-1 – Directions du mode de prédiction Intra_4x4 (pour information)

Soient intra4x4PredModeA et intra4x4PredModeB les variables qui spécifient les modes d'intraprédiction pour les blocs luma 4x4 du voisinage.

Intra4x4PredMode[luma4x4BlkIdx] est déduit comme suit.

- Le processus spécifié au § 6.4.7.3 est invoqué avec luma4x4BlkIdx donné en entrée et le résultat alloué à mbAddrA, luma4x4BlkIdxA, mbAddrB et luma4x4BlkIdxB.
- La variable dcOnlyPredictionFlag est déduite comme suit:
 - Si une des conditions suivantes est Vraie, dcOnlyPredictionFlag est mise égale à 1
 - le macrobloc avec l'adresse mbAddrA n'est pas disponible
 - le macrobloc avec l'adresse mbAddrB n'est pas disponible
 - le macrobloc avec l'adresse mbAddrA est disponible et codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1
 - le macrobloc avec l'adresse mbAddrB est disponible et codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1
 - Autrement, dcOnlyPredictionFlag est mise égale à 0.
- Pour N remplacé par A ou B, les variables intra4x4PredModeN sont déduites comme suit.
 - Si dcOnlyPredictionFlag est égal à 1 ou que le macrobloc avec l'adresse mbAddrN n'est pas codé en mode de prédiction de macrobloc Intra_4x4, intra4x4PredModeN est mis égal à 2 (mode de prédiction Intra_4x4_DC).
 - Autrement (dcOnlyPredictionFlag est égal à 0 et le macrobloc avec l'adresse mbAddrN est codé en mode de prédiction de macrobloc Intra_4x4), intra4x4PredModeN est mis égal à Intra4x4PredMode[luma4x4BlkIdxN], où Intra4x4PredMode est la matrice de variables allouée au macrobloc mbAddrN.
- Intra4x4PredMode[luma4x4BlkIdx] est déduit en appliquant la procédure suivante:

$$\begin{aligned}
 & \text{predIntra4x4PredMode} = \text{Min}(\text{intra4x4PredModeA}, \text{intra4x4PredModeB}) \\
 & \text{predIntra4x4PredMode} = \text{Min}(\text{intra4x4PredModeA}, \text{intra4x4PredModeB}) \\
 & \text{si}(\text{prev_intra4x4_pred_mode_flag}[\text{luma4x4BlkIdx}]) \\
 & \quad \text{Intra4x4PredMode}[\text{luma4x4BlkIdx}] = \text{predIntra4x4PredMode} \\
 & \text{ou} \tag{8-42} \\
 & \quad \text{si}(\text{rem_intra4x4_pred_mode}[\text{luma4x4BlkIdx}] < \text{predIntra4x4PredMode}) \\
 & \quad \quad \text{Intra4x4PredMode}[\text{luma4x4BlkIdx}] = \text{rem_intra4x4_pred_mode}[\text{luma4x4BlkIdx}] \\
 & \text{ou} \\
 & \quad \text{Intra4x4PredMode}[\text{luma4x4BlkIdx}] = \text{rem_intra4x4_pred_mode}[\text{luma4x4BlkIdx}] + 1
 \end{aligned}$$

8.3.1.2 Prédiction d'échantillon Intra_4x4

Ce processus est invoqué pour chaque bloc luma 4x4 d'un macrobloc avec mode de prédiction égal à Intra_4x4 suivi par le processus de décodage de la transformée et le processus de construction d'image avant le démontage de bloc pour chaque bloc luma 4x4.

Les entrées pour ce processus sont l'indice du bloc luma 4x4 avec l'indice luma4x4BlkIdx et les échantillons construits antérieurement (dans l'ordre de décodage) au processus de filtrage de démontage de blocs à partir des blocs luma adjacents.

Les résultats de ce processus sont les échantillons de prédiction $\text{pred4x4}_L[x, y]$, avec $x, y = 0..3$ pour le bloc luma 4x4 avec l'indice luma4x4BlkIdx.

La position de l'échantillon supérieur gauche d'un bloc luma 4x4 avec l'indice luma4x4BlkIdx à l'intérieur du macrobloc en cours est déduite en invoquant le processus de balayage de bloc luma 4x4 inverse du § 6.4.3 avec luma4x4BlkIdx en entrée et le résultat étant alloué à (xO, yO).

Les 13 échantillons voisins $p[x, y]$ qui sont des échantillons luma construits avant le processus de filtrage de démontage de blocs, avec $x = -1, y = -1..3$ et $x = 0..7, y = -1$, sont déduits comme suit.

- La localisation luma (xN, yN) est spécifié par

$$xN = xO + x \tag{8-43}$$

$$y_N = y_O + y \quad (8-44)$$

- Le processus de déduction pour les localisations du voisinage du § 6.4.8 est invoqué pour les localisations luma avec (x_N, y_N) en entrée et mbAddrN et (x_W, y_W) comme résultat.
- Chaque échantillon $p[x, y]$ avec $x = -1, y = -1..3$ et $x = 0..7, y = -1$ est déduit comme suit.
 - Si une des conditions suivantes est vraie, l'échantillon $p[x, y]$ est marqué comme "indisponible pour la prédiction Intra_4x4"
 - mbAddrN n'est pas disponible,
 - le macrobloc mbAddrN est codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1,
 - le macrobloc mbAddrN a mb_type égal à SI et constrained_intra_pred_flag est égal à 1 et le macrobloc en cours n'a pas mb_type égal à SI,
 - x est supérieur à 3 et luma4x4BlkIdx est égal à 3 ou 11.
 - Autrement, l'échantillon $p[x, y]$ est marqué comme "disponible pour la prédiction Intra_4x4" et l'échantillon luma à la localisation luma (x_W, y_W) à l'intérieur du macrobloc mbAddrN est alloué à $p[x, y]$.

Lorsque les échantillons $p[x, -1]$, avec $x = 4..7$, sont marqués comme "indisponible pour la prédiction Intra_4x4" et que l'échantillon $p[3, -1]$ est marqué comme "disponible pour la prédiction Intra_4x4," la valeur d'échantillon de $p[3, -1]$ est substituée aux valeurs d'échantillon $p[x, -1]$, avec $x = 4..7$ et les échantillons $p[x, -1]$, avec $x = 4..7$, sont marqués comme "disponible pour la prédiction Intra_4x4".

NOTE – Chaque bloc est supposé être construit dans une trame avant de décoder le bloc suivant.

En fonction de Intra4x4PredMode[luma4x4BlkIdx], un des modes de prédiction Intra_4x4 spécifiés aux § 8.3.1.2.1 à 8.3.1.2.9 doit être utilisé.

8.3.1.2.1 Spécification du mode de prédiction Intra_4x4_Vertical

Ce mode de prédiction Intra_4x4 doit être utilisé lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 0.

Ce mode ne doit être utilisé que lorsque les échantillons $p[x, -1]$ avec $x = 0..3$ sont marqués comme "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction $pred4x4_L[x, y]$, avec $x, y = 0..3$, sont calculées par

$$pred4x4_L[x, y] = p[x, -1], \text{ avec } x, y = 0..3 \quad (8-45)$$

8.3.1.2.2 Spécification du mode de prédiction Intra_4x4_Horizontal

Ce mode de prédiction Intra_4x4 doit être utilisé lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 1.

Ce mode ne doit être utilisé que lorsque les échantillons $p[-1, y]$, avec $y = 0..3$, sont marqués comme "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction $pred4x4_L[x, y]$, avec $x, y = 0..3$, sont calculées par

$$pred4x4_L[x, y] = p[-1, y], \text{ avec } x, y = 0..3 \quad (8-46)$$

8.3.1.2.3 Spécification du mode de prédiction Intra_4x4_DC

Ce mode de prédiction Intra_4x4 doit être utilisé lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 2.

Les valeurs des échantillons de prédiction $pred4x4_L[x, y]$, avec $x, y = 0..3$, sont calculées comme suit.

- Si tous les échantillons $p[x, -1]$, avec $x = 0..3$ et $p[-1, y]$, avec $y = 0..3$, sont marqués comme "disponible pour la prédiction Intra_4x4", les valeurs des échantillons de prédiction $pred4x4_L[x, y]$, avec $x, y = 0..3$, sont calculées par

$$pred4x4_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + 4) \gg 3 \quad (8-47)$$

- Autrement, si des échantillons $p[x, -1]$, avec $x = 0..3$ sont marqués comme "indisponible pour la prédiction Intra_4x4" et tous les échantillons $p[-1, y]$, avec $y = 0..3$ sont marqués comme "disponible pour la prédiction Intra_4x4", les valeurs des échantillons de prédiction $pred4x4_L[x, y]$, avec $x, y = 0..3$, sont calculées par

$$pred4x4_L[x, y] = (p[-1, 0] + p[-1, 1] + p[-1, 2] + p[-1, 3] + 2) \gg 2 \quad (8-48)$$

- Autrement, si des échantillons $p[-1, y]$, avec $y = 0..3$ sont marqués comme "indisponible pour la prédiction Intra_4x4" et tous les échantillons $p[x, -1]$, avec $x = 0..3$ sont marqués comme "disponible pour la prédiction Intra_4x4", les valeurs des échantillons de prédiction $pred4x4_L[x, y]$, avec $x, y = 0..3$ sont calculées par

$$pred4x4_L[x, y] = (p[0, -1] + p[1, -1] + p[2, -1] + p[3, -1] + 2) \gg 2 \quad (8-49)$$

- Autrement (certains échantillons $p[x, -1]$, avec $x = 0..3$ et certains échantillons $p[-1, y]$, avec $y = 0..3$, sont marqués comme "indisponible pour la prédiction Intra_4x4"), les valeurs des échantillons de prédiction $pred4x4_L[x, y]$, avec $x, y = 0..3$, sont calculées par

$$pred4x4_L[x, y] = 128 \quad (8-50)$$

NOTE – Un bloc luma 4x4 peut toujours être prédit en utilisant ce mode.

8.3.1.2.4 Spécification du mode de prédiction Intra_4x4_Diagonal_Down_Left

Ce mode de prédiction Intra_4x4 doit être utilisé lorsque $Intra4x4PredMode[luma4x4BlkIdx]$ est égal à 3.

Ce mode ne doit être utilisé que lorsque les échantillons $p[x, -1]$ avec $x = 0..7$ sont marqués comme "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction $pred4x4_L[x, y]$, avec $x, y = 0..3$, sont calculées comme suit.

- Si x est égal à 3 et y est égal à 3

$$pred4x4_L[x, y] = (p[6, -1] + 3 * p[7, -1] + 2) \gg 2 \quad (8-51)$$

- Autrement (x n'est pas égal à 3 ou y n'est pas égal à 3)

$$pred4x4_L[x, y] = (p[x + y, -1] + 2 * p[x + y + 1, -1] + p[x + y + 2, -1] + 2) \gg 2 \quad (8-52)$$

8.3.1.2.5 Spécification du mode de prédiction Intra_4x4_Diagonal_Down_Right

Ce mode de prédiction Intra_4x4 doit être utilisé lorsque $Intra4x4PredMode[luma4x4BlkIdx]$ est égal à 4.

Ce mode ne doit être utilisé que lorsque les échantillons $p[x, -1]$ avec $x = 0..3$ et $p[-1, y]$ avec $y = -1..3$ sont marqués comme "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction $pred4x4_L[x, y]$, avec $x, y = 0..3$, sont calculées comme suit.

- Si x est supérieur à y ,

$$pred4x4_L[x, y] = (p[x - y - 2, -1] + 2 * p[x - y - 1, -1] + p[x - y, -1] + 2) \gg 2 \quad (8-53)$$

- Autrement si x est inférieur à y ,

$$pred4x4_L[x, y] = (p[-1, y - x - 2] + 2 * p[-1, y - x - 1] + p[-1, y - x] + 2) \gg 2 \quad (8-54)$$

- Autrement (x est égal à y),

$$\text{pred4x4}_L[x, y] = (p[0, -1] + 2 * p[-1, -1] + p[-1, 0] + 2) \gg 2 \quad (8-55)$$

8.3.1.2.6 Spécification du mode de prédiction Intra_4x4_Vertical_Right

Ce mode de prédiction Intra_4x4 doit être utilisé lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 5.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..3 et p[-1, y] avec y = -1..3 sont marqués comme "disponible pour la prédiction Intra_4x4".

Soit la variable zVR mise égale à 2 * x - y.

Les valeurs des échantillons de prédiction pred4x4_L[x, y], avec x, y = 0..3, sont calculées comme suit.

- Si zVR est égal à 0, 2, 4 ou 6,

$$\text{pred4x4}_L[x, y] = (p[x - (y \gg 1) - 1, -1] + p[x - (y \gg 1), -1] + 1) \gg 1 \quad (8-56)$$

- Autrement, si zVR est égal à 1, 3 ou 5,

$$\text{pred4x4}_L[x, y] = (p[x - (y \gg 1) - 2, -1] + 2 * p[x - (y \gg 1) - 1, -1] + p[x - (y \gg 1), -1] + 2) \gg 2 \quad (8-57)$$

- Autrement, si zVR est égal à -1,

$$\text{pred4x4}_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) \gg 2 \quad (8-58)$$

- Autrement (zVR est égal à -2 ou -3),

$$\text{pred4x4}_L[x, y] = (p[-1, y - 1] + 2 * p[-1, y - 2] + p[-1, y - 3] + 2) \gg 2 \quad (8-59)$$

8.3.1.2.7 Spécification du mode de prédiction Intra_4x4_Horizontal_Down

Ce mode de prédiction Intra_4x4 doit être utilisé lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 6.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..3 et p[-1, y] avec y = -1..3 sont marqués comme "disponible pour la prédiction Intra_4x4".

Soit la variable zHD mise égale à 2 * y - x.

Les valeurs des échantillons de prédiction pred4x4_L[x, y], avec x, y = 0..3 sont calculées comme suit.

- Si zHD est égal à 0, 2, 4 ou 6,

$$\text{pred4x4}_L[x, y] = (p[-1, y - (x \gg 1) - 1] + p[-1, y - (x \gg 1)] + 1) \gg 1 \quad (8-60)$$

- Autrement, si zHD est égal à 1, 3 ou 5,

$$\text{pred4x4}_L[x, y] = (p[-1, y - (x \gg 1) - 2] + 2 * p[-1, y - (x \gg 1) - 1] + p[-1, y - (x \gg 1)] + 2) \gg 2 \quad (8-61)$$

- Autrement, si zHD est égal à -1,

$$\text{pred4x4}_L[x, y] = (p[-1, 0] + 2 * p[-1, -1] + p[0, -1] + 2) \gg 2 \quad (8-62)$$

- Autrement (zHD est égal à -2 ou -3),

$$\text{pred4x4}_L[x, y] = (p[x - 1, -1] + 2 * p[x - 2, -1] + p[x - 3, -1] + 2) \gg 2 \quad (8-63)$$

8.3.1.2.8 Spécification du mode de prédiction Intra_4x4_Vertical_Left

Ce mode de prédiction Intra_4x4 doit être utilisé lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 7.

Ce mode ne doit être utilisé que lorsque les échantillons p[x, -1] avec x = 0..7 sont marqués comme "disponible pour la prédiction Intra_4x4".

Les valeurs des échantillons de prédiction pred4x4_L[x, y], avec x, y = 0..3 sont calculées comme suit.

- Si y est égal à 0 ou 2,

$$\text{pred4x4}_L[x, y] = (p[x + (y \gg 1), -1] + p[x + (y \gg 1) + 1, -1] + 1) \gg 1 \quad (8-64)$$

- Autrement (y est égal à 1 ou 3),

$$\text{pred4x4}_L[x, y] = (p[x + (y \gg 1), -1] + 2 * p[x + (y \gg 1) + 1, -1] + p[x + (y \gg 1) + 2, -1] + 2) \gg 2 \quad (8-65)$$

8.3.1.2.9 Spécification du mode de prédiction Intra_4x4_Horizontal_Up

Ce mode de prédiction Intra_4x4 doit être utilisé lorsque Intra4x4PredMode[luma4x4BlkIdx] est égal à 8.

Ce mode ne doit être utilisé que lorsque les échantillons $p[-1, y]$ avec $y = 0..3$ sont marqués comme "disponible pour la prédiction Intra_4x4".

Soit la variable zHU mise égale à $x + 2 * y$.

Les valeurs des échantillons de prédiction $\text{pred4x4}_L[x, y]$, avec $x, y = 0..3$ sont calculées comme suit:

- Si zHU est égal à 0, 2 ou 4

$$\text{pred4x4}_L[x, y] = (p[-1, y + (x \gg 1)] + p[-1, y + (x \gg 1) + 1] + 1) \gg 1 \quad (8-66)$$

- Autrement, si zHU est égal à 1 ou 3

$$\text{pred4x4}_L[x, y] = (p[-1, y + (x \gg 1)] + 2 * p[-1, y + (x \gg 1) + 1] + p[-1, y + (x \gg 1) + 2] + 2) \gg 2 \quad (8-67)$$

- Autrement, si zHU est égal à 5,

$$\text{pred4x4}_L[x, y] = (p[-1, 2] + 3 * p[-1, 3] + 2) \gg 2 \quad (8-68)$$

- Autrement (zHU est supérieur à 5),

$$\text{pred4x4}_L[x, y] = p[-1, 3] \quad (8-69)$$

8.3.2 Processus de prédiction Intra_16x16 pour échantillons luma

Ce processus est invoqué lorsque le mode de prédiction de macrobloc est égal à Intra_16x16. Il spécifie comment sont déduits les échantillons luma d'intraprédiction pour le macrobloc en cours.

L'entrée pour ce processus est constituée des échantillons construits avant le processus de démontage de blocs à partir des blocs luma du voisinage (si disponibles).

Les résultats de ce processus sont les échantillons luma d'intraprédiction pour le macrobloc en cours $\text{pred}_L[x, y]$.

Les 33 échantillons $p[x, y]$ du voisinage qui sont des échantillons luma construits avant le processus de filtrage de démontage de blocs, avec $x = -1, y = -1..15$ et avec $x = 0..15, y = -1$, sont déduits comme suit.

- Le processus de déduction pour les localisations du voisinage du § 6.4.8 est invoqué pour les localisations luma avec (x, y) alloué à (xN, yN) comme entrée et $mbAddrN$ et (xW, yW) comme résultat.
- Chaque échantillon $p[x, y]$ avec $x = -1, y = -1..15$ et avec $x = 0..15, y = -1$ est déduit comme suit.
 - Si l'une des conditions suivantes est vraie, l'échantillon $p[x, y]$ est marqué comme "indisponible pour la prédiction Intra_16x16":
 - $mbAddrN$ n'est pas disponible;
 - le macrobloc $mbAddrN$ est codé en mode d'interprédiction et $\text{constrained_intra_pred_flag}$ est égal à 1;

- le macrobloc mbAddrN a mb_type égal à SI et constrained_intra_pred_flag est égal à 1.
- Autrement, l'échantillon p[x, y] est marqué comme "disponible pour la prédiction Intra_16x16" et l'échantillon luma à la localisation luma (xW, yW) à l'intérieur du macrobloc mbAddrN est mis à p[x, y].

Soit $pred_L[x, y]$ avec $x, y = 0..15$ pour noter les échantillons de prédiction pour les échantillons de bloc luma 16x16.

Les modes de prédiction Intra_16x16 sont spécifiés au Tableau 8-3.

Tableau 8-3 – Spécification de Intra16x16PredMode et des noms associés

Intra16x16PredMode	Nom de Intra16x16PredMode
0	Intra_16x16_Vertical (mode de prédiction)
1	Intra_16x16_Horizontal (mode de prédiction)
2	Intra_16x16_DC (mode de prédiction)
3	Intra_16x16_Plane (mode de prédiction)

En fonction de Intra16x16PredMode, un des modes de prédiction Intra_16x16 spécifiés aux § 8.3.2.1 à 8.3.2.4 doit être utilisé.

8.3.2.1 Spécification du mode de prédiction Intra_16x16_Vertical

Ce mode de prédiction Intra_16x16 ne doit être utilisé que lorsque les échantillons p[x, -1] avec $x = 0..15$ sont marqués comme "disponible pour la prédiction Intra_16x16".

$$pred_L[x, y] = p[x, -1], \text{ con } y = 0..15 \quad (8-70)$$

8.3.2.2 Spécification du mode de prédiction Intra_16x16_Horizontal

Ce mode de prédiction Intra_16x16 ne doit être utilisé que lorsque les échantillons p[-1, y] avec $y = 0..15$ sont marqués comme "disponible pour la prédiction Intra_16x16".

$$pred_L[x, y] = p[-1, y], \text{ con } y = 0..15 \quad (8-71)$$

8.3.2.3 Spécification du mode de prédiction Intra_16x16_DC

Ce mode de prédiction Intra_16x16 ne doit être utilisé que si les échantillons du voisinage sont marqués comme "disponible pour la prédiction Intra_16x16" comme suit.

- Si tous les échantillons p[x, -1], avec $x = 0..15$ et p[-1, y], avec $y = 0..15$ du voisinage sont marqués comme "disponible pour la prédiction Intra_16x16", la prédiction pour tous les échantillons luma dans le macrobloc est donnée par:

$$pred_L[x, y] = \left(\sum_{x'=0}^{15} p[x', -1] + \sum_{y'=0}^{15} p[-1, y'] + 16 \right) \gg 5, \text{ avec } = 0..15 \quad (8-72)$$

- Autrement, si un ou plusieurs des échantillons p[x, -1] du voisinage, avec $x = 0..15$ sont marqués comme "indisponible pour la prédiction Intra_16x16" et si tous les échantillons p[-1, y] du voisinage, avec $y = 0..15$ sont marqués comme "disponible pour la prédiction Intra_16x16", la prédiction pour tous les échantillons luma dans le macrobloc est donnée par:

$$pred_L[x, y] = \left(\sum_{y'=0}^{15} p[-1, y'] + 8 \right) \gg 4, \text{ avec } y = 0..15 \quad (8-73)$$

- Autrement, si un ou plusieurs des échantillons $p[-1, y]$ du voisinage, avec $y = 0..15$ sont marqués comme "indisponible pour la prédiction Intra_16x16 et que tous les échantillons $p[x, -1]$ du voisinage, avec $x = 0..15$ sont marqués comme "disponible pour la prédiction Intra_16x16", la prédiction pour tous les échantillons luma dans le macrobloc est donnée par:

$$\text{pred}_L[x, y] = \left(\sum_{x'=0}^{15} p[x', -1] + 8 \right) \gg 4 \text{ avec } x, y = 0..15 \quad (8-74)$$

- Autrement (certains des échantillons $p[x, -1]$ du voisinage, avec $x = 0..15$ et certains des échantillons $p[-1, y]$ du voisinage, avec $y = 0..15$ sont marqués comme "indisponible pour la prédiction Intra_16x16"), la prédiction pour tous les échantillons luma dans le macrobloc est donnée par:

$$\text{pred}_L[x, y] = 128, \text{ con } y = 0..15 \quad (8-75)$$

8.3.2.4 Spécification du mode de prédiction Intra_16x16_Plane

Ce mode de prédiction Intra_16x16 ne doit être utilisé que lorsque les échantillons $p[x, -1]$ avec $x = -1..15$ et $p[-1, y]$ avec $y = 0..15$ sont marqués comme "disponible pour la prédiction Intra_16x16".

$$\text{pred}_L[x, y] = \text{Clip1}((a + b * (x - 7) + c * (y - 7) + 16) \gg 5), \text{ avec } y = 0..15, \quad (8-76)$$

où:

$$a = 16 * (p[-1, 15] + p[15, -1]) \quad (8-77)$$

$$b = (5 * H + 32) \gg 6 \quad (8-78)$$

$$c = (5 * V + 32) \gg 6 \quad (8-79)$$

et H et V sont spécifiés aux équations 8-80 et 8-81.

$$H = \sum_{x'=0}^7 (x'+1) * (p[8+x', -1] - p[6-x', -1]) \quad (8-80)$$

$$V = \sum_{y'=0}^7 (y'+1) * (p[-1, 8+y'] - p[-1, 6-y']) \quad (8-81)$$

8.3.3 Processus d'intraprédiction pour les échantillons chroma

Ce processus est invoqué pour les types de macroblocs I et SI. Il spécifie comment sont déduits les échantillons chroma d'intraprédiction pour le macrobloc en cours.

Les entrées pour ce processus sont les échantillons construits antérieurement au processus de démontage de bloc à partir des blocs chroma du voisinage (si disponibles).

Les résultats de ce processus sont les échantillons chroma d'intraprédiction pour le macrobloc en cours $\text{pred}_{Cb}[x, y]$ et $\text{pred}_{Cr}[x, y]$.

Les deux blocs chroma (Cb et Cr) du macrobloc doivent utiliser le même mode de prédiction. Le mode de prédiction est appliqué à chaque bloc chroma séparément. Le processus spécifié au présent paragraphe est invoqué pour chaque bloc chroma. Dans le reste du présent paragraphe, bloc chroma se réfère à un des deux blocs chroma et la mention C est utilisée en remplacement de la mention Cb ou Cr.

Les 17 échantillons $p[x, y]$ du voisinage qui sont des échantillons chroma construits antérieurement au processus de filtrage de démontage de blocs, avec $x = -1, y = -1..7$ et avec $x = 0..7, y = -1$, sont déduits comme suit.

- Le processus de déduction pour les localisations du voisinage du § 6.4.8 est invoqué pour les localisations chroma avec (x, y) alloué à (xN, yN) comme entrée et mbAddrN et (xW, yW) comme résultat.
- Chaque échantillon p[x, y] est déduit comme suit.
 - Si l'une des conditions suivantes est vraie, l'échantillon p[x, y] est marqué comme "indisponible pour l'intraprédiction chroma":
 - mbAddrN n'est pas disponible;
 - le macrobloc mbAddrN est codé en mode d'interprédiction et constrained_intra_pred_flag est égal à 1;
 - le macrobloc mbAddrN a mb_type égal à SI et constrained_intra_pred_flag est égal à 1 et le macrobloc en cours n'a pas mb_type égal à SI.
 - Autrement, l'échantillon p[x, y] est marqué comme "disponible pour l'intraprédiction chroma" et l'échantillon chroma de composant C à la localisation chroma (xW, yW) au sein du macrobloc mbAddrN est alloué à p[x, y].

Soit $pred_C[x, y]$ avec $x, y = 0..7$ qui note les échantillons de prédiction pour les échantillons du bloc chroma.

Les modes d'intraprédiction chroma sont spécifiés au Tableau 8-4.

Tableau 8-4 – Spécification des modes d'intraprédiction chroma et des noms associés

intra_chroma_pred_mode	Nom de intra_chroma_pred_mode
0	Intra_Chroma_DC (mode de prédiction)
1	Intra_Chroma_Horizontal (mode de prédiction)
2	Intra_Chroma_Vertical (mode de prédiction)
3	Intra_Chroma_Plane (mode de prédiction)

En fonction de `intra_chroma_pred_mode`, un des modes d'intraprédiction chroma spécifiés aux § 8.3.3.1 à 8.3.3.4 doit être utilisé.

8.3.3.1 Spécification du mode de prédiction Intra_Chroma_DC

Les valeurs des échantillons de prédiction $pred_C[x, y]$ avec $x = 0..3$ et $y = 0..3$ sont calculées comme suit.

- Si tous les échantillons $p[x, -1]$, avec $x = 0..3$ et tous les échantillons $p[-1, y]$, avec $y = 0..3$ sont marqués comme "disponible pour l'intraprédiction chroma",

$$pred_C[x, y] = \left(\sum_{x'=0}^3 p[x', -1] + \sum_{y'=0}^3 p[-1, y'] + 4 \right) \gg 3, \text{ avec } x = 0..3 \text{ et } y = 0..3 \quad (8-82)$$

- Autrement, si tous les échantillons $p[x, -1]$, avec $x = 0..3$ sont marqués comme "disponible pour l'intraprédiction chroma" et qu'un ou plusieurs échantillons $p[-1, y]$, avec $y = 0..3$ sont marqués comme "indisponible pour l'intraprédiction chroma",

$$pred_C[x, y] = \left(\sum_{x'=0}^3 p[x', -1] + 2 \right) \gg 2, \text{ avec } x = 0..3 \text{ et } y = 0..3 \quad (8-83)$$

- Autrement, si un ou plusieurs échantillons $p[x, -1]$, avec $x = 0..3$ sont marqués comme "indisponible pour l'intraprédiction chroma" et tous les échantillons $p[-1, y]$, avec $y = 0..3$ sont marqués comme "disponible pour l'intraprédiction chroma",

$$\text{pred}_C[x, y] = \left(\sum_{y'=0}^3 p[-1, y'] + 2 \right) \gg 2, \text{ avec } x = 0..3 \text{ et } y = 0..3 \quad (8-84)$$

- Autrement (certains échantillons $p[x, -1]$, avec $x = 0..3$ et certains échantillons $p[-1, y]$, avec $y = 0..3$ sont marqués comme "indisponible pour l'intraprédiction chroma"),
 $\text{pred}_C[x, y] = 128$, avec $x = 0..3$ et $y = 0..3$ (8-85)

Les valeurs des échantillons de prédiction $\text{pred}_C[x, y]$ avec $x = 4..7$ et $y = 0..3$ sont calculées comme suit.

- Si tous les échantillons $p[x, -1]$, avec $x = 4..7$ sont marqués comme "disponible pour l'intraprédiction chroma",

$$\text{pred}_C[x, y] = \left(\sum_{x'=4}^7 p[x', -1] + 2 \right) \gg 2, \text{ avec } x = 4..7 \text{ et } y = 0..3 \quad (8-86)$$

- Autrement, si tous les échantillons $p[-1, y]$, avec $y = 0..3$ sont marqués comme "disponible pour l'intraprédiction chroma",

$$\text{pred}_C[x, y] = \left(\sum_{y'=0}^3 p[-1, y'] + 2 \right) \gg 2, \text{ avec } x = 4..7 \text{ et } y = 0..3 \quad (8-87)$$

- Autrement (certains échantillons $p[x, -1]$, avec $x = 4..7$ et certains échantillons $p[-1, y]$, avec $y = 0..3$ sont marqués comme "indisponible pour l'intraprédiction chroma"),
 $\text{pred}_C[x, y] = 128$, avec $x = 4..7$ et $y = 0..3$ (8-88)

Les valeurs des échantillons de prédiction $\text{pred}_C[x, y]$ avec $x = 0..3$ et $y = 4..7$ sont calculées comme suit.

- Si tous les échantillons $p[-1, y]$, avec $y = 4..7$ sont marqués comme "disponible pour l'intraprédiction chroma",

$$\text{pred}_C[x, y] = \left(\sum_{y'=4}^7 p[-1, y'] + 2 \right) \gg 2, \text{ avec } x = 0..3 \text{ et } y = 4..7 \quad (8-89)$$

- Autrement, si tous les échantillons $p[x, -1]$, avec $x = 0..3$ sont marqués comme "disponible pour l'intraprédiction chroma",

$$\text{pred}_C[x, y] = \left(\sum_{x'=0}^3 p[x', -1] + 2 \right) \gg 2, \text{ avec } x = 0..3 \text{ et } y = 4..7 \quad (8-90)$$

- Autrement (certains échantillons $p[x, -1]$, avec $x = 0..3$ et certains échantillons $p[-1, y]$, avec $y = 4..7$ sont marqués comme "indisponible pour l'intraprédiction chroma"),
 $\text{pred}_C[x, y] = 128$, avec $x = 0..3$ et $y = 4..7$ (8-91)

Les valeurs des échantillons de prédiction $\text{pred}_C[x, y]$ avec $x = 4..7$ et $y = 4..7$ sont calculées comme suit.

- Si tous les échantillons $p[x, -1]$, avec $x = 4..7$ et tous les échantillons $p[-1, y]$, avec $y = 4..7$ sont marqués comme "disponible pour l'intraprédiction chroma",

$$\text{pred}_C[x, y] = \left(\sum_{x'=4}^7 p[x', -1] + \sum_{y'=4}^7 p[-1, y'] + 4 \right) \gg 3, \text{ avec } x = 4..7 \text{ et } y = 4..7 \quad (8-92)$$

- Autrement, si tous les échantillons $p[x, -1]$ avec $x = 4..7$ sont marqués comme "disponible pour l'intraprédiction chroma" et qu'un ou plusieurs échantillons $p[-1, y]$, avec $y = 4..7$ sont marqués comme "indisponible pour l'intraprédiction chroma",

$$\text{pred}_C[x, y] = \left(\sum_{x'=4}^7 p[x', -1] + 2 \right) \gg 2, \text{ avec } x = 4..7 \text{ et } y = 4..7 \quad (8-93)$$

- Autrement, si un ou plusieurs échantillons $p[x, -1]$, avec $x = 4..7$ sont marqués comme "indisponible pour l'intraprédiction chroma" et tous les échantillons $p[-1, y]$, avec $y = 4..7$ sont marqués comme "disponible pour l'intraprédiction chroma",

$$\text{pred}_C[x, y] = \left(\sum_{y'=4}^7 p[-1, y'] + 2 \right) \gg 2, \text{ avec } x = 4..7 \text{ et } y = 4..7 \quad (8-94)$$

- Autrement (certains échantillons $p[x, -1]$, avec $x = 4..7$ et certains échantillons $p[-1, y]$, avec $y = 4..7$ sont marqués comme "indisponible pour l'intraprédiction chroma"),

$$\text{pred}_C[x, y] = 128, \text{ avec } x = 4..7 \text{ et } y = 4..7 \quad (8-95)$$

8.3.3.2 Spécification du mode de prédiction Intra_Chroma_Horizontal

Ce mode ne doit être utilisé que lorsque les échantillons $p[-1, y]$, avec $y = 0..7$ sont marqués comme "disponible pour l'intraprédiction chroma".

Les valeurs des échantillons de prédiction $\text{pred}_C[x, y]$ sont calculées comme suit.

$$\text{pred}_C[x, y] = p[-1, y], \text{ avec } x, y = 0..7 \quad (8-96)$$

8.3.3.3 Spécification du mode de prédiction Intra_Chroma_Vertical

Ce mode ne doit être utilisé que lorsque les échantillons $p[x, -1]$, avec $x = 0..7$ sont marqués comme "disponible pour l'intraprédiction chroma".

Les valeurs des échantillons de prédiction $\text{pred}_C[x, y]$ sont calculées comme suit.

$$\text{pred}_C[x, y] = p[x, -1], \text{ avec } x, y = 0..7 \quad (8-97)$$

8.3.3.4 Spécification du mode de prédiction Intra_Chroma_Plane

Ce mode ne doit être utilisé que lorsque les échantillons $p[x, -1]$, avec $x = 0..7$ et $p[-1, y]$, avec $y = -1..7$ sont marqués comme "disponible pour l'intraprédiction chroma".

Les valeurs des échantillons de prédiction $\text{pred}_C[x, y]$ sont calculées comme suit.

$$\text{pred}_C[x, y] = \text{Clip1}((a + b * (x - 3) + c * (y - 3) + 16) \gg 5), \text{ avec } x, y = 0..7 \quad (8-98)$$

où:

$$a = 16 * (p[-1, 7] + p[7, -1]) \quad (8-99)$$

$$b = (17 * H + 16) \gg 5 \quad (8-100)$$

$$c = (17 * V + 16) \gg 5 \quad (8-101)$$

et H et V sont spécifiés comme suit.

$$H = \sum_{x'=0}^3 (x'+1) * (p[4+x', -1] - p[2-x', -1]) \quad (8-102)$$

$$V = \sum_{y'=0}^3 (y'+1) * (p[-1, 4+y'] - p[-1, 2-y']) \quad (8-103)$$

8.3.4 Processus de construction des échantillons pour macroblocs I_PCM

Ce processus est invoqué lorsque mb_type est égal à I_PCM.

Les résultats de ce processus sont des échantillons de macrobloc S'_L , S'_{Cb} et S'_{Cr} construits antérieurement au processus de filtrage de démontage de blocs.

La variable dy est déduite comme suit.

- Si MbaffFrameFlag est égal à 1 et que le macrobloc en cours est un macrobloc de champ, dy est mise égale à 2.
- Autrement (MbaffFrameFlag est égal à 0 ou le macrobloc en cours est un macrobloc de trame), dy est mise égale à 1.

La position de l'échantillon luma supérieur gauche du macrobloc en cours est déduite en invoquant le processus de balayage de macrobloc inverse du § 6.4.1 avec CurrMbAddr comme entrée et le résultat étant alloué à (xP, yP).

Les échantillons construits antérieurement au processus de démontage de bloc sont générés comme spécifié par:

$$\text{pour}(i = 0; i < 256; i++) \\ S'_L[xP + (i \% 16), yP + dy * (i / 16)] = \text{pcm_byte}[i] \quad (8-104)$$

$$\text{pour}(i = 0; i < 64; i++) \{ \\ S'_{Cb}[(xP >> 1) + (i \% 8), ((yP + 1) >> 1) + dy * (i / 8)] = \text{pcm_byte}[i + 256] \\ S'_{Cr}[(xP >> 1) + (i \% 8), ((yP + 1) >> 1) + dy * (i / 8)] = \text{pcm_byte}[i + 320] \\ \} \quad (8-105)$$

8.4 Processus d'interprédiction

Ce processus est invoqué lors du décodage des types de macrobloc P et B.

Les résultats de ce processus sont des échantillons d'interprédiction pour le macrobloc en cours qui sont une matrice 16x16 pred_L d'échantillons luma et deux matrices 8x8 pred_{Cr} et pred_{Cb} d'échantillons chroma, une pour chacun des composants chroma Cb et Cr.

La subdivision d'un macrobloc est spécifiée par mb_type. Chaque subdivision de macrobloc est désignée par mbPartIdx. Lorsque la subdivision de macrobloc consiste en partitions qui sont égales à des sous-macroblots, chaque sous-macrobloc peut encore être partagé en subdivisions de sous-macrobloc comme spécifié par sub_mb_type. Chaque subdivision de sous-macrobloc est désignée par subMbPartIdx. Lorsque la subdivision de macrobloc ne comporte pas de sous-macroblots, subMbPartIdx est mis égal à 0.

Les étapes suivantes sont spécifiées pour chaque subdivision de macrobloc ou pour chaque subdivision de sous-macrobloc.

Les fonctions MbPartWidth(), MbPartHeight(), SubMbPartWidth() et SubMbPartHeight() décrivant la largeur et la hauteur des subdivisions de macrobloc et subdivisions de sous-macrobloc sont spécifiées au Tableau 7-10, au Tableau 7-11, au Tableau 7-14 et au Tableau 7-15.

Les variables partWidth et partHeight sont déduites comme suit.

- Si mb_type n'est pas égal à P_8x8 ou P_8x8ref0 ou B_8x8, on applique ce qui suit.
$$\text{partWidth} = \text{MbPartWidth}(\text{mb_type}) \quad (8-106)$$

$$\text{partHeight} = \text{MbPartHeight}(\text{mb_type}) \quad (8-107)$$

- Autrement (mb_type est égal à P_8x8 ou P_8x8ref0 ou B_8x8),
$$\text{partWidth} = \text{SubMbPartWidth}(\text{sub_mb_type}[\text{mbPartIdx}]) \quad (8-108)$$

$$\text{partHeight} = \text{SubMbPartHeight}(\text{sub_mb_type}[\text{mbPartIdx}]). \quad (8-109)$$

Lorsque mb_type est égal à B_Skip ou B_Direct_16x16 ou sub_mb_type[mbPartIdx] est égal à B_Direct_8x8, le processus d'interprédiction est spécifié pour

$$\text{partWidth} = 4 \quad (8-110)$$

$$\text{partHeight} = 4 \quad (8-111)$$

avec mbPartIdx traitant des valeurs 0..3. Pour chaque sous-macrobloc indexé par mbPartIdx, subMbPartIdx traite des valeurs 0..3.

Soit la variable $MvCnt$ initialement égale à 0 avant toute invocation du § 8.4.1 pour le macrobloc.

Le processus d'interprédiction pour une subdivision de macrobloc $mbPartIdx$ et une subdivision de sous-macrobloc $subMbPartIdx$ comporte les étapes ordonnées suivantes:

1. Processus de déduction pour les composantes de vecteur cinétique et les indices de référence comme spécifié au § 8.4.1.

Les entrées pour ce processus sont:

- une subdivision de macrobloc $mbPartIdx$;
- une subdivision de sous-macrobloc $subMbPartIdx$.

Les résultats de ce processus sont:

- les vecteurs cinétiques luma $mvL0$ et $mvL1$ et les vecteurs cinétiques chroma $mvCL0$ et $mvCL1$;
- les indices de référence $refIdxL0$ et $refIdxL1$;
- les fanions d'utilisation de liste de prédiction $predFlagL0$ et $predFlagL1$;
- le compte de vecteur cinétique de subdivision de sous-macrobloc $subMvCnt$.

2. La variable $MvCnt$ est incrémentée de $subMvCnt$.

3. Processus de décodage pour les échantillons d'interprédiction comme spécifié au § 8.4.2.

Les entrées pour ce processus sont:

- une subdivision de macrobloc $mbPartIdx$;
- une subdivision de sous-macrobloc $subMbPartIdx$;
- les variables spécifiant la largeur et la hauteur de la subdivision, $partWidth$ et $partHeight$;
- les vecteurs cinétiques luma $mvL0$ et $mvL1$ et les vecteurs cinétiques chroma $mvCL0$ et $mvCL1$;
- les indices de référence $refIdxL0$ et $refIdxL1$;
- les fanions d'utilisation de liste de prédiction $predFlagL0$ et $predFlagL1$.

Les résultats de ce processus sont:

- des échantillons d'interprédiction ($pred$); qui sont une matrice $(partWidth) \times (partHeight)$ $predPart_L$ d'échantillons luma de prédiction et deux matrices $(partWidth/2) \times (partHeight/2)$ $predPart_{Cr}$ et $predPart_{Cb}$ d'échantillons chroma de prédiction, une pour chacun des composants chroma Cb et Cr .

Pour l'utilisation dans les processus de déduction de variables invoquées ultérieurement dans le processus de décodage, on fait les allocations suivantes:

$$MvL0[mbPartIdx][subMbPartIdx] = mvL0 \quad (8-112)$$

$$MvL1[mbPartIdx][subMbPartIdx] = mvL1 \quad (8-113)$$

$$RefIdxL0[mbPartIdx] = refIdxL0 \quad (8-114)$$

$$RefIdxL1[mbPartIdx] = refIdxL1 \quad (8-115)$$

$$PredFlagL0[mbPartIdx] = predFlagL0 \quad (8-116)$$

$$PredFlagL1[mbPartIdx] = predFlagL1 \quad (8-117)$$

La localisation de l'échantillon supérieur gauche de la subdivision relative à l'échantillon supérieur gauche du macrobloc est déduite en invoquant le processus de balayage inverse de subdivision de macrobloc comme décrit au § 6.4.2.1 avec $mbPartIdx$ comme entrée et (xP, yP) comme résultat.

La localisation de l'échantillon supérieur gauche de la subdivision de macrobloc relative à l'échantillon supérieur gauche de la subdivision de macrobloc est déduite en invoquant le processus

de balayage inverse de division de sous-macrobloc comme décrit au § 6.4.2.2 avec `subMbPartIdx` comme entrée et (xS, yS) comme résultat.

La prédiction de macrobloc est formée en plaçant comme suit les échantillons de prédiction de subdivision ou de subdivision de sous-macrobloc dans leurs positions relatives correctes dans le macrobloc.

La variable $pred_L[xP + xS + x, yP + yS + y]$, avec $x = 0 .. partWidth - 1$, $y = 0 .. partHeight - 1$, est calculée par

$$pred_L[xP + xS + x, yP + yS + y] = predPart_L[x, y] \quad (8-118)$$

La variable $pred_C[xP / 2 + xS / 2 + x, yP / 2 + yS / 2 + y]$, avec $x = 0 .. partWidth/2 - 1$, $y = 0 .. partHeight/2 - 1$ et C remplacé par Cb ou Cr, est calculée par

$$pred_C[xP / 2 + xS / 2 + x, yP / 2 + yS / 2 + y] = predPart_C[x, y] \quad (8-119)$$

8.4.1 Processus de déduction pour composantes de vecteur cinétique et indices de référence

Les entrées pour ce processus sont:

- une subdivision de macrobloc `mbPartIdx`;
- une subdivision de sous-macrobloc `subMbPartIdx`.

Les résultats de ce processus sont:

- des vecteurs cinétiques luma `mvL0` et `mvL1` ainsi que des vecteurs cinétiques chroma `mvCL0` et `mvCL1`;
- des indices de référence `refIdxL0` et `refIdxL1`;
- des fanions d'utilisation de liste de prédiction `predFlagL0` et `predFlagL1`;
- une variable compte de vecteur cinétique de subdivision de sous-macrobloc `subMvCnt`.

Pour la déduction des variables `mvL0` et `mvL1` ainsi que `refIdxL0` et `refIdxL1`, on applique ce qui suit:

- si `mb_type` est égal à `P_Skip`, le processus de déduction pour les vecteurs cinétiques luma pour des macroblocs sautés dans des tranches P et SP du § 8.4.1.1 est invoqué avec comme résultat des vecteurs cinétiques luma `mvL0` et des indices de référence `refIdxL0` et `predFlagL0` est mis égal à 1. `mvL1` et `refIdxL1` sont marqués comme indisponibles et `predFlagL1` est mis égal à 0. La variable compte de vecteur cinétique de subdivision de sous-macrobloc `subMvCnt` est mise égale à 1.
- Autrement, si `mb_type` est égal à `B_Skip` ou si `B_Direct_16x16` ou `sub_mb_type[mbPartIdx]` est égal à `B_Direct_8x8`, le processus de déduction pour les vecteurs cinétiques luma pour `B_Skip`, `B_Direct_16x16` et `B_Direct_8x8` dans les tranches B du § 8.4.1.2 est invoqué avec `mbPartIdx` et `subMbPartIdx` comme entrée, le résultat étant des vecteurs cinétiques luma `mvL0`, `mvL1`, des indices de référence `refIdxL0`, `refIdxL1`, le compte de vecteur cinétique de subdivision `subMvCnt`, et les fanions d'utilisation de prédiction `predFlagL0` et `predFlagL1`.
- Autrement, pour X remplacé par 0 ou 1 dans les variables `predFlagLX`, `mvLX`, `refIdxLX` et dans `Pred_LX` et dans les éléments syntaxiques `ref_idx_IX` et `mvd_IX` et on applique ce qui suit.
- Les variables `refIdxLX` et `predFlagLX` sont calculées comme suit.

- Si `MbPartPredMode(mb_type, mbPartIdx)` ou `SubMbPredMode(sub_mb_type[mbPartIdx])` est égal à `Pred_LX` ou à `BiPred`,

$$refIdxLX = ref_idx_IX[mbPartIdx] \quad (8-120)$$

$$predFlagLX = 1 \quad (8-121)$$

- Autrement,
 - les variables `refIdxLX` et `predFlagLX` sont spécifiées par

$$\text{refIdxLX} = -1 \quad (8-122)$$

$$\text{predFlagLX} = 0 \quad (8-123)$$
- La variable `subMvCnt` pour le compte de vecteur cinétique de subdivision est mise égale à `predFlagL0 + predFlagL1`.
- Lorsque `predFlagLX` est égal à 1, le processus de déduction pour la prédiction de vecteur cinétique luma du § 8.4.1.3 est invoqué avec `mbPartIdx`, `subMbPartIdx`, `refIdxLX` et le suffixe de liste LX comme entrée et le résultat étant `mvpLX`. Les vecteurs cinétiques luma sont déduits par:

$$\text{mvLX}[0] = \text{mvpLX}[0] + \text{mvd_IX}[\text{mbPartIdx}][\text{subMbPartIdx}][0] \quad (8-124)$$

$$\text{mvLX}[1] = \text{mvpLX}[1] + \text{mvd_IX}[\text{mbPartIdx}][\text{subMbPartIdx}][1] \quad (8-125)$$

Pour la déduction des variables pour les vecteurs cinétiques chroma, on applique ce qui suit. Lorsque `predFlagLX` (avec X étant 0 ou 1) est égal à 1, le processus de déduction pour des vecteurs cinétiques chroma du § 8.4.1.4 est invoqué avec `mvLX` et `refIdxLX` comme entrée et le résultat étant `mvCLX`.

8.4.1.1 Processus de déduction des vecteurs cinétiques luma pour macroblocs sautés en tranches P et SP

Ce processus est invoqué lorsque `mb_type` est égal à `P_Skip`.

Les résultats de ce processus sont le vecteur cinétique `mvL0` et l'indice de référence `refIdxL0`.

L'indice de référence `refIdxL0` pour un macrobloc sauté est déduit comme suit.

$$\text{refIdxL0} = 0 \quad (8-126)$$

Pour la déduction du vecteur cinétique `mvL0` d'un type de macrobloc `P_Skip`, on applique ce qui suit.

- Soit `currSubMbType` égal à `sub_mb_type[0]`. Le processus spécifié au § 8.4.1.3.2 est invoqué avec `mbPartIdx` mis égal à 0, `currSubMbType`, `subMbPartIdx` mis égal à 0 et le suffixe de liste L0 comme entrées et le résultat est alloué à `mbAddrA`, `mbAddrB`, `mvL0A`, `mvL0B`, `refIdxL0A` et `refIdxL0B`.
- La variable `mvL0` est spécifiée comme suit.
 - Si une des conditions suivantes est vraie, les deux composantes du vecteur cinétique `mvL0` sont mises égales à 0:
 - `mbAddrA` n'est pas disponible;
 - `mbAddrB` n'est pas disponible;
 - `refIdxL0A` est égal à 0 et les deux composantes de `mvL0A` sont égales à 0;
 - `refIdxL0B` est égal à 0 et les deux composantes de `mvL0B` sont égales à 0.
 - Autrement, le processus de déduction pour la prédiction de vecteur cinétique luma, comme spécifié au § 8.4.1.3, est invoqué avec `mbPartIdx = 0`, `subMbPartIdx = 0`, `refIdxL0` et le suffixe de liste L0 comme entrée et le résultat est alloué à `mvL0`.

NOTE – Le résultat est directement alloué à `mvL0`, dans la mesure où le prédicant est égal au vecteur cinétique réel.

8.4.1.2 Processus de déduction des vecteurs cinétiques luma pour B_Skip, B_Direct_16x16 et B_Direct_8x8

Ce processus est invoqué lorsque `mb_type` est égal à `B_Skip` ou `B_Direct_16x16` ou que `sub_mb_type[mbPartIdx]` est égal à `B_Direct_8x8`.

Les entrées pour ce processus sont mbPartIdx et subMbPartIdx.

Les résultats de ce processus sont les indices de référence refIdxL0, refIdxL1, les vecteurs cinétiques mvL0 et mvL1, le compte vecteur cinétique de subdivision subMvCnt, et les fanions d'utilisation de liste de prédiction, predFlagL0 et predFlagL1.

Le processus de déduction dépend de la valeur de direct_spatial_mv_pred_flag, qui est présente dans le flux binaire dans la syntaxe d'en-tête de tranche comme spécifié au § 7.3.3 et il est spécifié comme suit.

- Si direct_spatial_mv_pred_flag est égal à 1, le mode dans lequel sont déduits les résultats de ce processus est désigné sous le nom de mode de prédiction direct spatial.
- Autrement (direct_spatial_mv_pred_flag est égal à 0), le mode dans lequel sont déduits les résultats de ce processus est désigné sous le nom de mode de prédiction direct temporel.

Les modes de prédiction directs spatial et temporel utilisent tous deux les vecteurs cinétiques et indices de référence colocalisés, comme spécifié au § 8.4.1.2.1.

Les vecteurs cinétiques et indices de référence sont déduits comme suit.

- Si le mode de prédiction direct spatial est utilisé, le mode de prédiction direct de vecteur cinétique et d'indice de référence spécifié au § 8.4.1.2.2 est utilisé, avec subMvCnt comme résultat.
- Autrement (le mode de prédiction direct temporel est utilisé), le mode de prédiction direct de vecteur cinétique et d'indice de référence spécifié au § 8.4.1.2.3 est utilisé, et la variable subMvCnt est calculée comme suit.
 - Si subMbPartIdx est égal à 0, subMvCnt est mis à 2.
 - Autrement (subMbPartIdx est différent de 0), subMvCnt est mis à 0.

8.4.1.2.1 Processus de déduction pour la subdivision de sous-macrobloccs 4x4 colocalisés

Les entrées pour ce processus sont mbPartIdx et subMbPartIdx.

Les résultats de ce processus sont l'image colPic, le macrobloc colocalisé mbAddrCol, le vecteur cinétique mvCol, l'indice de référence refIdxCol et la variable vertMvScale (qui peut être One_To_One, Frm_To_Fld ou Fld_To_Frm).

Soit firstRefPicL1 l'image de référence mentionnée par RefPicList1[0].

Lorsque firstRefPicL1 est une trame ou une paire de champs complémentaires, soient firstRefPicL1Top et firstRefPicL1Bottom les champs supérieur et inférieur de firstRefPicL1 et soient les variables suivantes spécifiées comme

$$\text{topAbsDiffPOC} = \text{Abs}(\text{DiffPicOrderCnt}(\text{firstRefPicL1Top}, \text{CurrPic})) \quad (8-127)$$

$$\text{bottomAbsDiffPOC} = \text{Abs}(\text{DiffPicOrderCnt}(\text{firstRefPicL1Bottom}, \text{CurrPic})) \quad (8-128)$$

La variable colPic spécifie l'image qui contient le macrobloc colocalisé comme spécifié au Tableau 8-5.

Tableau 8-5 – Spécification de la variable colPic

field_pic_flag	La première entrée dans RefPicList1 est ...	mb_field_decoding_flag	Condition supplémentaire	colPic
1	un champ d'une trame décodée			la trame contenant firstRefPicL1
	un champ décodé			firstRefPicL1
0	une trame décodée	0	topAbsDiffPOC < bottomAbsDiffPOC	le champ supérieur de firstRefPicL1
			topAbsDiffPOC >= bottomAbsDiffPOC	le champ inférieur de firstRefPicL1
	une paire de champs complémentaires	1	(CurrMbAddr & 1) == 0	le champ supérieur de firstRefPicL1
			(CurrMbAddr & 1) != 0	le champ inférieur de firstRefPicL1

Lorsque direct_8x8_inference_flag est égal à 1, subMbPartIdx est mis comme suit.

$$\text{subMbPartIdx} = \text{mbPartIdx} \quad (8-129)$$

Soit PicCodingStruct(X) une fonction avec l'argument X CurrPic ou colPic. Elle est spécifiée au Tableau 8-6.

Tableau 8-6 – Spécification de PicCodingStruct(X)

X est codé avec field_pic_flag égal à ...	mb_adaptive_frame_field_flag	PicCodingStruct(X)
1		FLD
0	0	FRM
0	1	AFRM

Avec luma4x4BlkIdx = mbPartIdx * 4 + subMbPartIdx, le processus de balayage de bloc luma 4x4 inverse comme spécifié au § 6.4.3 est invoqué avec luma4x4BlkIdx comme entrée et (x, y) alloué à (xCol, yCol) comme résultat.

Le Tableau 8-7 spécifie en deux étapes l'adresse de colocalisation de macrobloc mbAddrCol, yM et la variable vertMvScale:

- spécification d'une adresse de macrobloc mbAddrX en fonction de PicCodingStruct(CurrPic) et de PicCodingStruct(colPic).

NOTE – Il n'est pas possible pour les types de codage d'image CurrPic et colPic d'être (FRM, AFRM) ou (AFRM, FRM) parce que ces types de codage d'image doivent être séparés par une image IDR;

- spécification de mbAddrCol, yM et vertMvScale en fonction de mb_field_decoding_flag et de la variable fieldDecodingFlagX, qui est déduite comme suit.

- Si le macrobloc mbAddrX dans l'image colPic est un macrobloc de champ, fieldDecodingFlagX est mise égale à 1.
- Autrement (le macrobloc mbAddrX dans l'image colPic est un macrobloc de trame), fieldDecodingFlagX est mise égale à 0.

Les valeurs non spécifiées dans le Tableau 8-7 indiquent que la valeur de la variable correspondante n'est pas pertinente pour la rangée en cours du tableau.

$$\text{mbAddrCol1} = 2 * \text{PicWidthInMbs} * (\text{CurrMbAddr} / \text{PicWidthInMbs}) + (\text{CurrMbAddr} \% \text{PicWidthInMbs}) + \text{PicWidthInMbs} * (\text{yCol} / 8) \quad (8-130)$$

$$\text{mbAddrCol2} = 2 * \text{CurrMbAddr} + (\text{yCol} / 8) \quad (8-131)$$

$$\text{mbAddrCol3} = 2 * \text{CurrMbAddr} + \text{bottom_field_flag} \quad (8-132)$$

$$\text{mbAddrCol4} = \text{PicWidthInMbs} * (\text{CurrMbAddr} / (2 * \text{PicWidthInMbs})) + (\text{CurrMbAddr} \% \text{PicWidthInMbs}) \quad (8-133)$$

$$\text{mbAddrCol5} = \text{CurrMbAddr} / 2 \quad (8-134)$$

$$\text{mbAddrCol6} = 2 * (\text{CurrMbAddr} / 2) + ((\text{topAbsDiffPOC} < \text{bottomAbsDiffPOC}) ? 0 : 1) \quad (8-135)$$

$$\text{mbAddrCol7} = 2 * (\text{CurrMbAddr} / 2) + (\text{yCol} / 8) \quad (8-136)$$

Tableau 8-7 – Spécification de mbAddrCol, yM et vertMvScale

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

Soient mbPartIdxCol l'indice de subdivision de macrobloc de la subdivision colocalisée et subMbPartIdxCol l'indice de subdivision de sous-macrobloc de la subdivision colocalisée de sous-macrobloc. La subdivision dans le macrobloc mbAddrCol au sein de l'image colPic couvrant l'échantillon (xCol, yM) doit être allouée à mbPartIdxCol et la subdivision de sous-macrobloc au sein de la subdivision mbPartIdxCol couvrant l'échantillon (xCol, yM) dans le macrobloc mbAddrCol au sein de l'image colPic doit être allouée à subMbPartIdxCol.

Les fanions d'utilisation de prédiction predFlagL0Col et predFlagL1Col sont respectivement mis égaux à PredFlagL0[mbPartIdxCol] et PredFlagL1[mbPartIdxCol], qui sont les fanions d'utilisation de prédiction qui ont été alloués à la subdivision de macrobloc mbAddrCol\mbPartIdxCol au sein de l'image colPic.

Le vecteur cinétique mvCol et l'indice de référence refIdxCol sont déduits comme suit.

- Si le macrobloc mbAddrCol est codé en mode de prédiction intramacrobloc ou que les deux fanions d'utilisation de prédiction, predFlagL0Col et predFlagL1Col, sont égaux à 0, les deux composantes de mvCol sont mises égales à 0 et refIdxCol est mis égal à -1.
- Autrement, on applique ce qui suit.
 - Si predFlagL0Col est égal à 1, le vecteur cinétique mvCol et l'indice de référence refIdxCol sont mis respectivement égaux à MvL0[mbPartIdxCol][subMbPartIdxCol] et RefIdxL0[mbPartIdxCol], qui sont le vecteur cinétique mvL0 et l'indice de référence refIdxL0 qui ont été alloués à la subdivision de (sous-)macrobloc mbAddrCol\mbPartIdxCol\subMbPartIdxCol au sein de l'image colPic.
 - Autrement (predFlagL0Col est égal à 0 et predFlagL1Col est égal à 1), le vecteur cinétique mvCol et l'indice de référence refIdxCol sont respectivement mis égaux à MvL1[mbPartIdxCol][subMbPartIdxCol] et RefIdxL1[mbPartIdxCol], qui sont le vecteur cinétique mvL1 et l'indice de référence refIdxL1 qui ont été alloués à la subdivision de (sous-)macrobloc mbAddrCol\mbPartIdxCol\subMbPartIdxCol au sein de l'image colPic.

8.4.1.2.2 Processus de déduction pour le mode de prédiction de vecteur cinétique et d'indice de référence luma direct spatial

Ce processus est invoqué lorsque direct_spatial_mv_pred_flag est égal à 1 et qu'une des conditions suivantes est Vraie.

- mb_type est égal à B_Skip
- mb_type est égal à B_Direct_16x16
- sub_mb_type[mbPartIdx] est égal à B_Direct_8x8.

Les entrées pour ce processus sont mbPartIdx, subMbPartIdx.

Les résultats de ce processus sont les indices de référence refIdxL0, refIdxL1, les vecteurs cinétiques mvL0 et mvL1, le compte de vecteur cinétique de subdivision subMvCnt, et les fanions d'utilisation de liste de prédiction, predFlagL0 et predFlagL1.

Les indices de référence refIdxL0 et refIdxL1 et la variable directZeroPredictionFlag sont déduits en appliquant les étapes ordonnées suivantes:

1. Soit la variable currSubMbType égale à sub_mb_type[mbPartIdx].
2. Le processus spécifié au § 8.4.1.3.2 est invoqué avec mbPartIdx = 0, subMbPartIdx = 0, currSubMbType et le suffixe de liste L0 comme entrées et le résultat est alloué aux vecteurs cinétiques mvL0N et aux indices de référence refIdxL0N avec N remplacé par A, B ou C.
3. Le processus spécifié au § 8.4.1.3.2 est invoqué avec mbPartIdx = 0, subMbPartIdx = 0, currSubMbType et le suffixe de liste L1 comme entrées et le résultat est alloué aux vecteurs cinétiques mvL1N et aux indices de référence refIdxL1N avec N remplacé par A, B ou C.

NOTE – Les vecteurs cinétiques mvL0N, mvL1N et les indices de référence refIdxL0N, refIdxL1N sont identiques pour toutes les subdivisions de sous-macrobloc 4x4 d'un macrobloc.

4. Les indices de référence refIdxL0, refIdxL1 et directZeroPredictionFlag sont déduits par:

$$\text{refIdxL0} = \text{MinPositive}(\text{refIdxL0A}, \text{MinPositive}(\text{refIdxL0B}, \text{refIdxL0C})) \quad (8-137)$$

$$\text{refIdxL1} = \text{MinPositive}(\text{refIdxL1A}, \text{MinPositive}(\text{refIdxL1B}, \text{refIdxL1C})) \quad (8-138)$$

$$\text{directZeroPredictionFlag} = 0 \quad (8-139)$$

où

$$\text{MinPositive}(x, y) = \begin{cases} \text{Min}(x, y) & \text{si } x \geq 0 \text{ et } y \geq 0 \\ \text{Max}(x, y) & \text{autrement} \end{cases} \quad (8-140)$$

5. Lorsque les deux indices de référence refIdxL0 et refIdxL1 sont inférieurs à 0:

$$\text{refIdxL0} = 0 \quad (8-141)$$

$$\text{refIdxL1} = 0 \quad (8-142)$$

$$\text{directZeroPredictionFlag} = 1 \quad (8-143)$$

Le processus spécifié au § 8.4.1.2.1 est invoqué avec mbPartIdx, subMbPartIdx est donné en entrée et le résultat est alloué à refIdxCol et mvCol.

La variable colZeroFlag est déduite comme suit.

- Si toutes les conditions suivantes sont Vraies, colZeroFlag est mis égal à 1.
 - l'image de référence désignée par RefPicList1[0] est une image de référence à court terme
 - refIdxCol est égal à 0
 - les deux composantes de vecteur cinétique mvCol[0] et mvCol[1] sont dans la gamme de -1 à 1 en unités spécifiées comme suit.
 - si le macrobloc colocalisé est un macrobloc de trame, les unités de mvCol[0] et mvCol[1] sont des unités de quart d'échantillon de trame luma;
 - autrement (le macrobloc colocalisé est un macrobloc de champ), les unités de mvCol[0] et mvCol[1] sont des unités de quart d'échantillon de champ luma.

NOTE – Pour les besoins de détermination de la condition ci-dessus, la valeur mvCol[1] n'est pas étalonnée pour une utilisation des unités d'un vecteur cinétique pour le macrobloc en cours dans les cas où le macrobloc en cours est un macrobloc de trame et que le macrobloc colocalisé est un macrobloc de champ ou lorsque le macrobloc en cours est un macrobloc de champ et que le macrobloc colocalisé est un macrobloc de trame. Cet aspect diffère de l'utilisation de mvCol[1] dans le mode direct temporel comme spécifié au § 8.4.1.2.3, qui applique l'étalonnage au vecteur cinétique du macrobloc colocalisé pour utiliser les mêmes unités que celles d'un vecteur cinétique pour le macrobloc en cours, utilisant dans ces cas l'équation 8-146 ou l'équation 8-147.

- Autrement, colZeroFlag est mis égal à 0.

Les vecteur cinétiques mvLX (X étant 0 ou 1) sont déduits comme suit.

- Si l'une des conditions suivantes est Vraie, les deux composantes du vecteur cinétique mvLX sont mises égales à 0.
 - directZeroPredictionFlag est égal à 1
 - refIdxLX est inférieur à 0
 - refIdxLX est égal à 0 et colZeroFlag est égal à 1
- Autrement, le processus spécifié au § 8.4.1.3 est invoqué avec mbPartIdx = 0, subMbPartIdx = 0, refIdxLX et le suffixe de liste LX en entrée et le résultat est alloué à mvLX.

NOTE – Dans le cas immédiatement ci-dessus, le vecteur cinétique mvLX de retour est identique pour toutes les subdivisions 4x4 de sous-macrobloc d'un macrobloc.

Les fanions d'utilisation de prédiction predFlagL0 et predFlagL1 doivent être déduits comme spécifié en utilisant le Tableau 8-8.

Tableau 8-8 – Allocation des fanions d'utilisation de prédiction

refIdxL0	refIdxL1	predFlagL0	predFlagL1
>= 0	>= 0	1	1
>= 0	< 0	1	0
< 0	>= 0	0	1

La variable subMvCnt est calculée comme suit.

- Si subMbPartIdx est différent de 0 ou direct_8x8_inference_flag est égal à 0, subMvCnt est mis égal à 0.
- Autrement (subMbPartIdx est égal à 0 et direct_8x8_inference_flag est égal à 1), subMvCnt est mis égal à predFlagL0 + predFlagL1.

8.4.1.2.3 Processus de déduction pour mode de prédiction direct temporel de vecteur cinétique et d'indice de référence luma

Ce processus est invoqué lorsque direct_spatial_mv_pred_flag est égal à 0 et si une des conditions suivantes est Vraie:

- mb_type est égal à B_Skip
- mb_type est égal à B_Direct_16x16
- sub_mb_type[mbPartIdx] est égal à B_Direct_8x8.

Les entrées pour ce processus sont mbPartIdx et subMbPartIdx.

Les résultats de ce processus sont les vecteurs cinétiques mvL0 et mvL1, les indices de référence refIdxL0 et refIdxL1 et les fanions d'utilisation de liste de prédiction, predFlagL0 et predFlagL1.

Le processus spécifié au § 8.4.1.2.1 est invoqué avec mbPartIdx, subMbPartIdx est donné en entrée et le résultat est alloué à colPic, mbAddrCol, mvCol, refIdxCol et vertMvScale.

Les indices de référence refIdxL0 et refIdxL1 sont déduits comme suit.

$$\text{refIdxL0} = ((\text{refIdxCol} < 0) ? 0 : \text{MapColToList0}(\text{refIdxCol})) \quad (8-144)$$

$$\text{refIdxL1} = 0 \quad (8-145)$$

NOTE – Si le macrobloc en cours est un macrobloc de champ, refIdxL0 et refIdxL1 sont les indices d'une liste de champs; autrement (le macrobloc en cours est un macrobloc de trame), refIdxL0 et refIdxL1 sont les indices d'une liste de trames ou paire de champs de référence complémentaires.

Soit refPicCol une trame, un champ ou une paire de champs complémentaires qui sont désignés par l'indice de référence refIdxCol lors du décodage du macrobloc colocalisé mbAddrCol au sein de l'image colPic. La fonction MapColToList0(refIdxCol) est spécifiée comme suit:

- Si vertMvScale est égal à One_To_One, on applique ce qui suit.
 - Si field_pic_flag est égal à 0 et si le macrobloc en cours est un macrobloc de champ, on applique ce qui suit.
 - Soit refIdxL0Frm la plus faible valeur d'indice de référence dans la liste d'images de référence en cours RefPicList0 qui mentionne la trame ou paire des champs complémentaires qui contient le champ refPicCol. RefPicList0 doit contenir la variable PicNum ou LongTermPicNum qui mentionne la trame ou paire de champs complémentaires qui contient refPicCol. La valeur retournée de MapColToList0() est spécifiée comme suit.
 - Si le champ auquel se réfère refIdxCol a la même parité que le macrobloc en cours, MapColToList0(refIdxCol) retourne l'indice de référence (refIdxL0Frm << 1).
 - Autrement (le champ auquel se réfère refIdxCol a la parité opposée à celle du macrobloc en cours), MapColToList0(refIdxCol) retourne l'indice de référence ((refIdxL0Frm << 1) + 1).

- Autrement (`field_pic_flag` est égal à 1 ou le macrobloc en cours est un macrobloc de trame), `MapColToList0(refIdxCol)` retourne la plus faible valeur d'indice de référence `refIdxL0` dans la liste d'images de référence en cours `RefPicList0` qui mentionne `refPicCol`. `RefPicList0` doit contenir une variable `PicNum` ou `LongTermPicNum` qui mentionne `refPicCol`.
- Autrement, si `vertMvScale` est égal à `Frm_To_Fld`, on applique ce qui suit.
 - Si `field_pic_flag` est égal à 0, soit `refIdxL0Frm` la plus faible valeur d'indice de référence dans la liste d'images de référence en cours `RefPicList0` qui mentionne `refPicCol`. `MapColToList0(refIdxCol)` retourne l'indice de référence (`refIdxL0Frm << 1`). `RefPicList0` doit contenir une variable `PicNum` ou `LongTermPicNum` qui mentionne `refPicCol`.
 - Autrement (`field_pic_flag` est égal à 1), `MapColToList0(refIdxCol)` retourne la plus faible valeur d'indice de référence `refIdxL0` dans la liste d'images de référence en cours `RefPicList0` qui mentionne le champ de `refPicCol` avec la même parité que l'image en cours `CurrPic`. `RefPicList0` doit contenir une variable `PicNum` ou `LongTermPicNum` qui mentionne le champ de `refPicCol` avec la même parité que l'image en cours `CurrPic`.
- Autrement (`vertMvScale` est égal à `Fld_To_Frm`), `MapColToList0(refIdxCol)` retourne la plus faible valeur d'indice de référence `refIdxL0` dans la liste d'images de référence en cours `RefPicList0` qui mentionne la trame ou paire de champs complémentaires qui contient `refPicCol`. `RefPicList0` doit contenir la variable `PicNum` ou `LongTermPicNum` qui mentionne la trame ou paire de champs complémentaires qui contient `refPicCol`.

NOTE – Une image décodée de référence, qui était marquée comme "utilisé pour référence à court terme" lorsqu'elle était mentionnée dans le processus de décodage de l'image contenant le macrobloc colocalisé, peut avoir été modifiée pour être marquée comme "utilisé pour référence à long terme" avant d'être utilisée pour référence pour l'interprédiction utilisant le mode de prédiction direct pour le macrobloc en cours.

En fonction de la valeur de `vertMvScale`, la composante verticale de `mvCol` est modifiée comme suit.

- Si `vertMvScale` est égal à `Frm_To_Fld`

$$mvCol[1] = mvCol[1] / 2 \quad (8-146)$$
- Autrement, si `vertMvScale` est égal à `Fld_To_Frm`

$$mvCol[1] = mvCol[1] * 2 \quad (8-147)$$
- Autrement (`vertMvScale` est égal à `One_To_One`), `mvCol[1]` reste inchangée.

Les variables `currPicOrField`, `pic0` et `pic1`, sont calculées comme suit:

- Si `field_pic_flag` est égal à 0 et si le macrobloc en cours est un macrobloc de champ, on applique ce qui suit.
 - `currPicOrField` est le champ de l'image en cours `CurrPic` qui a la même parité que le macrobloc en cours.
 - `pic1` est le champ de `RefPicList1[0]` qui a la même parité que le macrobloc en cours.
 - Soit `frame0` la trame ou paire de champs complémentaires mentionnée par `RefPicList0[refIdxL0/2]`.
 - La variable `pic0` est calculée comme suit.
 - Si `refIdxL0 % 2` est égal à 0, `pic0` est le champ de `frame0` qui a la même parité que le macrobloc en cours.
 - Autrement (`refIdxL0 % 2` est différent de 0), `pic0` est le champ de `frame0` qui a la parité opposée à celle du macrobloc en cours.

- Autrement (`field_pic_flag` est égal à 1 ou le macrobloc en cours est un macrobloc de trame), `currPicOrField` est l'image en cours `CurrPic`, `pic1` est l'image de référence décodée mentionnée par `RefPicList1[0]`, et `pic0` est l'image de référence décodée mentionnée par `RefPicList0[refIdxL0]`.

Les deux vecteurs cinétiques `mvL0` et `mvL1` sont déduits comme suit pour chaque subdivision 4x4 de sous-macrobloc du macrobloc en cours.

NOTE – Il arrive fréquemment que de nombreuses subdivisions 4x4 de sous-macrobloccs partagent les mêmes vecteurs cinétiques et images de référence. Dans ces cas, la compensation de mouvement de mode direct temporel peut calculer les valeurs d'échantillon d'interprédiction dans des unités plus grandes que les blocs d'échantillon luma 4x4. Par exemple, lorsque `direct_8x8_inference_flag` est égal à 1, au moins chaque quadrant d'échantillon 8x8 luma du macrobloc partage les mêmes vecteurs cinétiques et images de référence.

- Si l'indice de référence `refIdxL0` se réfère à une image à long terme ou si `DiffPicOrderCnt(pic1, pic0)` est égal à 0, les vecteurs cinétiques `mvL0`, `mvL1` pour la subdivision de mode direct sont déduits par

$$mvL0 = mvCol \quad (8-148)$$

$$mvL1 = 0 \quad (8-149)$$

- Autrement, les vecteurs cinétiques `mvL0`, `mvL1` sont déduits comme versions étalonnées du vecteur cinétique `mvCol` de la subdivision colocalisée de sous-macrobloc comme spécifié ci-dessous (voir la Figure 8-2)

$$tx = (16384 + Abs(td / 2)) / td \quad (8-150)$$

$$DistScaleFactor = Clip3(-1024, 1023, (tb * tx + 32) \gg 6) \quad (8-151)$$

$$mvL0 = (DistScaleFactor * mvCol + 128) \gg 8 \quad (8-152)$$

$$mvL1 = mvL0 - mvCol \quad (8-153)$$

où `tb` et `td` sont calculés comme suit.

$$tb = Clip3(-128, 127, DiffPicOrderCnt(currPicOrField, pic0)) \quad (8-154)$$

$$td = Clip3(-128, 127, DiffPicOrderCnt(pic1, pic0)) \quad (8-155)$$

NOTE – `mvL0` et `mvL1` ne peuvent pas excéder les gammes spécifiées à l'Annexe A.

Les fanions d'utilisation de prédiction `predFlagL0` et `predFlagL1` sont tous deux mis égaux à 1.

La Figure 8-2 illustre la déduction du vecteur cinétique temporel en mode direct lorsque l'image en cours est temporellement entre l'image de référence de liste 0 et l'image de référence de liste 1.

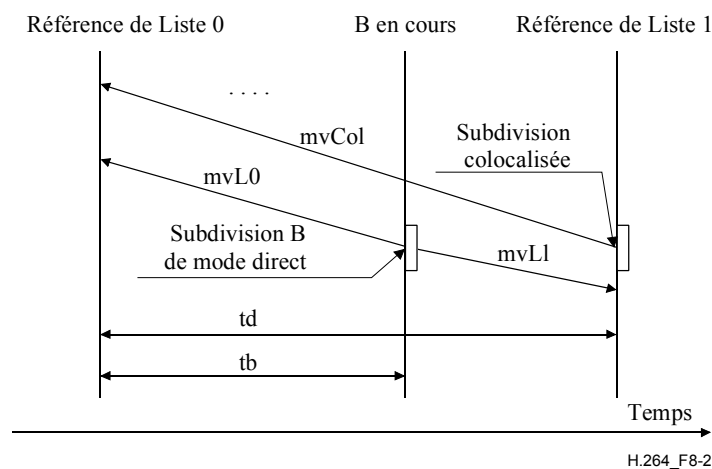


Figure 8-2 – Exemple de déduction de vecteur cinétique temporel en mode direct (pour information)

8.4.1.3 Processus de déduction pour la prédiction de vecteur cinétique luma

Les entrées pour ce processus sont:

- l'indice de subdivision de macrobloc mbPartIdx;
- l'indice de subdivision de sous-macrobloc subMbPartIdx;
- le suffixe de liste LX;
- l'indice de référence de la subdivision en cours refIdxLX.

Le résultat de ce processus est la prédiction.mvpLX du vecteur cinétique mvLX.

Soit currSubMbType mis égal à sub_mb_type[mbPartIdx].

Le processus de déduction pour les blocs du voisinage pour les données de mouvement du § 8.4.1.3.2 est invoqué avec mbPartIdx, subMbPartIdx, currSubMbType et le suffixe de liste LX comme entrée et avec mbAddrN\mbPartIdxN\subMbPartIdxN, les indices de référence refIdxLXN et les vecteurs cinétiques mvLXN avec N remplacé par A, B ou C comme résultat.

Le processus de déduction pour la prédiction de vecteur cinétique luma médian du § 8.4.1.3.1 est invoqué avec mbAddrN\mbPartIdxN\subMbPartIdxN, mvLXN, refIdxLXN avec N remplacé par A, B ou C et refIdxLX comme entrée et.mvpLX comme résultat, à moins qu'une des conditions suivantes ne soit Vraie:

- MbPartWidth(mb_type) est égal à 16, MbPartHeight(mb_type) est égal à 8, mbPartIdx est égal à 0 et refIdxLXB est égal à refIdxLX,

$$\text{mvpLX} = \text{mvLXB} \quad (8-156)$$

- MbPartWidth(mb_type) est égal à 16, MbPartHeight(mb_type) est égal à 8, mbPartIdx est égal à 1 et refIdxLXA est égal à refIdxLX,

$$\text{mvpLX} = \text{mvLXA} \quad (8-157)$$

- MbPartWidth(mb_type) est égal à 8, MbPartHeight(mb_type) est égal à 16, mbPartIdx est égal à 0 et refIdxLXA est égal à refIdxLX,

$$\text{mvpLX} = \text{mvLXA} \quad (8-158)$$

- MbPartWidth(mb_type) est égal à 8, MbPartHeight(mb_type) est égal à 16, mbPartIdx est égal à 1 et refIdxLXC est égal à refIdxLX,

$$\text{mvpLX} = \text{mvLXC} \quad (8-159)$$

La Figure 8-3 illustre la prédiction non médiane comme décrit ci-dessus.

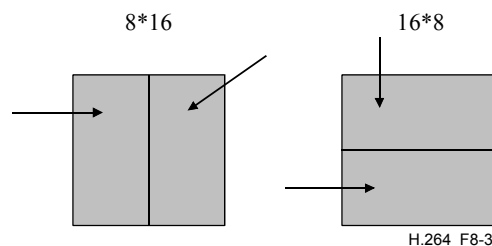


Figure 8-3 – Prédiction de segmentation directionnelle (pour information)

8.4.1.3.1 Processus de déduction pour la prédiction de vecteur cinétique luma médian

Les entrées pour ce processus sont:

- les subdivisions voisines mbAddrN\mbPartIdxN\subMbPartIdxN (avec N remplacé par A, B ou C);
- les vecteurs cinétiques mvLXN (avec N remplacé par A, B ou C) des subdivisions voisines;

- les indices de référence refIdxLXN (avec N remplacé par A, B ou C) des subdivisions voisines;
- l'indice de référence refIdxLX de la subdivision en cours.

Le résultat de ce processus est la prédiction de vecteur cinétique mvPLX.

La variable mvPLX est déduite comme suit:

- Lorsque les deux subdivisions mbAddrB\mbPartIdxB\subMbPartIdxB et mbAddrC\mbPartIdxC\subMbPartIdxC sont indisponibles et que mbAddrA\mbPartIdxA\subMbPartIdxA est disponible

$$mvLXB = mvLXA \quad (8-160)$$

$$mvLXC = mvLXA \quad (8-161)$$

$$refIdxLXB = refIdxLXA \quad (8-162)$$

$$refIdxLXC = refIdxLXA \quad (8-163)$$

- En fonction des indices de référence refIdxLXA, refIdxLXB ou refIdxLXC, on applique ce qui suit.
 - Si un et seulement un des indices de référence refIdxLXA, refIdxLXB ou refIdxLXC est égal à l'indice de référence refIdxLX de la subdivision en cours, on applique ce qui suit. Soit refIdxLXN l'indice de référence qui est égal à refIdxLX, le vecteur cinétique mvLXN est alloué à la prédiction de vecteur cinétique mvPLX:

$$mvPLX = mvLXN \quad (8-164)$$

- Autrement, chaque composant de la prédiction de vecteur cinétique mvPLX est donné par la médiane des composantes mvLXA, mvLXB et mvLXC du vecteur correspondant du vecteur cinétique:

$$mvPLX[0] = \text{Median}(mvLXA[0], mvLXB[0], mvLXC[0]) \quad (8-165)$$

$$mvPLX[1] = \text{Median}(mvLXA[1], mvLXB[1], mvLXC[1]) \quad (8-166)$$

8.4.1.3.2 Processus de déduction pour les données de mouvement des subdivisions du voisinage

Les entrées pour ce processus sont:

- l'indice de subdivision de macrobloc mbPartIdx;
- l'indice de subdivision de sous-macrobloc subMbPartIdx;
- le type de sous-macrobloc en cours currSubMbType;
- le suffixe de liste LX.

Les résultats de ce processus sont (avec N remplacé par A, B ou C)

- mbAddrN\mbPartIdxN\subMbPartIdxN spécifiant les subdivisions du voisinage;
- les vecteurs cinétiques mvLXN des subdivisions du voisinage;
- les indices de référence refIdxLXN des subdivisions du voisinage.

Les subdivisions mbAddrN\mbPartIdxN\subMbPartIdxN avec N étant A, B ou C sont déduites selon les étapes ordonnées suivantes:

1. Soit mbAddrD\mbPartIdxD\subMbPartIdxD des variables spécifiant une subdivision supplémentaire du voisinage.
2. Le processus du § 6.4.7.5 est invoqué avec mbPartIdx, currSubMbType et subMbPartIdx comme entrées et le résultat est alloué à mbAddrN\mbPartIdxN\subMbPartIdxN avec N remplacé par A, B, C ou D.

3. Lorsque la subdivision $mbAddrC\mbPartIdxC\subMbPartIdxC$ n'est pas disponible, on applique ce qui suit:

$$mbAddrC = mbAddrD \quad (8-167)$$

$$mbPartIdxC = mbPartIdxD \quad (8-168)$$

$$subMbPartIdxC = subMbPartIdxD \quad (8-169)$$

Les vecteurs cinétiques $mvLXN$ et les indices de référence $refIdxLXN$ (avec N étant A , B ou C) sont déduits comme suit.

– Si la subdivision de macrobloc ou subdivision de sous-macrobloc $mbAddrN\mbPartIdxN\subMbPartIdxN$ n'est pas disponible ou que $mbAddrN$ est codé en mode d'intraprédiction ou que $predFlagLX$ de $mbAddrN\mbPartIdxN\subMbPartIdxN$ est égal à 0, les deux composantes de $mvLXN$ sont mises égales à 0 et $refIdxLXN$ est mis égal à -1 .

– Autrement, on applique ce qui suit.

– Le vecteur cinétique $mvLXN$ et l'indice de référence $refIdxLXN$ sont respectivement mis égaux à $MvLX[mbPartIdxN][subMbPartIdxN]$ et $RefIdxLX[mbPartIdxN]$, qui sont le vecteur cinétique $mvLX$ et l'indice de référence $refIdxLX$ qui ont été alloués à la subdivision de (sous-)macrobloc $mbAddrN\mbPartIdxN\subMbPartIdxN$.

– Les variables $mvLXN[1]$ et $refIdxLXN$ sont ensuite traitées comme suit.

– Si le macrobloc en cours est un macrobloc de champ et que le macrobloc $mbAddrN$ est un macrobloc de trame

$$mvLXN[1] = mvLXN[1] / 2 \quad (8-170)$$

$$refIdxLXN = refIdxLXN * 2 \quad (8-171)$$

– Autrement, si le macrobloc en cours est un macrobloc de trame et que le macrobloc $mbAddrN$ est un macrobloc de champ

$$mvLXN[1] = mvLXN[1] * 2 \quad (8-172)$$

$$refIdxLXN = refIdxLXN / 2 \quad (8-173)$$

– Autrement, la composante verticale de vecteur cinétique $mvLXN[1]$ et l'indice de référence $refIdxLXN$ restent inchangés.

8.4.1.4 Processus de déduction pour des vecteurs cinétiques chroma

Les entrées pour ce processus sont un vecteur cinétique luma $mvLX$ et un indice de référence $refIdxLX$.

Les résultats de ce processus sont un vecteur cinétique chroma $mvCLX$.

Un vecteur cinétique chroma est déduit du vecteur cinétique luma correspondant. Dans la mesure où la précision des vecteurs cinétiques luma est d'un quart d'échantillon et où les échantillons chroma ont la moitié de la résolution de celle des luma, la précision des vecteurs cinétiques chroma est d'un huitième d'échantillon, c'est-à-dire qu'une valeur de 1 pour le vecteur cinétique chroma se réfère à un déplacement d'un huitième d'échantillon.

NOTE – Par exemple lorsque le vecteur luma s'applique à des échantillons luma 8×16 , le vecteur chroma correspondant s'applique aux échantillons chroma 4×8 et lorsque le vecteur luma s'applique aux échantillons luma 4×4 , le vecteur chroma correspondant s'applique aux échantillons chroma 2×2 .

Pour la déduction du vecteur cinétique $mvCLX$, on applique ce qui suit.

- Si le macrobloc en cours est un macrobloc de trame, les composantes horizontales et verticales du vecteur cinétique chroma $mvCLX$ sont déduites en multipliant les composantes correspondantes du vecteur cinétique luma $mvLX$ par 2, en mappant des unités d'un quart d'échantillon $mvLX$ à des unités d'un huitième d'échantillon $mvCLX$.

$$mvCLX[0] = mvLX[0] \quad (8-174)$$

$$mvCLX[1] = mvLX[1] \quad (8-175)$$

- Autrement (le macrobloc en cours est un macrobloc de champ), seule la composante horizontale du vecteur cinétique chroma $mvCLX[0]$ est déduite en utilisant l'équation 8-174. La composante verticale du vecteur cinétique chroma $mvCLX[1]$ dépend de la parité du champ en cours ou du macrobloc en cours et l'image de référence, qui est désignée par l'indice de référence $refIdxLX$. $mvCLX[1]$ est déduite de $mvLX[1]$ conformément au Tableau 8-9.

Tableau 8-9 – Déduction de la composante verticale du vecteur chroma dans le mode de codage de champ

Conditions de parité		$mvCLX[1]$
Image de référence ($refIdxLX$)	Champ en cours (image/macrobloc)	
Champ supérieur	Champ inférieur	$mvLX[1] + 2$
Champ inférieur	Champ supérieur	$mvLX[1] - 2$
Autrement		$mvLX[1]$

8.4.2 Processus de décodage pour échantillons d'interprédiction

Les entrées pour ce processus sont

- une subdivision de macrobloc $mbPartIdx$,
- une subdivision de sous-macrobloc $subMbPartIdx$
- des variables spécifiant la largeur et hauteur de subdivision, $partWidth$ et $partHeight$
- des vecteurs cinétiques luma $mvL0$ et $mvL1$ et des vecteurs cinétiques chroma $mvCL0$ et $mvCL1$
- des indices de référence $refIdxL0$ et $refIdxL1$
- des fanions d'utilisation de liste de prédiction, $predFlagL0$ et $predFlagL1$

Les résultats de ce processus sont

- les échantillons d'interprédiction $predPart$, qui sont une matrice $(partWidth) \times (partHeight)$ $predPart_L$ d'échantillons luma de prédiction, et deux matrices $(partWidth/2) \times (partHeight/2)$ $predPart_{Cb}$, $predPart_{Cr}$ d'échantillons chroma de prédiction, une pour chacun des composants chroma Cb et Cr .

Soient $predPartL0_L$ et $predPartL1_L$ des matrices $(partWidth) \times (partHeight)$ pour les valeurs d'échantillon luma prédits et $predPartL0_{Cb}$, $predPartL1_{Cb}$, $predPartL0_{Cr}$, et $predPartL1_{Cr}$ les matrices $(partWidth/2) \times (partHeight/2)$ des valeurs d'échantillon chroma prédits.

Pour LX remplacé par $L0$ ou $L1$ dans les variables $predFlagLX$, $RefPicListX$, $refIdxLX$, $refPicLX$, $predPartLX$, on spécifie ce qui suit.

Lorsque $predFlagLX$ est égal à 1, on applique ce qui suit.

- La trame de référence consistant en une matrice ordonnée à deux dimensions $refPicLX_L$ d'échantillons luma et deux matrices ordonnées à deux dimensions $refPicLX_{Cb}$ et $refPicLX_{Cr}$ d'échantillons chroma est déduite en invoquant le processus spécifié au § 8.4.2.1 avec $refIdxLX$ et $RefPicListX$ donnés en entrée.

- Les matrices predPartLX_L , predPartLX_{Cb} , et predPartLX_{Cr} sont déduites en invoquant le processus spécifié au § 8.4.2.2 avec la subdivision en cours spécifiée par $\text{mbPartIdx}\backslash\text{subMbPartIdx}$, les vecteurs cinétiques mvLX , mvCLX , et les matrices de référence avec refPicLX_L , refPicLX_{Cb} , et refPicLX_{Cr} données en entrée.

Pour C remplacé par L, Cb, ou Cr, la matrice predPart_C des échantillons de prédiction du composant C est déduite en invoquant le processus spécifié au § 8.4.2.3 avec la subdivision en cours spécifiée par mbPartIdx et subMbPartIdx et les matrices predPartL0_C et predPartL1_C ainsi que predFlagL0 et predFlagL1 données en entrée.

8.4.2.1 Processus de sélection de l'image de référence

L'entrée pour ce processus est un indice de référence refIdxLX .

Le résultat de ce processus est une image de référence consistant en une matrice à deux dimensions d'échantillons luma refPicLX_L et deux matrices à deux dimensions d'échantillons chroma refPicLX_{Cb} et refPicLX_{Cr} .

La liste d'images de référence RefPicListX est une liste de variables PicNum (pour des images de référence à court terme) et LongTermPicNum (pour des images de référence à long terme) de trames de référence précédemment décodées, de paires de champs de référence complémentaires, ou de champs de référence non appariés qui ont été marqués comme "utilisé pour référence" comme spécifié au § 8.2.5.

En fonction de field_pic_flag , la signification de PicNum et LongTermPicNum est spécifiée comme suit:

- Si field_pic_flag est égal à 1, toutes les entrées de la RefPicListX sont des variables PicNum et LongTermPicNum de champs de référence décodés ou de champs de trames de références décodées.
- Autrement (field_pic_flag est égal à 0), toutes les entrées de RefPicListX sont des variables PicNum et LongTermPicNum de trames de référence décodées ou de paires de champs de référence complémentaires.

La liste d'images de référence RefPicListX est déduite comme spécifié au § 8.2.4.

Pour la déduction de l'image de référence, on applique ce qui suit.

- Si field_pic_flag est égal à 1, le champ de référence ou champ d'une trame de référence désigné par $\text{PicNum} = \text{RefPicListX}[\text{refIdxLX}]$ ou $\text{LongTermPicNum} = \text{RefPicListX}[\text{refIdxLX}]$ doit être le résultat. Le champ de référence ou champ d'une trame de référence résultant consiste en une matrice $(\text{PicWidthInSamples}_L) \times (\text{PicHeightInSamples}_L)$ d'échantillons luma refPicLX_L et deux matrices $(\text{PicWidthInSamples}_C) \times (\text{PicHeightInSamples}_C)$ d'échantillons chroma refPicLX_{Cb} et refPicLX_{Cr} .
- Autrement (field_pic_flag est égal à 0), on applique ce qui suit.
 - Si le macrobloc en cours est un macrobloc de trame, la trame de référence ou paire de champs de référence complémentaires désignée par $\text{PicNum} = \text{RefPicListX}[\text{refIdxLX}]$ ou $\text{LongTermPicNum} = \text{RefPicListX}[\text{refIdxLX}]$ doit être le résultat. La trame de référence ou paire de champs de référence complémentaires résultante consiste en une matrice $(\text{PicWidthInSamples}_L) \times (\text{PicHeightInSamples}_L)$ d'échantillons luma refPicLX_L et deux matrices $(\text{PicWidthInSamples}_C) \times (\text{PicHeightInSamples}_C)$ d'échantillons chroma refPicLX_{Cb} et refPicLX_{Cr} .
 - Autrement (le macrobloc en cours est un macrobloc de champ), on applique ce qui suit.
 - Soit refFrame la trame de référence ou paire de champs de référence complémentaires qui est désignée par $\text{PicNum} = \text{RefPicListX}[\text{refIdxLX} / 2]$ ou $\text{LongTermPicNum} = \text{RefPicListX}[\text{refIdxLX} / 2]$.

- Le champ de refFrame est choisi comme suit.
 - Si refIdxLX % 2 est égal à 0, le champ de refFrame qui a la même parité que le macrobloc en cours doit être le résultat.
 - Autrement (refIdxLX % 2 est égal à 1), le champ de refFrame qui a la parité opposée à celle du macrobloc en cours doit être le résultat.
- Le champ de référence ou champ d'une trame de référence résultant consiste en une matrice $(\text{PicWidthInSamples}_L) \times (\text{PicHeightInSamples}_L/2)$ d'échantillons luma refPicLX_L et deux matrices $(\text{PicWidthInSamples}_C) \times (\text{PicHeightInSamples}_C/2)$ d'échantillons chroma refPicLX_{Cb} et refPicLX_{Cr}.

Les matrices d'échantillons d'image de référence refPicLX_L, refPicLX_{Cb}, refPicLX_{Cr} correspondent aux matrices d'échantillons décodés S_L, S_{Cb}, S_{Cr} déduites au § 8.7 pour les images décodées précédentes.

8.4.2.2 Processus d'interpolation d'échantillon fractionnaire

Les entrées pour ce processus sont

- la subdivision en cours donnée par son indice de subdivision mbPartIdx et son indice de subdivision de sous-macrobloc subMbPartIdx;
- la largeur partWidth, et la hauteur partHeight de cette subdivision en unités d'échantillon luma;
- un vecteur cinétique luma mvLX donné en unités de quart d'échantillon luma;
- un vecteur cinétique chroma mvCLX donné en unités de huitième d'échantillon chroma;
- les matrices d'échantillons d'image de référence choisies refPicLX_L, refPicLX_{Cb}, et refPicLX_{Cr}.

Les résultats de ce processus sont

- une matrice $(\text{partWidth}) \times (\text{partHeight})$ predPartLX_L de valeurs d'échantillons luma de prédiction; et
- deux matrices $(\text{partWidth}/2) \times (\text{partHeight}/2)$ predPartLX_{Cb}, et predPartLX_{Cr} de valeurs d'échantillons chroma de prédiction.

Soit (x_{A_L}, y_{A_L}) la localisation donnée en unités d'échantillons complets de l'échantillon luma supérieur gauche de la subdivision en cours donnée par mbPartIdx\subMbPartIdx relative à la localisation de l'échantillon luma supérieur gauche de la matrice bidimensionnelle donnée d'échantillons luma.

Soit (x_{Int_L}, y_{Int_L}) la localisation luma donnée en unités d'échantillons complets et (x_{Frac_L}, y_{Frac_L}) un décalage donné en unités de quart d'échantillon. Ces variables ne sont utilisées qu'au sein du présent paragraphe pour spécifier les localisations générales d'échantillon fractionnaire au sein des matrices d'échantillons de référence refPicLX_L, refPicLX_{Cb}, et refPicLX_{Cr}.

Pour chaque localisation d'échantillon luma $(0 \leq x_L < \text{partWidth}, 0 \leq y_L < \text{partHeight})$ au sein de la matrice de prédiction d'échantillons luma predLX_L, la valeur correspondante d'échantillon luma de prédiction predLX_L[x_L, y_L] est calculée comme suit:

$$x_{Int_L} = x_{A_L} + (\text{mvLX}[0] \gg 2) + x_L \quad (8-176)$$

$$y_{Int_L} = y_{A_L} + (\text{mvLX}[1] \gg 2) + y_L \quad (8-177)$$

$$x_{Frac_L} = \text{mvLX}[0] \& 3 \quad (8-178)$$

$$y_{Frac_L} = \text{mvLX}[1] \& 3 \quad (8-179)$$

- La valeur d'échantillon de prédiction predLX_L[x_L, y_L] est déduite en invoquant le processus spécifié au § 8.4.2.2.1 avec (x_{Int_L}, y_{Int_L}) , (x_{Frac_L}, y_{Frac_L}) et refPicLX_L donnés en entrée.

Soit (x_{Int_C}, y_{Int_C}) une localisation chroma donnée en unités d'échantillons complets et (x_{Frac_C}, y_{Frac_C}) un décalage donné en unités d'un huitième d'échantillon. Ces variables ne sont utilisées qu'au sein du présent paragraphe pour spécifier les localisations générales d'échantillon fractionnaire au sein des matrices d'échantillons de référence $refPicLX_{Cb}$, et $refPicLX_{Cr}$.

Pour chaque localisation d'échantillon chroma ($0 \leq x_C < partWidth/2$, $0 \leq y_C < partHeight/2$) au sein des matrices d'échantillons chroma de prédiction $predPartLX_{Cb}$ et $predPartLX_{Cr}$, les valeurs correspondantes d'échantillons chroma de prédiction $predPartLX_{Cb}[x_C, y_C]$ et $predPartLX_{Cr}[x_C, y_C]$ sont calculées comme suit:

$$x_{Int_C} = (xA_L \gg 1) + (mvCLX[0] \gg 3) + x_C \quad (8-180)$$

$$y_{Int_C} = (yA_L \gg 1) + (mvCLX[1] \gg 3) + y_C \quad (8-181)$$

$$xFrac_C = mvCLX[0] \& 7 \quad (8-182)$$

$$yFrac_C = mvCLX[1] \& 7 \quad (8-183)$$

- La valeur d'échantillon de prédiction $predPartLX_{Cb}[x_C, y_C]$ est déduite en invoquant le processus spécifié au § 8.4.2.2.2 avec (x_{Int_C}, y_{Int_C}) , $(xFrac_C, yFrac_C)$ et $refPicLX_{Cb}$ donnés en entrée.
- La valeur d'échantillon de prédiction $predPartLX_{Cr}[x_C, y_C]$ est calculée en invoquant le processus spécifié au § 8.4.2.2.2 avec (x_{Int_C}, y_{Int_C}) , $(xFrac_C, yFrac_C)$ et $refPicLX_{Cr}$ donnés en entrée.

8.4.2.2.1 Processus d'interpolation d'échantillon luma

Les entrées pour ce processus sont

- une localisation luma en unités d'échantillon complets (x_{Int_L}, y_{Int_L}) ,
- un décalage de localisation luma en unités d'échantillons fractionnaires $(xFrac_L, yFrac_L)$,
- la matrice d'échantillons luma de l'image de référence choisie $refPicLX_L$,

Le résultat de ce processus est une valeur prédite d'échantillon luma $predPartLX_L[x_L, y_L]$.

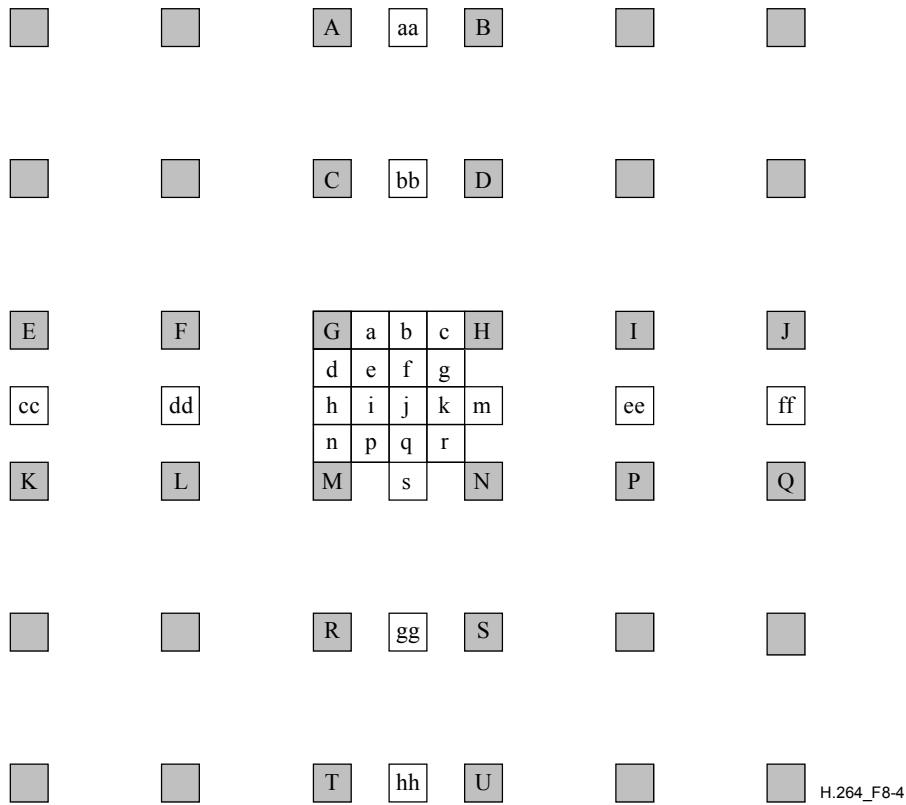


Figure 8-4 – Echantillons entiers (blocs ombrés avec lettres majuscules) et positions d'échantillons fractionnaires (blocs non ombrés avec lettres minuscules) pour interpolation de quarts d'échantillons luma.

La variable $refPicHeightEffective_L$, qui est la hauteur de la matrice luma d'image de référence effective, est calculée comme suit.

- Si $MbaffFrameFlag$ est égal à 0 ou $mb_field_decoding_flag$ est égal à 0, $refPicHeightEffective_L$ est mis égal à $PicHeightInSamples_L$.
- Autrement ($MbaffFrameFlag$ est égal à 1 et $mb_field_decoding_flag$ est égal à 1), $refPicHeightEffective_L$ est mis égal à $PicHeightInSamples_L / 2$.

A la Figure 8-4, les positions désignées avec des lettres minuscules au sein de blocs ombrés représentent des échantillons luma à des localisations d'échantillons entiers à l'intérieur de la matrice bidimensionnelle donnée $refPicLX_L$ d'échantillons luma. Ces échantillons peuvent être utilisés pour générer la valeur d'échantillon luma prédite $predPartLX_L[x_L, y_L]$. Les localisations (x_{Z_L}, y_{Z_L}) pour chacun des échantillons luma correspondants Z, où Z peut être A, B, C, D, E, F, G, H, I, J, K, L, M, N, P, Q, R, S, T, ou U, au sein de la matrice donnée $refPicLX_L$ d'échantillons luma sont déduites comme suit:

$$\begin{aligned} x_{Z_L} &= Clip3(0, PicWidthInSamples_L - 1, xInt_L + xDZ_L) \\ y_{Z_L} &= Clip3(0, refPicHeightEffective_L - 1, yInt_L + yDZ_L) \end{aligned} \quad (8-184)$$

Le Tableau 8-10 spécifie (xDZ_L, yDZ_L) pour différents remplacements de Z.

Tableau 8-10 – Localisation différentielle d'échantillons luma entiers

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

Etant donné les échantillons luma 'A' à 'U' à des localisations d'échantillons entiers (x_{A_L}, y_{A_L}) à (x_{U_L}, y_{U_L}), les échantillons luma 'a' à 's' à des positions d'échantillons fractionnaires sont déduits par les règles suivantes. Les valeurs de prédiction luma à des positions de demi-échantillon doivent être déduites en appliquant un filtre à 6 branches avec des valeurs de branche (1, -5, 20, 20, -5, 1). Les valeurs de prédiction luma à des positions de quart d'échantillon doivent être déduites en moyennant les échantillons à des positions de demi-échantillon et d'échantillon entier. Le processus pour chaque position fractionnaire est décrit ci-dessous.

- Les échantillons à des positions de demi-échantillon désignées par b doivent être déduits en calculant d'abord les valeurs intermédiaires notées b_1 en appliquant le filtre à six branches aux échantillons de position entière la plus proche dans la direction horizontale. Les échantillons à des positions de demi-échantillon désignées h doivent être déduits en calculant d'abord les valeurs intermédiaires notées h_1 en appliquant le filtre à six branches aux échantillons de position entière la plus proche dans la direction verticale:

$$b_1 = (E - 5 * F + 20 * G + 20 * H - 5 * I + J) \quad (8-185)$$

$$h_1 = (A - 5 * C + 20 * G + 20 * M - 5 * R + T) \quad (8-186)$$

Les valeurs de prédiction finales b et h doivent être déduites en utilisant:

$$b = \text{Clip1}((b_1 + 16) \gg 5) \quad (8-187)$$

$$h = \text{Clip1}((h_1 + 16) \gg 5) \quad (8-188)$$

- Les échantillons à une position de demi-échantillon désignées j doivent être déduits en calculant d'abord la valeur intermédiaire notée j_1 en appliquant le filtre à six branches aux valeurs intermédiaires des positions de demi-échantillon les plus proches dans la direction verticale ou horizontale parce que cela donne un résultat équivalent.

$$j_1 = cc - 5 * dd + 20 * h_1 + 20 * m_1 - 5 * ee + ff, \text{ ou} \quad (8-189)$$

$$j_1 = aa - 5 * bb + 20 * b_1 + 20 * s_1 - 5 * gg + hh \quad (8-190)$$

où les valeurs intermédiaires notées aa, bb, gg, s_1 et hh doivent être déduites en appliquant horizontalement le filtre à six branches de la même façon que pour la déduction de b_1 et les valeurs intermédiaires notées cc, dd, ee, m_1 et ff doivent être déduites en appliquant verticalement le filtre à six branches de la même façon que pour la déduction de h_1 . La valeur de prédiction finale j doit être calculée en utilisant:

$$j = \text{Clip1}((j_1 + 512) \gg 10) \quad (8-191)$$

- Les valeurs de prédiction finales s et m doivent être déduites de s_1 et m_1 de la même façon que pour la déduction de b et h, telles que donnée par:

$$s = \text{Clip1}((s_1 + 16) \gg 5) \quad (8-192)$$

$$m = \text{Clip1}((m_1 + 16) \gg 5) \quad (8-193)$$

- Les échantillons à des positions de quart d'échantillon désignées par a, c, d, n, f, i, k, et q doivent être déduits en moyennant avec arrondi supérieur aux deux échantillons les plus proches à des positions d'échantillons entiers et de moitié en utilisant:

$$a = (G + b + 1) \gg 1 \quad (8-194)$$

$$c = (H + b + 1) \gg 1 \quad (8-195)$$

$$d = (G + h + 1) \gg 1 \quad (8-196)$$

$$n = (M + h + 1) \gg 1 \quad (8-197)$$

$$f = (b + j + 1) \gg 1 \quad (8-198)$$

$$i = (h + j + 1) \gg 1 \quad (8-199)$$

$$k = (j + m + 1) \gg 1 \quad (8-200)$$

$$q = (j + s + 1) \gg 1 \quad (8-201)$$

- Les échantillons à des positions de quart d'échantillon désignées par e, g, p, et r doivent être déduits en moyennant avec arrondi supérieur aux deux échantillons les plus proches à des positions de demi-échantillon dans la direction de la diagonale en utilisant

$$e = (b + h + 1) \gg 1 \quad (8-202)$$

$$g = (b + m + 1) \gg 1 \quad (8-203)$$

$$p = (h + s + 1) \gg 1 \quad (8-204)$$

$$r = (m + s + 1) \gg 1 \quad (8-205)$$

Le décalage de localisation luma en unités d'échantillons fractionnaires ($xFrac_L$, $yFrac_L$) spécifie lequel des échantillons luma générés à des localisations d'échantillon entier et d'échantillon fractionnaire est alloué à la valeur d'échantillon luma prédite $predPartLX_L[x_L, y_L]$. Cette allocation est faite conformément au Tableau 8-11. La valeur de $predPartLX_L[x_L, y_L]$ doit être le résultat.

Tableau 8-11 – Allocation de l'échantillon de prédiction luma $predPartLX_L[x_L, y_L]$

$xFrac_L$	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 Processus d'interpolation d'échantillon chroma

Les entrées pour ce processus sont

- une localisation chroma en unités d'échantillon entier ($xInt_C$, $yInt_C$),
- un décalage de localisation chroma en unités d'échantillon fractionnaire ($xFrac_C$, $yFrac_C$),
- des échantillons de composant chroma provenant de l'image de référence choisie $refPicLX_C$.

Le résultat de ce processus est une valeur prédite d'échantillon chroma $predPartLX_C[x_C, y_C]$.

Dans la Figure 8-5, les positions désignées par A, B, C, et D représentent des échantillons chroma à des localisations d'échantillon entier dans le tableau bidimensionnel donné $refPicLX_C$ d'échantillons chroma.

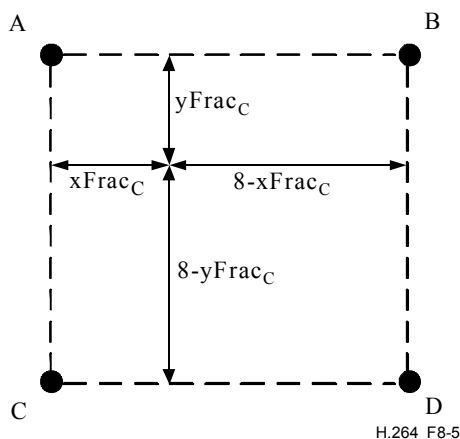


Figure 8-5 – Variables selon la position d'échantillon fractionnaire en interpolation chroma avec les échantillons A, B, C, et D de position entière qui les entourent

La variable $refPicHeightEffective_C$, qui est la hauteur de la matrice chroma d'image de référence effective, est calculée comme suit.

- Si $MbaffFrameFlag$ est égal à 0 ou $mb_field_decoding_flag$ est égal à 0, $refPicHeightEffective_C$ est mis égal à $PicHeightInSamples_C$.
- Autrement ($MbaffFrameFlag$ est égal à 1 and $mb_field_decoding_flag$ est égal à 1), $refPicHeightEffective_C$ est mis égal à $PicHeightInSamples_C / 2$.

Les coordonnées des échantillons spécifiées dans les équations 8-206 à 8-213 sont utilisées pour générer la valeur d'échantillon chroma prédite $predPartLX_C[x_C, y_C]$.

$$xA_C = \text{Clip3}(0, \text{PicWidthInSamples}_C - 1, xInt_C) \quad (8-206)$$

$$xB_C = \text{Clip3}(0, \text{PicWidthInSamples}_C - 1, xInt_C + 1) \quad (8-207)$$

$$xC_C = \text{Clip3}(0, \text{PicWidthInSamples}_C - 1, xInt_C) \quad (8-208)$$

$$xD_C = \text{Clip3}(0, \text{PicWidthInSamples}_C - 1, xInt_C + 1) \quad (8-209)$$

$$yA_C = \text{Clip3}(0, \text{refPicHeightEffective}_C - 1, yInt_C) \quad (8-210)$$

$$yB_C = \text{Clip3}(0, \text{refPicHeightEffective}_C - 1, yInt_C) \quad (8-211)$$

$$yC_C = \text{Clip3}(0, \text{refPicHeightEffective}_C - 1, yInt_C + 1) \quad (8-212)$$

$$yD_C = \text{Clip3}(0, \text{refPicHeightEffective}_C - 1, yInt_C + 1) \quad (8-213)$$

Etant donné les échantillons chroma A, B, C, et D aux localisations d'échantillon entier spécifiées dans les équations 8-206 à 8-213, la valeur prédite d'échantillon chroma $\text{predPartLXC}[x_C, y_C]$ est calculée comme suit:

$$\text{predPartLXC}[x_C, y_C] = ((8 - xFrac_C) * (8 - yFrac_C) * A + xFrac_C * (8 - yFrac_C) * B + (8 - xFrac_C) * yFrac_C * C + xFrac_C * yFrac_C * D + 32) \gg 6 \quad (8-214)$$

8.4.2.3 Processus de prédiction d'échantillon pondéré

Les entrées pour ce processus sont

- mbPartIdx : la subdivision en cours donnée par l'indice de subdivision
- subMbPartIdx : l'indice de subdivision de sous-macrobloc
- predFlagL0 et predFlagL1 : des fanions d'utilisation de liste de prédiction
- predPartLXL : une matrice $(\text{partWidth}) \times (\text{partHeight})$ d'échantillons de prédiction luma (avec LX remplacé par L0 ou L1 en fonction de predFlagL0 et predFlagL1)
- predPartLXC_b et predPartLXC_r : des matrices $(\text{partWidth}/2) \times (\text{partHeight}/2)$ d'échantillons de prédiction chroma, une pour chacun des composants chroma Cb et Cr (avec LX remplacé par L0 ou L1 en fonction de predFlagL0 et predFlagL1).

Les résultats de ce processus sont

- predPartL : une matrice $(\text{partWidth}) \times (\text{partHeight})$ d'échantillons de prédiction luma et
- predPartCb , et predPartCr : des matrices $(\text{partWidth}/2) \times (\text{partHeight}/2)$ d'échantillons de prédiction chroma, une pour chacun des composants chroma Cb et Cr.

Pour les macroblocs ou les subdivisions avec predFlagL0 égal à 1 dans les tranches P et SP, on applique ce qui suit:

- Si $\text{weighted_pred_flag}$ est égal à 0, le processus de prédiction d'échantillon pondéré par défaut comme décrit au § 8.4.2.3.1 est invoqué avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.
- Autrement ($\text{weighted_pred_flag}$ est égal à 1), le processus explicite de prédiction pondéré comme décrit au § 8.4.2.3.2 est invoqué avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.

Pour les macroblocs ou partitions avec predFlagL0 ou predFlagL1 égal à 1 dans les tranches B, on applique ce qui suit:

- Si $\text{weighted_bipred_idc}$ est égal à 0, le processus de prédiction d'échantillon pondéré par défaut comme décrit au § 8.4.2.3.1 est invoqué avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.
- Autrement, si $\text{weighted_bipred_idc}$ est égal à 1, le processus explicite de prédiction d'échantillon pondéré comme décrit au § 8.4.2.3.2, pour les macroblocs ou partitions avec predFlagL0 ou predFlagL1 égal à 1 avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.
- Autrement ($\text{weighted_bipred_idc}$ est égal à 2), on applique ce qui suit:

- Si predFlagL0 est égal à 1 et predFlagL1 est égal à 1, la prédiction implicite d'échantillon pondéré comme décrit au § 8.4.2.3.2 est invoquée avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.
- Autrement (predFlagL0 ou predFlagL1 sont égaux à 1 mais pas tous les deux), le processus de prédiction d'échantillon pondéré par défaut comme décrit au § 8.4.2.3.1 est invoqué avec les mêmes entrées et résultats que le processus décrit dans le présent paragraphe.

8.4.2.3.1 Processus de prédiction d'échantillon pondéré par défaut

Les entrées pour ce processus sont les mêmes que spécifié au § 8.4.2.3.

Les résultats de ce processus sont les mêmes que spécifié au § 8.4.2.3.

En fonction du composant pour lequel le bloc de prédiction est déduit, on applique ce qui suit.

- Si les valeurs de prédiction d'échantillon luma $\text{predPart}_L[x, y]$ sont calculées, on applique ce qui suit avec C mis égal à L, x mis égal à $0 \dots \text{partWidth} - 1$, et y mis égal à $0 \dots \text{partHeight} - 1$.
- Autrement, si les valeurs de prédiction d'échantillon de composant chroma Cb $\text{predPart}_{Cb}[x, y]$ sont calculées, on applique ce qui suit avec C mis égal à Cb, x mis égal à $0 \dots \text{partWidth} / 2 - 1$, et y mis égal à $0 \dots \text{partHeight} / 2 - 1$.
- Autrement (les valeurs de prédiction d'échantillon de composant chroma Cr $\text{predPart}_{Cr}[x, y]$ sont calculées), on applique ce qui suit avec C mis égal à Cr, x mis égal à $0 \dots \text{partWidth} / 2 - 1$, et y mis égal à $0 \dots \text{partHeight} / 2 - 1$.

Les valeurs d'échantillon de prédiction sont calculées comme suit.

- Si predFlagL0 est égal à 1 et predFlagL1 est égal à 0 pour la subdivision en cours,

$$\text{predPart}_c[x, y] = \text{predPart}_{L0c}[x, y] \quad (8-215)$$
- Autrement, si predFlagL0 est égal à 0 et predFlagL1 est égal à 1 pour la subdivision en cours,

$$\text{predPart}_c[x, y] = \text{predPart}_{L1c}[x, y] \quad (8-216)$$
- Autrement (predFlagL0 et predFlagL1 sont égaux à 1 pour la subdivision en cours),

$$\text{predPart}_c[x, y] = (\text{predPart}_{L0c}[x, y] + \text{predPart}_{L1c}[x, y] + 1) \gg 1. \quad (8-217)$$

8.4.2.3.2 Processus de prédiction d'échantillon pondéré

Les entrées pour ce processus sont les mêmes que spécifié au § 8.4.2.3.

Les résultats de ce processus sont les mêmes que spécifié au § 8.4.2.3.

En fonction du composant pour lequel est déduit le bloc de prédiction, on applique ce qui suit:

- Si les valeurs de prédiction d'échantillon luma $\text{predPart}_L[x, y]$ sont calculées, on applique ce qui suit avec C mis égal à L, x mis égal à $0 \dots \text{partWidth} - 1$, et y mis égal à $0 \dots \text{partHeight} - 1$.
- Autrement, si les valeurs de prédiction d'échantillon de composant chroma Cb $\text{predPart}_{Cb}[x, y]$ sont calculées, on applique ce qui suit avec C mis égal à Cb, x mis égal à $0 \dots \text{partWidth} / 2 - 1$, et y mis égal à $0 \dots \text{partHeight} / 2 - 1$.
- Autrement (les valeurs de prédiction d'échantillon de composant chroma Cr $\text{predPart}_{Cr}[x, y]$ sont calculées), on applique ce qui suit avec C mis égal à Cr, x mis égal à $0 \dots \text{partWidth} / 2 - 1$, et y mis égal à $0 \dots \text{partHeight} / 2 - 1$.

Les valeurs d'échantillon de prédiction sont calculées comme suit.

- Si la subdivision $mbPartIdx \setminus subMbPartIdx$ a $predFlagL0$ égal à 1 et $predFlagL1$ égal à 0, les valeurs prédites d'échantillon finales $predPart_C[x, y]$ sont calculées par

$$\begin{aligned} & \text{si}(\log WD \geq 1) \\ & \quad predPart_C[x, y] = Clip1(((predPartL0_C[x, y] * w_0 + 2^{\log WD - 1}) \gg \log WD) + o_0) \\ & \text{ou} \\ & \quad predPart_C[x, y] = Clip1(predPartL0_C[x, y] * w_0 + o_0) \end{aligned} \quad (8-218)$$

- Autrement, si la subdivision $mbPartIdx \setminus subMbPartIdx$ a $predFlagL0$ égal à 0 et $predFlagL1$ égal à 1, les valeurs prédites d'échantillon finales $predPart_C[x, y]$ sont calculées par

$$\begin{aligned} & \text{si}(\log WD \geq 1) \\ & \quad predPart_C[x, y] = Clip1(((predPartL1_C[x, y] * w_1 + 2^{\log WD - 1}) \gg \log WD) + o_1) \\ & \text{ou} \\ & \quad predPart_C[x, y] = Clip1(predPartL1_C[x, y] * w_1 + o_1) \end{aligned} \quad (8-219)$$

- Autrement (la subdivision $mbPartIdx \setminus subMbPartIdx$ a les deux $predFlagL0$ et $predFlagL1$ égaux à 1), les valeurs prédites d'échantillon finales $predPart_C[x, y]$ sont calculées par

$$predPart_C[x, y] = Clip1(((predPartL0_C[x, y] * w_0 + predPartL1_C[x, y] * w_1 + 2^{\log WD}) \gg (\log WD + 1)) + ((o_0 + o_1 + 1) \gg 1)) \quad (8-220)$$

Les variables des déductions ci-dessus pour les échantillons de prédiction sont calculées comme suit.

- Si $weighted_bipred_idc$ est égal à 2 et que le $slice_type$ est égal à B, la prédiction pondérée en mode implicite est utilisée comme suit.

$$\log WD = 5 \quad (8-221)$$

$$o_0 = 0 \quad (8-222)$$

$$o_1 = 0 \quad (8-223)$$

et w_0 et w_1 sont déduits comme suit.

- Les variables $currPicOrField$, $pic0$ et $pic1$ sont calculées comme suit:
 - si $field_pic_flag$ est égal à 0 et si le macrobloc en cours est un macrobloc de champ, on applique ce qui suit.
 - $currPicOrField$ est le champ de l'image en cours $CurrPic$ qui a la même parité que le macrobloc en cours.
 - Soit $frame0$ la trame ou paire de champs complémentaires mentionnée par $RefPicList0[refIdxL0 / 2]$.
 - La variable $pic0$ est calculée comme suit.
 - Si $refIdxL0 \% 2$ est égal à 0, $pic0$ est le champ de $frame0$ qui a la même parité que le macrobloc en cours.
 - Autrement ($refIdxL0 \% 2$ est différent de 0), $pic0$ est le champ $frame0$ qui a la parité opposée à celle du macrobloc en cours.
 - Soit $frame1$ la trame ou paire de champs complémentaires mentionnée par $RefPicList1[refIdxL1 / 2]$.
 - La variable $pic1$ est calculée comme suit.
 - Si $refIdxL1 \% 2$ est égal à 0, $pic1$ est le champ de $frame1$ qui a la même parité que le macrobloc en cours.
 - Autrement ($refIdxL1 \% 2$ est différent de 0), $pic1$ est le champ $frame1$ qui a la parité opposée à celle du macrobloc en cours.
 - Autrement ($field_pic_flag$ est égal à 1 ou le macrobloc en cours est un macrobloc de trame), $currPicOrField$ est l'image en cours $CurrPic$, $pic1$ est l'image de référence décodée mentionnée par $RefPicList1[refIdxL1]$, et $pic0$ est l'image de référence décodée mentionnée par $RefPicList0[refIdxL0]$.

– Les variables tb , td , tx , et $DistScaleFactor$ sont calculées d'après les valeurs de $currPicOrField$, $pic0$, $pic1$ en utilisant les équations 8-154, 8-155, 8-150 et 8-151, respectivement.

– Si $DiffPicOrderCnt(pic1, pic0)$ est égal à 0 $pic1$ ou/et $pic0$ sont des images de référence à long terme ou si $(DistScaleFactor \gg 2) < -64$ ou si $(DistScaleFactor \gg 2) > 128$, w_0 et w_1 sont calculés comme suit.

$$w_0 = 32 \quad (8-224)$$

$$w_1 = 32 \quad (8-225)$$

– Autrement,

$$w_0 = 64 - (DistScaleFactor \gg 2) \quad (8-226)$$

$$w_1 = DistScaleFactor \gg 2 \quad (8-227)$$

– Autrement ($weighted_pred_flag$ est égal à 1 dans les tranches P ou SP ou si $weighted_bipred_idc$ est égal à 1 dans les tranches B), la prédiction pondérée de mode explicite est utilisée comme suit.

– Les variables $refIdxL0WP$ et $refIdxL1WP$ sont déduites comme suit.

– Si $MbaffFrameFlag$ est égal à 1 et si le macrobloc en cours est un macrobloc de champ

$$refIdxL0WP = refIdxL0 \gg 1 \quad (8-228)$$

$$refIdxL1WP = refIdxL1 \gg 1 \quad (8-229)$$

– Autrement ($MbaffFrameFlag$ est égal à 0 ou le macrobloc en cours est un macrobloc de trame),

$$refIdxL0WP = refIdxL0 \quad (8-230)$$

$$refIdxL1WP = refIdxL1 \quad (8-231)$$

– Les variables $\log WD$, w_0 , w_1 , o_0 , et o_1 sont déduites comme suit.

– Si C dans $predPart_C[x, y]$ est remplacé par L pour les échantillons luma

$$\log WD = luma_log2_weight_denom \quad (8-232)$$

$$w_0 = luma_weight_10[refIdxL0WP] \quad (8-233)$$

$$w_1 = luma_weight_11[refIdxL1WP] \quad (8-234)$$

$$o_0 = luma_offset_10[refIdxL0WP] \quad (8-235)$$

$$o_1 = luma_offset_11[refIdxL1WP] \quad (8-236)$$

– Autrement (C dans $predPart_C[x, y]$ est remplacé par Cb ou Cr pour les échantillons chroma, avec $iCbCr = 0$ pour Cb, $iCbCr = 1$ pour Cr),

$$\log WD = chroma_log2_weight_denom \quad (8-237)$$

$$w_0 = chroma_weight_10[refIdxL0WP][iCbCr] \quad (8-238)$$

$$w_1 = chroma_weight_11[refIdxL1WP][iCbCr] \quad (8-239)$$

$$o_0 = chroma_offset_10[refIdxL0WP][iCbCr] \quad (8-240)$$

$$o_1 = chroma_offset_11[refIdxL1WP][iCbCr] \quad (8-241)$$

Lorsque la prédiction pondérée en mode explicite est utilisée et que la subdivision $mbPartIdx \setminus subMbPartIdx$ a à la fois $predFlagL0$ et $predFlagL1$ égaux à 1, on doit respecter les contraintes suivantes:

$$-128 \leq w_0 + w_1 \leq ((\log WD == 7) ? 127 : 128) \quad (8-242)$$

NOTE – Pour la prédiction pondérée en mode implicite, les pondérations w_0 et w_1 sont chacune obligatoirement dans la gamme de $-64 \dots 128$ et la contrainte exprimée dans l'équation 8-242, bien que non explicitement imposée, doit toujours être respectée. Pour la prédiction pondérée en mode explicite avec $\log_{2}WD$ égal à 7, lorsqu'une des deux pondérations w_0 ou w_1 est supposée être égale à 128 (du fait que $\text{luma_weight_l0_flag}$, $\text{luma_weight_l1_flag}$, $\text{chroma_weight_l0_flag}$, ou $\text{chroma_weight_l1_flag}$ est égal à 0), l'autre pondération (w_1 ou w_0) doit avoir une valeur négative pour que la contrainte exprimée dans l'équation (8-242) se vérifie (et par conséquent l'autre $\text{luma_weight_l0_flag}$, $\text{luma_weight_l1_flag}$, $\text{chroma_weight_l0_flag}$, ou $\text{chroma_weight_l1_flag}$ doit être égal à 1).

8.5 Processus de décodage du coefficient de transformée et processus de construction d'image antérieurement au processus de filtrage de démontage de blocs

Les entrées pour ce processus sont Intra16x16DCLevel (si disponible), Intra16x16ACLevel (si disponible), LumaLevel (si disponible), ChromaDCLevel , ChromaACLevel , et les matrices d'échantillons d'inter ou d'intraprédiction disponibles pour le macrobloc en cours pour le composant applicable pred_L , pred_{Cb} , ou pred_{Cr} .

NOTE – Lors du décodage d'un macrobloc en mode de prédiction Intra_{4x4} , le composant luma du tableau de prédiction de macrobloc peut n'être pas complet, dans la mesure où pour chaque bloc luma $4x4$, le processus de prédiction Intra_{4x4} pour les échantillons luma, comme spécifié au § 8.3.1, et le processus spécifié au présent paragraphe, font l'objet d'une itération.

Les résultats de ce processus sont les matrices d'échantillons construites antérieurement au processus de filtrage de démontage de blocs pour les composants applicables S'_L , S'_{Cb} , ou S'_{Cr} .

NOTE – Lors du décodage d'un macrobloc en mode de prédiction Intra_{4x4} , le composant luma des matrices d'échantillons construites antérieurement au processus de filtrage de démontage de blocs peut n'être pas complet, dans la mesure où pour chaque bloc luma $4x4$, le processus de prédiction Intra_{4x4} pour les échantillons luma, comme spécifié au § 8.3.1, et le processus spécifié au présent paragraphe, font l'objet d'une itération.

Le présent paragraphe spécifie le décodage du coefficient de transformée et la construction d'image antérieurement au processus de filtrage de démontage de blocs.

Lorsque le macrobloc en cours est codé P_Skip ou B_Skip , toutes les valeurs de LumaLevel , ChromaDCLevel , ChromaACLevel sont mises égales à 0 pour le macrobloc en cours.

8.5.1 Spécification du processus de décodage de transformée pour les blocs résiduels

Lorsque le mode de prédiction du macrobloc en cours n'est pas égal à Intra_{16x16} , la variable LumaLevel contient les niveaux pour les coefficients de transformée luma. Pour un bloc luma $4x4$ indexé par $\text{luma4x4BlkIdx} = 0 \dots 15$ sont spécifiées les étapes ordonnées suivantes.

1. Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.4, est invoqué avec $\text{LumaLevel}[\text{luma4x4BlkIdx}]$ comme entrée et le tableau bidimensionnel c comme résultat.
2. Le processus d'étalonnage et de transformation pour les blocs $4x4$ résiduels, comme spécifié au § 8.5.8, est invoqué avec c comme entrée et r comme résultat.
3. La position de l'échantillon supérieur gauche d'un bloc luma $4x4$ avec l'indice luma4x4BlkIdx au sein du macrobloc est déduite en invoquant le processus inverse de balayage de bloc luma $4x4$ du § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (xO, yO) .
4. La matrice $4x4$ u avec les éléments u_{ij} pour $i, j = 0 \dots 3$ est déduite avec

$$u_{ij} = \text{Clip1}(\text{pred}_L[xO + j, yO + i] + r_{ij}) \quad (8-243)$$
5. Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.9 est invoqué avec luma4x4BlkIdx , u comme entrée et S' comme résultat.

8.5.2 Spécification du processus de décodage de transformée pour échantillons luma du mode de prédiction de macrobloc Intra_16x16

Lorsque le mode de prédiction du macrobloc en cours est égal à Intra_16x16, les variables Intra16x16DCLevel et Intra16x16ACLevel contiennent les niveaux pour les coefficients de transformée luma. Le décodage du coefficient de transformée s'effectue selon les étapes ordonnées suivantes:

1. Les coefficients de transformée 4x4 luma DC de tous les blocs luma 4x4 du macrobloc sont décodés.
 - a. Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.4, est invoqué avec Intra16x16DCLevel comme entrée et la matrice bidimensionnelle c comme résultat.
 - b. Le processus d'étalonnage et de transformation pour les coefficients de transformée luma DC pour le type de macrobloc Intra_16x16, comme spécifié au § 8.5.6, est invoqué avec c comme entrée et dcY comme résultat.
2. Pour un bloc luma 4x4 indexé par luma4x4BlkIdx = 0..15 sont spécifiées les étapes ordonnées suivantes.
 - a. La variable lumaList, qui est une liste de 16 entrées, est déduite. La première entrée de lumaList est la valeur correspondante provenant de la matrice dcY. La Figure 8-6 montre l'allocation des indices de la matrice dcY aux luma4x4BlkIdx. Les deux chiffres dans les petits carrés se réfèrent aux indices i et j de dcY_{ij}, et les chiffres dans les grands carrés se réfèrent à luma4x4BlkIdx.

<small>00</small> 0	<small>01</small> 1	<small>02</small> 4	<small>03</small> 5
<small>10</small> 2	<small>11</small> 3	<small>12</small> 6	<small>13</small> 7
<small>21</small> 8	<small>21</small> 9	<small>22</small> 12	<small>23</small> 13
<small>30</small> 10	<small>31</small> 11	<small>32</small> 14	<small>33</small> 15

H.264_F8-6

Figure 8-6 – Allocation des indices de dcY à luma4x4BlkIdx

Les éléments de lumaList avec les indices k = 1..15 sont spécifiés comme:

$$\text{lumaList}[k] = \text{Intra16x16ACLevel}[\text{luma4x4BlkIdx}][k - 1] \quad (8-244)$$

- b. Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.4, est invoqué avec lumaList comme entrée et la matrice bidimensionnelle c comme résultat.
- c. Le processus d'étalonnage et de transformation pour les blocs 4x4 résiduels, comme spécifié au § 8.5.8, est invoqué avec c comme entrée et r comme résultat.
- d. La position de l'échantillon supérieur gauche d'un bloc luma 4x4 avec l'indice luma4x4BlkIdx au sein du macrobloc est déduite en invoquant le processus inverse de balayage de bloc luma 4x4 du § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (xO, yO).
- e. La matrice 4x4 u avec les éléments u_{ij} pour i, j = 0..3, est déduite comme

$$u_{ij} = \text{Clip1}(\text{pred}_L[xO + j, yO + i] + r_{ij}) \quad (8-245)$$

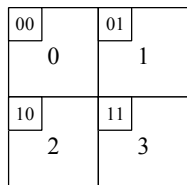
- f. Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.9 est invoqué avec luma4x4BlkIdx , u comme entrée et S' comme résultat.

8.5.3 Spécification du processus de décodage de transformée pour échantillons chroma

Pour chaque composant chroma, les variables $\text{ChromaDCLevel}[iCbCr]$ et $\text{ChromaACLevel}[iCbCr]$, avec $iCbCr$ mis égal à 0 pour Cb et $iCbCr$ mis égal à 1 pour Cr, contiennent les niveaux pour les deux composants des coefficients de transformée chroma. Pour chaque composant chroma, le décodage de transformée traite séparément les étapes ordonnées suivantes:

1. On décode les coefficients de transformées 2x2 chroma DC des blocs chroma 4x4 du composant du macrobloc indexé par $iCbCr$.
 - a. La matrice c 2x2 est déduite en utilisant le processus inverse de balayage de trame appliqué à ChromaDCLevel comme suit

$$c = \begin{bmatrix} \text{ChromaDCLevel}[iCbCr][0] & \text{ChromaDCLevel}[iCbCr][1] \\ \text{ChromaDCLevel}[iCbCr][2] & \text{ChromaDCLevel}[iCbCr][3] \end{bmatrix} \quad (8-246)$$
 - b. Le processus d'étalonnage et de transformation pour les coefficients de transformée chroma DC comme spécifié au § 8.5.7 est invoqué avec c comme entrée et dcC comme résultat.
2. Pour chaque bloc chroma 4x4 indexé par $\text{chroma4x4BlkIdx} = 0..3$ du composant indexé par $iCbCr$, on spécifie les étapes ordonnées suivantes.
 - a. On déduit la variable chromaList , qui est une liste de 16 entrées. La première entrée de chromaList est la valeur correspondante provenant de la matrice dcC . La Figure 8-7 montre l'allocation des indices de la matrice dcC à chroma4x4BlkIdx . Les deux chiffres dans les petits carrés se réfèrent aux indices i et j dans dcC_{ij} , et les chiffres dans les grands carrés se réfèrent à chroma4x4BlkIdx .



H.264_F87

Figure 8-7 – Allocation des indices de dcC à chroma4x4BlkIdx

Les éléments de chromaList avec l'indice $k = 1..15$ sont spécifiés comme

$$\text{chromaList}[k] = \text{ChromaACLevel}[\text{chroma4x4BlkIdx}][k-1] \quad (8-247)$$

- b. Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.4, est invoqué avec chromaList comme entrée et la matrice bidimensionnelle c comme résultat.
- c. Le processus d'étalonnage et de transformation pour les blocs résiduels 4x4, comme spécifié au § 8.5.8, est invoqué avec c comme entrée et r comme résultat.
- d. La position de l'échantillon supérieur gauche d'un bloc chroma 4x4 avec l'indice chroma4x4BlkIdx au sein du macrobloc est déduite comme suit

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (8-248)$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (8-249)$$

e. La matrice 4x4 u avec les éléments u_{ij} pour $i, j = 0..3$ est déduite comme

$$u_{ij} = \text{Clip1}(\text{pred}_c[xO + j, yO + i] + r_{ij}) \quad (8-250)$$

f. Le processus de construction d'image avant le processus de filtrage de démontage de blocs du § 8.5.9 est invoqué avec chroma4x4BlkIdx , u comme entrée et S' comme résultat.

8.5.4 Processus de balayage inverse pour les coefficients de transformée

L'entrée pour ce processus est une liste de 16 valeurs.

Le résultat de ce processus est une variable c contenant une matrice bidimensionnelle de valeurs 4x4 avec le niveau alloué aux localisations dans le bloc de transformée.

Le processus de décodage fait correspondre la séquence de niveaux de coefficients de transformée aux positions des niveaux de coefficient de transformée. Pour ce mappage, on utilise les deux schémas de balayage inverse montrés à la Figure 8-8.

Le balayage en zigzag inverse doit être utilisé pour les macroblocs de trame et le balayage inverse de champ doit être utilisé pour les macroblocs de champ.

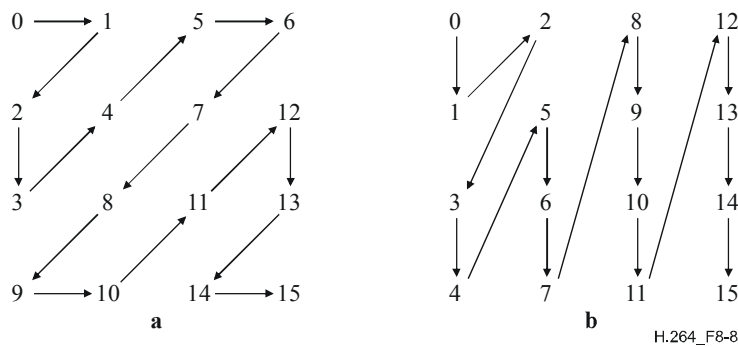


Figure 8-8 – a) Balayage en zigzag. b) Balayage de champ

Le Tableau 8-12 donne le mappage entre l'indice idx de la liste d'entrées de 16 éléments et les indices i et j de la matrice bidimensionnelle c.

Tableau 8-12 – Spécification du mappage des indices à c_{ij} pour balayage en zigzag et de champ

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}
champ	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}

8.5.5 Processus de déduction pour les paramètres de quantification chroma et la fonction d'étalonnage

Les résultats de ce processus sont:

- QP_C : le paramètre de quantification chroma
- QS_C : le paramètre supplémentaire de quantification chroma nécessaire pour décoder les tranches SP et SI (le cas échéant).

NOTE – Les valeurs QP de paramètre de quantification QP_Y , QP_C , QS_Y , et QS_C sont toujours dans la gamme de 0 à 51 inclus.

La valeur de QP_C pour chroma est déterminée à partir de la valeur courante de QP_Y et de la valeur de $\text{chroma_qp_index_offset}$.

NOTE – Les équations d'étalonnage sont spécifiées de telle sorte que le facteur d'étalonnage des niveaux des coefficients de transformée équivalent double pour chaque incrément de 6 dans QP_Y . Ainsi, il y a un accroissement du facteur utilisé pour l'étalonnage d'approximativement 12 % pour chaque accroissement de 1 de la valeur de QP_Y .

La valeur de QP_C doit être déterminée comme spécifié au Tableau 8-13 sur la base de l'index désigné par qP_I . La valeur de qP_I doit être déduite comme suit.

$$qP_I = \text{Clip3}(0, 51, QP_Y + \text{chroma_qp_index_offset}) \quad (8-251)$$

Tableau 8-13 – Spécification de QP_C en fonction de qP_I

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

Lorsque la tranche en cours est une tranche SP ou SI, QS_C est déduit en utilisant le processus ci-dessus, en substituant QP_Y à QS_Y et QP_C à QS_C .

La fonction $\text{LevelScale}(m, i, j)$ est spécifiée comme suit:

$$\text{LevelScale}(m, i, j) = \begin{cases} v_{m0} & \text{pour } (i, j) \in \{(0,0), (0,2), (2,0), (2,2)\}, \\ v_{m1} & \text{pour } (i, j) \in \{(1,1), (1,3), (3,1), (3,3)\}, \\ v_{m2} & \text{autrement;} \end{cases} \quad (8-252)$$

où le premier et le second indice de v sont respectivement les indices de rangée et de colonne, de la matrice spécifiée comme:

$$v = \begin{bmatrix} 10 & 16 & 13 \\ 11 & 18 & 14 \\ 13 & 20 & 16 \\ 14 & 23 & 18 \\ 16 & 25 & 20 \\ 18 & 29 & 23 \end{bmatrix}. \quad (8-253)$$

8.5.6 Processus d'étalonnage et de transformation pour coefficients de transformée luma DC pour type de macrobloc Intra_16x16

Les entrées pour ce processus sont les valeurs de niveau de coefficient de transformée pour les coefficients de transformée luma DC de macroblocs Intra_16x16 comme une matrice c 4x4 avec des éléments c_{ij} , où i et j forment un indice de fréquence à deux dimensions;

Les résultats de ce processus sont 16 valeurs DC étalonnées pour des blocs luma 4x4 de macroblocs Intra_16x16 comme une matrice 4x4 dcY avec des éléments dcY_{ij} .

La transformée inverse pour les coefficients de transformée 4x4 luma DC est spécifié par:

$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}. \quad (8-254)$$

Un flux binaire conforme à la présente Recommandation | Norme internationale ne doit pas contenir de données qui feraient qu'un élément f_{ij} de f excède la gamme de valeurs entières de -2^{15} à $2^{15}-1$, inclus.

Après la transformée inverse, l'étalonnage est effectué comme suit.

- Si QP_Y est supérieur ou égal à 12, le résultat étalonné doit être calculé comme

$$dcY_{ij} = (f_{ij} * \text{LevelScale}(QP_Y \% 6, 0, 0)) \ll (QP_Y / 6 - 2), \text{ avec } i, j = 0..3 \quad (8-255)$$

- Autrement (QP_Y est inférieur à 12), le résultat étalonné doit être calculé comme

$$dcY_{ij} = (f_{ij} * \text{LevelScale}(QP_Y \% 6, 0, 0) + 2^{1-QP_Y/6}) \gg (2 - QP_Y / 6), \text{ avec } i, j = 0..3 \quad (8-256)$$

Un flux binaire conforme à la présente Recommandation | Norme internationale ne doit pas contenir de données qui feraient qu'un élément dcY_{ij} de dcY excède la gamme de valeurs entières de -2^{15} à $2^{15}-1$, inclus.

NOTE – Lorsque `entropy_coding_mode_flag` est égal à 0 et que QP_Y est inférieur à 10, la gamme des valeurs qui peuvent être représentées pour les éléments c_{ij} de c est insuffisante pour représenter la gamme complète des valeurs des éléments dcY_{ij} de dcY qui pourrait être nécessaire pour former une approximation fidèle du contenu de toute image source possible par l'utilisation du type de macroblocs `Intra_16x16`.

NOTE – Etant donné que la limitation de la gamme des valeurs imposée aux éléments dcY_{ij} de dcY est imposée après le décalage à droite dans l'équation 8-256, une plus large gamme de valeurs doit être admise dans le décodeur avant le décalage à droite.

8.5.7 Processus d'étalonnage et de transformation pour coefficients de transformée chroma DC

Les entrées pour ce processus sont des valeurs de niveau de coefficient de transformée pour des coefficients de transformée chroma DC d'un composant chroma du macrobloc en une matrice 2×2 c avec des éléments c_{ij} , où i et j forment un indice de fréquence à deux dimensions.

Les résultats de ce processus sont 4 valeurs DC étalonnées en une matrice 2×2 dcC avec des éléments dcC_{ij} .

La transformée inverse pour les coefficients de transformée 2×2 chroma DC est spécifiée par:

$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8-257)$$

Un flux binaire conforme à la présente Recommandation | Norme internationale ne doit pas contenir de données qui feraient qu'un élément f_{ij} de f excède la gamme de valeurs entières de -2^{15} à $2^{15}-1$, inclus.

Après la transformée inverse, l'étalonnage est effectué comme suit:

- Si QP_C est supérieur ou égal à 6, le résultat de l'étalonnage doit être calculé comme:

$$dcC_{ij} = (f_{ij} * \text{LevelScale}(QP_C \% 6, 0, 0)) \ll (QP_C / 6 - 1), \text{ avec } i, j = 0, 1 \quad (8-258)$$

- Autrement (QP_C est inférieur à 6), le résultat de l'étalonnage doit être calculé comme:

$$dcC_{ij} = (f_{ij} * \text{LevelScale}(QP_C \% 6, 0, 0)) \gg 1, \text{ avec } i, j = 0, 1 \quad (8-259)$$

Un flux binaire conforme à la présente Recommandation | Norme internationale ne doit pas contenir de données qui feraient qu'un élément dcC_{ij} de dcC excède la gamme de valeurs entières de -2^{15} à $2^{15}-1$ inclus.

NOTE – Lorsque `entropy_coding_mode_flag` est égal à 0 et que QP_Y est inférieur à 4, la gamme des valeurs qui peuvent être représentées pour les éléments C_{ij} de C est insuffisante pour représenter la gamme complète des valeurs des éléments dcC_{ij} de dcC qui pourrait être nécessaire pour former une approximation fidèle du contenu de toute image source possible.

NOTE – Etant donné que la limitation de la gamme des valeurs imposée aux éléments dcC_{ij} de dcC est imposée après le décalage à droite dans l'équation 8-259, une plus large gamme de valeurs doit être admise dans le décodeur avant le décalage à droite.

8.5.8 Processus d'étalonnage et de transformation pour blocs 4x4 résiduels

L'entrée pour ce processus est une matrice 4x4 c avec des éléments c_{ij} qui sont soit une matrice se rapportant au bloc résiduel du composant luma soit une matrice se rapportant au bloc résiduel d'un composant chroma.

Les résultats de ce processus sont des valeurs résiduelles d'échantillon en une matrice 4x4 r avec des éléments r_{ij} .

La variable sMbFlag est déduite comme suit.

- Si mb_type est égal à SI ou si le mode de prédiction de macrobloc est égal à Inter dans une tranche SP, sMbFlag est mise égale à 1.
- Autrement (mb_type n'est pas égal à SI et le mode de prédiction de macrobloc n'est pas égal à Inter dans une tranche SP), sMbFlag est mise égale à 0.

La variable qP est déduite comme suit.

- Si la matrice de résultat c se rapporte à un bloc luma résiduel et si sMbFlag est égale à 0
$$qP = QP_Y \quad (8-260)$$
- Autrement, si la matrice de résultat c se rapporte à un bloc luma résiduel et si sMbFlag est égale à 1
$$qP = QS_Y \quad (8-261)$$
- Autrement, si la matrice de résultat c se rapporte à un bloc chroma résiduel et si sMbFlag est égale à 0
$$qP = QP_C \quad (8-262)$$
- Autrement (la matrice de résultat c se rapporte à un bloc chroma résiduel et sMbFlag est égale à 1)
$$qP = QS_C \quad (8-263)$$

L'étalonnage des niveaux de coefficient de transformée de bloc 4x4 c_{ij} s'effectue comme suit.

- Si toutes les conditions suivantes sont Vraies
 - i est égal à 0
 - j est égal à 0
 - c se rapporte à un bloc luma résiduel codé en utilisant le mode de prédiction Intra_16x16 ou c se rapporte à un bloc résiduel chroma

la variable d_{00} est déduite comme:

$$d_{00} = c_{00} \quad (8-264)$$

- Autrement,

$$d_{ij} = (c_{ij} * \text{LevelScale}(qP \% 6, i, j)) \ll (qP / 6), \quad \text{avec } i, j = 0..3 \text{ excepté comme noté ci-dessus} \quad (8-265)$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément d_{ij} de d avec $i, j = 0..3$ excède la gamme de valeurs entières allant de -2^{15} à $2^{15}-1$ inclus.

Le processus de transformée doit convertir le bloc de coefficients de transformée étalonnés en un bloc d'échantillons de résultat d'une manière mathématiquement équivalente à ce qui suit.

D'abord, chaque rangée (horizontale) de coefficients de transformée étalonnés est transformée en utilisant comme suit une transformée inverse unidimensionnelle.

Un ensemble de valeurs intermédiaires est calculé comme suit.

$$e_{i0} = d_{i0} + d_{i2}, \quad \text{avec } i = 0..3 \quad (8-266)$$

$$e_{i1} = d_{i0} - d_{i2}, \quad \text{avec } i = 0..3 \quad (8-267)$$

$$e_{i2} = (d_{i1} \gg 1) - d_{i3}, \text{ avec } i = 0..3 \quad (8-268)$$

$$e_{i3} = d_{i1} + (d_{i3} \gg 1), \text{ avec } i = 0..3 \quad (8-269)$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément e_{ij} de e avec $i, j = 0..3$ excède la gamme des valeurs entières allant de -2^{15} à $2^{15}-1$ inclus.

Alors, le résultat de la transformée est calculé à partir de ces valeurs intermédiaires comme suit.

$$f_{i0} = e_{i0} + e_{i3}, \text{ avec } i = 0..3 \quad (8-270)$$

$$f_{i1} = e_{i1} + e_{i2}, \text{ avec } i = 0..3 \quad (8-271)$$

$$f_{i2} = e_{i1} - e_{i2}, \text{ avec } i = 0..3 \quad (8-272)$$

$$f_{i3} = e_{i0} - e_{i3}, \text{ avec } i = 0..3 \quad (8-273)$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément f_{ij} de f avec $i, j = 0..3$ excède la gamme de valeurs entières allant de -2^{15} à $2^{15}-1$ inclus.

Puis, chaque colonne (verticale) de la matrice résultante est transformée en utilisant la même transformée inverse unidimensionnelle comme suit.

On calcule comme suit un ensemble de valeurs intermédiaires.

$$g_{0j} = f_{0j} + f_{2j}, \text{ avec } j = 0..3 \quad (8-274)$$

$$g_{1j} = f_{0j} - f_{2j}, \text{ avec } j = 0..3 \quad (8-275)$$

$$g_{2j} = (f_{1j} \gg 1) - f_{3j}, \text{ avec } j = 0..3 \quad (8-276)$$

$$g_{3j} = f_{1j} + (f_{3j} \gg 1), \text{ avec } j = 0..3 \quad (8-277)$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément g_{ij} de g avec $i, j = 0..3$ excède la gamme de valeurs entières allant de -2^{15} à $2^{15}-1$ inclus.

Puis, le résultat transformé est calculé comme suit à partir de ces valeurs intermédiaires.

$$h_{0j} = g_{0j} + g_{3j}, \text{ avec } j = 0..3 \quad (8-278)$$

$$h_{1j} = g_{1j} + g_{2j}, \text{ avec } j = 0..3 \quad (8-279)$$

$$h_{2j} = g_{1j} - g_{2j}, \text{ avec } j = 0..3 \quad (8-280)$$

$$h_{3j} = g_{0j} - g_{3j}, \text{ avec } j = 0..3 \quad (8-281)$$

Le flux binaire ne doit contenir aucune donnée dont le résultat ferait qu'un élément h_{ij} of h avec $i, j = 0..3$ excède la gamme de valeurs entières allant de -2^{15} à $2^{15}-33$ inclus.

Après avoir effectué les deux transformées inverses unidimensionnelles, l'horizontale et la verticale, pour produire une matrice d'échantillons de transformée, les valeurs finales d'échantillon résiduel construit doivent être calculées comme:

$$r_{ij} = (h_{ij} + 2^5) \gg 6 \text{ avec } i, j = 0..3 \quad (8-282)$$

8.5.9 Processus de construction d'image antérieurement au processus de filtrage de démontage de blocs

Les entrées pour ce processus sont:

- luma4x4BlkIdx ou chroma4x4BlkIdx
- une matrice d'échantillon 4x4 u avec des éléments u_{ij} qui est un bloc luma ou chroma.

Les résultats de ce processus sont des blocs d'échantillon s construits antérieurement au processus de filtrage de démontage de blocs.

La position de l'échantillon luma supérieur gauche du macrobloc en cours est déduite en invoquant le processus de balayage inverse de macrobloc du § 6.4.1 avec CurrMbAddr en entrée et le résultat étant alloué à (xP, yP).

Lorsque u est un bloc luma, pour chaque échantillon u_{ij} du bloc luma 4x4, on applique ce qui suit.

- La position de l'échantillon supérieur gauche d'un bloc luma 4x4 avec l'indice luma4x4BlkIdx au sein du macrobloc est déduite en invoquant le processus de balayage inverse de bloc luma 4x4 du § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (xO, yO).
- En fonction de la variable MbaffFrameFlag, on applique ce qui suit.
 - Si MbaffFrameFlag est égal à 1 et que le macrobloc en cours est un macrobloc de champ

$$S'_L[xP + xO + j, yP + 2 * (yO + i)] = u_{ij} \quad \text{avec } i, j = 0..3 \quad (8-283)$$
 - Autrement (MbaffFrameFlag est égal à 0 ou le macrobloc en cours est un macrobloc de trame)

$$S'_L[xP + xO + j, yP + yO + i] = u_{ij} \quad \text{avec } i, j = 0..3 \quad (8-284)$$

Lorsque u est un bloc chroma, pour chaque échantillon u_{ij} du bloc chroma 4x4, on applique ce qui suit.

- L'indice C dans les variables S'_C et $pred_C$ est remplacé par Cb pour le composant chroma Cb et par Cr pour le composant chroma Cr.
- La position de l'échantillon supérieur gauche d'un bloc chroma 4x4 avec l'indice chroma4x4BlkIdx au sein du macrobloc est déduite comme suit:

$$xO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (8-285)$$

$$yO = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (8-286)$$
- En fonction de la variable MbaffFrameFlag, on applique ce qui suit:
 - Si MbaffFrameFlag est égal à 1 et que le macrobloc en cours est un macrobloc de champ

$$S'_C[(xP \gg 1) + xO + j, ((yP + 1) \gg 1) + 2 * (yO + i)] = u_{ij} \quad \text{avec } i, j = 0..3 \quad (8-287)$$
 - Autrement (MbaffFrameFlag est égal à 0 ou le macrobloc en cours est un macrobloc de trame)

$$S'_C[(xP \gg 1) + xO + j, ((yP + 1) \gg 1) + yO + i] = u_{ij} \quad \text{avec } i, j = 0..3 \quad (8-288)$$

8.6 Processus de décodage pour macroblocs P dans des tranches SP ou des macroblocs SI

Ce processus est invoqué lors du décodage de types de macrobloc P dans une tranche de type SP ou un macrobloc de type SI dans des tranches SI.

Les entrées pour ce processus sont les niveaux de coefficient de transformée résiduelle de prédiction et les échantillons prédits pour le macrobloc en cours.

Les résultats de ce processus sont les échantillons du macrobloc en cours décodés antérieurement au processus de filtrage de démontage de blocs.

Le présent paragraphe spécifie le processus de décodage du coefficient de transformée et le processus de construction d'image pour les macroblocs de type P dans les tranches SP et les macroblocs de type SI dans les tranches SI.

NOTE – Les tranches SP utilisent le codage interprédicatif pour exploiter les redondances temporelles dans la séquence, d'une manière similaire au codage de tranche P. A la différence du codage de tranche P, le codage de tranche SP permet cependant une reconstruction à l'identique d'une tranche même lorsque des images de référence différentes sont utilisées. Les tranches SI utilisent la prédiction spatiale, d'une façon similaire aux tranches I. Le codage de tranche SI permet une reconstruction à l'identique dans une tranche SP correspondante. Les propriétés des tranches SP et SI aident à fournir les fonctionnalités pour commuter les flux binaires, faire des raccords, donner un accès aléatoire, un accès rapide vers l'avant ou vers l'arrière, et pour la résistance aux erreurs ou leur correction.

Une tranche SP se compose de macroblocs codés comme macroblocs de type I ou macroblocs de type P.

Une tranche SI se compose de macroblocs codés comme macroblocs de type I ou macroblocs de type SI.

Le processus de décodage du coefficient de transformée et le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs pour les macroblocs de type I dans les tranches SI doivent être invoqués comme spécifié au § 8.5. Les macroblocs de type SI doivent être décodés comme décrit ci-dessous.

Lorsque le macrobloc en cours est codé P_Skip, toutes les valeurs de LumaLevel, ChromaDCLevel, ChromaACLevel sont mises égales à 0 pour le macrobloc en cours.

8.6.1 Processus de décodage SP pour images non commutatives

Ce processus est invoqué, lors du décodage de macroblocs de type P dans des tranches SP dans lesquelles sp_for_switch_flag est égal à 0.

Les entrées pour ce processus sont des échantillons d'interprédiction pour le macrobloc en cours, selon le § 8.4, et les niveaux de coefficient de transformée résiduelle de prédiction.

Les résultats de ce processus sont les échantillons du macrobloc en cours décodés avant le processus de filtrage de démontage de blocs.

Le présent paragraphe s'applique à tous les macroblocs dans les tranches SP dans lesquelles sp_for_switch_flag est égal à 0, excepté celles avec le mode de prédiction de macrobloc égal à Intra_4x4 ou Intra_16x16. Il ne s'applique pas aux tranches SI.

8.6.1.1 Processus de décodage de coefficient de transformée luma

Les entrées pour ce processus sont des échantillons luma d'interprédiction pour le macrobloc en cours pred_L d'après le § 8.4, les niveaux de coefficient de transformée résiduelle de prédiction, LumaLevel, et l'indice du bloc luma 4x4 luma4x4BlkIdx.

Les résultats de ce processus sont les échantillons luma du macrobloc en cours décodés antérieurement au processus de filtrage de démontage de blocs.

La position de l'échantillon supérieur gauche du bloc luma 4x4 avec l'indice luma4x4BlkIdx au sein du macrobloc en cours est déduite en invoquant le processus de balayage inverse de bloc luma 4x4 du § 6.4.3 avec luma4x4BlkIdx comme entrée et le résultat étant alloué à (x, y).

Soit la variable p une matrice 4x4 d'échantillons de prédiction avec les éléments p_{ij} qu'on calcule comme suit.

$$p_{ij} = \text{pred}_L[x + j, y + i] \quad \text{avec } i, j = 0..3 \quad (8-289)$$

La variable p est transformée, ce qui produit des coefficients de transformée c^p conformément à :

$$c^p = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix} \quad (8-290)$$

Le processus inverse de balayage de coefficient de transformée comme décrit au § 8.5.4 est invoqué avec LumaLevel[luma4x4BlkIdx] comme entrée et la matrice bidimensionnelle c^f avec les éléments c_{ij}^f comme résultat.

Les coefficients de transformée de prédiction résiduels c^f sont étalonnés en utilisant le paramètre de quantification QP_Y, et ajoutés au coefficient de transformées du bloc de prédiction c^p avec i, j = 0..3 comme suit.

$$c_{ij}^s = c_{ij}^p + (((c_{ij}^f * \text{LevelScale}(QP_Y \% 6, i, j) * A_{ij}) \ll (QP_Y / 6)) \gg 6) \quad (8-291)$$

où $\text{LevelScale}(m, i, j)$ est spécifié dans l'équation 8-252, et où A_{ij} est spécifié comme:

$$A_{ij} = \begin{cases} 16 & \text{pour } (i, j) \in \{(0,0), (0,2), (2,0), (2,2)\}, \\ 25 & \text{pour } (i, j) \in \{(1,1), (1,3), (3,1), (3,3)\}, \\ 20 & \text{autrement;} \end{cases} \quad (8-292)$$

La fonction $\text{LevelScale2}(m, i, j)$, utilisée dans les formules ci-dessous, est spécifiée comme:

$$\text{LevelScale2}(m, i, j) = \begin{cases} w_{m0} & \text{pour } (i, j) \in \{(0,0), (0,2), (2,0), (2,2)\}, \\ w_{m1} & \text{pour } (i, j) \in \{(1,1), (1,3), (3,1), (3,3)\}, \\ w_{m2} & \text{autrement;} \end{cases} \quad (8-293)$$

où le premier et le second indice de w sont les indices respectivement de rangée et de colonne de la matrice spécifiée comme:

$$w = \begin{bmatrix} 13107 & 5243 & 8066 \\ 11916 & 4660 & 7490 \\ 10082 & 4194 & 6554 \\ 9362 & 3647 & 5825 \\ 8192 & 3355 & 5243 \\ 7282 & 2893 & 4559 \end{bmatrix} \quad (8-294)$$

La somme résultante, c^s , est quantifiée avec un paramètre de quantification QS_Y et avec $i, j = 0..3$ comme suit.

$$c_{ij} = (\text{Sign}(c_{ij}^s) * (\text{Abs}(c_{ij}^s) * \text{LevelScale2}(QS_Y \% 6, i, j) + (1 \ll (14 + QS_Y / 6)))) \gg (15 + QS_Y / 6) \quad (8-295)$$

Le processus d'étalonnage et de transformation pour les blocs 4x4 résiduels, comme spécifié au § 8.5.8, est invoqué avec c comme entrée et r comme résultat.

La matrice 4x4 u avec les éléments u_{ij} est déduite comme suit.

$$u_{ij} = \text{Clip1}(r_{ij}) \text{ avec } i, j = 0..3 \quad (8-296)$$

Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.9 est invoqué avec luma4x4BlkIdx , u comme entrée et S' comme résultat.

8.6.1.2 Processus de décodage du coefficient de transformée chroma

Les entrées pour ce processus sont les échantillons chroma d'interprédiction pour le macrobloc en cours du § 8.4 et les niveaux de coefficient de transformée de prédiction résiduel, ChromaDCLevel et ChromaACLevel .

Les résultats de ce processus sont les échantillons chroma du macrobloc en cours décodés antérieurement au processus de filtrage de démontage de blocs.

Ce processus est invoqué deux fois: une fois pour le composant Cb et une fois pour le composant Cr. Le composant est mentionné en remplaçant C par Cb pour le composant Cb et C par Cr pour le composant Cr. Soit $iCbCr$ qui désigne le composant chroma en cours.

Pour chaque bloc 4x4 du composant chroma en cours indexé en utilisant chroma4x4BlkIdx avec chroma4x4BlkIdx égal à 0..3, on applique ce qui suit:

– La position de l'échantillon supérieur gauche d'un bloc chroma 4x4 avec l'indice chroma4x4BlkIdx au sein du macrobloc est déduite comme suit.

$$x = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 0) \quad (8-297)$$

$$y = \text{InverseRasterScan}(\text{chroma4x4BlkIdx}, 4, 4, 8, 1) \quad (8-298)$$

- Soit p une matrice 4x4 d'échantillons de prédiction avec des éléments p_{ij} qu'on calcule comme suit.

$$p_{ij} = \text{pred}_c[x + j, y + i] \text{ avec } i, j = 0..3 \quad (8-299)$$

- La matrice 4x4 p est transformée pour produire les coefficients de transformée $c^p(\text{chroma4x4BlkIdx})$ en utilisant l'équation 8-290.
- La variable chromaList , qui est une liste de 16 entrées, est déduite. $\text{chromaList}[0]$ est mise égale à 0. Les $\text{chromaList}[k]$ avec les indices $k = 1..15$ sont spécifiées comme suit.

$$\text{chromaList}[k] = \text{ChromaACLevel}[\text{iCbCr}][\text{chroma4x4BlkIdx}][k - 1] \quad (8-300)$$

- Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.4, est invoqué avec chromaList comme entrée et la matrice 4x4 c^r comme résultat.
- Les coefficients de transformée de prédiction résiduels c^r sont étalonnés en utilisant le paramètre de quantification QP_C , et ajoutés aux coefficients de transformée du bloc de prédiction c^p avec $i, j = 0..3$, sauf pour la combinaison $i = 0, j = 0$, comme suit.

$$c_{ij}^s = c_{ij}^p(\text{chroma4x4BlkIdx}) + (((c_{ij}^r * \text{LevelScale}(QP_C \% 6, i, j) * A_{ij}) \ll (QP_C / 6)) \gg 6) \quad (8-301)$$

- La somme résultante, c^s , est quantifiée avec un paramètre de quantification QS_C et avec $i, j = 0..3$, sauf pour la combinaison $i = 0, j = 0$, comme suit. La déduction de $c_{00}(\text{chroma4x4BlkIdx})$ est décrite plus loin dans le présent paragraphe.

$$c_{ij}(\text{chroma4x4BlkIdx}) = (\text{Sign}(c_{ij}^s) * (\text{Abs}(c_{ij}^s) * \text{LevelScale2}(QS_C \% 6, i, j) + (1 \ll (14 + QS_C / 6)))) \gg (15 + QS_C / 6) \quad (8-302)$$

- Le processus d'étalonnage et de transformation pour les blocs 4x4 résiduels, comme spécifié au § 8.5.8, est invoqué avec $c(\text{chroma4x4BlkIdx})$ comme entrée et r comme résultat.
 - La matrice 4x4 u avec les éléments u_{ij} est déduite comme suit.
- $$u_{ij} = \text{Clip1}(r_{ij}) \text{ avec } i, j = 0..3 \quad (8-303)$$
- Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.9 est invoqué avec chroma4x4BlkIdx , u comme entrée et S' comme résultat.

Le calcul du niveau de coefficient de transformée DC $c_{00}(\text{chroma4x4BlkIdx})$ est spécifié comme suit. Les coefficients de transformée DC des quatre blocs 4x4 chroma de prédiction du composant en cours du macrobloc sont assemblés en une matrice 2x2 avec des éléments $c_{00}^p(\text{chroma4x4BlkIdx})$ et une transformée 2x2 est appliquée aux coefficients de transformée DC comme suit.

$$dc^p = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} c_{00}^p(0) & c_{00}^p(1) \\ c_{00}^p(2) & c_{00}^p(3) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8-304)$$

Les niveaux de coefficient de transformée résiduel de prédiction DC chroma , $\text{ChromaDCLevel}[\text{iCbCr}][k]$ avec $k = 0..3$ sont étalonnés en utilisant le paramètre de quantification QP , et ajoutés aux coefficients de transformée DC de prédiction comme suit.

$$dc_{ij}^s = dc_{ij}^p + (((\text{ChromaDCLevel}[\text{iCbCr}][j * 2 + i] * \text{LevelScale}(QP_C \% 6, 0, 0) * A_{00}) \ll (QP_C / 6)) \gg 5) \text{ avec } i, j = 0, 1 \quad (8-305)$$

La matrice 2x2 dc^s , est quantifiée en utilisant le paramètre de quantification QS_C comme suit:

$$dc_{ij}^r = (\text{Sign}(dc_{ij}^s) * (\text{Abs}(dc_{ij}^s) * \text{LevelScale2}(QS_C \% 6, 0, 0) + (1 \ll (15 + QS_C / 6)))) \gg (16 + QS_C / 6) \text{ avec } i, j = 0, 1 \quad (8-306)$$

La matrice 2x2 f avec les éléments f_{ij} et $i, j = 0..1$ est déduite comme suit.

$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} dc_{00}^r & dc_{01}^r \\ dc_{10}^r & dc_{11}^r \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8-307)$$

L'étalonnage des éléments f_{ij} de f est effectué comme suit:

– Si QS_C est supérieur ou égal à 6, les $c_{00}()$ sont déduits par

$$c_{00}(j * 2 + i) = (f_{ij} * \text{LevelScale}(QS_C \% 6, 0, 0)) \ll (QS_C / 6 - 1) \text{ avec } i, j = 0, 1 \quad (8-308)$$

– Autrement (QS_C est inférieur à 6), les $c_{00}()$ sont déduits par

$$c_{00}(j * 2 + i) = (f_{ij} * \text{LevelScale}(QS_C \% 6, 0, 0)) \gg 1 \text{ avec } i, j = 0, 1 \quad (8-309)$$

8.6.2 Processus de décodage de tranche SP et SI pour images commutantes

Ce processus est invoqué, lors du décodage de macroblocs de type P dans des tranches SP dans lesquelles `sp_for_switch_flag` est égal à 1 et lors du décodage de macroblocs de type SI dans des tranches SI.

Les entrées pour ce processus sont les niveaux de coefficient de transformée résiduel de prédiction et les matrices d'échantillons de prédiction $pred_L$, $pred_{Cb}$, $pred_{Cr}$ pour le macrobloc en cours.

Les résultats de ce processus sont les échantillons du macrobloc en cours décodés antérieurement au processus de filtrage de démontage de blocs.

8.6.2.1 Processus de décodage de coefficient de transformée luma

Les entrées pour ce processus sont des échantillons luma de prédiction $pred_L$ et les niveaux de coefficient de transformée résiduel de prédiction, `LumaLevel`.

Les résultats de ce processus sont les échantillons luma du macrobloc en cours décodés antérieurement au processus de filtrage de démontage de blocs.

La matrice 4x4 p avec des éléments p_{ij} avec $i, j = 0..3$, est calculée comme au § 8.6.1.1. Elle est transformée conformément à l'équation 8-290 pour produire les coefficients de transformée c^p . Ces coefficients de transformée sont alors quantifiés avec le paramètre de quantification QS_Y , comme suit:

$$c_{ij}^s = (\text{Sign}(c_{ij}^p) * (\text{Abs}(c_{ij}^p) * \text{LevelScale2}(QS_Y \% 6, i, j) + (1 \ll (14 + QS_Y / 6)))) \gg (15 + QS_Y / 6) \text{ avec } i, j = 0..3 \quad (8-310)$$

Le processus inverse de balayage de coefficient de transformée décrit au § 8.5.4 est invoqué avec `LumaLevel[luma4x4BlkIdx]` comme entrée et la matrice bidimensionnelle c^r avec les éléments c_{ij}^r comme résultat.

La matrice 4x4 c avec les éléments c_{ij} avec $i, j = 0..3$ est déduite comme suit.

$$c_{ij} = c_{ij}^r + c_{ij}^s \text{ avec } i, j = 0..3 \quad (8-311)$$

Le processus d'étalonnage et de transformation pour blocs 4x4 résiduels, comme spécifié au § 8.5.8, est invoqué avec c comme entrée et r comme résultat.

La matrice 4x4 u avec les éléments u_{ij} est déduite comme suit.

$$u_{ij} = \text{Clip1}(r_{ij}) \text{ avec } i, j = 0..3 \quad (8-312)$$

Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.9 est invoqué avec `luma4x4BlkIdx`, u comme entrée et S' comme résultat.

8.6.2.2 Processus de décodage du coefficient de transformée chroma

Les entrées pour ce processus sont des échantillons chroma prédits pour le macrobloc en cours selon le § 8.4 et les niveaux de coefficient de transformée résiduel de prédiction, `ChromaDCLevel` et `ChromaACLevel`.

Les résultats de ce processus sont les échantillons chroma du macrobloc en cours décodés antérieurement au processus de filtrage de démontage de blocs.

Ce processus est invoqué deux fois: une fois pour le composant Cb et une fois pour le composant Cr. Le composant est mentionné en remplaçant C par Cb pour le composant Cb et C par Cr pour le composant Cr. Soit iCbCr qui choisit le composant chroma en cours.

Pour chaque bloc 4x4 du composant chroma en cours indexé en utilisant chroma4x4BlkIdx avec chroma4x4BlkIdx égal à 0..3, on applique ce qui suit.

- La matrice 4x4 p avec les éléments p_{ij} avec $i, j = 0..3$ est déduite comme au § 8.6.1.2. Elle est transformée conformément à l'équation 8-290 pour produire des coefficients de transformée $c_{ij}^p(\text{chroma4x4BlkIdx})$. Ces coefficients de transformée sont alors quantifiés avec le paramètre de quantification QS_C , avec $i, j = 0..3$, sauf pour la combinaison $i = 0, j = 0$ comme suit. Le traitement de $c_{00}^p(\text{chroma4x4BlkIdx})$ est décrit ci-après dans le présent paragraphe.

$$c_{ij}^s = (\text{Sign}(c_{ij}^p(\text{chroma4x4BlkIdx})) * (\text{Abs}(c_{ij}^p(\text{chroma4x4BlkIdx})) * \text{LevelScale2}(QS_C \% 6, i, j) + (1 \ll (14 + QS_C / 6)))) \gg (15 + QS_C / 6) \quad (8-313)$$

- On calcule la variable chromaList, qui est une liste de 16 entrées. chromaList[0] est mise égale à 0. Les chromaList[k] avec l'indice $k = 1..15$ sont spécifiées comme suit:

$$\text{chromaList}[k] = \text{ChromaACLevel}[iCbCr][\text{chroma4x4BlkIdx}][k - 1] \quad (8-314)$$

- Le processus inverse de balayage de coefficient de transformée, comme décrit au § 8.5.4, est invoqué avec chromaList comme entrée et la matrice bidimensionnelle $c^r(\text{chroma4x4BlkIdx})$ avec les éléments $c_{ij}^r(\text{chroma4x4BlkIdx})$ comme résultat.

- La matrice 4x4 c(chroma4x4BlkIdx) avec les éléments $c_{ij}(\text{chroma4x4BlkIdx})$ avec $i, j = 0..3$, sauf pour la combinaison $i = 0, j = 0$ est déduite comme suit. La déduction de $c_{00}(\text{chroma4x4BlkIdx})$ est décrite ci-dessous.

$$c_{ij}(\text{chroma4x4BlkIdx}) = c_{ij}^r(\text{chroma4x4BlkIdx}) + c_{ij}^s \quad (8-315)$$

- Le processus d'étalonnage et de transformation pour les blocs 4x4 résiduels, comme spécifié au § 8.5.8, est invoqué avec c(chroma4x4BlkIdx) comme entrée et r comme résultat.

- La matrice 4x4 u avec les éléments u_{ij} est déduite comme suit:

$$u_{ij} = \text{Clip1}(r_{ij}) \text{ avec } i, j = 0..3 \quad (8-316)$$

- Le processus de construction d'image antérieurement au processus de filtrage de démontage de blocs du § 8.5.9 est invoqué avec chroma4x4BlkIdx, u comme entrée et S' comme résultat.

Le calcul du niveau DC de coefficient de transformée $c_{00}(\text{chroma4x4BlkIdx})$ est spécifié comme suit. Les coefficients de transformée DC des 4 blocs 4x4 de prédiction chroma du composant en cours du macrobloc, $c_{00}^p(\text{chroma4x4BlkIdx})$, sont assemblés en une matrice 2x2, et une transformée 2x2 est appliquée aux coefficients de transformée DC de ces blocs conformément à l'équation (8-304) résultant en coefficients de transformée DC dc_{ij}^p .

Ces coefficients de transformée DC sont alors quantifiés avec le paramètre de quantification QS_C , tel que donné par:

$$dc_{ij}^s = (\text{Sign}(dc_{ij}^p) * (\text{Abs}(dc_{ij}^p) * \text{LevelScale2}(QS_C \% 6, 0, 0) + (1 \ll (15 + QS_C / 6)))) \gg (16 + QS_C / 6) \quad \text{avec } i, j = 0, 1 \quad (8-317)$$

Les coefficients de transformée résiduels de prédiction DC chroma analysés grammaticalement, ChromaDCLevel[iCbCr][k] avec $k = 0..3$, sont ajoutés à ces coefficients de transformée DC quantifiés du bloc de prédiction, comme donné par:

$$dc_{ij}^r = dc_{ij}^s + \text{ChromaDCLevel}[iCbCr][j * 2 + i] \text{ avec } i, j = 0, 1 \quad (8-318)$$

La matrice 2x2 f avec les éléments f_{ij} et $i, j = 0..1$ est calculée en utilisant l'équation 8-307.

La matrice 2x2 f avec les éléments f_{ij} et $i, j = 0..1$ est copiée comme suit.

8.7 Processus de filtrage de démontage de blocs

On doit appliquer un filtrage conditionnel à toutes les bordures de bloc 4x4 d'une image, sauf les bordures qui sont à la frontière de l'image et toute bordure pour laquelle le processus de filtrage de démontage de blocs est désactivé par `disable_deblocking_filter_idc`, comme spécifié ci-dessous. Ce processus de filtrage est effectué sur la base du macrobloc une fois terminé le processus de construction d'image précédant le processus de filtrage de démontage de blocs (spécifié aux § 8.5 et 8.6) pour l'image décodée entière, avec tous les macroblocs d'une image traités dans l'ordre croissant des adresses de macroblocs.

NOTE – Avant d'effectuer le processus de filtrage de démontage de blocs pour chaque macrobloc, les échantillons sortis des blocs du macrobloc ou de la paire de macroblocs de dessus (s'il en est) et du macrobloc ou de la paire de macroblocs à gauche (s'il en est) du macrobloc en cours, sont toujours disponibles du fait que le processus de filtrage de démontage de blocs est effectué une fois terminé le processus de construction d'image précédant le processus de filtrage de démontage de blocs pour l'image décodée entière.

Le processus de filtrage de démontage de blocs est invoqué séparément pour les composants luma et chroma. Pour chaque macrobloc, on filtre d'abord les bordures verticales, de gauche à droite, puis on filtre les bordures horizontales du haut en bas. Le processus de filtrage de démontage de blocs luma est effectué sur quatre bordures de 16 échantillons et le processus de filtrage de démontage de blocs pour chaque composant chroma est effectué sur deux bordures de 8 échantillons, pour la direction horizontale, comme indiqué sur le côté gauche de la Figure 8-9 et pour la direction verticale, comme indiqué sur le côté droit de la Figure 8-9. Les valeurs d'échantillon au-dessus et à gauche du macrobloc en cours qui ont déjà été modifiés par l'opération du processus de filtrage de démontage de blocs sur des macroblocs précédents doivent être utilisés en entrée du processus de filtrage de démontage de blocs sur le macrobloc en cours et peuvent encore être modifiés pendant le filtrage du macrobloc en cours. Les valeurs d'échantillon modifiées pendant le filtrage des bordures verticales sont utilisées en entrée pour le filtrage des bordures horizontales pour le même macrobloc.

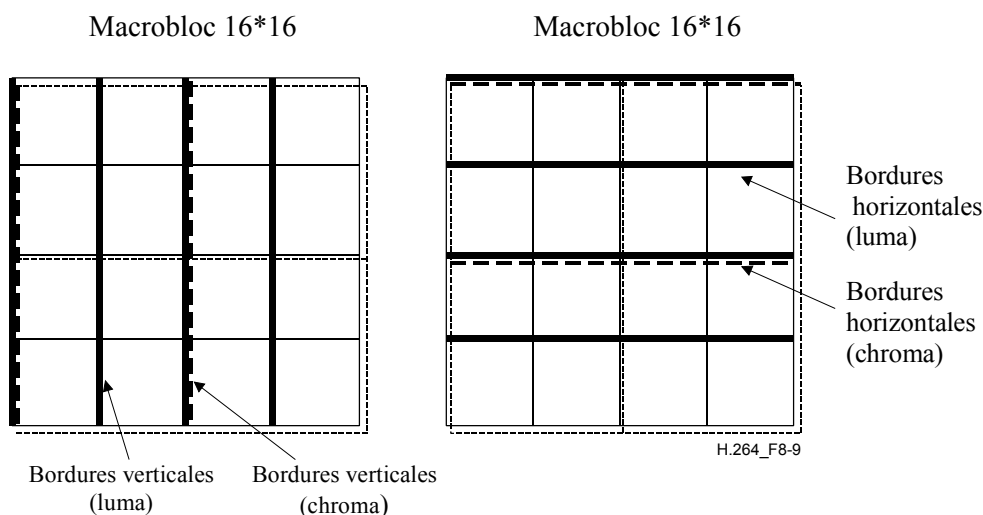


Figure 8-9 – Frontières dans un macrobloc à filtrer (frontières luma montrées avec lignes continues et frontières chroma avec des pointillés)

Pour chaque macrobloc en ordre croissant de `mbAddr`, on applique ce qui suit:

1. Les variables `fieldModeMbFlag`, `filterInternalEdgesFlag`, `filterLeftMbEdgeFlag` et `filterTopMbEdgeFlag` sont déduites comme suit.
 - La variable `fieldModeMbFlag` est déduite comme suit.

- Si l'une des conditions suivantes est Vraie, fieldModeMbFlag est mise égale à 1.
 - field_pic_flag est égal à 1.
 - MbaffFrameFlag est égal à 1 et le macrobloc mbAddr est un macrobloc de champ.
 - Autrement, fieldModeMbFlag est mis égal à 0.
- La variable filterInternalEdgesFlag est déduite comme suit.
 - si disable_deblocking_filter_idc pour la tranche qui contient le macrobloc mbAddr est égal à 1, la variable filterInternalEdgesFlag est mise égale à 0;
 - autrement (disable_deblocking_filter_idc pour la tranche qui contient le macrobloc mbAddr n'est pas égal à 1), la variable filterInternalEdgesFlag est mise égale à 1.
- La variable filterLeftMbEdgeFlag est déduite comme suit.
 - Si l'une des conditions suivantes est vraie, la variable filterLeftMbEdgeFlag est mise égale à 0.
 - La bordure verticale gauche de macrobloc du mbAddr du macrobloc représente une frontière d'image.
 - Pour la tranche qui contient le mbAddr du macrobloc, disable_deblocking_filter_idc est égal à 1.
 - Pour la tranche qui contient le mbAddr du macrobloc, disable_deblocking_filter_idc est égal à 2 et la bordure verticale gauche de macrobloc du mbAddr du macrobloc représente une frontière de tranche.
 - Autrement, la variable filterLeftMbEdgeFlag est mise égale à 1.
- La variable filterTopMbEdgeFlag est déduite comme suit.
 - Si l'une des conditions suivantes est vraie, la variable filterTopMbEdgeFlag est mise égale à 0.
 - La bordure horizontale supérieure de macrobloc du mbAddr du macrobloc représente une frontière d'image.
 - Pour la tranche qui contient le mbAddr du macrobloc, disable_deblocking_filter_idc est égal à 1.
 - Pour la tranche qui contient le mbAddr du macrobloc, disable_deblocking_filter_idc est égal à 2 et la bordure horizontale supérieure de macrobloc du mbAddr du macrobloc représente une frontière de tranche.
 - Autrement, la variable filterTopMbEdgeFlag est mise égale à 1.

2. Etant donné les variables fieldModeMbFlag, filterInternalEdgesFlag, filterLeftMbEdgeFlag et filterTopMbEdgeFlag, le filtrage de démontage de blocs est commandé comme suit.

- Lorsque filterLeftMbEdgeFlag est égal à 1, le filtrage de la bordure luma verticale gauche est spécifié comme suit:
 - Le processus spécifié au § 8.7.1 est invoqué avec mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (0, k)$ avec $k = 0..15$ en entrée et S'_L comme résultat.
- Lorsque filterInternalEdgesFlag est égal à 1, le filtrage des bordures luma verticales internes est spécifié comme suit.
 - le processus spécifié au § 8.7.1 est invoqué avec mbAddr, chromaEdgeFlag = 0, verticalEdgeFlag = 1, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (4, k)$ avec $k = 0..15$ en entrée et S'_L comme résultat;

- le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 0$, $verticalEdgeFlag = 1$, $fieldModeFilteringFlag = fieldModeMbFlag$, et $(xE_k, yE_k) = (8, k)$ avec $k = 0..15$ en entrée et S'_L comme résultat;
- le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 0$, $verticalEdgeFlag = 1$, $fieldModeFilteringFlag = fieldModeMbFlag$, et $(xE_k, yE_k) = (12, k)$ avec $k = 0..15$ en entrée et S'_L comme résultat.
- Lorsque $filterTopMbEdgeFlag$ est égal à 1, le filtrage de la bordure luma horizontale supérieure est spécifié comme suit.
 - Si $MbaffFrameFlag$ est égal à 1, $(mbAddr \% 2)$ est égal à 0, $mbAddr$ est supérieur ou égal à $2 * PicWidthInMbs$, le macrobloc $mbAddr$ est un macrobloc de trame, et le macrobloc $(mbAddr - 2 * PicWidthInMbs + 1)$ est un macrobloc de champ, on applique ce qui suit:
 - le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 0$, $verticalEdgeFlag = 0$, $fieldModeFilteringFlag = 1$, et $(xE_k, yE_k) = (k, 0)$ avec $k = 0..15$ en entrée et S'_L comme résultat;
 - le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 0$, $verticalEdgeFlag = 0$, $fieldModeFilteringFlag = 1$, et $(xE_k, yE_k) = (k, 1)$ avec $k = 0..15$ en entrée et S'_L comme résultat.
 - Autrement, le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 0$, $verticalEdgeFlag = 0$, $fieldModeFilteringFlag = fieldModeMbFlag$, et $(xE_k, yE_k) = (k, 0)$ avec $k = 0..15$ en entrée et S'_L comme résultat.
- Lorsque $filterInternalEdgesFlag$ est égal à 1, le filtrage des bordures luma horizontales internes est spécifié comme suit.
 - le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 0$, $verticalEdgeFlag = 0$, $fieldModeFilteringFlag = fieldModeMbFlag$, et $(xE_k, yE_k) = (k, 4)$ avec $k = 0..15$ en entrée et S'_L comme résultat;
 - le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 0$, $verticalEdgeFlag = 0$, $fieldModeFilteringFlag = fieldModeMbFlag$, et $(xE_k, yE_k) = (k, 8)$ avec $k = 0..15$ en entrée et S'_L comme résultat;
 - le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 0$, $verticalEdgeFlag = 0$, $fieldModeFilteringFlag = fieldModeMbFlag$, et $(xE_k, yE_k) = (k, 12)$ avec $k = 0..15$ en entrée et S'_L comme résultat.
- Pour les deux composants chroma $iCbCr = 0$ et 1, on applique ce qui suit.
 - Lorsque $filterLeftMbEdgeFlag$ est égal à 1, le filtrage de la bordure verticale gauche est spécifié comme suit.
 - le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 1$, $iCbCr$, $verticalEdgeFlag = 1$, $fieldModeFilteringFlag = 1$, et $(xE_k, yE_k) = (0, k)$ avec $k = 0..7$ en entrée, et S'_C avec C remplacé par Cb pour $iCbCr = 0$ et C remplacé par Cr pour $iCbCr = 1$ comme résultat.
 - Lorsque $filterInternalEdgesFlag$ est égal à 1, le filtrage de la bordure chroma verticale interne est spécifié comme suit.
 - le processus spécifié au § 8.7.1 est invoqué avec $mbAddr$, $chromaEdgeFlag = 1$, $iCbCr$, $verticalEdgeFlag = 1$, $fieldModeFilteringFlag = fieldModeMbFlag$, et $(xE_k, yE_k) = (4, k)$ avec $k = 0..7$ en entrée, et S'_C avec C remplacé par Cb pour $iCbCr = 0$ et C remplacé par Cr pour $iCbCr = 1$ comme résultat.
- Lorsque $filterTopMbEdgeFlag$ est égal à 1, le filtrage de la bordure chroma horizontale supérieure est spécifié comme suit.

- si MbaffFrameFlag est égal à 1, $(mbAddr \% 2)$ est égal à 0, mbAddr est supérieur ou égal à $2 * PicWidthInMbs$, le macrobloc mbAddr est un macrobloc de trame, et le macrobloc $(mbAddr - 2 * PicWidthInMbs + 1)$ est un macrobloc de champ, on applique ce qui suit.
 - le processus spécifié au § 8.7.1 est invoqué avec mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFilteringFlag = 1, et $(xE_k, yE_k) = (k, 0)$ avec $k = 0..7$ en entrée, et S'_C avec C remplacé par Cb pour iCbCr = 0 et C remplacé par Cr pour iCbCr = 1 comme résultat;
 - le processus spécifié au § 8.7.1 est invoqué avec mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFilteringFlag = 1, et $(xE_k, yE_k) = (k, 1)$ avec $k = 0..7$ en entrée, et S'_C avec C remplacé par Cb pour iCbCr = 0 et C remplacé par Cr pour iCbCr = 1 comme résultat.
- Autrement, le processus spécifié au § 8.7.1 est invoqué avec mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 0)$ avec $k = 0..7$ en entrée, et S'_C avec C remplacé par Cb pour iCbCr = 0 et C remplacé par Cr pour iCbCr = 1 comme résultat.
- Lorsque filterInternalEdgesFlag est égal à 1, le filtrage de la bordure chroma horizontale interne est spécifié comme suit.
 - le processus spécifié au § 8.7.1 est invoqué avec mbAddr, chromaEdgeFlag = 1, iCbCr, verticalEdgeFlag = 0, fieldModeFilteringFlag = fieldModeMbFlag, et $(xE_k, yE_k) = (k, 4)$ avec $k = 0..7$ en entrée, et S'_C avec C remplacé par Cb pour iCbCr = 0 et C remplacé par Cr pour iCbCr = 1 comme résultat.

NOTE – Lorsque le filtrage de mode de champ (fieldModeFilteringFlag est égal à 1) est appliqué sur les bordures horizontales supérieures d'un macrobloc de trame, ce filtrage vertical sur la frontière supérieure ou inférieure du macrobloc peut impliquer certains échantillons qui s'étendent au-delà de la bordure interne d'un bloc qui est aussi filtré en interne en mode de trame.

NOTE – Dans tous les cas sont filtrées trois bordures luma horizontales, une bordure chroma horizontale pour Cb, et une bordure chroma horizontale pour Cr, qui sont internes à un macrobloc. Lorsque le filtrage de mode de champ (fieldModeFilteringFlag est égal à 1) est appliqué aux bordures supérieures d'un macrobloc de trame, on filtre deux bordures luma horizontales, deux bordures chroma horizontales pour Cb, et deux bordures chroma horizontales pour Cr entre le macrobloc de trame et la paire de macroblocs ci-dessus, en utilisant le filtrage de mode de champ pour au total jusqu'à cinq bordures luma horizontales, trois bordures chroma horizontales pour Cb, et trois bordures chroma horizontales pour Cr filtrées qui sont considérées comme contrôlées par le macrobloc de trame. Dans tous les autres cas, au plus quatre bordures luma horizontales, deux chroma horizontales pour Cb, et deux chroma horizontales pour Cr sont filtrées et considérées comme contrôlées par un macrobloc particulier.

Finalement, les matrices S'_L , S'_{Cb} , S'_{Cr} sont respectivement alloués aux matrices S_L , S_{Cb} , S_{Cr} (qui représentent l'image décodée).

8.7.1 Processus de filtrage pour bordures de bloc

Les entrées pour ce processus sont mbAddr, chromaEdgeFlag, l'indice de composant chroma iCbCr (lorsque chromaEdgeFlag est égal à 1), verticalEdgeFlag, fieldModeFilteringFlag, et un ensemble de seize localisations d'échantillon luma (lorsque chromaEdgeFlag est égal à 0) ou huit localisations d'échantillon chroma (lorsque chromaEdgeFlag est égal à 1) (xE_k, yE_k) , avec $k = 0 .. nE - 1$,

exprimé par rapport au coin supérieur gauche du macrobloc mbAddr. L'ensemble de localisations d'échantillon (xE_k, yE_k) représente les localisations d'échantillon immédiatement à la droite d'une bordure verticale (lorsque verticalEdgeFlag est égal à 1) ou immédiatement au-dessous d'une bordure horizontale (lorsque verticalEdgeFlag est égal à 0).

La variable nE est déduite comme suit:

- si chromaEdgeFlag est égal à 0, nE est 16;
- autrement (chromaEdgeFlag est égal à 1), nE est 8.

Soit s' une variable spécifiant une matrice d'échantillons luma ou chroma, calculée comme suit:

- si chromaEdgeFlag est égal à 0, s' représente la matrice d'échantillons luma S'_L de l'image en cours;
- autrement, si chromaEdgeFlag est égal à 1 et iCbCr est égal à 0, s' représente la matrice d'échantillons chroma S'_{Cb} du composant chroma Cb de l'image en cours;
- autrement (chromaEdgeFlag est égal à 1 et iCbCr est égal à 1), s' représente la matrice d'échantillons chroma S'_{Cr} du composant chroma Cr de l'image en cours.

La variable dy est déduite comme suit:

- si fieldModeFilteringFlag est égal à 1 et MbaffFrameFlag est égal à 1, dy est mise égale à 2;
- autrement (fieldModeFilteringFlag est égal à 0 ou MbaffFrameFlag est égal à 0), dy est mise égale à 1.

La position de l'échantillon luma supérieur gauche du macrobloc mbAddr est déduite en invoquant le processus de balayage inverse de macrobloc du § 6.4.1 avec mbAddr en entrée et le résultat étant alloué à (xP, yP) .

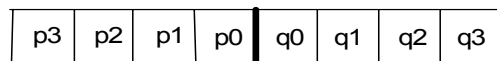


Figure 8-10 – Convention pour décrire les échantillons sur une frontière de bloc 4x4 horizontale ou verticale

Pour chaque localisation d'échantillon (xE_k, yE_k) , $k = 0 .. nE - 1$, on applique ce qui suit.

- Le processus de filtrage s'applique à un ensemble de huit échantillons sur une bordure horizontale ou verticale de bloc 4x4 noté p_i et q_i avec $i = 0..3$ comme indiqué à la Figure 8-10 avec la bordure se tenant entre p_0 et q_0 . p_i et q_i avec $i = 0..3$ sont spécifiés comme suit:
 - si verticalEdgeFlag est égal à 1,

$$q_i = s'[xP + xE_k + i, yP + yE_k] \quad (8-320)$$

$$p_i = s'[xP + xE_k - i - 1, yP + yE_k] \quad (8-321)$$
 - autrement (verticalEdgeFlag est égal à 0),

$$q_i = s'[xP + xE_k, yP + dy * (yE_k + i) - (yE_k \% 2)] \quad (8-322)$$

$$p_i = s'[xP + xE_k, yP + dy * (yE_k - i - 1) - (yE_k \% 2)] \quad (8-323)$$
- Le processus spécifié au § 8.7.2 est invoqué avec les valeurs d'échantillon p_i et q_i ($i = 0..3$), chromaEdgeFlag, verticalEdgeFlag, et fieldModeFilteringFlag en entrée, et le résultat est alloué aux valeurs d'échantillon résultantes filtrées p'_i et q'_i avec $i = 0..2$.
- Les valeurs d'échantillon d'entrée p_i et q_i avec $i = 0..2$ sont remplacées par les valeurs d'échantillon résultantes filtrées correspondantes p'_i et q'_i avec $i = 0..2$ au sein de la matrice d'échantillon s' comme suit:

– si verticalEdgeFlag est égal à 1,

$$s[xP + xE_k + i, yP + yE_k] = q'_i \quad (8-324)$$

$$s[xP + xE_k - i - 1, yP + yE_k] = p'_i \quad (8-325)$$

– autrement (verticalEdgeFlag est égal à 0),

$$s[xP + xE_k, yP + dy * (yE_k + i) - (yE_k \% 2)] = q'_i \quad (8-326)$$

$$s[xP + xE_k, yP + dy * (yE_k - i - 1) - (yE_k \% 2)] = p'_i \quad (8-327)$$

8.7.2 Processus de filtrage pour un ensemble d'échantillons sur une bordure horizontale ou verticale de bloc

Les entrées pour ce processus sont les valeurs d'échantillon d'entrée p_i et q_i avec i dans la gamme de 0..3 d'un ensemble unique d'échantillons sur une bordure à filtrer, chromaEdgeFlag, verticalEdgeFlag, et fieldModeFilteringFlag.

Les résultats de ce processus sont les valeurs d'échantillon résultantes filtrées p'_i et q'_i avec i dans la gamme de 0..2.

La variable dépendante du contenu de force du filtrage de frontière bS est déduite comme suit.

- si chromaEdgeFlag est égal à 0, le processus de déduction pour la force du filtrage de frontière dépendant du contenu spécifiée au § 8.7.2.1 est invoqué avec p_0 , q_0 , et verticalEdgeFlag en entrée, et le résultat est alloué à bS;
- autrement (chromaEdgeFlag est égal à 1), on applique ce qui suit.
 - Si fieldModeFilteringFlag est égal à 0, la variable bS utilisée pour le filtrage d'un ensemble d'échantillons d'une bordure chroma horizontale ou verticale doit être mise égale à la valeur de bS pour le filtrage d'un ensemble d'échantillons d'une bordure luma respectivement horizontale ou verticale qui contient l'échantillon luma à la localisation $(2 * x, 2 * y)$ au sein de la matrice luma de la trame, où (x, y) est la localisation de l'échantillon chroma q_0 au sein de la matrice chroma pour cette trame;
 - Autrement (fieldModeFilteringFlag est égal à 1), la variable bS utilisée pour le filtrage d'un ensemble d'échantillons d'une bordure chroma horizontale ou verticale doit être mise égale à la valeur de bS pour le filtrage d'un ensemble d'échantillons d'une bordure luma respectivement horizontale ou verticale qui contient l'échantillon luma à la localisation $(2 * x, 2 * y)$ au sein de la matrice luma du même champ, où (x, y) est la localisation de l'échantillon chroma q_0 au sein de la matrice chroma pour ce champ.

Le processus spécifié au § 8.7.2.2 est invoqué avec p_0 , q_0 , p_1 , q_1 , chromaEdgeFlag, et bS en entrée, et le résultat est alloué à filterSamplesFlag, indexA, α , et β .

En fonction de la variable filterSamplesFlag, on applique ce qui suit.

- si filterSamplesFlag est égal à 1, on applique ce qui suit.
 - si bS est inférieur à 4, le processus spécifié au § 8.7.2.3 est invoqué avec p_i et q_i ($i = 0..3$), chromaEdgeFlag, bS, β , et indexA donnés en entrée, et le résultat est alloué à p'_i et q'_i ($i = 0..2$);
 - autrement (bS est égal à 4), le processus spécifié au § 8.7.2.4 est invoqué avec p_i et q_i ($i = 0..3$), chromaEdgeFlag, α , et β donnés en entrée, et le résultat est alloué à p'_i et q'_i ($i = 0..2$).
- Autrement (filterSamplesFlag est égal à 0), les échantillons résultants filtrés p'_i et q'_i ($i = 0..2$) sont remplacés par les échantillons d'entrée correspondants p_i et q_i :

pour $i = 0..2$, $p'_i = p_i$ (8-328)

pour $i = 0..2$, $q'_i = q_i$ (8-329)

8.7.2.1 Processus de déduction pour force de filtrage de frontière luma dépendant du contenu

Les entrées pour ce processus sont les valeurs d'échantillon d'entrée p_0 et q_0 d'un ensemble unique d'échantillons sur une bordure à filtrer et `verticalEdgeFlag`.

Le résultat de ce processus est la variable `bS`.

Soit la variable `mixedModeEdgeFlag` à calculer comme suit:

- si `MbaffFrameFlag` est égal à 1 et que les échantillons p_0 et q_0 sont dans différentes paires de macroblocs, dont l'une est une paire de macroblocs de champ et l'autre une paire de macroblocs de trame, `mixedModeEdgeFlag` est mise égale à 1;
- autrement, `mixedModeEdgeFlag` est mise égale à 0.

La variable `bS` est déduite comme suit.

- Si la bordure de bloc est aussi une bordure de macrobloc et si une des conditions suivantes est vraie, une valeur de `bS` égale à 4 doit être le résultat:
 - les échantillons p_0 et q_0 sont tous deux dans des macroblocs de trame et l'un ou l'autre des échantillons p_0 ou q_0 , ou les deux, sont dans un macrobloc codé en utilisant un mode d'intraprédiction de macrobloc;
 - les échantillons p_0 et q_0 sont tous deux dans des macroblocs de trame et l'un ou l'autre des échantillons p_0 ou q_0 , ou les deux, sont dans un macrobloc qui est dans une tranche avec `slice_type` égal à SP ou SI;
 - `MbaffFrameFlag` est égal à 1 ou `field_pic_flag` est égal à 1, et `verticalEdgeFlag` est égal à 1, et l'un ou l'autre des échantillons p_0 ou q_0 , ou les deux, sont dans un macrobloc codé en utilisant un mode d'intraprédiction de macrobloc.
 - `MbaffFrameFlag` est égal à 1 ou `field_pic_flag` est égal à 1, et `verticalEdgeFlag` est égal à 1, et l'un ou l'autre des échantillons p_0 ou q_0 , ou les deux, sont dans un macrobloc qui est dans une tranche avec `slice_type` égal à SP ou SI;
- Autrement, si une des conditions suivantes est Vraie, une valeur de `bS` égale à 3 doit être le résultat:
 - `mixedModeEdgeFlag` est égal à 0 et l'un ou l'autre des échantillons p_0 ou q_0 , ou les deux, sont dans un macrobloc codé en utilisant un mode d'intraprédiction de macrobloc;
 - `mixedModeEdgeFlag` est égal à 0 et l'un ou l'autre des échantillons p_0 ou q_0 , ou les deux, sont dans un macrobloc qui est dans une tranche avec `slice_type` égal à SP ou SI.
 - `mixedModeEdgeFlag` est égal à 1, `verticalEdgeFlag` est égal à 0, et l'un ou l'autre des échantillons p_0 ou q_0 ou les deux, sont dans un macrobloc codé en utilisant un mode d'intraprédiction de macrobloc.
 - `mixedModeEdgeFlag` est égal à 1, `verticalEdgeFlag` est égal à 0, et l'un ou l'autre des échantillons p_0 ou q_0 , ou les deux, sont dans un macrobloc qui est dans une tranche avec `slice_type` égal à SP ou SI.
- Autrement, si la condition suivante est Vraie, une valeur de `bS` égale à 2 doit être le résultat:
 - le bloc luma 4x4 contenant l'échantillon p_0 ou le bloc luma 4x4 contenant l'échantillon q_0 contient des niveaux de coefficient de transformée différents de zéro.
- Autrement, si une des conditions suivantes est Vraie, une valeur de `bS` égale à 1 doit être le résultat:
 - `mixedModeEdgeFlag` est égal à 1;

- mixedModeEdgeFlag est égal à 0 et on utilise pour la prédiction de la subdivision de macrobloc/sous-macrobloc contenant l'échantillon p_0 des images de référence différentes ou des numéros de vecteurs cinétiques différents de ceux de la prédiction de la subdivision de macrobloc/sous-macrobloc contenant l'échantillon q_0 ,

NOTE – La question de savoir si les images de référence utilisées pour les deux subdivisions de macrobloc/sous-macrobloc sont identiques ou différentes est uniquement fonction des images qui sont mentionnées, indépendamment de savoir, d'une part, si une prédiction est établie au moyen d'un indice de la liste 0 ou d'un indice de la liste 1 et, d'autre part, si la position d'indice dans une liste d'images de référence est différente ou non.

- mixedModeEdgeFlag est égal à 0 et un vecteur cinétique est utilisé pour prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon p_0 et un vecteur cinétique est utilisé pour prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon q_0 et la différence absolue entre la composante horizontale ou verticale du vecteur cinétique utilisé est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma;
- mixedModeEdgeFlag est égal à 0 et deux vecteurs cinétiques et deux images de référence différentes sont utilisés pour prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon p_0 et deux vecteurs cinétiques pour les deux mêmes images de référence sont utilisés pour prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon q_0 et la différence absolue entre la composante horizontale ou verticale d'un vecteur cinétique utilisé dans la prédiction des deux macroblocs/subdivision de sous-macroblocks pour la même image de référence est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma;
- mixedModeEdgeFlag est égal à 0 et deux vecteurs cinétiques pour la même image de référence sont utilisés pour prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon p_0 et deux vecteurs cinétiques pour la même image de référence qu'utilisée pour prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon p_0 sont utilisés pour prédire le macrobloc/la subdivision de sous-macrobloc contenant l'échantillon q_0 et les deux conditions suivantes sont Vraies:
 - la différence absolue entre la composante horizontale ou verticale des vecteurs cinétiques de liste 0 utilisés dans la prédiction des deux macroblocs/subdivisions de sous-macroblocks est supérieure ou égale à 4 en quarts d'échantillon de trame luma ou la différence absolue entre la composante horizontale ou verticale des vecteurs cinétiques de liste 1 utilisés dans la prédiction des deux macroblocs/subdivisions de sous-macroblocks est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma;
 - la différence absolue entre la composante horizontale ou verticale du vecteur cinétique de liste 0 utilisé dans la prédiction du macrobloc/de la subdivision de sous-macrobloc contenant l'échantillon p_0 et le vecteur cinétique de liste 1 utilisé dans la prédiction du macrobloc/de la subdivision de sous-macrobloc contenant l'échantillon q_0 est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma ou la différence absolue entre la composante horizontale ou verticale du vecteur cinétique de liste 1 utilisé dans la prédiction du macrobloc/de la subdivision de sous-macrobloc contenant l'échantillon p_0 et le vecteur cinétique de liste 0 utilisé dans la prédiction du macrobloc/de la subdivision de sous-macrobloc contenant l'échantillon q_0 est supérieure ou égale à 4 en unités de quart d'échantillon de trame luma.

NOTE – Une différence verticale de 4 en unités de quart d'échantillon de trame luma est une différence de 2 en unités de quart d'échantillon de champ luma.

- Autrement, une valeur de bS égale à 0 doit être le résultat.

8.7.2.2 Processus de déduction pour les seuils de chaque bordure de bloc

Les entrées pour ce processus sont les valeurs d'échantillon d'entrée p_0 , q_0 , p_1 et q_1 d'un ensemble d'échantillons sur une bordure à filtrer, chromaEdgeFlag , et bS , pour l'ensemble d'échantillons d'entrée, comme spécifié au § 8.7.2.

Les résultats de ce processus sont la variable filterSamplesFlag , qui indique si les échantillons d'entrée sont filtrés, la valeur de indexA , et les valeurs des variables de seuil α et β .

Soit qP_p et qP_q les variables spécifiant les valeurs des paramètres de quantification pour les macroblocs contenant respectivement les échantillons p_0 et q_0 . Les variables qP_z (avec z remplacé par p ou q) sont déduites comme suit:

- Si chromaEdgeFlag est égal à 0, on applique ce qui suit.
 - Si le macrobloc contenant l'échantillon z_0 est un macrobloc I_PCM, qP_z est mis à 0;
 - Autrement (le macrobloc contenant l'échantillon z_0 n'est pas un macrobloc I_PCM), qP_z est mis à la valeur de QP_Y du macrobloc contenant l'échantillon z_0 .
- Autrement (chromaEdgeFlag est égal à 1), on applique ce qui suit.
 - Si le macrobloc contenant l'échantillon z_0 est un macrobloc I_PCM, qP_z est mis à la valeur de QP_C qui correspond à une valeur de 0 pour QP_Y comme spécifié au § 8.5.5;
 - Autrement (le macrobloc contenant l'échantillon z_0 n'est pas un macrobloc I_PCM), qP_z est mis à la valeur de QP_C qui correspond à la valeur QP_Y du macrobloc contenant l'échantillon z_0 comme spécifié au § 8.5.5.

Soit qP_{av} une variable spécifiant un paramètre de quantification moyen. Elle est déduite comme suit.

$$qP_{av} = (qP_p + qP_q + 1) \gg 1 \quad (8-330)$$

NOTE – Dans les tranches SP et SI, qP_{av} est calculée de la même façon que dans les autres types de tranche. QS_Y de l'équation 7-17 n'est pas utilisé dans le filtre de démontage de blocs.

Soit indexA une variable qui est utilisée pour accéder au tableau α (Tableau 8-14) aussi bien qu'au tableau t_{C0} (Tableau 8-15), qui est utilisé pour filtrer les bordures avec bS inférieur à 4 comme spécifié au § 8.7.2.3, et soit indexB une variable qui est utilisée pour accéder au tableau β (Tableau 8-14). Les variables indexA et indexB sont déduites comme suit, lorsque les valeurs de FilterOffsetA et FilterOffsetB sont les valeurs de ces variables spécifiées au § 7.4.3 pour la tranche qui contient le macrobloc contenant l'échantillon q_0 .

$$\text{indexA} = \text{Clip3}(0, 51, qP_{av} + \text{FilterOffsetA}) \quad (8-331)$$

$$\text{indexB} = \text{Clip3}(0, 51, qP_{av} + \text{FilterOffsetB}) \quad (8-332)$$

Les variables de seuil α et β sont spécifiées au Tableau 8-14 en fonction des valeurs de indexA et indexB .

La variable filterSamplesFlag est calculée par

$$\text{filterSamplesFlag} = (bS \neq 0 \ \&\& \ \text{Abs}(p_0 - q_0) < \alpha \ \&\& \ \text{Abs}(p_1 - p_0) < \beta \ \&\& \ \text{Abs}(q_1 - q_0) < \beta) \quad (8-333)$$

Tableau 8-14 – Calcul de indexA et indexB à partir des variables de seuil dépendantes du décalage α et β

		indexA (pour α) ou indexB (pour 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

Tableau 8-14 (fin) – Calcul de indexA et indexB à partir des variables de seuil dépendantes du décalage α et β

		indexA (pour α) ou indexB (pour 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 Processus de filtrage pour bordures avec bS inférieur à 4

Les entrées pour ce processus sont les valeurs d'échantillon d'entrée p_i et q_i ($i = 0..2$) d'un ensemble d'échantillons sur une bordure à filtrer, chromaEdgeFlag, bS, β , et indexA, pour l'ensemble d'échantillons d'entrée, comme spécifié au 8.7.2.

Les résultats de ce processus sont les valeurs d'échantillon filtré résultant p'_i et q'_i ($i = 0..2$) pour l'ensemble de valeurs d'échantillon d'entrée.

Les échantillons filtrés résultants p'_0 et q'_0 sont calculés par:

$$\Delta = \text{Clip3}(-t_c, t_c, (((q_0 - p_0) \ll 2) + (p_1 - q_1) + 4) \gg 3)) \quad (8-334)$$

$$p'_0 = \text{Clip1}(p_0 + \Delta) \quad (8-335)$$

$$q'_0 = \text{Clip1}(q_0 - \Delta) \quad (8-336)$$

où le seuil t_c est déterminé comme suit.

– Si chromaEdgeFlag est égal à 0,

$$t_c = t_{c0} + ((a_p < \beta) ? 1 : 0) + ((a_q < \beta) ? 1 : 0) \quad (8-337)$$

– Autrement (chromaEdgeFlag est égal à 1),

$$t_c = t_{c0} + 1 \quad (8-338)$$

Le seuil t_{c0} est spécifié au Tableau 8-15 en fonction des valeurs de indexA et bS.

Soit a_p et a_q les deux variables de seuil spécifiées par

$$a_p = \text{Abs}(p_2 - p_0) \quad (8-339)$$

$$a_q = \text{Abs}(q_2 - q_0) \quad (8-340)$$

L'échantillon filtré résultant p'_1 est calculé comme suit.

– Si chromaEdgeFlag est égal à 0 et que a_p est inférieur à β ,

$$p'_1 = p_1 + \text{Clip3}(-t_{c0}, t_{c0}, (p_2 + ((p_0 + q_0 + 1) \gg 1) - (p_1 \ll 1)) \gg 1) \quad (8-341)$$

– Autrement (chromaEdgeFlag est égal à 1 ou a_p est supérieur ou égal à β),

$$p'_1 = p_1 \quad (8-342)$$

L'échantillon filtré résultant q'_1 est calculé comme suit.

– Si chromaEdgeFlag est égal à 0 et que a_q est inférieur à β ,

$$q'_1 = q_1 + \text{Clip3}(-t_{c0}, t_{c0}, (q_2 + ((p_0 + q_0 + 1) \gg 1) - (q_1 \ll 1)) \gg 1) \quad (8-343)$$

– Autrement (chromaEdgeFlag est égal à 1 ou a_q est supérieur ou égal à β),

$$q'_1 = q_1 \quad (8-344)$$

Les échantillons filtrés résultants p'_2 et q'_2 sont toujours mis égaux aux échantillons d'entrée p_2 et q_2 :

$$p'_2 = p_2 \quad (8-345)$$

$$q'_2 = q_2 \quad (8-346)$$

Tableau 8-15 – Valeur de la variable d'écrêtage de filtre t_{C0} en fonction de indexA et bS

	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	1	1	1	1	1	1	1	1	1	1

Tableau 8-15 (fin) – Valeur de la variable d'écrêtage de filtre t_{C0} en fonction de indexA et bS

	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 Processus de filtrage pour bordures avec bS égal à 4

Les entrées pour ce processus sont les valeurs d'échantillon d'entrée p_i et q_i ($i = 0..3$) d'un ensemble d'échantillons sur une bordure à filtrer, la variable `chromaEdgeFlag`, et les valeurs des variables de seuil α et β pour l'ensemble d'échantillons, comme spécifié au § 8.7.2.

Les résultats de ce processus sont les valeurs d'échantillon filtré résultant p'_i et q'_i ($i = 0..2$) pour l'ensemble de valeurs d'échantillon d'entrée.

Soit a_p et a_q deux variables de seuil comme spécifié respectivement par les équations 8-339 et 8-340 au § 8.7.2.3.

Les échantillons filtrés résultants p'_i ($i = 0..2$) sont calculés comme suit.

- Si `chromaEdgeFlag` est égal à 0 et que la condition suivante est satisfaite,

$$a_p < \beta \ \&\& \ \text{Abs}(p_0 - q_0) < ((\alpha \gg 2) + 2) \quad (8-347)$$

les variables p'_0 , p'_1 , et p'_2 sont alors calculées par

$$p'_0 = (p_2 + 2*p_1 + 2*p_0 + 2*q_0 + q_1 + 4) \gg 3 \quad (8-348)$$

$$p'_1 = (p_2 + p_1 + p_0 + q_0 + 2) \gg 2 \quad (8-349)$$

$$p'_2 = (2*p_3 + 3*p_2 + p_1 + p_0 + q_0 + 4) \gg 3 \quad (8-350)$$

- Autrement (`chromaEdgeFlag` est égal à 1 ou la condition de l'équation 8-347 n'est pas satisfaite), les variables p'_0 , p'_1 , et p'_2 sont calculées par:

$$p'_0 = (2*p_1 + p_0 + q_1 + 2) \gg 2 \quad (8-351)$$

$$p'_1 = p_1 \quad (8-352)$$

$$p'_2 = p_2 \quad (8-353)$$

Les échantillons filtrés résultants q'_i ($i = 0..2$) sont déduits comme suit.

- Si `chromaEdgeFlag` est égal à 0 et que la condition suivante est satisfaite,

$$a_q < \beta \ \&\& \ \text{Abs}(p_0 - q_0) < ((\alpha \gg 2) + 2) \quad (8-354)$$

les variables q'_0 , q'_1 , et q'_2 sont alors calculées par

$$q'_0 = (p_1 + 2*p_0 + 2*q_0 + 2*q_1 + q_2 + 4) \gg 3 \quad (8-355)$$

$$q'_1 = (p_0 + q_0 + q_1 + q_2 + 2) \gg 2 \quad (8-356)$$

$$q'_2 = (2*q_3 + 3*q_2 + q_1 + q_0 + p_0 + 4) \gg 3 \quad (8-357)$$

- Autrement (`chromaEdgeFlag` est égal à 1 ou la condition de l'équation 8-354 n'est pas satisfaite), les variables q'_0 , q'_1 , et q'_2 sont calculées par:

$$q'_0 = (2 * q_1 + q_0 + p_1 + 2) \gg 2 \quad (8-358)$$

$$q'_1 = q_1 \quad (8-359)$$

$$q'_2 = q_2 \quad (8-360)$$

9 Processus d'analyse grammaticale

Les entrées pour ce processus sont des bits provenant de la charge utile RBSP.

Les résultats de ce processus sont des valeurs d'éléments syntaxiques.

Ce processus est invoqué lorsque le descripteur d'un élément syntaxique dans les tableaux syntaxiques du § 7.3 est égal à ue(v), me(v), se(v), te(v) (voir le § 9.1), ce(v) (voir le § 9.2), ou ae(v) (voir le § 9.3).

9.1 Processus d'analyse grammaticale pour les codes Exp-Golomb

Ce processus est invoqué lorsque le descripteur d'un élément syntaxique dans les tableaux syntaxiques du § 7.3 est égal à ue(v), me(v), se(v), ou te(v). Pour les éléments syntaxiques des § 7.3.4 et 7.3.5, ce processus n'est invoqué que lorsque entropy_coding_mode_flag est égal à 0.

Les entrées pour ce processus sont des bits provenant de la charge utile RBSP.

Les résultats de ce processus sont des valeurs d'éléments syntaxiques.

Les éléments syntaxiques codés ue(v), me(v), ou se(v) sont des Exp-Golomb codés. Les éléments syntaxiques codés te(v) sont des Exp-Golomb codés tronqués. Le processus d'analyse grammaticale pour ces éléments syntaxiques commence par la lecture des bits débutants à la localisation en cours dans le flux binaire jusqu'à et y compris le premier bit différent de zéro, et en comptant le nombre de bits directeurs qui sont égaux à 0. Ce processus doit être équivalent à ce qui suit:

```

leadingZeroBits = -1
pour(b = 0; !b; leadingZeroBits++)
    b = read_bits(1)

```

La variable codeNum est alors allouée comme suit:

$$\text{codeNum} = 2^{\text{leadingZeroBits}} - 1 + \text{read_bits}(\text{leadingZeroBits})$$

où la valeur résultant de ces read_bits(leadingZeroBits) est interprétée comme une représentation binaire d'un entier algébrique avec le bit de plus fort poids écrit en premier.

Le Tableau 9-1 illustre la structure du code Exp-Golomb en séparant la chaîne binaire en bits de "préfixe" et de "suffixe". Les bits de "préfixe" sont les bits qui sont analysés grammaticalement dans le pseudo-code ci-dessus pour le calcul des leadingZeroBits, et sont indiqués comme 0 ou 1 dans la colonne Forme de chaîne binaire du Tableau 9-1. Les bits de "suffixe" sont les bits qui sont analysés grammaticalement dans le calcul de codeNum et sont indiqués comme x_i dans le Tableau 9-1, avec i dans la gamme de 0 à leadingZeroBits - 1, inclus. Chaque x_i peut prendre les valeurs 0 ou 1.

Tableau 9-1 – Chaînes binaires avec des bits de "préfixe" et de "suffixe" et allocation aux gammes de codeNum (pour information)

Forme de chaîne binaire	Gamme de codeNum
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
...	...

Le Tableau 9-2 illustre explicitement l'allocation des chaînes binaires aux valeurs de codeNum.

Tableau 9-2 – Chaînes binaires Exp-Golomb et codeNum en forme explicite et utilisées comme ue(v) (pour information)

Chaîne binaire	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
...	...

En fonction du descripteur, la valeur d'un élément syntaxique est déduite comme suit.

- Si l'élément syntaxique est codé comme ue(v), la valeur de l'élément syntaxique est égale à codeNum.
- Autrement, si l'élément syntaxique est codé comme se(v), la valeur de l'élément syntaxique est déduite en invoquant le processus de mappage pour les codes Exp-Golomb algébriques comme spécifié au § 9.1.1 avec codeNum comme entrée.
- Autrement, si l'élément syntaxique est codé comme me(v), la valeur de l'élément syntaxique est déduite en invoquant le processus de mappage pour schéma de bloc codé comme spécifié au § 9.1.2 avec codeNum comme entrée.
- Autrement (l'élément syntaxique est codé comme te(v)), la gamme de l'élément syntaxique doit d'abord être déterminée. La gamme de cet élément syntaxique peut être entre 0 et x, avec x supérieur ou égal à 1 et est utilisée dans le calcul de la valeur d'un élément syntaxique comme suit:
 - si x est supérieur à 1, codeNum et la valeur de l'élément syntaxique doit être calculée de la même façon que pour les éléments syntaxiques codés ue(v);

- autrement (x est égal à 1), le processus d'analyse grammaticale pour `codeNum` qui est égal à la valeur de l'élément syntaxique est donnée par un processus équivalent à:

```
b = read_bits( 1 )
codeNum = !b
```

9.1.1 Processus de mappage pour les codes Exp-Golomb algébriques

L'entrée pour ce processus est `codeNum` comme spécifié au § 9.1.

Le résultat de ce processus est une valeur d'un élément syntaxique codé `se(v)`.

L'élément syntaxique est alloué au `codeNum` en ordonnant l'élément syntaxique par sa valeur absolue en ordre croissant et en représentant la valeur positive pour une valeur absolue donnée par le plus bas `codeNum`. Le Tableau 9-3 donne la règle d'allocation.

Tableau 9-3 – Allocation d'élément syntaxique à `codeNum` pour les éléments syntaxiques codés en Exp-Golomb algébrique `se(v)`

<code>codeNum</code>	Valeur d'élément syntaxique
0	0
1	1
2	-1
3	2
4	-2
5	3
6	-3
k	$(-1)^{k+1} \text{Ceil}(k \div 2)$

9.1.2 Processus de mappage pour schéma de bloc codé

L'entrée pour ce processus est `codeNum` comme spécifié au § 9.1.

Le résultat de ce processus est une valeur de l'élément syntaxique `coded_block_pattern` codé `me(v)`.

Le Tableau 9-4 montre l'allocation de `coded_block_pattern` à `codeNum` selon que le mode de prédiction de macrobloc est égal à `Intra_4x4` ou `Inter`.

Tableau 9-4 – Allocation de `codeNum` aux valeurs de `coded_block_pattern` pour les modes de prédiction de macrobloc

<code>codeNum</code>	<code>coded_block_pattern</code>	
	<code>Intra_4x4</code>	<code>Inter</code>
0	47	0
1	31	16
2	15	1
3	0	2
4	23	4
5	27	8
6	29	32

Tableau 9-4 – Allocation de codeNum aux valeurs de coded_block_pattern pour les modes de prédiction de macrobloc

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
44	36	22
45	40	25
46	38	38
47	41	41

9.2 Processus d'analyse grammaticale CAVLC pour les niveaux de coefficient de transformée

Ce processus est invoqué lorsque les éléments syntaxiques d'analyse grammaticale avec descripteur sont égaux à $ce(v)$ du § 7.3.5.3.1 et lorsque $entropy_coding_mode_flag$ est égal à 0.

Les entrées pour ce processus sont des bits provenant des données de tranche, un nombre maximal des niveaux de coefficient de transformée différents de zéro $maxNumCoeff$, l'indice de bloc luma $luma4x4BlkIdx$ ou l'indice de bloc chroma $chroma4x4BlkIdx$ du bloc en cours de niveaux de coefficient de transformée.

Le résultat de ce processus est la liste $coeffLevel$ contenant les niveaux de coefficient de transformée du bloc luma avec l'indice de bloc $luma4x4BlkIdx$ ou du bloc chroma avec l'indice de bloc $chroma4x4BlkIdx$.

Le processus est spécifié selon les étapes ordonnées suivantes:

1. Tous les niveaux de coefficient de transformée, avec les indices de 0 à $maxNumCoeff - 1$, dans la liste $coeffLevel$ sont mis égaux à 0.
2. Le nombre total des niveaux de coefficient de transformée différents de zéro $TotalCoeff(coeff_token)$ et le nombre de niveaux de coefficient de transformée de traîne $TrailingOnes(coeff_token)$ sont calculés par l'analyse grammaticale de $coeff_token$ (voir le § 9.2.1) comme suit.
 - Si le nombre des niveaux de coefficient de transformée différents de zéro $TotalCoeff(coeff_token)$ est égal à 0, la liste $coeffLevel$ contenant des valeurs 0 est retournée et aucune autre étape n'est effectuée;
 - Autrement, les étapes suivantes sont effectuées.
 - a. Les niveaux de coefficient de transformée différents de zéro sont calculés en analysant grammaticalement $trailing_ones_sign_flag$, $level_prefix$, et $level_suffix$ (voir le § 9.2.2).
 - b. Les exécutions de niveaux de coefficient de transformée zéro avant chaque de niveau de coefficient de transformée différent de zéro sont calculées en analysant grammaticalement $total_zeros$ et run_before (voir le § 9.2.3).
 - c. Les informations de niveau et d'occurrence sont combinées dans la liste $coeffLevel$ (voir le § 9.2.4).

9.2.1 Processus d'analyse grammaticale pour le nombre total de niveaux de coefficient de transformée et de traînes

Les entrées pour ce processus sont des bits provenant des données de tranche, un nombre maximal de niveaux de coefficient de transformée différents de zéro $maxNumCoeff$, l'indice de bloc luma $luma4x4BlkIdx$ ou l'indice de bloc chroma $chroma4x4BlkIdx$ du bloc de transformées en cours.

Les résultats de ce processus sont $TotalCoeff(coeff_token)$ et $TrailingOnes(coeff_token)$.

L'élément syntaxique $coeff_token$ est décodé en utilisant un des cinq codages VLC spécifiés dans les cinq colonnes les plus à droite du Tableau 9-5. Chaque codage VLC spécifie à la fois $TotalCoeff(coeff_token)$ et $TrailingOnes(coeff_token)$ pour un mot de code $coeff_token$ donné. Le choix du codage VLC dépend d'une variable nC qui est déduite comme suit.

- si le processus d'analyse grammaticale CAVLC est invoqué pour $ChromaDCLevel$, nC est mise égale à -1 .
- autrement, on applique ce qui suit.
 - lorsque le processus d'analyse grammaticale CAVLC est invoqué pour $Intra16x16DCLevel$, $luma4x4BlkIdx$ est mise égal à 0;

- les variables blkA et blkB sont déduites comme suit.
 - Si le processus d'analyse grammaticale CAVLC est invoqué pour Intra16x16DCLevel, Intra16x16ACLevel, ou LumaLevel, le processus spécifié au § 6.4.7.3 est invoqué avec luma4x4BlkIdx comme entrée, et le résultat est alloué à mbAddrA, mbAddrB, luma4x4BlkIdxA, et luma4x4BlkIdxB. Le bloc luma 4x4 spécifié par mbAddrA\luma4x4BlkIdxA est alloué à blkA, et le bloc luma 4x4 spécifié par mbAddrB\luma4x4BlkIdxB est alloué à blkB.
 - Autrement (le processus d'analyse grammaticale CAVLC est invoqué pour ChromaACLevel), le processus spécifié au § 6.4.7.4 est invoqué avec chroma4x4BlkIdx en entrée, et le résultat est alloué à mbAddrA, mbAddrB, chroma4x4BlkIdxA, et chroma4x4BlkIdxB. Le bloc chroma 4x4 spécifié par mbAddrA\iCbCr\chroma4x4BlkIdxA est alloué à blkA, et le bloc chroma 4x4 spécifié par mbAddrB\iCbCr\chroma4x4BlkIdxB est alloué à blkB.
- Soit nA et nB le nombre des niveaux de coefficient de transformée différents de zéro (donné par TotalCoeff(coeff_token)) respectivement dans le bloc de niveaux de coefficient de transformée blkA localisé à gauche du bloc en cours et dans le bloc de niveaux de coefficient de transformée blkB localisé au-dessus du bloc en cours.
- Avec N remplacé par A et B, dans mbAddrN, blkN, et nN, on applique ce qui suit.
 - Si l'une des conditions suivantes est Vraie, nN est mise égale à 0.
 - mbAddrN n'est pas disponible.
 - Le macrobloc en cours est codé en utilisant un mode d'intraprédiction, constrained_intra_pred_flag est égal à 1, mbAddrN est codé en utilisant l'interprédiction et la subdivision de données de tranche est utilisée (nal_unit_type est dans la gamme de 2 à 4, inclus).
 - Le macrobloc mbAddrN a mb_type égal à P_Skip ou B_Skip.
 - Tous les niveaux de coefficient de transformée AC résiduel du bloc voisin blkN sont égaux à 0 du fait du bit correspondant de CodedBlockPatternLuma ou CodedBlockPatternChroma qui est égal à 0.
 - Autrement, si mbAddrN est un macrobloc I_PCM, nN est mise égale à 16.
 - Autrement, nN est mise égale à la valeur TotalCoeff(coeff_token) du bloc voisin blkN.

NOTE – Les valeurs nA et nB qui sont calculées en utilisant TotalCoeff(coeff_token) n'incluent pas les niveaux de coefficient de transformée DC dans les macroblocs Intra 16x16 ou les niveaux de coefficient de transformée DC dans les blocs chroma, parce que ces niveaux de coefficient de transformée sont décodés séparément. Lorsque le bloc au-dessus ou à gauche appartient à un macrobloc Intra 16x16, ou est un bloc chroma, nA et nB est le nombre de niveaux de coefficient de transformée AC décodés différents de zéro.

NOTE – Lors de l'analyse grammaticale de Intra16x16DCLevel, les valeurs nA et nB sont fondées sur le nombre des niveaux de coefficient de transformée différents de zéro dans les blocs 4x4 adjacents et non sur le nombre de niveaux de coefficient de transformée DC différents de zéro dans les blocs 16x16 adjacents.
- Etant donné les valeurs de nA et nB, la variable nC est déduite comme suit.
 - Si mbAddrA et mbAddrB sont tous deux disponibles, la variable nC est mise égale à $(nA + nB + 1) \ggg 1$.
 - Autrement (mbAddrA n'est pas disponible ou mbAddrB n'est pas disponible), la variable nC est mise égale à nA + nB.

La valeur de TotalCoeff(coeff_token) résultant du décodage de coeff_token doit être dans la gamme de 0 à maxNumCoeff, inclus.

Tableau 9-5 – Mappage de coeff_token avec TotalCoeff(coeff_token) et TrailingOnes(coeff_token)

TrailingOnes (coeff_token)	TotalCoeff (coeff_token)	0 <= nC < 2	2 <= nC < 4	4 <= nC < 8	8 <= nC	nC == -1
0	0	1	11	1111	0000 11	01
0	1	0001 01	0010 11	0011 11	0000 00	0001 11
1	1	01	10	1110	0000 01	1
0	2	0000 0111	0001 11	0010 11	0001 00	0001 00
1	2	0001 00	0011 1	0111 1	0001 01	0001 10
2	2	001	011	1101	0001 10	001
0	3	0000 0011 1	0000 111	0010 00	0010 00	0000 11
1	3	0000 0110	0010 10	0110 0	0010 01	0000 011
2	3	0000 101	0010 01	0111 0	0010 10	0000 010
3	3	0001 1	0101	1100	0010 11	0001 01
0	4	0000 0001 11	0000 0111	0001 111	0011 00	0000 10
1	4	0000 0011 0	0001 10	0101 0	0011 01	0000 0011
2	4	0000 0101	0001 01	0101 1	0011 10	0000 0010
3	4	0000 11	0100	1011	0011 11	0000 000
0	5	0000 0000 111	0000 0100	0001 011	0100 00	–
1	5	0000 0001 10	0000 110	0100 0	0100 01	–
2	5	0000 0010 1	0000 101	0100 1	0100 10	–
3	5	0000 100	0011 0	1010	0100 11	–
0	6	0000 0000 0111 1	0000 0011 1	0001 001	0101 00	–
1	6	0000 0000 110	0000 0110	0011 10	0101 01	–
2	6	0000 0001 01	0000 0101	0011 01	0101 10	–
3	6	0000 0100	0010 00	1001	0101 11	–
0	7	0000 0000 0101 1	0000 0001 111	0001 000	0110 00	–
1	7	0000 0000 0111 0	0000 0011 0	0010 10	0110 01	–
2	7	0000 0000 101	0000 0010 1	0010 01	0110 10	–
3	7	0000 0010 0	0001 00	1000	0110 11	–
0	8	0000 0000 0100 0	0000 0001 011	0000 1111	0111 00	–
1	8	0000 0000 0101 0	0000 0001 110	0001 110	0111 01	–
2	8	0000 0000 0110 1	0000 0001 101	0001 101	0111 10	–
3	8	0000 0001 00	0000 100	0110 1	0111 11	–
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	–
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	–

**Tableau 9-5 – Mappage de coeff_token avec TotalCoeff(coeff_token)
et TrailingOnes(coeff_token)**

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 Processus d'analyse grammaticale pour les informations de niveau

Les entrées pour ce processus sont les bits provenant des données de tranche, le nombre des niveaux de coefficient de transformée différents de zéro TotalCoeff(coeff_token), et le nombre de niveaux de coefficient de transformée de traîne TrailingOnes(coeff_token).

Le résultat de ce processus est une liste avec le niveau de nom qui contient les niveaux de coefficient de transformée.

Un indice *i* est mis égal à 0 au départ. Puis la procédure itérative suivante est appliquée TrailingOnes(coeff_token) fois pour décoder les niveaux de coefficient de transformée de traîne (le cas échéant):

- un élément syntaxique à un bit trailing_ones_sign_flag est décodé et évalué comme suit.
 - Si trailing_ones_sign_flag est égal à 0, la valeur +1 est allouée au niveau [*i*].
 - Autrement (trailing_ones_sign_flag est égal à 1), la valeur –1 est allouée au niveau [*i*].
- l'indice *i* est incrémenté de 1.

A la suite du décodage des niveaux de coefficient de transformée de traîne, on initialise comme suit une variable suffixLength.

- Si TotalCoeff(coeff_token) est supérieur à 10 et que TrailingOnes(coeff_token) est inférieur à 3, suffixLength est mise égale à 1.
- Autrement (TotalCoeff(coeff_token) est inférieur ou égal à 10 ou TrailingOnes(coeff_token) est égal à 3), suffixLength est mise égale à 0.

On applique alors la procédure itérative suivante (TotalCoeff(coeff_token) – TrailingOnes(coeff_token)) fois pour décoder les niveaux restants (le cas échéant):

- l'élément syntaxique `level_prefix` est décodé en utilisant le codage VLC spécifié au Tableau 9-6;
- la variable `levelSuffixSize` est mise égale à la variable `suffixLength` à l'exception des deux cas suivants;
 - lorsque `level_prefix` est égal à 14 et que `suffixLength` est égal à 0, `levelSuffixSize` est mise égale à 4;
 - lorsque `level_prefix` est égal à 15, `levelSuffixSize` est mise égale à 12;
- l'élément syntaxique `level_suffix` est décodé comme suit.
 - Si `levelSuffixSize` est supérieur à 0, l'élément syntaxique `level_suffix` est décodé comme représentation d'un entier algébrique $u(v)$ avec `levelSuffixSize` bits.
 - Autrement (`levelSuffixSize` est égal à 0), l'élément syntaxique `level_suffix` doit être supposé égal à 0.
- Une variable `levelCode` est mise égale à $(\text{level_prefix} \ll \text{suffixLength}) + \text{level_suffix}$.
- Lorsque `level_prefix` est égal à 15 et que `suffixLength` est égal à 0, `levelCode` est incrémentée de 15.
- Lorsque l'indice i est égal à `TrailingOnes(coeff_token)` et que `TrailingOnes(coeff_token)` est plus petit que 3, `levelCode` est incrémentée de 2.
- La variable `level[i]` est déduite comme suit.
 - Si `levelCode` est un nombre pair, la valeur $(\text{levelCode} + 2) \gg 1$ est allouée au niveau $[i]$.
 - Autrement, la valeur $(-\text{levelCode} - 1) \gg 1$ est allouée au niveau $[i]$.
- Lorsque `suffixLength` est égal à 0, `suffixLength` est mise égale à 1.
- Lorsque la valeur absolue de `niveau [i]` est supérieure à $(3 \ll (\text{suffixLength} - 1))$ et que `suffixLength` est inférieure à 6, `suffixLength` est incrémentée de 1.
- L'indice i est incrémenté de 1.

Tableau 9-6 – Tableau des mots de code pour level_prefix

level_prefix	Chaîne binaire
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 Processus d'analyse grammaticale pour information d'exécution

Les entrées pour ce processus sont des bits provenant de données de tranche, le nombre de niveaux de coefficient de transformée différents de zéro $TotalCoeff(coeff_token)$, et le nombre maximal de niveaux de coefficient de transformée différents de zéro $maxNumCoeff$.

Le résultat de ce processus est une liste des exécutions des niveaux de coefficient de transformée zéro précédant des niveaux de coefficient de transformée différents de zéro appelés exécutions (*run*).

Au départ, un indice i est mis égal à 0.

La variable $zerosLeft$ est déduite comme suit.

- Si le nombre de niveaux de coefficient de transformée différents de zéro $TotalCoeff(coeff_token)$ est égal au nombre maximal de niveaux de coefficient de transformée différents de zéro $maxNumCoeff$, une variable $zerosLeft$ est mise égale à 0.
- Autrement (le nombre de niveaux de coefficient de transformée différents de zéro $TotalCoeff(coeff_token)$ est inférieur au nombre maximal de niveaux de coefficient de transformée différents de zéro $maxNumCoeff$), $total_zeros$ est décodé et $zerosLeft$ est mis égal à sa valeur.

Le code VLC utilisé pour décodé $total_zeros$ est déduit comme suit:

- si $maxNumCoeff$ est égal à 4, un des codes VLC spécifié au Tableau 9-9 est utilisé;
- autrement ($maxNumCoeff$ n'est pas égal à 4), les codes VLC tirés du Tableau 9-7 et du Tableau 9-8 sont utilisés.

On applique alors la procédure itérative suivante ($TotalCoeff(coeff_token) - 1$) fois:

- La variable $run[i]$ est déduite comme suit.
 - Si $zerosLeft$ est supérieur à zero, une valeur run_before est décodée sur la base du Tableau 9-10 et $zerosLeft$. $run[i]$ est mis égal à run_before .

- Autrement (zerosLeft est égal à 0), run[i] est mis égal à 0.
- La valeur de run[i] est soustraite de zerosLeft et le résultat est alloué à zerosLeft. Le résultat de la soustraction doit être supérieur ou égal à 0.
- L'indice i est incrémenté de 1.

Finalement, la valeur de zerosLeft est allouée à run[i].

Tableau 9-7 – Tableaux de total_zeros pour blocs 4x4 avec TotalCoeff(coeff_token) de 1 à 7

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						

Tableau 9-8 – Tableaux de total_zeros pour blocs 4x4 avec TotalCoeff(coeff_token) de 8 à 15

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							

Tableau 9-9 – Tableaux de total_zeros pour blocs chroma DC 2x2

total_zeros	TotalCoeff(coeff_token)		
	1	2	3
0	1	1	1
1	01	01	0
2	001	00	
3	000		

Tableau 9-10 – Tableaux pour run_before

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 Combinaison des informations de niveau et d'exécution

Les entrées pour ce processus sont une liste de niveaux de coefficient de transformée appelés niveau (*level*), une liste d'exécutions appelées exécution (*run*) et le nombre de niveaux de coefficient de transformée différents de zéro TotalCoeff(coeff_token).

Le résultat de ce processus est une liste coeffLevel de niveaux de coefficient de transformée.

Une variable coeffNum est mise égale à -1 et un indice i est mis égal à (TotalCoeff(coeff_token) - 1). On applique la procédure itérative suivante TotalCoeff(coeff_token) fois:

- coeffNum est incrémenté de run[i] + 1;
- coeffLevel[coeffNum] est mis égal à level[i];
- l'indice i est décrémenté de 1.

9.3 Processus d'analyse grammaticale CABAC pour données de tranche

Ce processus est invoqué lors de l'analyse grammaticale d'éléments syntaxiques avec le descripteur ae(v) des § 7.3.4 et 7.3.5 lorsque entropy_coding_mode_flag est égal à 1.

Les entrées pour ce processus sont une demande de valeur d'élément syntaxique et les valeurs des éléments syntaxiques précédemment analysés grammaticalement.

Le résultat de ce processus est la valeur de l'élément syntaxique.

Lors du démarrage de l'analyse grammaticale des données de tranche d'une tranche du § 7.3.4, le processus d'initialisation du processus d'analyse grammaticale CABAC est invoqué comme spécifié au § 9.3.1.

L'analyse grammaticale des éléments syntaxiques s'effectue comme suit:

pour chaque valeur demandée d'un élément syntaxique est calculée une binairisation, comme décrit au § 9.3.2.

La binairisation pour l'élément syntaxique et la séquence de segments (*bins*) analysés grammaticalement détermine le flux du processus de décodage, comme décrit au § 9.3.3.

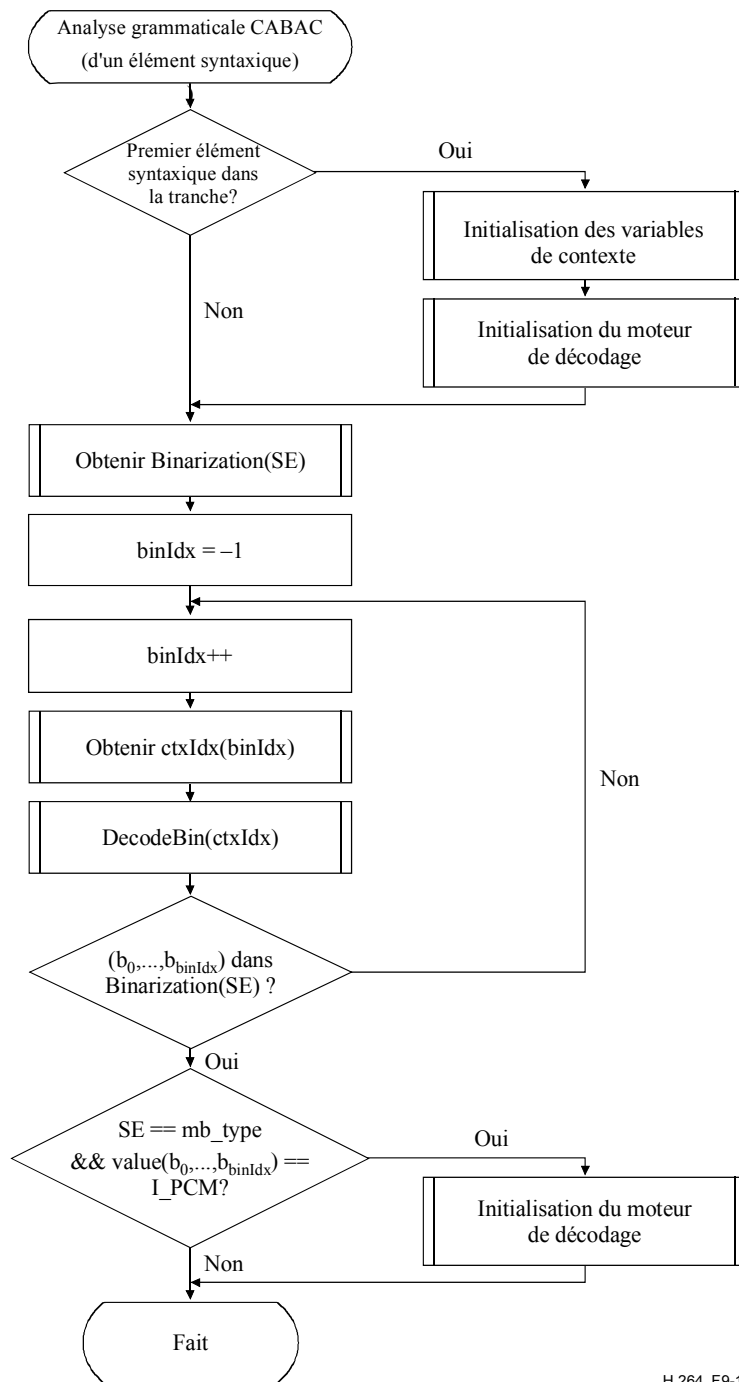
Pour chaque segment de la binairisation de l'élément syntaxique, qui est indexé par la variable *binIdx*, un indice de contexte *ctxIdx* est calculé comme spécifié au § 9.3.3.1.

Pour chaque *ctxIdx* le processus de décodage arithmétique est invoqué comme spécifié au § 9.3.3.2.

La séquence ($b_0 \dots b_{\text{binIdx}}$) résultante de bins analysés grammaticalement est comparée à l'ensemble de la chaîne de segments donnée par le processus de binairisation après le décodage de chaque segment. Lorsque la séquence correspond à une chaîne de segments dans l'ensemble donné, la valeur correspondante doit être allouée à l'élément syntaxique.

Dans le cas où la demande de valeur d'un élément syntaxique est effectuée pour l'élément syntaxique *mb_type* et où la valeur décodée de *mb_type* est *I_PCM*, le moteur de décodage doit être initialisé après le décodage du *pcm_alignment_zero_bit* et de toutes les données de *pcm_byte*, comme spécifié au § 9.3.1.2.

L'ensemble du processus d'analyse grammaticale CABAC est illustré dans l'organigramme de la Figure 9-1 avec l'abréviation SE pour élément syntaxique (SE, *syntax element*).



H.264_F9-1

Figure 9-1 – Illustration du processus d'analyse grammaticale CABAC pour un élément syntaxique SE (pour information)

9.3.1 Processus d'initialisation

Les résultats de ce processus sont des variables internes CABAC initialisées.

Les processus des § 9.3.1.1 et 9.3.1.2 sont invoqués lors du début de l'analyse grammaticale des données de tranche d'une tranche du § 7.3.4.

Le processus du § 9.3.1.2 est aussi invoqué après le décodage du pcm_alignment_zero_bit et de toutes les données de pcm_byte pour un macrobloc de type I_PCM.

9.3.1.1 Processus d'initialisation pour variables de contexte

Les résultats de ce processus sont les variables de contexte CABAC initialisées indexées par ctxIdx.

Du Tableau 9-12 au Tableau 9-23 figurent les valeurs des variables n et m utilisées dans l'initialisation des variables de contexte qui sont allouées à tous les éléments syntaxiques des § 7.3.4 et 7.3.5 excepté pour le fanion de fin de tranche.

Pour chaque variable de contexte, on initialise les deux variables $pStateIdx$ et $valMPS$.

NOTE – La variable $pStateIdx$ correspond à un indice de probabilité d'état et la variable $valMPS$ correspond à la valeur du symbole le plus probable, comme décrit plus loin au § 9.3.3.2.

Les deux valeurs allouées à $pStateIdx$ et $valMPS$ pour l'initialisation sont dérivées de $SliceQP_Y$, qui est calculé dans l'équation 7-16. Etant données les deux entrées de tableau (m , n):

1. $preCtxState = Clip3(1, 126, ((m * SliceQP_Y) \gg 4) + n)$
2. $si(preCtxState \leq 63) \{$
 - $pStateIdx = 63 - preCtxState$
 - $valMPS = 0$ $\} ou \{$
 - $pStateIdx = preCtxState - 64$
 - $valMPS = 1$ $\}$

Dans le Tableau 9-11 figure la liste des $ctxIdx$ pour lesquels l'initialisation est nécessaire pour chacun des types de tranche. Sont aussi listés les numéros de tableau qui incluent les valeurs de m et n nécessaires pour l'initialisation. Pour les types de tranche P, SP et B, l'initialisation dépend aussi de la valeur de l'élément syntaxique $cabac_init_idc$. Noter que les noms d'élément syntaxique n'affectent pas le processus d'initialisation.

Tableau 9-11 – Association des ctxIdx et des éléments syntaxiques pour chaque type de tranche dans le processus d'initialisation

	Élément syntaxique	Tableau	Type de tranche			
			SI	I	P, SP	B
slice_data()	mb_skip_flag	Tableau 9-13 Tableau 9-14			11-13	24-26
	mb_field_decoding_flag	Tableau 9-18	70-72	70-72	70-72	70-72
macroblock_layer()	mb_type	Tableau 9-12, Tableau 9-13, Tableau 9-14.	0-10	3-10	14-20	27-35
	coded_block_pattern (luma)	Tableau 9-18	73-76	73-76	73-76	73-76
	coded_block_pattern (chroma)	Tableau 9-18	77-84	77-84	77-84	77-84
	mb_qp_delta	Tableau 9-17	60-63	60-63	60-63	60-63
mb_pred()	prev_intra4x4_pred_mode_flag	Tableau 9-17	68	68	68	68
	rem_intra4x4_pred_mode	Tableau 9-17	69	69	69	69
	intra_chroma_pred_mode	Tableau 9-17	64-67	64-67	64-67	64-67
mb_pred() and sub_mb_pred()	ref_idx_l0	Tableau 9-16			54-59	54-59
	ref_idx_l1	Tableau 9-16				54-59
	mvd_l0[][0]	Tableau 9-15			40-46	40-46
	mvd_l1[][0]	Tableau 9-15				40-46
	mvd_l0[][1]	Tableau 9-15			47-53	47-53
	mvd_l1[][1]	Tableau 9-15				47-53
sub_mb_pred()	sub_mb_type	Tableau 9-13 Tableau 9-14			21-23	36-39
residual_block_cabac()	coded_block_flag	Tableau 9-18	85-104	85-104	85-104	85-104
	significant_coeff_flag[]	Tableau 9-19, Tableau 9-22.	105-165, 277-337	105-165, 277-337	105-165, 277-337	105-165, 277-337
	last_significant_coeff_flag[]	Tableau 9-20, Tableau 9-23.	166-226, 338-398	166-226, 338-398	166-226, 338-398	166-226, 338-398
	coeff_abs_level_minus1[]	Tableau 9-21	227-275	227-275	227-275	227-275

NOTE – ctxIdx égal à 276 est associé au fanion end_of_slice_flag et au segment de mb_type, qui spécifie le type de macrobloc I_PCM. Le processus de décodage spécifié au § 9.3.3.2.4 s'applique à ctxIdx égal à 276. Ce processus de décodage, peut cependant être aussi implémenté en utilisant le processus de décodage spécifié au § 9.3.3.2.1. Dans ce cas, les valeurs initiales associées à ctxIdx égal à 276 sont spécifiées comme pStateIdx = 63 et valMPS = 0, où pStateIdx = 63 représente un état de probabilité de non-adaptation.

Tableau 9-12 – Valeurs des variables m et n pour ctxIdx de 0 à 10

Variables d'initialisation	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

Tableau 9-13 – Valeurs des variables m et n pour ctxIdx de 11 à 23

Valeur de cabac_init_idc	Variables d'initialisation	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

Tableau 9-14 – Valeurs des variables m et n pour ctxIdx de 24 à 39

Valeur de cabac_init_idc	Variables d'initialisation	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

Tableau 9-15 – Valeurs des variables m et n pour ctxIdx de 40 à 53

Valeur de cabac_init_idc	Variables d'initialisation	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

Tableau 9-16 – Valeurs des variables m et n pour ctxIdx de 54 à 59

Valeur de cabac_init_idc	Variables d'initialisation	ctxIdx					
		54	55	56	57	58	59
0	m	-7	-5	-4	-5	-7	1
	n	67	74	74	80	72	58
1	m	-1	-1	1	-2	-5	0
	n	66	77	70	86	72	61
2	m	3	-4	-2	-12	-7	1
	n	55	79	75	97	50	60

Tableau 9-17 – Valeurs des variables m et n pour ctxIdx de 60 à 69

Variables d'initialisation	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

Tableau 9-18 – Valeurs des variables m et n pour ctxIdx de 70 à 104

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			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									

Tableau 9-19 – Valeurs des variables m et n pour ctxIdx de 105 à 165

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	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									

Tableau 9-20 – Valeurs des variables m et n pour ctxIdx de 166 à 226

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	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									

Tableau 9-21 – Valeurs des variables m et n pour ctxIdx de 227 à 275

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeurs de cabac_init_idc					
			0		1		2					0		1		2	
	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									

Tableau 9-22 – Valeurs des variables m et n pour ctxIdx de 277 à 337

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	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									

Tableau 9-23 – Valeurs des variables m et n pour ctxIdx de 338 à 398

ctxIdx	Tranches I et SI		Valeur de cabac_init_idc						ctxIdx	Tranches I et SI		Valeur de cabac_init_idc					
			0		1		2					0		1		2	
	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									

9.3.1.2 Processus d'initialisation pour le moteur de décodage arithmétique

Ce processus est invoqué avant le décodage du premier macrobloc d'une tranche ou après le décodage du bit pcm_alignment_zero_bit et de toutes les données de pcm_byte pour un macrobloc de type I_PCM.

Les résultats de ce processus sont les registres initialisés du moteur de décodage codIRange et codIOffset, tous deux avec une précision de registre de 16 bits.

L'état du moteur de décodage arithmétique est représenté par les variables codIRange et codIOffset. Dans la procédure d'initialisation du processus de décodage arithmétique, codIRange est mis égal à 0x01FE et codIOffset est mis égal à la valeur résultant de read_bits(9) interprété comme une représentation binaire de 9 bits d'un entier non signé avec le bit de plus fort poids écrit en premier.

NOTE – La description du moteur de décodage arithmétique utilise dans la présente Recommandation | Norme internationale une précision de registre de 16 bits. Cependant, la précision de registre minimale pour les variables `codIRange` et `codIOffset` est de 9 bits.

9.3.2 Processus de binairisation

L'entrée pour ce processus est une demande d'élément syntaxique.

Le résultat de ce processus est la binairisation de l'élément syntaxique, `maxBinIdxCtx`, `ctxIdxOffset`, et `bypassFlag`.

Le Tableau 9-24 spécifie le type de processus de binairisation, `maxBinIdxCtx`, et `ctxIdxOffset` associé à chaque élément syntaxique.

La spécification du processus de binairisation unaire (U), le processus de binairisation unaire tronqué (TU, *truncated unary*), le processus de binairisation Exp-Golomb de $k^{\text{ième}}$ ordre unaire concaténé (UEGk, *concatenated unary / k-th order Exp-Golomb*), et le processus de binairisation de longueur fixe (FL, *fixed-length*) sont donnés respectivement aux § 9.3.2.1 à 9.3.2.4. D'autres binairisations sont spécifiées aux § 9.3.2.5 à 9.3.2.7.

Excepté pour les tranches I, les binairisations pour l'élément syntaxique `mb_type`, comme spécifié au § 9.3.2.5, consistent en une chaîne de segments donnés par un enchaînement de chaînes binaires de préfixes et de suffixes. La binairisation UEGk, spécifiée au § 9.3.2.3, qui est utilisée pour la binairisation des éléments syntaxiques `mvd_IX` ($X = 0, 1$) et `coeff_abs_level_minus1`, et la binairisation du `coded_block_pattern` consistent aussi en un enchaînement de chaînes binaires de préfixes et de suffixes. Pour ces processus de binairisation, la chaîne binaire de préfixes et celle de suffixes sont indexées séparément en utilisant la variable `binIdx` comme spécifié plus loin au § 9.3.3. Les deux ensembles de chaînes binaires de préfixes et de suffixes sont désignés respectivement comme la partie préfixe de la binairisation et la partie suffixe de la binairisation.

Une valeur spécifique de la variable de décalage d'indice de contexte (`ctxIdxOffset`) et une valeur spécifique de la variable `maxBinIdxCtx` sont associées à chaque binairisation ou partie de binairisation d'un élément syntaxique, comme indiqué au Tableau 9-24. Lorsque deux valeurs pour chacune de ces variables sont spécifiées pour un élément syntaxique dans le Tableau 9-24, la valeur dans la rangée supérieure se rapporte à la partie préfixe tandis que la valeur dans la rangée inférieure se rapporte à la partie suffixe de la binairisation de l'élément syntaxique correspondant.

L'utilisation du processus `DecodeBypass` et de la variable `bypassFlag` est déduite comme suit.

- Si aucune valeur n'est allouée à `ctxIdxOffset` pour la binairisation ou partie de binairisation correspondante marquée "na" dans le Tableau 9-24, tous les segments de la chaîne binaire de la binairisation ou partie de préfixe/suffixe de binairisation correspondante doivent être décodés en invoquant le processus `DecodeBypass` comme spécifié au § 9.3.3.2.3. Dans un tel cas, `bypassFlag` est mis égal à 1, et `bypassFlag` est utilisé pour indiquer que pour l'analyse grammaticale de la valeur du segment à partir du flux binaire, le processus `DecodeBypass` doit être appliqué.
- Autrement, pour chaque valeur possible de `binIdx` jusqu'à la valeur spécifiée de `MaxBinIdxCtx` donnée au Tableau 9-24, une valeur spécifique de la variable `ctxIdx` est spécifiée plus en détail au § 9.3.3. `bypassFlag` est mis égal à 0.

Les valeurs possibles de l'indice de contexte `ctxIdx` sont dans la gamme de 0 à 398 inclus. La valeur allouée à `ctxIdxOffset` spécifie la valeur inférieure de la gamme de `ctxIdx` allouée à la binairisation ou partie de binairisation correspondante d'un élément syntaxique.

`ctxIdx = ctxIdxOffset = 276` est alloué à l'élément syntaxique `end_of_slice_flag` et au segment de `mb_type`, qui spécifie le type de macrobloc `I_PCM` comme spécifié plus en détail au § 9.3.3.1. Pour l'analyse grammaticale de la valeur du segment correspondant provenant du flux binaire, on doit

appliquer le processus arithmétique de décodage pour les décisions avant la terminaison (DecodeTerminate), comme spécifié au § 9.3.3.2.4.

NOTE – Les segments de mb_type dans les tranches I et les segments du suffixe pour mb_type dans les tranches SI qui correspondent à la même valeur de binIdx partagent le même ctxIdx. Le dernier segment du préfixe de mb_type et le premier segment du suffixe de mb_type dans les tranches P, SP, et B peuvent partager le même ctxIdx.

Tableau 9-24 – Eléments syntaxiques et types de binairisation associés, maxBinIdxCtx et ctxIdxOffset

Elément syntaxique	Type de binairisation	maxBinIdxCtx	ctxIdxOffset
mb_type (seulement tranches SI)	Préfixe et suffixe comme spécifié au § 9.3.2.5	Préfixe: 0 Suffixe: 6	Préfixe: 0 Suffixe: 3
mb_type (seulement tranches I)	comme spécifié au § 9.3.2.5	6	3
mb_skip_flag (seulement tranches P, SP)	FL, cMax=1	0	11
mb_type (seulement tranches P, SP)	Préfixe et suffixe comme spécifié au § 9.3.2.5	Préfixe: 2 Suffixe: 5	Préfixe: 14 Suffixe: 17
sub_mb_type (seulement tranches P, SP)	Comme spécifié au § 9.3.2.5	2	21
mb_skip_flag (seulement tranches B)	FL, cMax=1	0	24
mb_type (seulement tranches B)	Préfixe et suffixe comme spécifié au § 9.3.2.5	Préfixe: 3 Suffixe: 5	Préfixe: 27 Suffixe: 32
sub_mb_type (seulement tranches B)	Comme spécifié au § 9.3.2.5	3	36
mvd_10[][][0], mvd_11[][][0]	Préfixe et suffixe comme donné par UEG3, avec signedValFlag=1, uCoff=9	Préfixe: 4 Suffixe: na	Préfixe: 40 Suffixe: na (utilise DecodeBypass)
mvd_10[][][1], mvd_11[][][1]		Préfixe: 4 Suffixe: na	Préfixe: 47 Suffixe: na (utilise DecodeBypass)
ref_idx_10, ref_idx_11	U	2	54
mb_qp_delta	Comme spécifié au § 9.3.2.7	2	60
intra_chroma_pred_mode	TU, cMax=3	1	64
prev_intra4x4_pred_mode_flag	FL, cMax=1	0	68
rem_intra4x4_pred_mode	FL, cMax=7	0	69
mb_field_decoding_flag	FL, cMax=1	0	70
coded_block_pattern	Préfixe et suffixe comme spécifié au § 9.3.2.6	Préfixe: 3 Suffixe: 1	Préfixe: 73 Suffixe: 77
coded_block_flag	FL, cMax=1	0	85
significant_coeff_flag (seulement les blocs codés de trame)	FL, cMax=1	0	105

Tableau 9-24 – Éléments syntaxiques et types de binairisation associés, maxBinIdxCtx et ctxIdxOffset

Élément syntaxique	Type de binairisation	maxBinIdxCtx	ctxIdxOffset
last_significant_coeff_flag (seulement les blocs codés de trame)	FL, cMax=1	0	166
coeff_abs_level_minus1	Préfixe et suffixe comme donné par UEG0 avec signedValFlag=0, uCoff=14	Préfixe: 1 Suffixe: na	Préfixe: 227 Suffixe: na, (utilise DecodeBypass)
coeff_sign_flag	FL, cMax=1	0	na, (utilise DecodeBypass)
end_of_slice_flag	FL, cMax=1	0	276
significant_coeff_flag (seulement les blocs codés de champ)	FL, cMax=1	0	277
last_significant_coeff_flag (seulement les blocs codés de champ)	FL, cMax=1	0	338

9.3.2.1 Processus de binairisation unaire (U)

L'entrée pour ce processus est une demande de binairisation U pour un élément syntaxique.

Le résultat de ce processus est la binairisation U de l'élément syntaxique.

La chaîne de segments d'un élément syntaxique ayant une valeur (un entier arithmétique) synElVal est une chaîne binaire de longueur synElVal + 1 indexée par BinIdx. Les segments qui pour binIdx sont inférieurs à synElVal sont égaux à 1. Le segment avec binIdx égal à synElVal est égal à 0.

Le Tableau 9-25 illustre les chaînes de segments de la binairisation unaire pour un élément syntaxique.

Tableau 9-25 – Chaîne de segments de la binairisation unaire (pour information)

Valeur de l'élément syntaxique	Chaîne de segments					
0	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 Processus de binairisation unaire tronqué (TU)

L'entrée pour ce processus est une demande de binairisation TU pour un élément syntaxique et cMax.

Le résultat de ce processus est la binairisation TU de l'élément syntaxique.

Pour les valeurs d'élément syntaxique (entier non signé) inférieures à cMax, le processus de binarisation U est invoqué, comme spécifié au § 9.3.2.1. Pour la valeur d'élément syntaxique égale à cMax la chaîne de segments est une chaîne binaire de longueur cMax avec tous les segments égaux à 1.

NOTE – La binarisation TU est toujours invoquée avec une valeur de cMax égale à la plus grande valeur possible de l'élément syntaxique en cours de décodage.

9.3.2.3 Processus de binarisation Exp-Golomb de k^{ième} ordre unaire concaténé (UEGk)

L'entrée pour ce processus est une demande de binarisation UEGk pour un élément syntaxique, signedValFlag et uCoff.

Le résultat de ce processus est la binarisation UEGk de l'élément syntaxique.

Une chaîne de segments UEGk est un enchaînement d'une chaîne binaire de préfixe et d'une chaîne binaire de suffixe. Le préfixe de la binarisation est spécifié par l'invocation du processus de binarisation TU pour la partie préfixe $\text{Min}(\text{uCoff}, \text{Abs}(\text{synElVal}))$ d'une valeur d'élément syntaxique synElVal comme spécifié au § 9.3.2.2 avec $\text{cMax} = \text{uCoff}$, où $\text{uCoff} > 0$.

La chaîne de segments UEGk est déduite comme suit:

- Si une des conditions suivantes est Vraie, la chaîne de segments d'un élément syntaxique qui a une valeur synElVal ne comporte seulement qu'une chaîne binaire de préfixe,
 - signedValFlag est égal à 0 et la chaîne binaire préfixe n'est pas égale à la chaîne binaire de longueur uCoff avec tous les bits égaux à 1;
 - signedValFlag est égal à 1 et la chaîne binaire préfixe est égale à la chaîne binaire qui comporte un seul bit d'une valeur égale à 0.
- Autrement, la chaîne de segments de la partie suffixe UEGk d'une valeur d'un élément syntaxique synElVal est spécifiée par un processus équivalent au pseudo-code suivant:

```

si( Abs( synElVal ) >= uCoff ) {
  sufS = Abs( synElVal ) - uCoff
  stopLoop = 0
  faire {
    si( sufS >= ( 1 << k ) ) {
      mettre( 1 )
      sufS = sufS - ( 1 << k )
      k++
    } ou {
      mettre( 0 )
      tandis que( k-- )
        mettre( ( sufS >> k ) & 0x01 )
      stopLoop = 1
    }
  }
  } tandi que( !stopLoop )
}
si( signedValFlag && synElVal != 0 )
  si( synElVal > 0 )
    mettre( 0 )
  ou
    mettre( 1 )

```

NOTE – La spécification pour le code Exp-Golomb de k^{ième} ordre (EGk) utilise une signification inversée pour les 1 et les 0 pour la partie unaire du code Exp-Golomb d'ordre 0, comme spécifié au § 9.1.

9.3.2.4 Processus de binarisation de longueur fixe (FL)

L'entrée pour ce processus est une demande de binarisation FL pour un élément syntaxique et cMax.

Le résultat de ce processus est la binairisation FL de l'élément syntaxique.

La binairisation FL est construite en utilisant une chaîne de segments d'entiers arithmétiques `fixedLength-bit` de la valeur de l'élément syntaxique, où `fixedLength = Ceil(Log2(cMax + 1))`. L'indexation des segments pour la binairisation FL est telle que `binIdx = 0` se rapporte au bit de plus faible poids et que les valeurs croissantes de `binIdx` vont vers le bit de plus fort poids.

9.3.2.5 Processus de binairisation pour le type de macrobloc et de sous-macrobloc

L'entrée pour ce processus est une demande de binairisation pour les éléments syntaxiques `mb_type` ou `sub_mb_type`.

Le résultat de ce processus est la binairisation de l'élément syntaxique.

Le schéma de binairisation pour le décodage du type de macrobloc dans les tranches I est spécifié au Tableau 9-26.

Pour les types de macrobloc de tranches SI, la binairisation consiste en chaînes de segments spécifiées comme la concaténation d'une chaîne binaire de préfixe et de suffixe comme suit.

La chaîne binaire de préfixe consiste en un seul bit, qui est spécifié par $b_0 = ((mb_type == SI) ? 0 : 1)$. Pour la valeur de l'élément syntaxique pour laquelle b_0 est égal à 0, la chaîne de segments ne comporte que la chaîne binaire de préfixe. Pour la valeur d'élément syntaxique pour laquelle b_0 est égal à 1, la binairisation est donnée par la concaténation du préfixe b_0 et de la chaîne binaire de suffixe comme spécifié au Tableau 9-26 pour le type de macrobloc des tranches I indexé en soustrayant 1 de la valeur de `mb_type` dans les tranches SI.

Tableau 9-26 – Binairisation pour les types de macrobloc dans les tranches I

Valeur (nom) de mb_type	Chaîne de segments						
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

Les schémas de binairisation pour les types de macrobloc P dans les tranches P et SP et pour les macroblocs B dans les tranches B sont spécifiés au Tableau 9-27.

La chaîne de segments pour les types de macrobloc I dans les tranches P et SP correspondant aux valeurs mb_type de 5 à 30 consiste en une concaténation d'un préfixe, qui consiste en un seul bit avec une valeur égale à 1, comme spécifié au Tableau 9-27 et d'un suffixe, comme spécifié au Tableau 9-26, indexé en soustrayant 5 de la valeur de mb_type.

mb_type égal à 4 (P_8x8ref0) n'est pas autorisé.

Pour les types de macrobloc I dans les tranches B (valeurs de mb_type de 23 à 48) la binairisation consiste en chaînes de segments spécifiées comme concaténation d'une chaîne binaire de préfixe, comme spécifié au Tableau 9-27, et de chaînes binaires de suffixe, comme spécifié au Tableau 9-26, indexées en soustrayant 23 de la valeur de mb_type.

Tableau 9-27 – Binairisation pour les types de macrobloc dans les tranches P, SP, et B

Type de tranche	Valeur (nom) de mb_type	Chaîne de segments							
Tranche P, SP	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 à 30 (Intra, seulement préfixe)	1							
Tranche B	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 à 48 (Intra, seulement préfixe)	1	1	1	1	0	1			
binIdx	0	1	2	3	4	5	6		

Pour les tranches P, SP, et B la spécification de la binairisation pour sub_mb_type est donnée au Tableau 9-28.

Tableau 9-28 – Binairisation pour les types de sous-macrobloc dans les tranches P, SP, et B

Type de tranche	Valeur (nom) de sub_mb_type	Chaîne de segments					
Tranches P, SP	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			
Tranche B	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 Processus de binairisation pour schéma de bloc codé

L'entrée pour ce processus est une demande de binairisation pour l'élément syntaxique coded_block_pattern.

Le résultat de ce processus est la binairisation de l'élément syntaxique.

La binairisation de coded_block_pattern consiste en la concaténation d'une partie de préfixe et d'une partie de suffixe. La partie préfixe de la binairisation est donnée par la binairisation FL de CodedBlockPatternLuma avec cMax = 15. La partie suffixe consiste en la binairisation TU de CodedBlockPatternChroma avec cMax = 2. La relation entre la valeur de l'élément syntaxique coded_block_pattern et les valeurs de CodedBlockPatternLuma et CodedBlockPatternChroma est donnée comme spécifié au § 7.4.5.

9.3.2.7 Processus de binairisation pour mb_qp_delta

L'entrée pour ce processus est une demande de binairisation pour l'élément syntaxique mb_qp_delta.

Le résultat de ce processus est la binairisation de l'élément syntaxique.

La chaîne de segments de mb_qp_delta est déduite par la binairisation U de la valeur correspondante de l'élément syntaxique mb_qp_delta, où la règle d'allocation entre la valeur algébrique de mb_qp_delta et sa valeur correspondante est donnée comme spécifié au Tableau 9-3.

9.3.3 Flux de processus de décodage

L'entrée pour ce processus est une binairisation de l'élément syntaxique demandé, maxBinIdxCtx, bypassFlag et ctxIdxOffset comme spécifié au § 9.3.2.

Le résultat de ce processus est la valeur de l'élément syntaxique.

Ce processus spécifie comment chaque bit d'une chaîne binaire est analysé grammaticalement pour chaque élément syntaxique.

Après l'analyse grammaticale de chaque bit, la chaîne binaire résultante est comparée à toutes les chaînes de segments de la binairisation de l'élément syntaxique et on applique ce qui suit.

- Si la chaîne binaire est égale à une des chaînes de segments, la valeur correspondante de l'élément syntaxique est le résultat.
- Autrement (la chaîne binaire n'est pas égale à une des chaînes de segments), le bit suivant est analysé grammaticalement.

Lors de l'analyse grammaticale de chaque segment, la variable `binIdx` est incrémentée de 1 en commençant par `binIdx` mise égale à 0 pour le premier segment.

Lorsque la binairisation de l'élément syntaxique correspondant consiste en une partie de binairisation de préfixe et de suffixe, la variable `binIdx` est mise égale à 0 pour le premier segment de chaque partie de la chaîne de segments (partie préfixe ou partie suffixe). Dans ce cas, après l'analyse grammaticale de la chaîne binaire de préfixe, le processus d'analyse grammaticale de la chaîne binaire de suffixe se rapportant aux binairisations spécifiées aux § 9.3.2.3 et 9.3.2.5 est invoquée en fonction de la chaîne binaire de préfixe résultante, comme spécifié aux § 9.3.2.3 et 9.3.2.5. Noter que pour la binairisation de l'élément syntaxique `coded_block_pattern`, la chaîne binaire de suffixe est présente sans considération de la chaîne binaire de préfixe de longueur 4 comme spécifié au § 9.3.2.6.

En fonction de la variable `bypassFlag`, on applique ce qui suit.

- Si `bypassFlag` est égal à 1, on doit appliquer le processus de décodage de dérivation, comme spécifié au § 9.3.3.2.3, pour analyser grammaticalement la valeur des segments provenant du flux binaire.
- Autrement (`bypassFlag` est égal à 0), l'analyse grammaticale de chaque segment est spécifiée par les deux étapes ordonnées suivantes:
 1. étant donné `binIdx`, `maxBinIdxCtx` et `ctxIdxOffset`, `ctxIdx` est calculé comme spécifié au § 9.3.3.1;
 2. étant donné `ctxIdx`, on décode la valeur du segment provenant du flux binaire, comme spécifié au § 9.3.3.2.

9.3.3.1 Processus de déduction pour `ctxIdx`

Les entrées pour ce processus sont `binIdx`, `maxBinIdxCtx` et `ctxIdxOffset`.

Le résultat de ce processus est `ctxIdx`.

Le Tableau 9-29 donne les allocations des incréments de `ctxIdx` (`ctxIdxInc`) à `binIdx` pour toutes les valeurs de `ctxIdxOffset` excepté celles qui se rapportent aux éléments syntaxiques `coded_block_flag`, `significant_coeff_flag`, `last_significant_coeff_flag`, et `coeff_abs_level_minus1`.

Le `ctxIdx` à utiliser avec un `binIdx` spécifique est spécifié en déterminant d'abord le `ctxIdxOffset` associé à la chaîne de segments donnée ou sa partie. Le `ctxIdx` est déterminé comme suit.

- Si le `ctxIdxOffset` figure dans le Tableau 9-29, le `ctxIdx` pour un `binIdx` est la somme de `ctxIdxOffset` et de `ctxIdxInc`, qu'on trouve au Tableau 9-29. Lorsque plus d'une valeur figure dans le Tableau 9-29 pour un `binIdx`, le processus d'allocation pour `ctxIdxInc` pour ce `binIdx` est spécifiée plus en détail dans les paragraphes donnés entre parenthèses dans l'entrée de tableau correspondante.
- Autrement (`ctxIdxOffset` ne figure pas dans le Tableau 9-29), le `ctxIdx` est spécifié comme la somme des termes suivants: `ctxIdxOffset` et `ctxIdxBlockCatOffset(ctxBlockCat)` comme spécifié au Tableau 9-30 et `ctxIdxInc(ctxBlockCat)`. Le § 9.3.3.1.3 spécifie quel `ctxBlockCat` est utilisé. Le § 9.3.3.1.1.9 spécifie l'allocation de `ctxIdxInc(ctxBlockCat)`

pour `coded_block_flag`, et le § 9.3.3.1.3 spécifie l'allocation de `ctxIdxInc(ctxBlockCat)` pour `significant_coeff_flag`, `last_significant_coeff_flag`, et `coeff_abs_level_minus1`.

Tous les segments avec `binIdx` plus grand que `maxBinIdxCtx` sont analysés grammaticalement en utilisant `ctxIdx` alloué à `maxBinIdxCtx`.

Toutes les entrées du Tableau 9-29 marquées "na" correspondent aux valeurs de `binIdx` qui n'interviennent pas pour le `ctxIdxOffset` correspondant.

`ctxIdx = 276` est alloué au `binIdx` of `mb_type` indiquant le mode `I_PCM`. Pour analyser grammaticalement la valeur des segments correspondants provenant du flux binaire, on doit appliquer le processus de décodage arithmétique pour les décisions avant achèvement, comme spécifié au § 9.3.3.2.4.

Tableau 9-29 – Allocation de ctxIdxInc à binIdx pour toutes les valeurs de ctxIdxOffset excepté celles se rapportant aux éléments syntaxiques coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1

ctxIdxOffset	binIdx						
	0	1	2	3	4	5	>= 6
0	0,1,2 (§ 9.3.3.1.1.3)	na	na	na	na	na	na
3	0,1,2 (§ 9.3.3.1.1.3)	ctxIdx=276	3	4	5,6 (§ 9.3.3.1.2)	6,7 (§ 9.3.3.1.2)	7
11	0,1,2 (§ 9.3.3.1.1.1)	na	na	na	na	na	na
14	0	1	2,3 (§ 9.3.3.1.2)	na	na	na	na
17	0	ctxIdx=276	1	2	2,3 (§ 9.3.3.1.2)	3	3
21	0	1	2	na	na	na	na
24	0,1,2 (§ 9.3.3.1.1.1)	na	na	na	na	na	na
27	0,1,2 (§ 9.3.3.1.1.3)	3	4,5 (§ 9.3.3.1.2)	5	5	5	5
32	0	ctxIdx=276	1	2	2,3 (§ 9.3.3.1.2)	3	3
36	0	1	2,3 (§ 9.3.3.1.2)	3	3	3	na
40	0,1,2 (§ 9.3.3.1.1.7)	3	4	5	6	6	6
47	0,1,2 (§ 9.3.3.1.1.7)	3	4	5	6	6	6
54	0,1,2,3 (§ 9.3.3.1.1.6)	4	5	5	5	5	5
60	0,1 (§ 9.3.3.1.1.5)	2	3	3	3	3	3
64	0,1,2 (§ 9.3.3.1.1.8)	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)	na	na	na	na	na	na
73	0,1,2,3 (§ 9.3.3.1.1.4)	0,1,2,3 (§ 9.3.3.1.1.4)	0,1,2,3 (§ 9.3.3.1.1.4)	0,1,2,3 (§ 9.3.3.1.1.4)	na	na	na
77	0,1,2,3 (§ 9.3.3.1.1.4)	4,5,6,7 (§ 9.3.3.1.1.4)	na	na	na	na	na
276	0	na	na	na	na	na	na

Le Tableau 9-30 donne les valeurs de ctxIdxBlockCatOffset en fonction de ctxBlockCat pour les éléments syntaxiques coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1. La spécification de ctxBlockCat est donnée au Tableau 9-32.

Tableau 9-30 – Allocation de ctxIdxBlockCatOffset à ctxBlockCat pour les éléments syntaxiques coded_block_flag, significant_coeff_flag, last_significant_coeff_flag, et coeff_abs_level_minus1

Élément syntaxique	ctxBlockCat (comme spécifié au Tableau 9-32)				
	0	1	2	3	4
coded_block_flag	0	4	8	12	16
significant_coeff_flag	0	15	29	44	47
last_significant_coeff_flag	0	15	29	44	47
coeff_abs_level_minus1	0	10	20	30	39

9.3.3.1.1 Processus d'allocation de ctxIdxInc en utilisant les éléments syntaxiques du voisinage

Le § 9.3.3.1.1.1 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique mb_skip_flag.

Le § 9.3.3.1.1.2 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique mb_field_decoding_flag.

Le § 9.3.3.1.1.3 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique mb_type.

Le § 9.3.3.1.1.4 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique coded_block_pattern.

Le § 9.3.3.1.1.5 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique mb_qp_delta.

Le § 9.3.3.1.1.6 spécifie le processus de déduction de ctxIdxInc pour les éléments syntaxiques ref_idx_l0 et ref_idx_l1.

Le § 9.3.3.1.1.7 spécifie le processus de déduction de ctxIdxInc pour les éléments syntaxiques mvd_l0 et mvd_l1.

Le § 9.3.3.1.1.8 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique intra_chroma_pred_mode.

Le § 9.3.3.1.1.9 spécifie le processus de déduction de ctxIdxInc pour l'élément syntaxique coded_block_flag.

9.3.3.1.1.1 Processus de déduction de ctxIdxInc pour l'élément syntaxique mb_skip_flag

Le résultat de ce processus est ctxIdxInc.

Lorsque MbaffFrameFlag est égal à 1 et que mb_field_decoding_flag n'a pas été décodé (pas encore) pour la paire de macroblocs en cours avec l'adresse de macrobloc supérieur $2 * (\text{CurrMbAddr}/2)$, on doit appliquer la règle d'inférence pour l'élément syntaxique mb_field_decoding_flag, comme spécifié au § 7.4.4.

Le processus de déduction pour les macroblocs du voisinage spécifié au § 6.4.7.1 est invoqué et le résultat est alloué à mbAddrA et mbAddrB.

Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.

- Si mbAddrN n'est pas disponible ou si mb_skip_flag pour le macrobloc mbAddrN est égal à 1, condTermFlagN est mise égale à 0.
- Autrement (mbAddrN est disponible et mb_skip_flag pour le macrobloc mbAddrN est égal à 0), condTermFlagN est mise égale à 1.

La variable `ctxIdxInc` est calculée par

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-1)$$

9.3.3.1.1.2 Processus de déduction de `ctxIdxInc` pour l'élément syntaxique `mb_field_decoding_flag`

Le résultat de ce processus est `ctxIdxInc`.

On invoque le processus de déduction pour les adresses des macroblocs du voisinage et leur disponibilité dans les trames MBAFF, comme spécifié au § 6.4.6, et le résultat est alloué à `mbAddrA` et `mbAddrB`.

Lorsque les macroblocs `mbAddrN` et `mbAddrN + 1` ont tous deux `mb_type` égal à `P_Skip` ou `B_Skip`, on doit appliquer la règle d'inférence pour l'élément syntaxique `mb_field_decoding_flag`, comme spécifié au § 7.4.4, pour le macrobloc `mbAddrN`.

Soit la variable `condTermFlagN` (N étant A ou B) calculée comme suit.

- Si l'une des conditions suivantes est vraie, `condTermFlagN` est mise égale à 0,
 - `mbAddrN` n'est pas disponible
 - le macrobloc `mbAddrN` est un macrobloc de trame.
- Autrement, `condTermFlagN` est mise égale à 1.

La variable `ctxIdxInc` est calculée par

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-2)$$

9.3.3.1.1.3 Processus de déduction de `ctxIdxInc` pour l'élément syntaxique `mb_type`

L'entrée pour ce processus est `ctxIdxOffset`.

Le résultat de ce processus est `ctxIdxInc`.

Le processus de déduction pour les macroblocs du voisinage spécifié au § 6.4.7.1 est invoqué et le résultat est alloué à `mbAddrA` et `mbAddrB`.

Soit la variable `condTermFlagN` (N étant A ou B) calculée comme suit.

- Si l'une des conditions suivantes est Vraie, `condTermFlagN` est mis égal à 0
 - `mbAddrN` n'est pas disponible
 - `ctxIdxOffset` est égal à 0 et `mb_type` pour le macrobloc `mbAddrN` est égal à `SI`
 - `ctxIdxOffset` est égal à 3 et `mb_type` pour le macrobloc `mbAddrN` est égal à `I_4x4`
 - `ctxIdxOffset` est égal à 27 et le macrobloc `mbAddrN` est sauté
 - `ctxIdxOffset` est égal à 27 et `mb_type` pour le macrobloc `mbAddrN` est égal à `B_Direct_16x16`
- Autrement, `condTermFlagN` est mise égale à 1.

La variable `ctxIdxInc` est calculée par

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-3)$$

9.3.3.1.1.4 Processus de déduction de `ctxIdxInc` pour l'élément syntaxique `coded_block_pattern`

Les entrées pour ce processus sont `ctxIdxOffset` et `binIdx`.

Le résultat de ce processus est `ctxIdxInc`.

En fonction de la valeur de la variable `ctxIdxOffset`, on applique ce qui suit.

- Si `ctxIdxOffset` est égal à 73, on applique ce qui suit

- Le processus de déduction pour les blocs luma 8x8 du voisinage, spécifié au § 6.4.7.2, est invoqué avec `luma8x8BlkIdx = binIdx` en entrée et le résultat est alloué à `mbAddrA`, `mbAddrB`, `luma8x8BlkIdxA`, et `luma8x8BlkIdxB`.
- Soit la variable `condTermFlagN` (N étant A ou B) calculée comme suit.
- Si l'une des conditions suivantes est Vraie, `condTermFlagN` est mise égale à 0
 - `mbAddrN` n'est pas disponible
 - `mb_type` pour le macrobloc `mbAddrN` est égal à `I_PCM`
 - le macrobloc `mbAddrN` n'est pas sauté et $((\text{CodedBlockPatternLuma} \gg \text{luma8x8BlkIdxN}) \& 1)$ n'est pas égal à 0 pour le macrobloc `mbAddrN`
- Autrement, `condTermFlagN` est mise égale à 1.
 - La variable `ctxIdxInc` est calculée par

$$\text{ctxIdxInc} = \text{condTermFlagA} + 2 * \text{condTermFlagB} \quad (9-4)$$
- Autrement (`ctxIdxOffset` est égal à 77), on applique ce qui suit.
 - Le processus de déduction pour les macroblocs du voisinage spécifié au § 6.4.7.1 est invoqué et le résultat est alloué à `mbAddrA` et `mbAddrB`.
- Soit la variable `condTermFlagN` (N étant A ou B) calculée comme suit.
 - Si `mbAddrN` est disponible et que `mb_type` pour le macrobloc `mbAddrN` est égal à `I_PCM`, `condTermFlagN` est mise égale à 1
 - Autrement, si l'une des conditions suivantes est vraie, `condTermFlagN` est mise égale à 0
 - `mbAddrN` n'est pas disponible ou le macrobloc `mbAddrN` est sauté
 - `binIdx` est égal à 0 et `CodedBlockPatternChroma` pour le macrobloc `mbAddrN` est égal à 0
 - `binIdx` est égal à 1 et `CodedBlockPatternChroma` pour le macrobloc `mbAddrN` n'est pas égal à 2
 - Autrement, `condTermFlagN` est mise égale à 1.
- La variable `ctxIdxInc` est calculée par

$$\text{ctxIdxInc} = \text{condTermFlagA} + 2 * \text{condTermFlagB} + ((\text{binIdx} == 1) ? 4 : 0) \quad (9-5)$$

NOTE – Lorsque un macrobloc utilise un mode de prédiction `Intra_16x16`, les valeurs de `CodedBlockPatternLuma` et `CodedBlockPatternChroma` pour le macrobloc sont calculées à partir de `mb_type` comme spécifié au Tableau 7-8.

9.3.3.1.1.5 Processus de déduction de `ctxIdxInc` pour l'élément syntaxique `mb_qp_delta`

Le résultat de ce processus est `ctxIdxInc`.

Soit `prevMbAddr` l'adresse de macrobloc du macrobloc qui précède le macrobloc en cours dans l'ordre de décodage. Lorsque le macrobloc en cours est le premier macrobloc d'une tranche, `prevMbAddr` est marqué comme indisponible.

Soit la variable `ctxIdxInc` calculée comme suit.

- Si l'une des conditions suivantes est Vraie, `ctxIdxInc` est mise égale à 0
 - `prevMbAddr` n'est pas disponible ou le macrobloc `prevMbAddr` est sauté
 - `mb_type` du macrobloc `prevMbAddr` est égal à `I_PCM`
 - le macrobloc `prevMbAddr` n'est pas codé en mode de prédiction `Intra_16x16` et `CodedBlockPatternLuma` et `CodedBlockPatternChroma` sont tous deux égaux à 0 pour le macrobloc `prevMbAddr`

- mb_qp_delta est égal à 0 pour le macrobloc prevMbAddr
- Autrement, ctxIdxInc est mise égale à 1.

9.3.3.1.1.6 Processus de déduction de ctxIdxInc pour les éléments syntaxiques ref_idx_10 et ref_idx_11

Les entrées pour ce processus sont mbPartIdx et la liste de suffixes d'image de référence IX, où X = 0 ou 1.

Le résultat de ce processus est ctxIdxInc.

Soit currSubMbType mis égal à sub_mb_type[mbPartIdx].

Le processus de déduction pour les subdivisions du voisinage spécifiées au § 6.4.7.5 est invoqué avec mbPartIdx, currSubMbType et subMbPartIdx = 0 en entrée et le résultat est alloué à mbAddrA\mbPartIdxA et mbAddrB\mbPartIdxB.

Avec ref_idx_IX[mbPartIdxN] (N étant A ou B) spécifiant l'élément syntaxique pour le macrobloc mbAddrN, soit la variable refIdxZeroFlagN calculée comme suit.

- Si MbaffFrameFlag est égal à 1, le macrobloc en cours est un macrobloc de trame, et le macrobloc mbAddrN est un macrobloc de champ

$$\text{refIdxZeroFlagN} = ((\text{ref_idx_IX}[\text{mbPartIdxN}] > 1) ? 0 : 1) \quad (9-6)$$

- Autrement,

$$\text{refIdxZeroFlagN} = ((\text{ref_idx_IX}[\text{mbPartIdxN}] > 0) ? 0 : 1) \quad (9-7)$$

Soit la variable predModeEqualFlag spécifiée comme suit.

- Si le macrobloc mbAddrN a le mb_type égal à P_8x8 ou B_8x8, on applique ce qui suit.
 - Si SubMbPredMode(sub_mb_type[mbPartIdxN]) n'est pas égal à Pred_LX ni à BiPred, predModeEqualFlag est mise égale à 0, où sub_mb_type spécifie l'élément syntaxique pour le macrobloc mbAddrN.
 - Autrement, predModeEqualFlag est mise égale à 1.
- Autrement, on applique ce qui suit.
 - Si MbPartPredMode(mb_type, mbPartIdxN) n'est pas égal à Pred_LX ni à BiPred, predModeEqualFlag est mise égale à 0, où mb_type spécifie l'élément syntaxique pour le macrobloc mbAddrN.
 - Autrement, predModeEqualFlag est mise égale à 1.

Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.

- Si l'une des conditions suivantes est Vraie, condTermFlagN est mise égale à 0
 - mbAddrN n'est pas disponible
 - le macrobloc mbAddrN a le mb_type égal à P_Skip ou B_Skip
 - le macrobloc mbAddrN est codé en mode d'intraprédiction
 - predModeEqualFlag est égal à 0
 - refIdxZeroFlagN est égal à 1
- Autrement, condTermFlagN est mise égale à 1.

La variable ctxIdxInc est calculée par

$$\text{ctxIdxInc} = \text{condTermFlagA} + 2 * \text{condTermFlagB} \quad (9-8)$$

9.3.3.1.1.7 Processus de déduction de ctxIdxInc pour les éléments syntaxiques mvd_I0 et mvd_I1

Les entrées pour ce processus sont mbPartIdx, subMbPartIdx, le suffixe IX de liste d'image de référence, et ctxIdxOffset.

Le résultat de ce processus est ctxIdxInc.

Soit currSubMbType mis égal à sub_mb_type[mbPartIdx].

Le processus de déduction pour les subdivisions du voisinage, spécifié au § 6.4.7.5, est invoqué avec mbPartIdx, currSubMbType et subMbPartIdx en entrée et le résultat est alloué à mbAddrA\mbPartIdxA\subMbPartIdxA et mbAddrB\mbPartIdxB\subMbPartIdxB.

Soit la variable compIdx calculée comme suit.

- Si ctxIdxOffset est égal à 40, compIdx est mise égale à 0.
- Autrement (ctxIdxOffset est égal à 47), compIdx est mise égale à 1.

Soit la variable predModeEqualFlag spécifiée comme suit.

- Si le macrobloc mbAddrN a le mb_type égal à P_8x8 ou B_8x8, on applique ce qui suit.
 - Si SubMbPredMode(sub_mb_type[mbPartIdxN]) n'est pas égal à Pred_LX ni à BiPred, predModeEqualFlag est mise égale à 0, où sub_mb_type spécifie l'élément syntaxique pour le macrobloc mbAddrN.
 - Autrement, predModeEqualFlag est mise égale à 1.
- Autrement, on applique ce qui suit.
 - Si MbPartPredMode(mb_type, mbPartIdxN) n'est pas égal à Pred_LX ni à BiPred, predModeEqualFlag est mise égale à 0, où mb_type spécifie l'élément syntaxique pour le macrobloc mbAddrN.
 - Autrement, predModeEqualFlag est mise égale à 1.

Soit la variable absMvdCompN (N étant A ou B) calculée comme suit.

- Si l'une des conditions suivantes est Vraie, absMvdCompN est mise égale à 0
 - mbAddrN n'est pas disponible
 - le macrobloc mbAddrN a le mb_type égal à P_Skip ou B_Skip
 - le macrobloc mbAddrN est codé en mode d'intraprédiction
 - predModeEqualFlag est égale à 0
- Autrement, on applique ce qui suit
 - Si compIdx est égal à 1, MbaffFrameFlag est égal à 1, le macrobloc en cours est un macrobloc de trame, et le macrobloc mbAddrN est un macrobloc de champ
$$\text{absMvdCompN} = \text{Abs}(\text{mvd_IX}[\text{mbPartIdxN}] [\text{subMbPartIdxN}] [\text{compIdx}]) * 2 \quad (9-9)$$
 - Autrement, si compIdx est égal à 1, MbaffFrameFlag est égal à 1, le macrobloc en cours est un macrobloc de champ, et le macrobloc mbAddrN est un macrobloc de trame
$$\text{absMvdCompN} = \text{Abs}(\text{mvd_IX}[\text{mbPartIdxN}] [\text{subMbPartIdxN}] [\text{compIdx}]) / 2 \quad (9-10)$$
 - Autrement,
$$\text{absMvdCompN} = \text{Abs}(\text{mvd_IX}[\text{mbPartIdxN}] [\text{subMbPartIdxN}] [\text{compIdx}]) \quad (9-11)$$

La variable ctxIdxInc est déduite comme suit

- Si (absMvdCompA + absMvdCompB) est inférieur à 3, ctxIdxInc est mise égale à 0.
- Autrement, si (absMvdCompA + absMvdCompB) est supérieur à 32, ctxIdxInc est mise égale à 2.

- Autrement ((absMvdCompA + absMvdCompB) est dans la gamme de 3 à 32, inclus), ctxIdxInc est mise égale à 1.

9.3.3.1.1.8 Processus de déduction de ctxIdxInc pour l'élément syntaxique intra_chroma_pred_mode

Le résultat de ce processus est ctxIdxInc.

On invoque le processus de déduction pour les macroblocs du voisinage, spécifié au § 6.4.7.1, et le résultat est alloué à mbAddrA et mbAddrB.

Soit la variable condTermFlagN (avec N remplacé par A ou B) calculée comme suit.

- Si l'une des conditions suivantes est Vraie, condTermFlagN est mise égale à 0
 - mbAddrN n'est pas disponible;
 - Le macrobloc mbAddrN est codé en mode d'interprédiction;
 - mb_type pour le macrobloc mbAddrN est égal à I_PCM;
 - intra_chroma_pred_mode pour le macrobloc mbAddrN est égal à 0.
- Autrement, condTermFlagN est mise égale à 1.

La variable ctxIdxInc est calculée par

$$\text{ctxIdxInc} = \text{condTermFlagA} + \text{condTermFlagB} \quad (9-12)$$

9.3.3.1.1.9 Processus de déduction de ctxIdxInc pour l'élément syntaxique coded_block_flag

L'entrée pour ce processus est ctxBlockCat et une entrée supplémentaire est spécifiée comme suit.

- Si ctxBlockCat est égal à 0, pas d'entrée supplémentaire.
- Autrement, si ctxBlockCat est égal à 1 ou 2, luma4x4BlkIdx.
- Autrement, si ctxBlockCat est égal à 3, l'indice de composant chroma iCbCr.
- Autrement (ctxBlockCat est égal à 4), chroma4x4BlkIdx et l'indice de composant chroma compIdx.

Le résultat de ce processus est ctxIdxInc(ctxBlockCat).

Soit la variable transBlockN (N étant A ou B) calculée comme suit.

- Si ctxBlockCat est égal à 0, on applique ce qui suit.
 - On invoque le processus de déduction pour les macroblocs du voisinage spécifié au § 6.4.7.1 et le résultat est alloué à mbAddrN (N étant A ou B).
 - La variable transBlockN est déduite comme suit.
 - Si mbAddrN est disponible et que le macrobloc mbAddrN est codé en mode de prédiction Intra_16x16, le bloc DC luma du macrobloc mbAddrN est alloué à transBlockN.
 - Autrement, transBlockN est marquée comme indisponible.
- Autrement, si ctxBlockCat est égal à 1 ou 2, on applique ce qui suit.
 - Le processus de déduction pour les blocs luma 4x4 du voisinage, spécifié au § 6.4.7.3, est invoqué avec luma4x4BlkIdx en entrée et le résultat est alloué à mbAddrN, luma4x4BlkIdxN (N étant A ou B).
 - La variable transBlockN est déduite comme suit.
 - Si mbAddrN est disponible, le macrobloc mbAddrN n'est pas sauté, le mb_type pour le macrobloc mbAddrN n'est pas égal à I_PCM, et ((CodedBlockPatternLuma >> (luma4x4BlkIdxN >>> 2)) & 1) n'est pas égal à 0

pour le macrobloc mbAddrN, le bloc luma 4x4 avec luma4x4BlkIdxN du macrobloc mbAddrN est alloué à transBlockN.

- Autrement, transBlockN est marquée comme indisponible.
- Autrement, si ctxBlockCat est égal à 3, on applique ce qui suit.
 - Le processus de déduction pour les macroblocs du voisinage, spécifié au § 6.4.7.1, est invoqué et le résultat est alloué à mbAddrN (N étant A ou B).
 - La variable transBlockN est déduite comme suit.
 - Si mbAddrN est disponible, le macrobloc mbAddrN n'est pas sauté, le mb_type pour le macrobloc mbAddrN n'est pas égal à I_PCM, et CodedBlockPatternChroma n'est pas égal à 0 pour le macrobloc mbAddrN, le bloc DC chroma du composant chroma iCbCr du macrobloc mbAddrN est alloué à transBlockN.
 - Autrement, transBlockN est marquée comme indisponible.
- Autrement (ctxBlockCat est égal à 4), on applique ce qui suit.
 - Le processus de déduction pour les blocs chroma 4x4 voisins, spécifié au § 6.4.7.4, est invoqué avec chroma4x4BlkIdx en entrée et le résultat est alloué à mbAddrN, chroma4x4BlkIdxN (N étant A ou B).
 - La variable transBlockN est déduite comme suit.
 - Si mbAddrN est disponible, le macrobloc mbAddrN n'est pas sauté, le mb_type pour le macrobloc mbAddrN n'est pas égal à I_PCM, et CodedBlockPatternChroma est égal à 2 pour le macrobloc mbAddrN, le bloc chroma 4x4 avec chroma4x4BlkIdxN du composant chroma iCbCr du macrobloc mbAddrN est alloué à transBlockN.
 - Autrement, transBlockN est marqué comme indisponible.

Soit la variable condTermFlagN (N étant A ou B) calculée comme suit.

- Si l'une des conditions suivantes est Vraie, condTermFlagN est mise égale à 0
 - mbAddrN n'est pas disponible et le macrobloc en cours est codé en mode d'interprédiction;
 - mbAddrN est disponible et transBlockN n'est pas disponible et mb_type pour le macrobloc mbAddrN n'est pas égal à I_PCM;
 - Le macrobloc en cours est codé en mode d'intraprédiction, constrained_intra_pred_flag est égal à 1, le macrobloc mbAddrN est disponible et est codé en mode d'interprédiction, et la subdivision des données de tranche est utilisée (nal_unit_type est dans la gamme de 2 à 4, inclus).
- Autrement, si l'une des conditions suivantes est Vraie, condTermFlagN est mise égale à 1
 - mbAddrN n'est pas disponible et le macrobloc en cours est codé en mode d'intraprédiction;
 - mb_type pour le macrobloc mbAddrN est égal à I_PCM.
- Autrement, condTermFlagN est mise égale à la valeur du coded_block_flag du bloc de transformée transBlockN qui a été décodé pour le macrobloc mbAddrN.

La variable ctxIdxInc(ctxBlockCat) est calculée par

$$\text{ctxIdxInc}(\text{ctxBlockCat}) = \text{condTermFlagA} + 2 * \text{condTermFlagB} \quad (9-13)$$

9.3.3.1.2 Processus d'allocation de ctxIdxInc utilisant les valeurs de segments précédemment décodés

Les entrées pour ce processus sont ctxIdxOffset et binIdx.

Le résultat de ce processus est `ctxIdxInc`.

Le Tableau 9-31 contient la spécification de `ctxIdxInc` pour les valeurs données de `ctxIdxOffset` et `binIdx`.

Pour chaque valeur de `ctxIdxOffset` et `binIdx`, `ctxIdxInc` est calculé en utilisant certaines des valeurs des segments précédemment décodés ($b_0, b_1, b_2, \dots, b_k$), où la valeur de l'indice k est inférieure à la valeur de `binIdx`.

Tableau 9-31 – Spécification de `ctxIdxInc` pour des valeurs spécifiques de `ctxIdxOffset` et `binIdx`

Valeur (nom) de <code>ctxIdxOffset</code>	<code>binIdx</code>	<code>ctxIdxInc</code>
3	4	$(b_3 \neq 0) ? 5 : 6$
	5	$(b_3 \neq 0) ? 6 : 7$
14	2	$(b_1 \neq 1) ? 2 : 3$
17	4	$(b_3 \neq 0) ? 2 : 3$
27	2	$(b_1 \neq 0) ? 4 : 5$
32	4	$(b_3 \neq 0) ? 2 : 3$
36	2	$(b_1 \neq 0) ? 2 : 3$

9.3.3.1.3 Processus d'allocation de `ctxIdxInc` pour les éléments syntaxiques `significant_coeff_flag`, `last_significant_coeff_flag`, et `coeff_abs_level_minus1`

Les entrées pour ce processus sont `ctxIdxOffset` et `binIdx`.

Le résultat de ce processus est `ctxIdxInc`.

Le processus d'allocation de `ctxIdxInc` pour les éléments syntaxiques `significant_coeff_flag`, `last_significant_coeff_flag`, et `coeff_abs_level_minus1` ainsi que pour `coded_block_flag` dépend des catégories des différents blocs mentionnés par la variable `ctxBlockCat`. La spécification des catégories de ces blocs est donnée au Tableau 9-32.

Tableau 9-32 – Spécification de `ctxBlockCat` pour les différents blocs

Description du bloc	<code>maxNumCoeff</code>	<code>ctxBlockCat</code>
Bloc de niveaux de coefficient de transformée luma DC (pour macrobloc codé en mode de prédiction <code>Intra_16x16</code>)	16	0
Bloc de niveaux de coefficient de transformée luma AC (pour macrobloc codé en mode de prédiction <code>Intra_16x16</code>)	15	1
Bloc de niveaux de coefficient de transformée luma (pour macrobloc non codé en mode de prédiction <code>Intra_16x16</code>)	16	2
Bloc de niveaux de coefficient de transformée chroma DC	4	3
Bloc de niveaux de coefficient de transformée chroma AC	15	4

Pour les éléments syntaxiques `significant_coeff_flag` et `last_significant_coeff_flag` la position de balayage `scanningPos` au sein du bloc considéré est allouée à `ctxIdxInc`, où `scanningPos` va de 0 à `maxNumCoeff - 2`, inclus:

$$\text{ctxIdxInc} = \text{scanningPos} \quad (9-14)$$

La position de balayage pour les blocs codés de trame se rapporte au balayage en zigzag; la position de balayage pour les blocs codés de champ se rapporte au balayage de champ.

Soit numDecodAbsLevelEq1 la notation du nombre accumulé de niveaux décodés de coefficient de transformée avec une valeur absolue égale à 1, et soit numDecodAbsLevelGt1 la notation du nombre accumulé de niveaux décodés de coefficient de transformée avec une valeur absolue supérieure à 1. Les deux nombres se rapportent au même bloc de coefficient de transformée, dans lequel s'effectue le processus de décodage en cours. Puis, pour le décodage de coeff_abs_level_minus1, ctxIdxInc pour coeff_abs_level_minus1 est spécifié en fonction de binIdx comme suit:

– Si binIdx est égal à 0, ctxIdxInc est calculé par

$$\text{ctxIdxInc} = ((\text{numDecodAbsLevelGt1} \neq 0) ? 0 : \text{Min}(4, 1 + \text{numDecodAbsLevelEq1})) \quad (9-15)$$

– Autrement (binIdx est supérieur à 0), ctxIdxInc est calculé par

$$\text{ctxIdxInc} = 5 + \text{Min}(4, \text{numDecodAbsLevelGt1}) \quad (9-16)$$

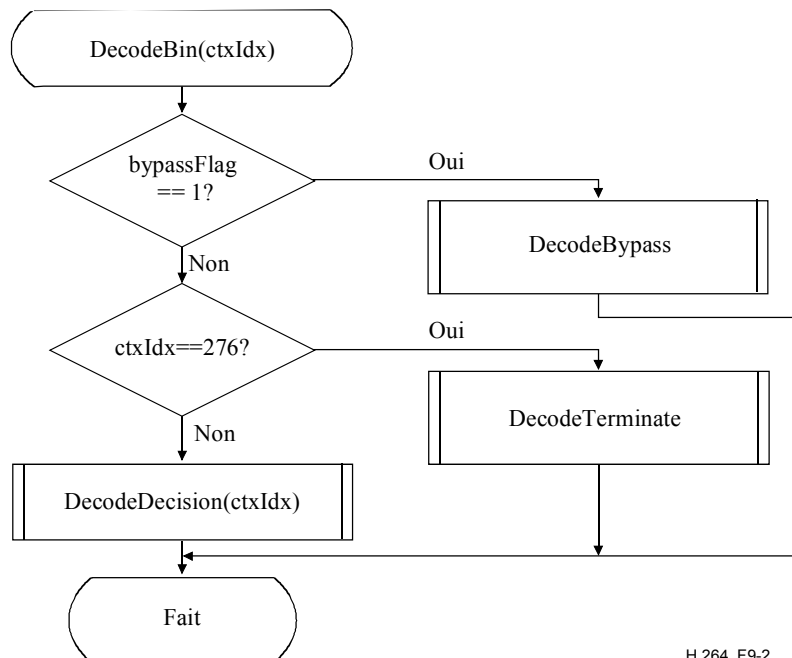
9.3.3.2 Processus de décodage arithmétique

Les entrées pour ce processus sont le bypassFlag, ctxIdx comme calculé au § 9.3.3.1, et les variables d'état codIRange et codIOffset du moteur de décodage arithmétique.

Le résultat de ce processus est la valeur du segment.

La Figure 9-2 illustre l'ensemble du processus de décodage arithmétique pour un seul segment. Pour décoder la valeur d'un segment, l'indice de contexte ctxIdx est passé au processus de décodage arithmétique DecodeBin(ctxIdx), qui est spécifié comme suit.

- Si bypassFlag est égal à 1, DecodeBypass() comme spécifié au § 9.3.3.2.3 est invoqué.
- Autrement, si bypassFlag est égal à 0 et que ctxIdx est égal à 276, DecodeTerminate() comme spécifié au § 9.3.3.2.4 est invoqué.
- Autrement (bypassFlag est égal à 0 et ctxIdx n'est pas égal à 276), DecodeDecision() comme spécifié au § 9.3.3.2.1 doit s'appliquer.



H.264_F9-2

Figure 9-2 – Aperçu général du processus de décodage arithmétique pour un seul segment (pour information)

NOTE – Le codage arithmétique se fonde sur le principe d'une subdivision à intervalle récursif. Etant donné une estimation de probabilité $p(0)$ et $p(1) = 1 - p(0)$ d'une décision binaire $(0, 1)$, un sous-intervalle de code donné initialement avec la gamme `codIRange` sera subdivisé en deux sous-intervalles ayant respectivement la gamme $p(0) * \text{codIRange}$ et $\text{codIRange} - p(0) * \text{codIRange}$. En fonction de la décision, qui a été observée, le sous-intervalle correspondant sera choisi comme le nouvel intervalle de code, et une chaîne de code binaire pointant dans cet intervalle représentera la séquence de décisions binaires observées. Il est utile de distinguer entre le symbole le plus probable (MPS, *most probable symbol*) et le symbole le moins probable (LPS, *least probable symbol*), de sorte que les décisions binaires doivent être identifiées comme MPS ou LPS, plutôt que comme 0 ou 1. Etant donné cette terminologie, chaque contexte est spécifié par la probabilité p_{LPS} du LPS et la valeur de MPS (`valMPS`), qui est 0 ou 1.

Le moteur de codage arithmétique de la présente Recommandation | Norme internationale a trois propriétés distinctes:

- l'estimation de probabilité est effectuée au moyen d'une machine à états finis avec un processus de transition fondé sur un tableau entre 64 états de probabilité représentatifs différents $\{ p_{\text{LPS}}(\text{pStateIdx}) \mid 0 \leq \text{pStateIdx} < 64 \}$ pour la probabilité LPS p_{LPS} . La numérotation des états est arrangée de telle sorte que l'état de probabilité avec l'indice $\text{pStateIdx} = 0$ correspond à une valeur de probabilité LPS de 0,5, avec une probabilité LPS décroissante vers les indices d'état les plus élevés;
- la gamme de `codIRange` représentant l'état du moteur de codification est quantifiée par un petit ensemble $\{Q_1, \dots, Q_4\}$ de valeurs de quantification préétablies avant le calcul de la nouvelle gamme d'intervalles. L'enregistrement d'un tableau contenant toutes les valeurs de produit précalculées de 64×4 de $Q_i * p_{\text{LPS}}(\text{pStateIdx})$ permet une approximation sans multiplication du produit $\text{codIRange} * p_{\text{LPS}}(\text{pStateIdx})$;
- pour des éléments syntaxiques, ou leurs parties, pour lesquels une distribution de probabilité approximativement uniforme est supposée être donnée, on utilise un processus de dérivation séparé et simplifié de codage et décodage.

9.3.3.2.1 Processus de décodage arithmétique pour une décision binaire

Les entrées pour ce processus sont `ctxIdx`, `codIRange`, et `codIOffset`.

Les résultats de ce processus sont la valeur décodée `binVal`, et les variables mises à jour `codIRange` et `codIOffset`.

La Figure 9-3 indique le diagramme de décodage d'une décision unique (`DecodeDecision`).

1. La valeur de la variable `codIRangeLPS` est déduite comme suit.
 - Etant donné la valeur en cours de `codIRange`, la variable `qCodIRangeIdx` est calculée par

$$\text{qCodIRangeIdx} = (\text{codIRange} \gg 6) \& 0x03 \quad (9-17)$$
 - Etant donné `qCodIRangeIdx` et `pStateIdx` associés à `ctxIdx`, la valeur de la variable `rangeTabLPS` comme spécifié au Tableau 9-33 est allouée à `codIRangeLPS`:

$$\text{codIRangeLPS} = \text{rangeTabLPS}[\text{pStateIdx}][\text{qCodIRangeIdx}] \quad (9-18)$$
2. La variable `codIRange` est mise égale à `codIRange - codIRangeLPS` et on applique ce qui suit.
 - Si `codIOffset` est supérieur ou égal à `codIRange`, la variable `binVal` est mise égale à $1 - \text{valMPS}$, `codIOffset` est décrémenté de `codIRange`, et `codIRange` est mis égal à `codIRangeLPS`.
 - Autrement, la variable `binVal` est mise égale à `valMPS`.

Etant donné la valeur de `binVal`, la transition d'état est effectuée comme spécifié au § 9.3.3.2.1.1. En fonction de la valeur en cours de `codIRange`, la renormalisation est effectuée comme spécifié au § 9.3.3.2.2.

9.3.3.2.1.1 Processus de transition d'état

Les entrées pour ce processus sont le pStateIdx en cours, la valeur décodée binVal et les valeurs valMPS de la variable de contexte associée à ctxIdx.

Les résultats de ce processus sont les pStateIdx et valMPS mis à jour de la variable de contexte associée à ctxIdx.

En fonction de la valeur décodée binVal, la mise à jour des deux variables pStateIdx et valMPS associées à ctxIdx est déduite comme suit:

```

si( binVal == valMPS )
  pStateIdx = transIdxMPS( pStateIdx )
ou {
  si ( pStateIdx == 0 )
    valMPS = 1 - valMPS
    pStateIdx = transIdxLPS( pStateIdx )
}
    
```

(9-19)

Le Tableau 9-34 spécifie les règles de transition transIdxMPS() et transIdxLPS() après le décodage de la valeur de valMPS et 1 - valMPS, respectivement.

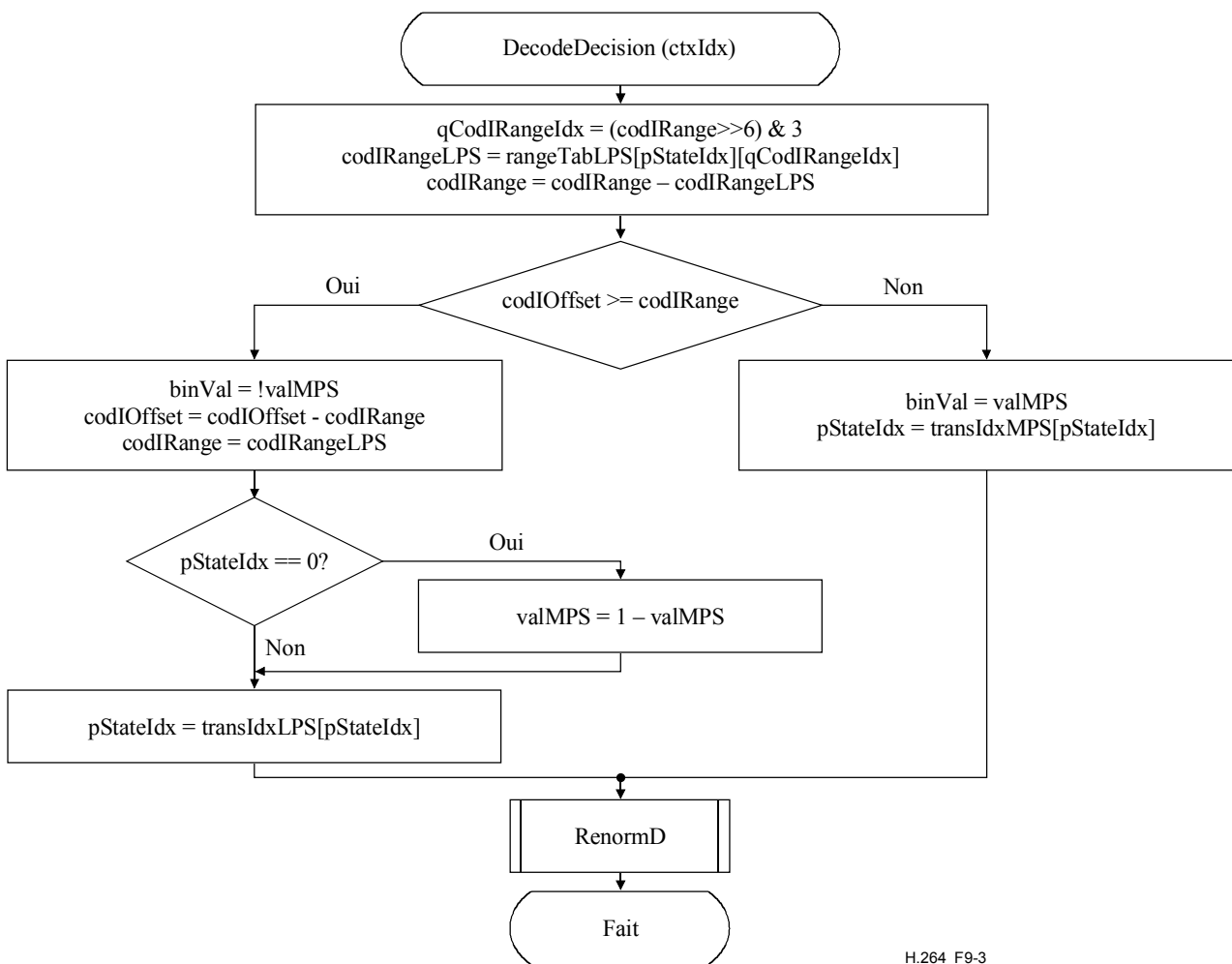


Figure 9-3 – Diagramme pour le décodage d'une décision

Tableau 9-33 – Spécification de rangeTabLPS en fonction de pStateIdx et qCodIRangeIdx

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

Tableau 9-34 – Tableau de transition d'état

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 Processus de renormalisation dans le moteur de décodage arithmétique

Les entrées pour ce processus sont des bits provenant de données de tranche et les variables codIRange et codIOffset.

Les résultats de ce processus sont les variables mises à jour codIRange et codIOffset.

Un diagramme de la renormalisation est donné à la Figure 9-4. La valeur en cours de codIRange est d'abord comparée à 0x0100 et les étapes suivantes sont spécifiées comme suit.

- Si codIRange est supérieur ou égal à 0x0100, aucune renormalisation n'est nécessaire et le processus RenormD est terminé.
- Autrement (codIRange est inférieur à 0x0100), on entre dans la boucle de renormalisation. Dans cette boucle, la valeur de codIRange est doublée, c'est-à-dire décalée à gauche de 1 et un seul bit est déplacé dans codIOffset en utilisant read_bits(1).

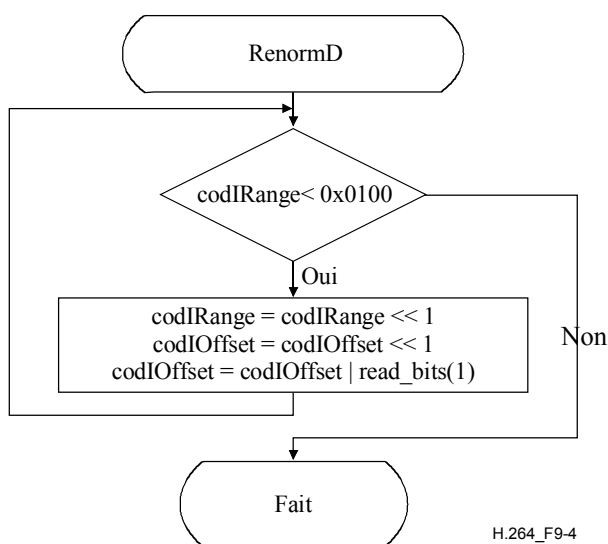


Figure 9-4 – Diagramme de renormalisation

9.3.3.2.3 Processus de décodage d'aiguillage pour décisions binaires

Les entrées pour ce processus sont des bits provenant de données de tranche et les variables `codIRange` et `codIOffset`.

Les résultats de ce processus sont la variable mise à jour `codIOffset` et la valeur décodée `binVal`.

Le processus de décodage d'aiguillage est invoqué lorsque `bypassFlag` est égal à 1. La Figure 9-5 donne un diagramme du processus correspondant.

D'abord, la valeur de `codIOffset` est doublée, c'est-à-dire, décalée à gauche de 1 et un seul bit est déplacé dans `codIOffset` en utilisant `read_bits(1)`. Puis, la valeur de `codIOffset` est comparée à la valeur de `codIRange` et les étapes suivantes sont spécifiées comme suit.

- Si `codIOffset` est supérieur ou égal à `codIRange`, la variable `binVal` est mise égale à 1 et `codIOffset` est décrémentée de `codIRange`.
- Autrement (`codIOffset` est inférieur à `codIRange`), la variable `binVal` est mise égale à 0.

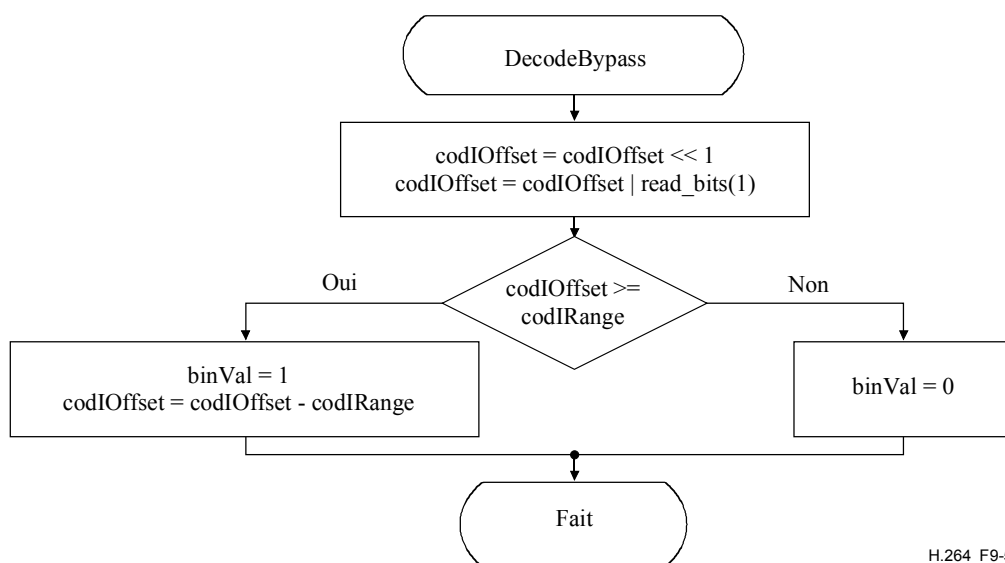


Figure 9-5 – Diagramme du processus de décodage d'aiguillage

9.3.3.2.4 Processus de décodage pour décisions binaires avant achèvement

Les entrées pour ce processus sont des bits provenant de données de tranche et les variables `codIRange` et `codIOffset`.

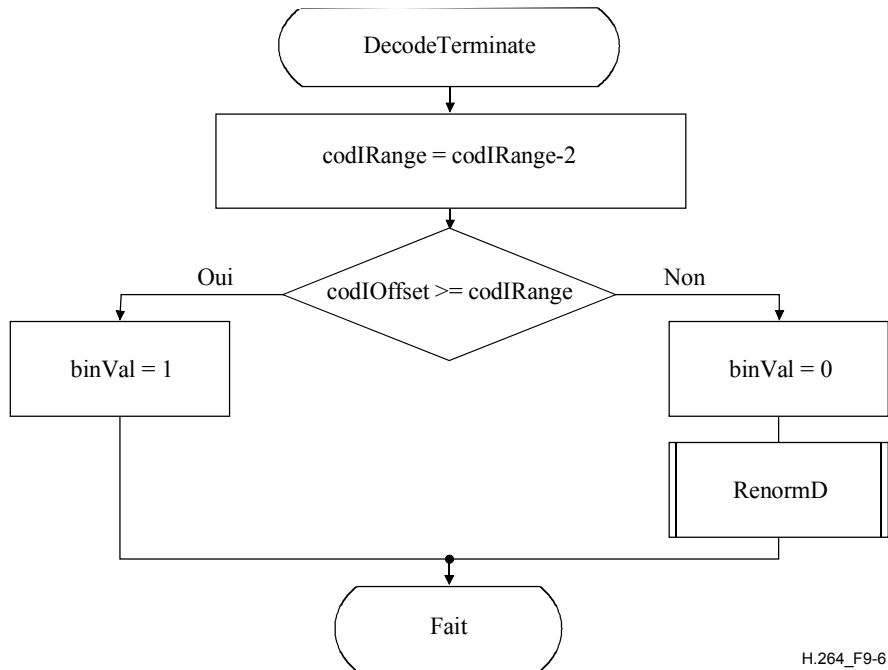
Les résultats de ce processus sont les variables mises à jour `codIRange` et `codIOffset`, et la valeur décodée `binVal`.

Ce programme spécial de décodage s'applique au décodage de `end_of_slice_flag` (*fanion de fin de tranche*) et du segment indiquant le mode `I_PCM` correspondant à `ctxIdx` égal à 276. La Figure 9-6 donne le diagramme du processus de décodage correspondant, qui est spécifié comme suit.

D'abord, la valeur de `codIRange` est décrémentée de 2. Puis, la valeur de `codIOffset` est comparée à la valeur de `codIRange` et les étapes suivantes sont spécifiées comme suit:

- Si `codIOffset` est supérieur ou égal à `codIRange`, la variable `binVal` est mise égale à 1, aucune renormalisation n'est effectuée, et le décodage CABAC s'achève. Le dernier bit inséré dans le registre `codIOffset` est égal à 1. Lors du décodage de `end_of_slice_flag`, ce dernier bit inséré dans le registre `codIOffset` est interprété comme `rbsp_stop_one_bit`.
- Autrement (`codIOffset` est inférieur à `codIRange`), la variable `binVal` est mise égale à 0 et la renormalisation est effectuée comme spécifié au § 9.3.3.2.2.

NOTE – Cette procédure peut aussi être implémentée en utilisant DecodeDecision(ctxIdx) avec $\text{ctxIdx} = 276$. Dans le cas où la valeur décodée est égale à 1, sept bits de plus seraient lus par DecodeDecision(ctxIdx) et un processus de décodage aurait à ajuster en conséquence son pointeur de flux binaire pour décoder correctement les éléments syntaxiques suivants.



H.264_F9-6

Figure 9-6 – Diagramme de décodage d'une décision avant achèvement

9.3.4 Processus de codage arithmétique (pour information)

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

Les entrées pour ce processus sont les décisions qui sont à coder et à écrire.

Les résultats de ce processus sont les bits qui sont écrits à la charge utile RBSP.

Ce paragraphe pour information décrit un moteur de codage arithmétique qui correspond au moteur de décodage arithmétique décrit au § 9.3.3.2. Le moteur de codage est essentiellement symétrique du moteur de décodage, c'est-à-dire que les procédures sont appelées dans le même ordre. Les procédures suivantes sont décrites dans la présente section: InitEncoder, EncodeDecision, EncodeBypass, EncodeTerminate, qui correspondent respectivement à InitDécodeur, DecodeDecision, DecodeBypass, et DecodeTerminate. L'état du moteur arithmétique de codage est représenté par une valeur de la variable codILow pointant sur l'extrémité inférieure d'un sous-intervalle et une valeur de la variable codIRange spécifiant la gamme correspondante de ce sous-intervalle.

9.3.4.1 Processus d'initialisation pour le moteur arithmétique de codage (pour information)

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

Ce processus est invoqué avant le codage du premier macrobloc d'une tranche, et après le codage de $\text{pcm_alignment_zero_bit}$ et de toutes les données pcm_byte pour un macrobloc de type I_PCM.

Les résultats de ce processus sont les valeurs codILow , codIRange , firstBitFlag , bitsOutstanding , et symCnt du moteur de codage arithmétique.

Dans la procédure d'initialisation de l'encodeur, `codILow` est mis égal à 0, et `codIRange` est mis égal à 0x01FE. De plus, un `firstBitFlag` est mis égal à 1, et les compteurs `bitsOutstanding` et `symCnt` sont mis égaux à 0.

NOTE – La précision minimale de registre requise pour `codILow` est de 10 bits et pour `CodIRange` elle est de 9 bits. La précision requise pour les compteurs `bitsOutstanding` et `symCnt` devrait être suffisamment grande pour empêcher le débordement des registres en question. Lorsque `MaxBinCountInSlice` note le nombre total maximal de décisions binaires à coder en une tranche, la précision minimale de registre requise pour les variables `bitsOutstanding` et `symCnt` est donnée par $\text{Ceil}(\text{Log}_2(\text{MaxBinCountInSlice} + 1))$.

9.3.4.2 Processus de codage pour une décision binaire (pour information)

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

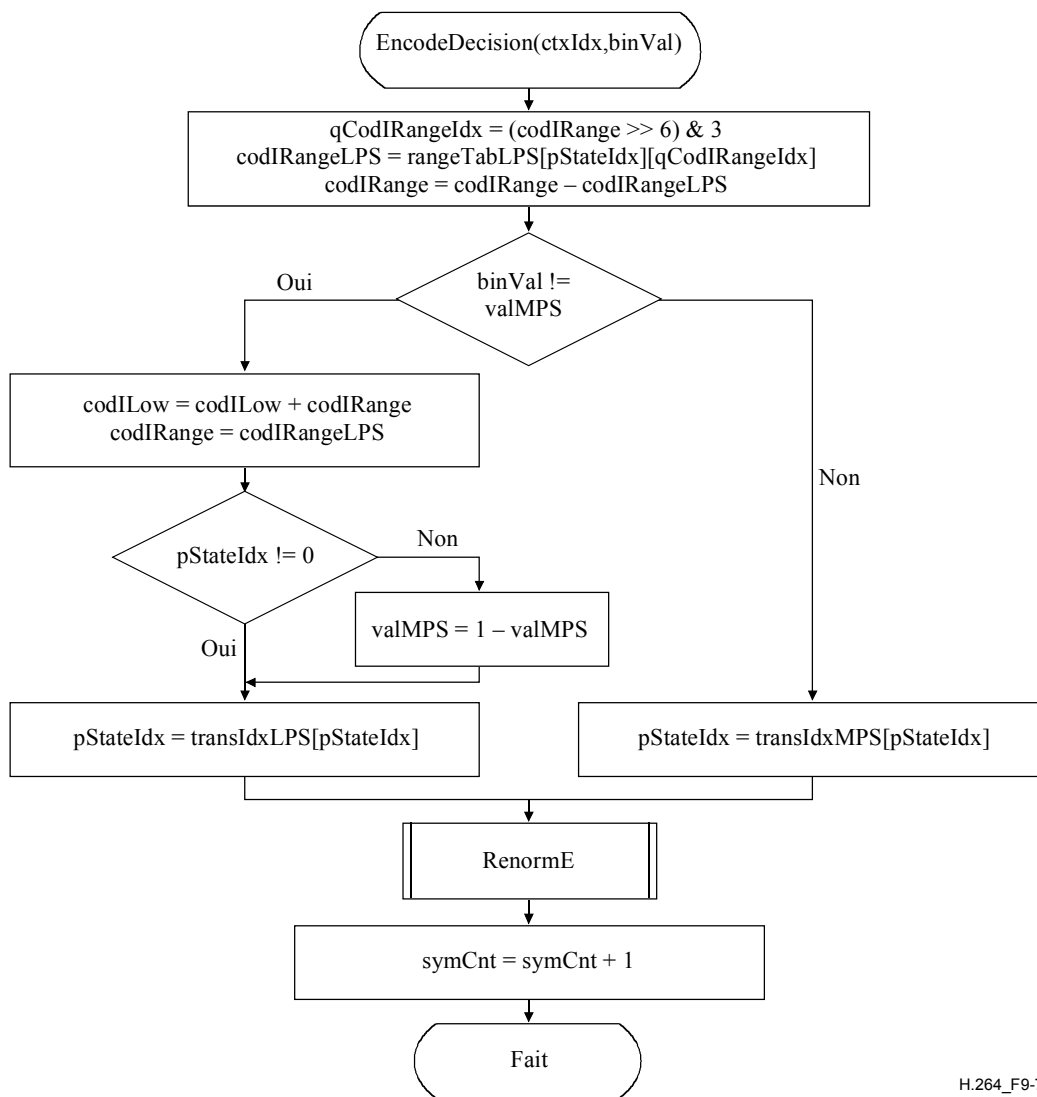
Les entrées pour ce processus sont l'indice de contexte `ctxIdx`, la valeur de `binVal` à coder, et les variables `codIRange`, `codILow` et `symCnt`.

Les résultats de ce processus sont les variables `codIRange`, `codILow`, et `symCnt`.

La Figure 9-7 donne le diagramme pour le codage d'une seule décision. Dans une première étape, la variable `codIRangeLPS` est déduite comme suit.

Etant donné la valeur en cours de `codIRange`, `codIRange` est mappé avec l'indice `qCodIRangeIdx` d'une valeur quantifiée de `codIRange` en utilisant l'équation 9-17. La valeur de `qCodIRangeIdx` et la valeur de `pStateIdx` associées à `ctxIdx` sont utilisées pour déterminer la valeur de la variable `rangeTabLPS` comme spécifié au Tableau 9-33, qui est alloué à `codIRangeLPS`. La valeur de `codIRange - codIRangeLPS` est allouée à `codIRange`.

Dans une seconde étape, la valeur de `binVal` est comparée à `valMPS` associé à `ctxIdx`. Lorsque `binVal` est différent de `valMPS`, `codIRange` est ajouté à `codILow` et `codIRange` est mis égal à la valeur `codIRangeLPS`. Etant donné la décision codée, la transition d'état est effectuée comme spécifié au § 9.3.3.2.1.1. En fonction de la valeur en cours de `codIRange`, la renormalisation est effectuée comme spécifié au § 9.3.4.3. Finalement, la variable `symCnt` est incrémentée de 1.



H.264_F9-7

Figure 9-7 – Diagramme pour le codage d'une décision

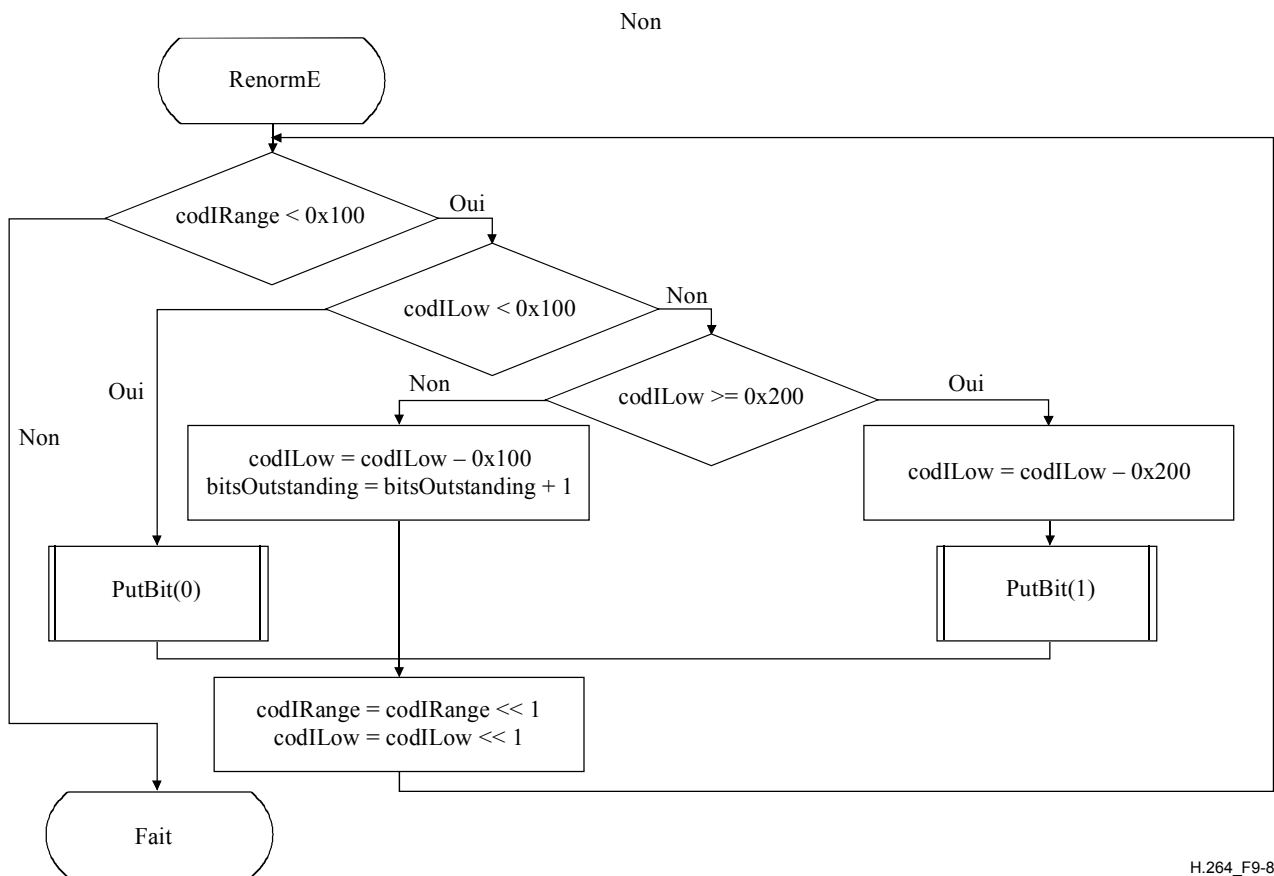
9.3.4.3 Processus de renormalisation dans le moteur de codage arithmétique (pour information)

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

Les entrées pour ce processus sont les variables `codIRange`, `codILow`, `firstBitFlag`, et `bitsOutstanding`.

Les résultats de ce processus sont zéro ou plusieurs bits écrits dans la charge utile RBSP et les variables mises à jour `codIRange`, `codILow`, `firstBitFlag`, et `bitsOutstanding`.

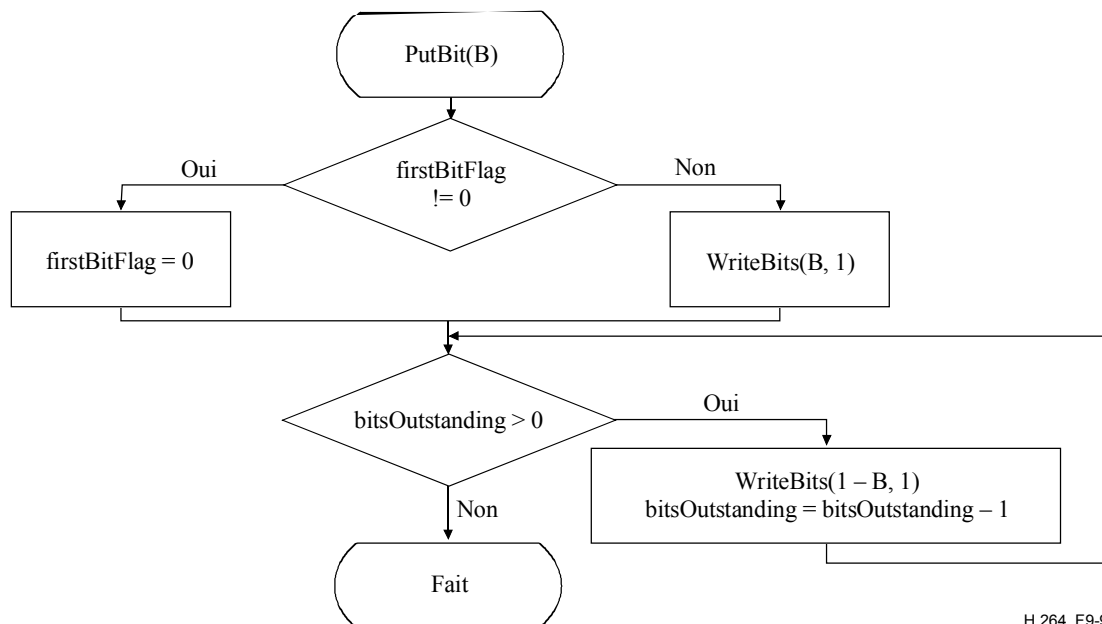
La renormalisation est illustrée à la Figure 9-8.



H.264_F9-8

Figure 9-8 – Diagramme de la renormalisation dans le codeur

La procédure PutBit() décrite à la Figure 9-9 donne la commande du report. Elle utilise la fonction WriteBits(B, N) qui écrit N bits avec la valeur B au flux binaire et avance le pointeur du flux binaire de N positions de bit. Cette fonction suppose l'existence d'un pointeur de flux binaire avec une indication de la position du prochain bit que le processus de codage doit écrire au flux binaire.



H.264_F9-9

Figure 9-9 – Diagramme de PutBit(B)

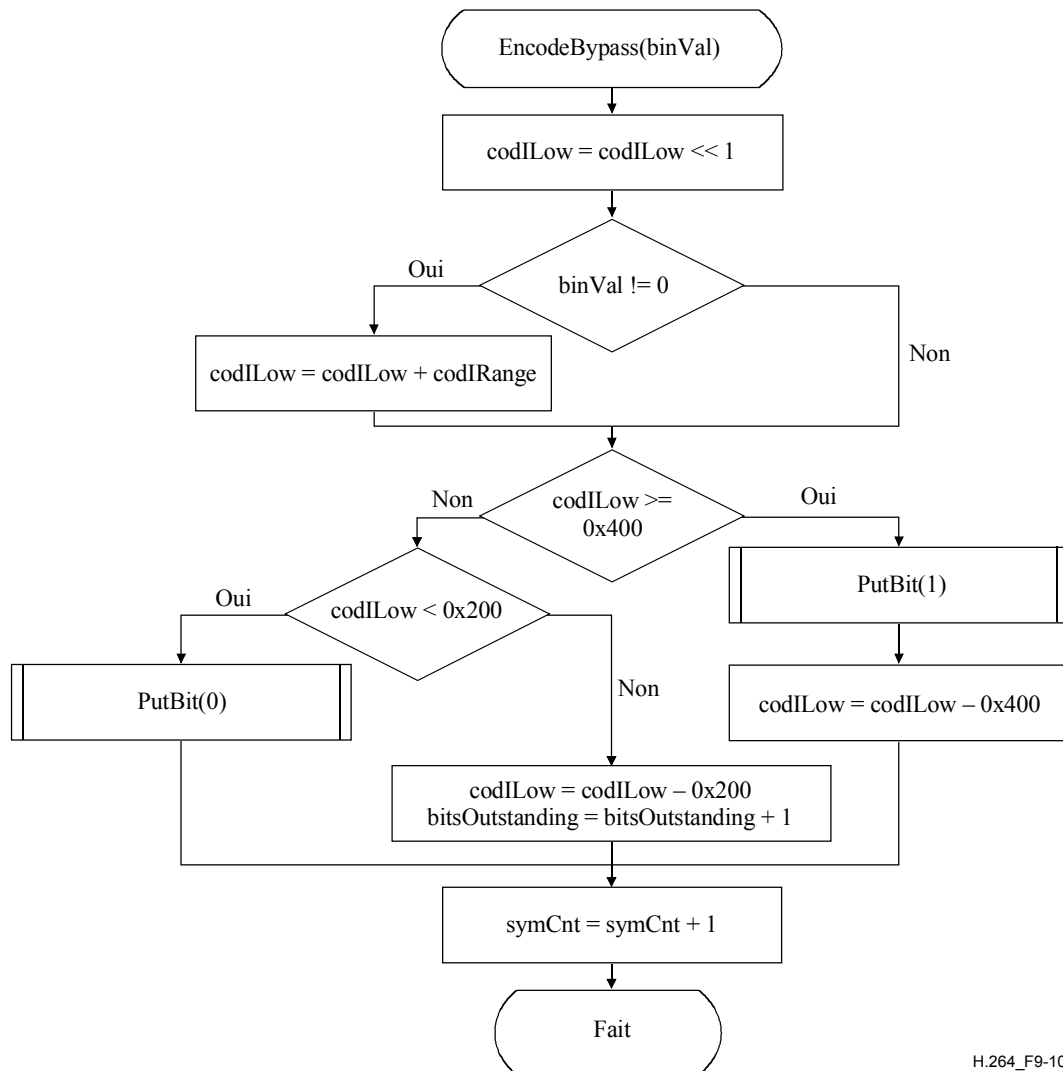
9.3.4.4 Processus de codage d'aiguillage pour décisions binaires (pour information)

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

Les entrées pour ce processus sont les variables binVal, codILow, codIRange, bitsOutstanding, et symCnt.

Le résultat de ce processus est un bit écrit à la charge utile RBSP et les variables mises à jour codILow, codIRange, bitsOutstanding, et symCnt.

Ce processus de codage s'applique à toutes les décisions binaires avec bypassFlag égal à 1. La renormalisation est incluse dans la spécification de ce processus comme indiqué à la Figure 9-10.



H.264_F9-10

Figure 9-10 – Diagramme de l'aiguillage de codage

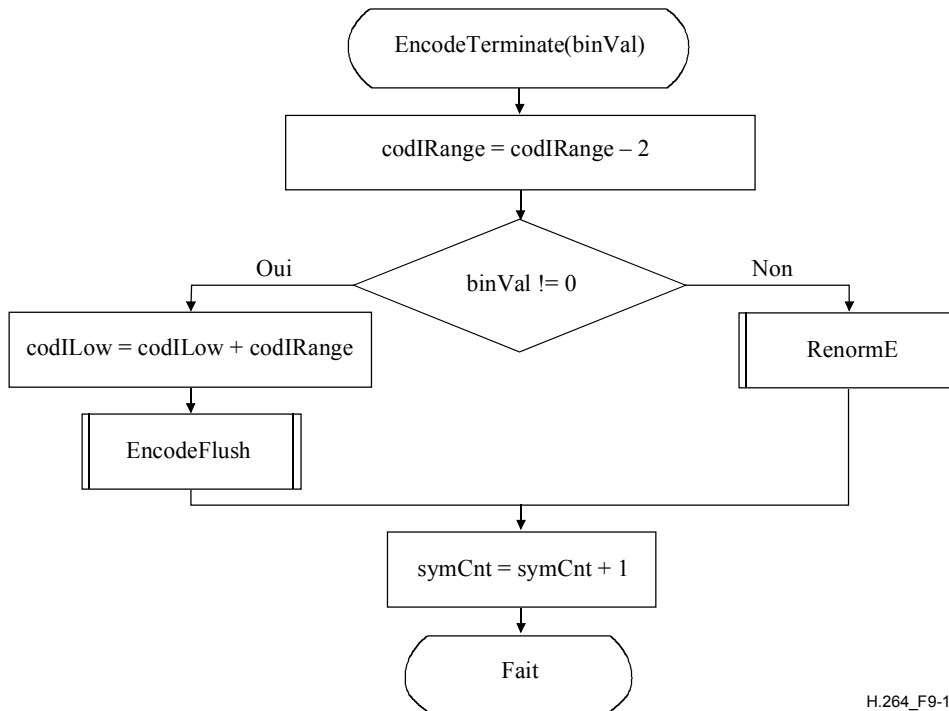
9.3.4.5 Processus de codage pour une décision binaire avant achèvement (pour information)

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

Les entrées pour ce processus sont les variables binVal, codIRange, codILow, bitsOutstanding, et symCnt.

Les résultats de ce processus sont zéro ou plusieurs bits écrits à la charge utile RBSP et les variables mises à jour `codILow`, `codIRange`, `bitsOutstanding`, et `symCnt`.

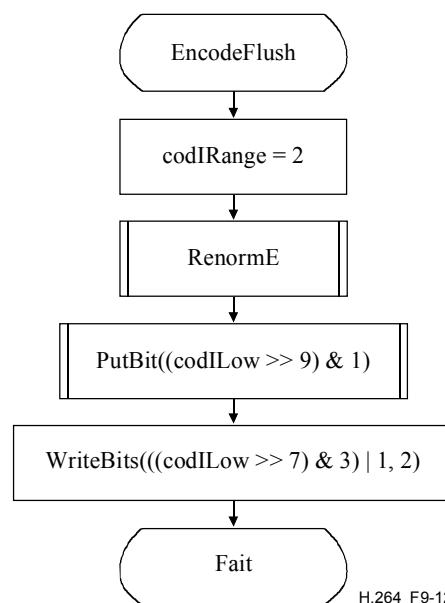
Ce programme de codage indiqué à la Figure 9-11 s'applique au codage de `end_of_slice_flag` et du segment indiquant le `mb_type I_PCM` tous deux associés à `ctxIdx` égal à 276.



H.264_F9-11

Figure 9-11 – Diagramme de codage d'une décision avant achèvement

Lorsque la valeur de `binVal` à coder est égale à 1, le codage CABAC est terminé et on applique la procédure de vidage indiquée à la Figure 9-12. Dans cette procédure de vidage, le dernier bit écrit par `WriteBits (B, N)` est égal à 1. Lors du codage de `end_of_slice_flag`, ce dernier bit est interprété comme le `rbstop_one_bit`.



H.264_F9-12

Figure 9-12 – Diagramme du vidage à achèvement

9.3.4.6 Processus de bourrage d'octet (pour information)

Le présent paragraphe ne constitue pas une partie intégrante de la présente Recommandation | Norme internationale.

Ce processus est invoqué après le codage du dernier macrobloc de la dernière tranche d'une image et après encapsulation.

Les entrées pour ce processus sont le nombre d'octets NumBytesInVclNALunits de toutes les unités NAL VCL d'une image, le nombre de macroblocs PicSizeInMbs dans l'image, et le nombre de symboles binaires BinCountsInNALunits résultant du codage du contenu de toutes les unités NAL VCL de l'image.

Les résultats de ce processus sont zéro ou plusieurs octets accolés à l'unité NAL.

Soit la variable k qui est mise égale à $\text{Ceil}(\text{Ceil}(3 * \text{BinCountsInNALunits} - 3 * 96 * \text{PicSizeInMbs})/32) - \text{NumBytesInVclNALunits} / 3$. En fonction de la variable k on applique ce qui suit.

- Si k est inférieure ou égale à 0, aucun cabac_zero_word n'est accolé à l'unité NAL.
- Autrement (k est supérieur à 0), la séquence 0x000003 de trois octets est accolée k fois à l'unité NAL après encapsulation, où les deux premiers octets 0x0000 représentent un cabac_zero_word et le troisième octet 0x03 représente un emulation_prevention_three_byte.

Annexe A

Profils et niveaux

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

Les profils et niveaux spécifient des restrictions sur les flux binaires et donc des limites sur les capacités nécessaires pour décoder les flux binaires. Les profils et niveaux peuvent aussi être utilisés pour indiquer les points d'interfonctionnement entre les implémentations de décodeurs individuels.

NOTE – La présente Recommandation | Norme internationale n'inclut pas d'"options" individuelles à choisir pour le décodeur, car cela augmenterait les difficultés d'interfonctionnement.

Chaque profil spécifie un sous-ensemble de caractéristiques d'algorithme et de limites qui doivent être acceptées par tous les décodeurs conformes à ce profil.

NOTE – Les codeurs ne sont pas tenus de faire usage d'un sous-ensemble particulier de caractéristiques constituant un profil.

Chaque niveau spécifie un ensemble de limites sur les valeurs que peuvent prendre les éléments syntaxiques de la présente Recommandation | Norme internationale. Le même ensemble de définitions de niveau est utilisé avec tous les profils, mais des implémentations individuelles peuvent accepter un niveau différent pour chaque profil accepté. Pour tout profil donné, les niveaux correspondent généralement à la charge de traitement du décodeur et à la capacité de mémorisation.

A.1 Exigences sur la capacité du décodeur vidéo

Les capacités des décodeurs vidéo conformes à la présente Recommandation | Norme internationale sont spécifiées en termes de capacité à décoder les flux vidéo conformes aux contraintes des profils et niveaux spécifiés dans la présente annexe. Pour chacun de ces profils, le niveau accepté pour ce profil doit aussi être exprimé.

Des valeurs spécifiques sont spécifiées dans la présente annexe pour les éléments syntaxiques `profile_idc` et `level_idc`. Toutes les autres valeurs de `profile_idc` et `level_idc` sont réservées pour être utilisées à l'avenir par l'UIT-T | ISO/CEI.

NOTE – Les décodeurs ne devraient pas conclure que, lorsqu'une valeur réservée de `profile_idc` ou `level_idc` tombe entre les valeurs spécifiées dans la présente Recommandation | Norme internationale, cela indique des capacités intermédiaires entre les profils ou niveaux spécifiés, car il n'y a pas de restrictions sur la méthode que choisira l'UIT-T | ISO/CEI pour l'utilisation de telles valeurs réservées pour l'avenir.

A.2 Profils

A.2.1 Profil de base

Les flux binaires conformes au profil de base doivent respecter les contraintes suivantes:

- Seuls les types de tranche I et P peuvent être présents.
- Les flux d'unités NAL ne doivent pas contenir de valeurs de `nal_unit_type` dans la gamme de 2 à 4, inclus.
- Les ensembles de paramètres de séquence doivent avoir `frame_mbs_only_flag` (*fanion de macroblocs de trame uniquement*) égal à 1.
- Les ensembles de paramètres d'image doivent avoir `weighted_pred_flag` (*fanion de prédiction pondérée*) et `weighted_bipred_idc` (*indice de biprédiction pondérée*) égaux tous deux à 0.
- Les ensembles de paramètres d'image doivent avoir `entropy_coding_mode_flag` (*fanion de mode de codage entropique*) égal à 0.

- Les ensembles de paramètres d'image doivent avoir `num_slice_groups_minus1` (*numéro de groupes de tranche moins 1*) dans la gamme de 0 à 7, inclus.
- La contrainte de niveau spécifiée pour le profil de base au § A.3 doit être respectée.

La conformité d'un flux binaire au profil de base est spécifiée par `profile_idc` égal à 66.

Les décodeurs conformes au profil de base à un niveau spécifique doivent être capables de décoder tous les flux binaires dans lesquels `profile_idc` est égal à 66 ou dans lesquels `constraint_set0_flag` est égal à 1 et dans lesquels `level_idc` représente un niveau inférieur ou égal au niveau spécifié.

A.2.2 Profil principal

Les flux binaires conformes au profil principal doivent respecter les contraintes suivantes:

- seuls les types de tranche I, P, et B peuvent être présents;
- les flux d'unités NAL ne doivent pas contenir de valeurs de `nal_unit_type` dans la gamme de 2 à 4 inclus;
- l'ordre arbitraire de tranche n'est pas admis;
- les ensembles de paramètres d'image doivent avoir `num_slice_groups_minus1` égal à 0 uniquement;
- les ensembles de paramètres d'image doivent avoir `redundant_pic_cnt_present_flag` égal à 0 uniquement;
- la contrainte de niveau spécifiée pour le profil principal au § A.3 doit être respectée.

La conformité d'un flux binaire au profil principal est spécifiée par `profile_idc` égal à 77.

Les décodeurs conformes au profil principal à un niveau spécifié doivent être capables de décoder tous les flux binaires dans lesquels `profile_idc` est égal à 77 ou dans lesquels `constraint_set1_flag` est égal à 1 et dans lesquels `level_idc` représente un niveau inférieur ou égal au niveau spécifié.

A.2.3 Profil étendu

Les flux binaires conformes au profil étendu doivent respecter les contraintes suivantes:

- les ensembles de paramètres de séquence doivent avoir `direct_8x8_inference_flag` égal à 1;
- les ensembles de paramètres d'image doivent avoir `entropy_coding_mode_flag` égal à 0;
- les ensembles de paramètres d'image doivent avoir `num_slice_groups_minus1` dans la gamme de 0 à 7 inclus;
- la contrainte de niveau spécifiée pour le profil étendu au § A.3 doit être respectée.

La conformité d'un flux binaire au profil étendu est spécifiée par `profile_idc` égal à 88.

Les décodeurs conformes au profil étendu à un niveau spécifié doivent être capables de décoder tous les flux binaires dans lesquels `profile_idc` est égal à 88 ou dans lesquels `constraint_set2_flag` est égal à 1 et dans lesquels `level_idc` représente un niveau inférieur ou égal au niveau spécifié.

Les décodeurs conformes au profil étendu à un niveau spécifié doivent être capables de décoder tous les flux binaires dans lesquels `profile_idc` est égal à 66 ou dans lesquels `constraint_set0_flag` est égal à 1, dans lesquels `level_idc` représente un niveau inférieur ou égal au niveau spécifié.

A.3 Niveaux

On spécifie ce qui suit pour exprimer les contraintes de la présente annexe.

- Soit l'unité d'accès n la $n^{\text{ième}}$ unité d'accès dans l'ordre de décodage, la première unité d'accès étant l'unité d'accès 0.
- Soit l'image n l'image codée primaire ou l'image décodée correspondante de l'unité d'accès n .

A.3.1 Limites de niveau indépendantes du profil

Soit la variable fR calculée comme suit.

- Si l'image n est une trame, fR est mise égale à $1 \div 172$.
- Autrement (l'image n est un champ), fR est mise égale à $1 \div (172 * 2)$.

Les flux binaires conformes à tout profil à un niveau spécifié doivent respecter les contraintes suivantes:

- a) Le délai nominal de retrait de l'unité d'accès n (avec $n > 0$) de la mémoire tampon CPB, comme spécifié au § C.1.2, satisfait à la contrainte que $t_{r,n}(n) - t_r(n-1)$ soit supérieur ou égal à $\text{Max}(\text{PicSizeInMbs} \div \text{MaxMBPS}, fR)$, où MaxMBPS est la valeur spécifiée au Tableau A-1 qui s'applique à l'image n , et PicSizeInMbs est le nombre de macroblocs dans l'image n .
- b) La différence entre les temps consécutifs de sortie des images de la mémoire tampon DPB, comme spécifié au § C.2.2, satisfait à la contrainte que $\Delta t_{o,dpb}(n) \geq \text{Max}(\text{PicSizeInMbs} \div \text{MaxMBPS}, fR)$, où MaxMBPS est la valeur spécifiée au Tableau A-1 pour l'image n , et PicSizeInMbs est le nombre de macroblocs de l'image n , pourvu que l'image n soit une image qui soit sortie et ne soit pas la dernière image du flux binaire qui est sorti.
- c) La somme des variables NumBytesInNALunit pour l'unité d'accès 0 est inférieure ou égale à $256 * \text{ChromaFormatFactor} * (\text{PicSizeInMbs} + \text{MaxMBPS} * (t_r(0) - t_{r,n}(0))) \div \text{MinCR}$, où MaxMBPS et MinCR sont les valeurs spécifiées au Tableau A-1 qui s'appliquent à l'image 0 et PicSizeInMbs est le nombre de macroblocs dans l'image 0.
- d) La somme des variables NumBytesInNALunit pour l'unité d'accès n (avec $n > 0$) est inférieure ou égale à $256 * \text{ChromaFormatFactor} * \text{MaxMBPS} * (t_r(n) - t_r(n-1)) \div \text{MinCR}$, où MaxMBPS et MinCR sont les valeurs spécifiées au Tableau A-1 qui s'appliquent à l'image n .
- e) $\text{PicWidthInMbs} * \text{FrameHeightInMbs} \leq \text{MaxFS}$, où MaxFS est spécifié au Tableau A-1.
- f) $\text{PicWidthInMbs} \leq \text{Sqrt}(\text{MaxFS} * 8)$.
- g) $\text{FrameHeightInMbs} \leq \text{Sqrt}(\text{MaxFS} * 8)$.
- h) $\text{max_dec_frame_buffering} \leq \text{MaxDpbSize}$, où MaxDpbSize est égal à $\text{Min}(1024 * \text{MaxDPB} / (\text{PicWidthInMbs} * \text{FrameHeightInMbs} * 256 * \text{ChromaFormatFactor}), 16)$ et MaxDPB est donné au Tableau A-1 en unités de 1024 octets. $\text{max_dec_frame_buffering}$ est aussi appelé taille DPB.
- i) Pour les paramètres VCL HRD, $\text{BitRate}[\text{SchedSelIdx}] \leq 1000 * \text{MaxBR}$ et $\text{CpbSize}[\text{SchedSelIdx}] \leq 1000 * \text{MaxCPB}$ pour au moins une valeur de SchedSelIdx , où $\text{BitRate}[\text{SchedSelIdx}]$ est donné par l'équation E-13 et $\text{CpbSize}[\text{SchedSelIdx}]$ est donné par l'équation E-14 lorsque $\text{vcl_hrd_parameters_present_flag}$ est égal à 1. MaxBR et MaxCPB sont spécifiés au Tableau A-1 en unités de 1000 bits/s et 1000 bits, respectivement. Le flux binaire doit satisfaire à ces conditions pour au moins une

- valeur de SchedSelIdx dans la gamme de 0 à `cpb_cnt_minus1`, inclus. `CpbSize[SchedSelIdx]` est aussi appelé taille CPB.
- j) Pour les paramètres NAL HRD, `BitRate[SchedSelIdx] ≤ 1200 * MaxBR` et `CpbSize[SchedSelIdx] ≤ 1200 * MaxCPB` pour au moins une valeur de SchedSelIdx, où `BitRate[SchedSelIdx]` est donné par l'équation E-13 et `CpbSize[SchedSelIdx]` est donné par l'équation E-14 lorsque `nal_hrd_parameters_present_flag` est égal à 1. MaxBR et MaxCPB sont spécifiés au Tableau A-1 en unités de 1200 bits/s et 1200 bits, respectivement. Le flux binaire doit satisfaire à ces conditions pour au moins une valeur de SchedSelIdx dans la gamme de 0 à `cpb_cnt_minus1`.
 - k) La gamme de composantes verticales de vecteur cinétique ne doit pas excéder MaxVmvR en unités d'échantillons de trame luma, où MaxVmvR est spécifié au Tableau A-1.
 - l) La gamme de vecteurs cinétiques horizontaux ne doit pas excéder la gamme de -2048 à 2047,75 inclus, en unités d'échantillons luma.
 - m) Le nombre de vecteurs cinétiques pour deux macroblocs consécutifs dans l'ordre de décodage (qui s'applique aussi au total depuis le dernier macrobloc d'une tranche et du premier macrobloc de la tranche suivante dans l'ordre de décodage) ne doit pas excéder MaxMvsPer2Mb, où MaxMvsPer2Mb est spécifié au Tableau A-1. Le nombre de vecteurs cinétiques pour chaque macrobloc correspond à la valeur de la variable MvCnt une fois achevé le processus d'intraprédiction ou d'interprédiction pour le macrobloc.
 - n) Le nombre de bits de données de `macrobloc_layer()` pour tout macrobloc n'est pas supérieur à $128 + 2048 * \text{ChromaFormatFactor}$. En fonction de `entropy_coding_mode_flag`, les bits de données de `macrobloc_layer()` sont comptés comme suit.
 - Si `entropy_coding_mode_flag` est égal à 0, le nombre de bits de données de `macrobloc_layer()` est donné par le nombre de bits dans la structure syntaxique de `macrobloc_layer()` pour un macrobloc.
 - Autrement (`entropy_coding_mode_flag` est égal à 1), le nombre de bits de données de `macrobloc_layer()` pour un macrobloc est donné par le nombre de fois que `read_bits(1)` est appelé aux § 9.3.3.2.2 et 9.3.3.2.3 lors de l'analyse grammaticale de `macrobloc_layer()` associé au macrobloc.

Le Tableau A-1 ci-dessous spécifie les limites pour chaque niveau. Les entrées marquées "-" dans le Tableau A-1 notent l'absence de limite correspondante.

La conformité à un niveau particulier doit être spécifiée en mettant l'élément syntaxique `level_idc` égal à une valeur de dix fois le numéro de niveau spécifié au Tableau A-1.

Tableau A-1 – Limites de niveau

Numéro de niveau	Taux maximal de traitement de macrobloc MaxMBPS (MB/s)	Taille max. de trame MaxFS (MBs)	Taille max. de tampon d'image décodée MaxDPB (1024 octets)	Débit binaire vidéo maximal MaxBR (1000 bit/s ou 1200 bit/s)	Taille maximale de CPB MaxCPB (1000 bits ou 1200 bits)	Gamme de composant MV vertical MaxVmvR (échantillons de trame luma)	Taux minimal de compression MinCR	Nombre maximal de vecteurs cinétiques pour deux MB consécutifs MaxMvsPer2Mb
1	1 485	99	148,5	64	175	[-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	491 520	8 192	12 288,0	50 000	62 500	[-512,+511,75]	2	16
5	589 824	22 080	41 310,0	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

Les niveaux avec des numéros de niveau non entiers dans le Tableau A-1 sont considérés comme "niveaux intermédiaires".

NOTE – Tous les niveaux ont le même statut, mais certaines applications peuvent choisir de n'utiliser que les niveaux à numéro entier.

Pour information, le § A.3.3 montre les effets de ces limites sur les débits de trame pour plusieurs exemples de formats d'image.

A.3.2 Limites de niveau spécifiques d'un profil

- Dans les flux binaires conformes au profil principal, le temps de retrait de l'unité d'accès 0 doit satisfaire à la contrainte que le nombre de tranches dans l'image 0 soit inférieur ou égal à $(PicSizeInMbs + MaxMBPS * (t_r(0) - t_{r,n}(0))) \div SliceRate$, où $SliceRate$ est la valeur spécifiée au Tableau A-3 qui s'applique à l'image 0.
- Dans les flux binaires conformes au profil principal, la différence entre le temps de retrait consécutif des unités d'accès n et $n - 1$ (avec $n > 0$) doit satisfaire à la contrainte que le nombre de tranches dans l'image n soit inférieur ou égal à $MaxMBPS * (t_r(n) - t_r(n - 1)) \div SliceRate$, où $SliceRate$ est la valeur spécifiée au Tableau A-3 qui s'applique à l'image n .
- Dans les flux binaires conformes au profil principal, les ensembles de paramètres de séquence doivent avoir `direct_8x8_inference_flag` égal à 1 pour les niveaux spécifiés au Tableau A-3.

NOTE – `direct_8x8_inference_flag` n'est pas pertinent pour le profil de base car il ne permet pas le type de tranche B (spécifié au § A.2.1), et `direct_8x8_inference_flag` est égal à 1 pour tous les niveaux du profil étendu (spécifié au § A.2.3).

- Dans les flux binaires conformes au profil principal et au profil étendu, les ensembles de paramètres de séquence doivent avoir `frame_mbs_only_flag` égal à 1 pour les niveaux spécifiés au Tableau A-3 pour le profil principal et au Tableau A-4 pour le profil étendu.

NOTE – frame_mbs_only_flag est égal à 1 pour tous les niveaux du profil de base (spécifié au § A.2.1).

- e) Dans les flux binaires conformes au profil principal et au profil étendu, la valeur de sub_mb_type dans les macroblocs B ne doit pas être égale à B_Bi_8x4, B_Bi_4x8, ou B_Bi_4x4 pour les niveaux dans lesquels MinLumaBiPredSize est indiqué comme 8x8 au Tableau A-3 pour le profil principal et au Tableau A-4 pour le profil étendu.
- f) Dans les flux binaires conformes au profil principal et au profil étendu, $(xInt_{max} - xInt_{min} + 6) * (yInt_{max} - yInt_{min} + 6) \leq MaxSubMbRectSize$ dans les macroblocs codés avec mb_type égal à P_8x8, P_8x8ref0 ou B_8x8 pour toutes les invocations du processus spécifié au § 8.4.2.2.1 utilisé pour générer le tableau d'échantillons luma prédits pour une seule liste (liste 0 ou liste 1) pour chaque sous-macrobloc 8x8, où NumSubMbPart(sub_mb_type) > 1, où MaxSubMbRectSize est spécifié au Tableau A-2 pour le profil de base et au Tableau A-4 pour le profil étendu et
- $xInt_{min}$ comme valeur minimale de $xInt_L$ parmi toutes les prédictions d'échantillon luma pour le sous-macrobloc
 - $xInt_{max}$ comme valeur maximale de $xInt_L$ parmi toutes les prédictions d'échantillon luma pour le sous-macrobloc
 - $yInt_{min}$ comme valeur minimale de $yInt_L$ parmi toutes les prédictions d'échantillon luma pour le sous-macrobloc
 - $yInt_{max}$ comme valeur maximale de $yInt_L$ parmi toutes les prédictions d'échantillon luma pour le sous-macrobloc.

A.3.2.1 Limites du profil de base

Le Tableau A-2 spécifie les limites pour chaque niveau qui est spécifique des flux binaires conformes au profil de base. Les entrées marquées "-" au Tableau A-2 notent l'absence de limite correspondante.

Tableau A-2 – Limites de niveau du profil de base

Numéro de niveau	MaxSubMbRectSize
1	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.3.2.2 Limites de profil principal

Le Tableau A-3 spécifie les limites pour chaque niveau qui est spécifique des flux binaires conformes au profil principal. Les entrées marquées "-" au Tableau A-3 notent l'absence de limite correspondante.

Tableau A-3 – Limites de niveau du profil principal

Numéro de niveau	SliceRate	MinLumaBiPredSize	direct_8x8_inference_flag	frame_mbs_only_flag
1	–	–	–	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.3.2.3 Limites de profil étendu

Le Tableau A-4 spécifie les limites pour chaque niveau qui est spécifique des flux binaires conformes au profil étendu. Les entrées marquées "-" au Tableau A-4 notent l'absence de limite correspondante.

Tableau A-4 – Limites de niveau du profil étendu

Numéro de niveau	MaxSubMbRectSize	MinLumaBiPredSize	frame_mbs_only_flag
1	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.3.3 Effet des limites de niveau sur le débit de trame (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

**Tableau A-5 – Débits de trame maximaux (trames par seconde)
pour quelques exemples de tailles de trame**

					1	1.1	1.2	1.3	2	2.1	2.2
Numéro de niveau:											
Taille max de trame (macroblocs):					99	396	396	396	396	792	1 620
Max macroblocs/seconde:					1 485	3 000	6 000	11 880	11 880	19 800	20 250
Taille de trame maximale (échantillons):					25 344	101 376	101 376	101 376	101 376	202 752	414 720
Max échantillons/seconde:					380 160	768 000	1 536 000	3 041 280	3 041 280	5 068 800	5 184 000
Format	Lar-geur luma	Hauteur luma	Total MB	Echantil-lons luma							
SQCIF	128	96	48	12 288	30,9	62,5	125,0	172,0	172,0	172,0	172,0
QCIF	176	144	99	25 344	15,0	30,3	60,6	120,0	120,0	172,0	172,0
QVGA	320	240	300	76 800	–	10,0	20,0	39,6	39,6	66,0	67,5
525 SIF	352	240	330	84 480	–	9,1	18,2	36,0	36,0	60,0	61,4
CIF	352	288	396	101 376	–	7,6	15,2	30,0	30,0	50,0	51,1
525 HHR	352	480	660	168 960	–	–	–	–	–	30,0	30,7
625 HHR	352	576	792	202 752	–	–	–	–	–	25,0	25,6
VGA	640	480	1 200	307 200	–	–	–	–	–	–	16,9
525 4SIF	704	480	1 320	337 920	–	–	–	–	–	–	15,3
525 SD	720	480	1 350	345 600	–	–	–	–	–	–	15,0
4CIF	704	576	1 584	405 504	–	–	–	–	–	–	12,8
625 SD	720	576	1 620	414 720	–	–	–	–	–	–	12,5
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	–	–	–	–	–	–	–
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	–	–	–	–	–	–	–

**Tableau A-5 (suite) – Débits de trame maximaux (trames par seconde)
pour quelques exemples de tailles de trame**

Numéro de niveau:					3	3.1	3.2	4	4.1	4.2
Taille max de trame (macrobloques):					1 620	3 600	5 120	8 192	8 192	8 192
Max macrobloques/seconde:					40 500	108 000	216 000	245 760	245 760	589 824
Taille max de trame (échantillons):					414 720	921 600	1 310 720	2 097 152	2 097 152	2 097 152
Max échantillons/seconde:					10 368 000	27 648 000	55 296 000	62 914 560	62 914 560	125 829 120
Format	Largeur luma	Hauteur luma	Total MB	Echantillons luma						
SQCIF	128	96	48	12 288	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
QVGA	320	240	300	76 800	135,0	172,0	172,0	172,0	172,0	172,0
525 SIF	352	240	330	84 480	122,7	172,0	172,0	172,0	172,0	172,0
CIF	352	288	396	101 376	102,3	172,0	172,0	172,0	172,0	172,0
525 HHR	352	480	660	168 960	61,4	163,6	172,0	172,0	172,0	172,0
625 HHR	352	576	792	202 752	51,1	136,4	172,0	172,0	172,0	172,0
VGA	640	480	1 200	307 200	33,8	90,0	172,0	172,0	172,0	172,0
525 4SIF	704	480	1 320	337 920	30,7	81,8	163,6	172,0	172,0	172,0
525 SD	720	480	1 350	345 600	30,0	80,0	160,0	172,0	172,0	172,0
4CIF	704	576	1 584	405 504	25,6	68,2	136,4	155,2	155,2	172,0
625 SD	720	576	1 620	414 720	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	160,0
720p HD	1280	720	3 600	921 600	–	30,0	60,0	68,3	68,3	136,5
4VGA	1280	960	4 800	1 228 800	–	–	45,0	51,2	51,2	102,4
SXGA	1280	1024	5 120	1 310 720	–	–	42,2	48,0	48,0	96,0
525 16SIF	1408	960	5 280	1 351 680	–	–	–	46,5	46,5	93,1
16CIF	1408	1152	6 336	1 622 016	–	–	–	38,8	38,8	77,6
4SVGA	1600	1200	7 500	1 920 000	–	–	–	32,8	32,8	65,5
1080 HD	1920	1088	8 160	2 088 960	–	–	–	30,1	30,1	60,2
2Kx1K	2048	1024	8 192	2 097 152	–	–	–	30,0	30,0	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	–	–	–	–	–	–

**Tableau A-5 (fin) – Débits de trame maximaux (trames par seconde)
pour quelques exemples de tailles de trame**

Numéro de niveau:					5	5.1
Taille de trame maximale (macroblocs):					21 696	36 864
Max macroblocs/seconde:					589 824	983 040
Taille de trame maximale (échantillons):					5 554 176	9 437 184
Max échantillons/seconde:					150 994 944	251 658 240
Format	Largeur luma	Hauteur luma	Total MB	Echantillons luma		
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
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

Il convient de noter ce qui suit:

- La présente Recommandation | Norme internationale est une spécification de trame à taille variable. Les tailles de trame spécifiques du Tableau A-5 sont uniquement données à titre d'exemple.
- Tel qu'utilisé dans le Tableau A-5, "525" se réfère à une utilisation typique dans les environnements utilisant 525 lignes de balayage analogique (dont environ 480 lignes contiennent la région visible de l'image), et "625" se réfère aux environnements utilisant 625 lignes de balayage analogique (dont environ 576 lignes contiennent la région visible de l'image).
- XGA est aussi connu sous le nom de (aka) XVGA, 4SVGA aka UXGA, 16XGA aka 4Kx3K, CIF aka 625 SIF, 625 HHR aka 2CIF aka demi 625 D-1, aka demi 625 UIT-R BT.601, 525 SD aka 525 D-1 aka 525 UIT-R BT.601, 625 SD aka 625 D-1 aka 625 UIT-R BT.601.
- Les débits de trame donnés sont corrects pour les modes de balayage progressifs. Les débits de trame sont aussi corrects pour le codage vidéo entrelacé pour les cas de hauteur de trame divisible par 32.

Annexe B

Format de flux d'octets

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe spécifie la syntaxe et la sémantique d'un format de flux d'octets spécifié pour être utilisé par des applications qui fournissent certains ou tous les flux d'unités NAL comme flux ordonnés d'octets ou de bits au sein desquels les localisations des frontières des unités NAL doivent être identifiables à partir des schémas inclus dans les données, tels que les systèmes de la Rec. UIT-T H.222.0 | ISO/CEI 13818-1 ou de la Rec. UIT-T H.320. Pour les applications orientées bit, l'ordre binaire pour le format de flux d'octet est spécifié comme commençant avec le bit de plus fort poids (MSB) du premier octet, puis en continuant jusqu'au bit de plus faible poids (LSB) du premier octet, suivi par le MSB du second octet, etc.

Le format de flux d'octets consiste en une séquence de structures syntaxiques d'unités NAL de flux d'octet. Chaque structure syntaxique d'unité NAL de flux d'octet contient un préfixe de code de déclenchement suivi d'une structure syntaxique `nal_unit(NumBytesInNALunit)`. Il peut (et dans certaines circonstances, il doit) aussi contenir un élément syntaxique `zero_byte` supplémentaire. Il peut aussi contenir un ou plusieurs éléments syntaxiques `trailing_zéro_8bits` supplémentaires. Lorsqu'il s'agit de la première unité NAL de flux d'octet du flux linéaire, il peut aussi contenir un ou plusieurs éléments syntaxiques `leading_zéro_8bits` supplémentaires.

B.1 Syntaxe et sémantique d'unité NAL de flux d'octet

B.1.1 Syntaxe d'unité NAL de flux d'octet

	C	Descripteur
<code>byte_stream_nal_unit(NumBytesInNALunit) {</code>		
<code> alors que(next_bits(24) != 0x000001 &&</code> <code> next_bits(32) != 0x00000001)</code>		
<code> leading_zéro_8bits /* égal à 0x00 */</code>		f(8)
<code> si(next_bits(24) != 0x000001)</code>		
<code> zero_byte /* égal à 0x00 */</code>		f(8)
<code> si(more_data_in_byte_stream()) {</code>		
<code> start_code_prefix_one_3bytes /* égal à 0x000001 */</code>		f(24)
<code> nal_unit(NumBytesInNALunit)</code>		
<code> }</code>		
<code> alors que(more_data_in_byte_stream() &&</code> <code> next_bits(24) != 0x000001 &&</code> <code> next_bits(32) != 0x00000001)</code>		
<code> trailing_zéro_8bits /* égal à 0x00 */</code>		f(8)
<code>}</code>		

B.1.2 Sémantique d'unité NAL de flux d'octet

L'ordre des unités NAL de flux d'octet dans le flux d'octet doit suivre l'ordre de décodage des unités NAL contenues dans le flux d'octet des unités NAL (voir le § 7.4.1.2). Le contenu de chaque unité NAL de flux d'octet est associé à la même unité d'accès que l'unité NAL contenue dans l'unité NAL de flux d'octet (voir le § 7.4.1.2.3).

leading_zéro_8bits est un octet égal à 0x00.

NOTE – L'élément syntaxique `leading_zéro_8bits` ne peut être présent que dans la première unité NAL de flux d'octet du flux binaire, étant donné que (comme il ressort du diagramme de syntaxe du § B.1.1) les octets égaux à 0x00 qui suivent une structure syntaxique d'unité NAL et qui précèdent la séquence de quatre octets 0x00000001 (qui doit être interprétée comme un élément `zero_byte` suivi d'un élément

start_code_prefix_one_3bytes) seront considérés comme des éléments syntaxiques trailing_zero_8bits qui font partie de l'unité NAL de flux d'octet précédente.

zero_byte est un seul octet égal à 0x00.

Lorsque l'une des conditions suivantes est remplie, les 'éléments syntaxiques zero_byte doivent être présents.

- le nal_unit_type au sein de nal_unit() est égal à 7 (ensemble de paramètres de séquence) ou 8 (ensemble de paramètres d'image);
- la structure syntaxique de l'unité NAL de flux d'octet contient la première unité NAL d'une unité d'accès dans l'ordre de décodage, comme spécifié au § 7.4.1.2.3.

start_code_prefix_one_3bytes est une séquence de valeur fixe de 3 octets égale à 0x000001. Cet élément syntaxique est appelé un préfixe de code de déclenchement.

trailing_zero_8bits est un octet égal à 0x00.

B.2 Processus de décodage d'unité NAL de flux d'octet

L'entrée pour ce processus consiste en un flux ordonné d'octets fait d'une séquence de structures syntaxiques d'unités NAL de flux d'octets. La sortie du processus est une réponse de structures syntaxiques d'unités NAL.

Au début du processus de décodage, le décodeur initialise sa position courante dans le flux d'octets au début de ce flux. Puis il extrait et supprime chaque élément syntaxique leading_zero_8bits (le cas échéant) en déplaçant la position courante dans le flux d'octets vers l'avant à raison d'un octet à la fois, jusqu'à ce que la position courante dans le flux d'octets soit telle que les quatre octets suivants du flux binaire forment la séquence de quatre octets 0x00000001.

Le décodeur effectue alors le processus par étapes de façon répétitive pour extraire et décoder chaque structure syntaxique d'unités NAL dans le flux d'octets jusqu'à ce qu'il arrive à la fin du flux d'octets (telle que déterminée par des moyens non spécifiés) et que la dernière unité NAL du flux binaire ait été décodée:

1. lorsque les quatre octets suivants dans le flux binaire forment la séquence de quatre octets 0x00000001, l'octet suivant dans le flux d'octets (qui est un élément syntaxique zero_byte) est extrait et supprimé et la position courante dans le flux d'octets est mise égale à la position de l'octet qui suit cet octet supprimé;
2. la prochaine séquence de trois octets dans le flux d'octets (qui est un élément syntaxique start_code_prefix_one_3bytes) est extraite et supprimée et la position courante dans le flux d'octet est mise égale à la position de l'octet suivant cette séquence de trois octets;
3. NumBytesInNALunit est mis égal au nombre d'octets commençant par l'octet à la position courante dans le flux d'octet, jusques et y compris le dernier octet qui précède la localisation de toute condition suivante:
 - a. une séquence suivante de trois octets alignés sur les octets égale à 0x000000, ou
 - b. une séquence suivante de trois octets alignés sur les octets égale à 0x000001, ou
 - c. la fin du flux d'octets, telle que déterminée par des moyens non spécifiés;
4. NumBytesInNALunit octets sont retirés du flux binaire et la position en cours dans le flux d'octets est avancée de NumBytesInNALunit octets. Cette séquence d'octets est nal_unit(NumBytesInNALunit) et elle est décodée en utilisant le processus de décodage d'unités NAL;
5. lorsque la position courante dans le flux d'octets ne correspond pas à la fin du flux d'octets (telle que déterminée par des moyens non spécifiés) et que les octets suivants dans le flux d'octets ne commencent pas par une séquence de trois octets égale à

0x000001 et que les octets suivants dans le flux d'octets ne commencent pas par une séquence de quatre octets égale à 0x00000001, le décodeur extrait et supprime chaque élément syntaxique `trailing_zero_8bits`, en déplaçant la position courante dans le flux d'octets à la fois, jusqu'à ce que la position courante dans le flux d'octets soit telle que les octets suivants dans le flux d'octets forment la séquence de quatre octets 0x00000001 où que l'on soit arrivé à la fin du flux d'octets (telle que déterminée par des moyens non spécifiés).

B.3 Récupération du verrouillage d'octet du décodeur (pour information)

Le présent paragraphe ne fait pas partie intégrante de la présente Recommandation | Norme internationale.

De nombreuses applications fournissent les données à un décodeur d'une façon qui est intrinsèquement alignée sur les octets, et donc il n'est aucun besoin de la procédure de détection du verrouillage d'octet orientée sur les bits décrite au présent paragraphe.

On dit d'un décodeur qu'il a un verrouillage d'octet avec un flux binaire lorsque ce décodeur est en mesure de déterminer si les positions des données dans le flux binaire sont ou non alignées sur les octets. Lorsqu'un décodeur n'a pas de verrouillage d'octet avec le flux d'octets du codeur, le décodeur peut examiner le flux binaire entrant pour chercher le schéma binaire '00000000 00000000 00000001' (31 bits consécutifs égaux à 0 suivis d'un bit égal à 1). Le bit suivant immédiatement ce schéma est le premier bit d'un octet verrouillé suivant un préfixe de code de déclenchement. A la détection de ce schéma, le décodeur sera verrouillé sur le codeur et positionné au début d'une unité NAL dans le flux d'octet.

Une fois le verrouillage d'octet effectué avec le codeur, le décodeur peut examiner le flux d'octet entrant pour trouver les séquences de trois octets 0x000001 et 0x000003 suivantes.

Lorsque la séquence de trois octets 0x000001 est détectée, il s'agit d'un préfixe de code de déclenchement.

Lorsque la séquence de trois octets 0x000003 est détectée, le troisième octet (0x03) est un `emulation_prevention_three_byte` à détruire comme spécifié au § 7.4.1.

La procédure de détection de verrouillage d'octet décrite au présent paragraphe est fonctionnellement équivalente à la recherche d'une séquence d'octets pour trois octets consécutifs de valeur zéro (0x000000), débutant à toute position de verrouillage. La détection de ce schéma indique que le prochain octet différent de zéro contient la fin d'un préfixe de code de déclenchement (car un flux d'octets conforme ne peut pas contenir plus de 23 bits consécutifs de valeur zéro sans contenir 31, ou plus, bits consécutifs de valeur zéro, permettant la détection du 0x000000 se rapportant à un début de position de verrouillage), et le premier bit de valeur différente de zéro dans ce prochain octet différent de zéro est le dernier bit d'un octet verrouillé et c'est le dernier bit d'un préfixe de code de déclenchement.

Annexe C

Décodeur fictif de référence

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe spécifie le décodeur fictif de référence (HRD) et son utilisation pour vérifier la conformité du flux binaire et du décodeur.

Deux types de flux binaires font l'objet de la vérification de conformité du HRD pour la présente Recommandation | Norme internationale. Le premier de ces types de flux binaire, appelé flux binaire de Type I, est un flux d'unités NAL ne contenant que des unités NAL VCL et des unités NAL de données de remplissage pour toutes les unités d'accès dans le flux binaire. Le second type de flux binaire, appelé flux binaire de Type II, contient, en plus des unités NAL VCL et des unités NAL de données de remplissage pour toutes les unités d'accès dans le flux binaire, au moins un des éléments suivants:

- des unités NAL non VCL supplémentaires autres que des unités NAL de données de remplissage;
- tous les éléments syntaxiques `leading_zero_8bits`, `zero_byte`, `start_code_prefix_one_3bytes` et `trailing_zero_8bits` qui forment un flux d'octets à partir du flux d'unités NAL (comme spécifié à l'Annexe B).

La Figure C-1 montre les types de points de conformité de flux binaire vérifiés par le HRD.

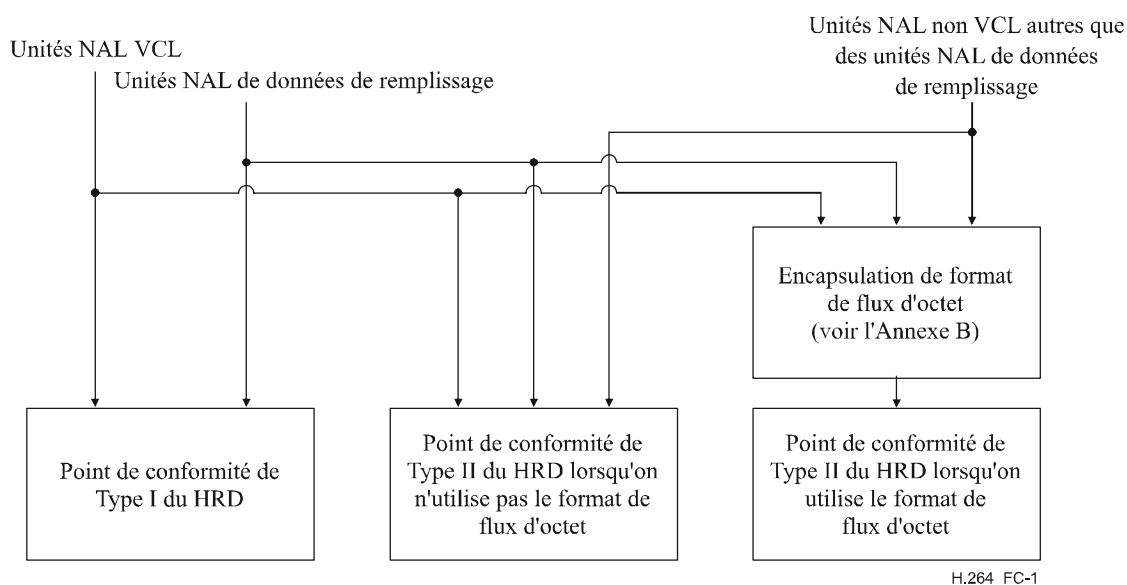


Figure C-1 – Structure des flux d'octets et des flux d'unités NAL pour les vérifications de conformité par le HRD

Les éléments syntaxiques des unités NAL non VCL (ou leurs valeurs par défaut pour certains éléments syntaxiques) requis pour le HRD sont spécifiés aux paragraphes de sémantique du § 7 et aux Annexes D et E.

Deux types d'ensembles de paramètres de HRD sont utilisés. Les ensembles de paramètres de HRD sont signalés par des informations d'utilisation vidéo, comme spécifié aux § E.1 et E.2, qui font partie de la structure syntaxique d'ensembles de paramètres de séquence.

Pour vérifier la conformité d'un flux binaire en utilisant le HRD, tous les ensembles de paramètres de séquences et les ensembles de paramètres d'image mentionnés dans les unités NAL VCL, et les messages SEI correspondants de période de mise en mémoire tampon et de temporisation d'image doivent être envoyés au HRD en temps utile, soit dans le flux binaire (par des unités NAL non VCL), soit par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale.

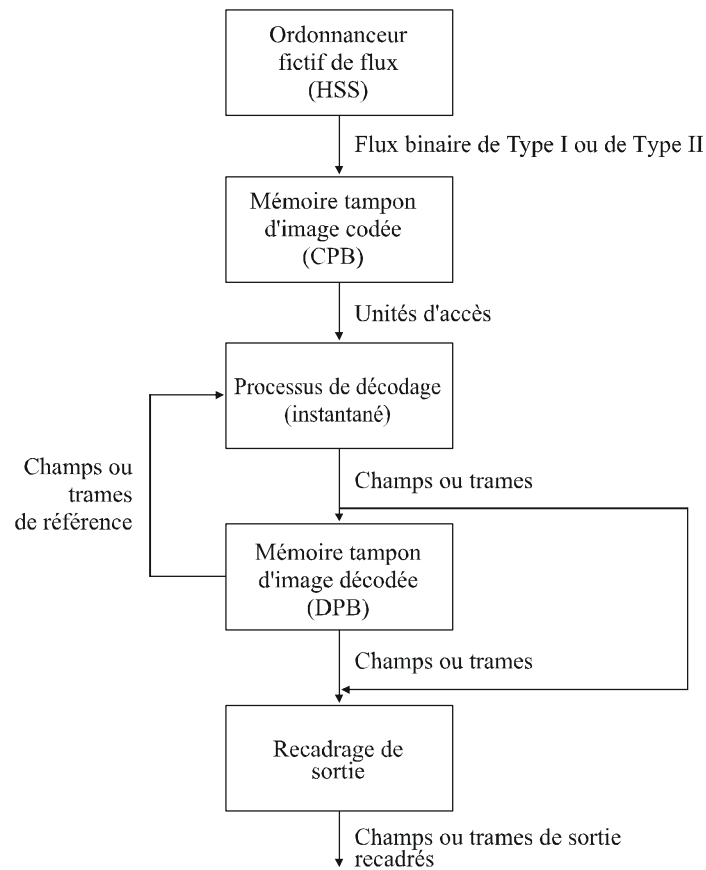
Dans les Annexes C, D et E, la spécification pour "présence" des unités NAL non VCL est aussi satisfaite lorsque ces unités NAL (ou seulement certaines d'entre elles) sont envoyées aux décodeurs (ou au HRD) par d'autres moyens non spécifiés par la présente Recommandation | Norme internationale. Pour les besoins du comptage des bits, seuls les bits appropriés qui sont réellement présents dans le flux binaire sont comptés.

NOTE – A titre d'exemple, la synchronisation d'une unité NAL non VCL, envoyée par des moyens autres que la présence dans le flux binaire, avec des unités NAL qui sont présentes dans le flux binaire, peut être réalisée en indiquant deux points dans le flux binaire, entre lesquels l'unité NAL non VCL aurait été présente dans le flux binaire, si le codeur avait décidé de la convoier dans le flux binaire.

Lorsque le contenu d'une unité NAL non VCL est envoyé pour être appliqué par des moyens autres que la présence au sein du flux binaire, la représentation du contenu de l'unité NAL non VCL n'est pas nécessaire pour utiliser la même syntaxe que celle spécifiée dans la présente annexe.

NOTE – Lorsque les informations du HRD sont contenues au sein du flux binaire, il est possible de vérifier la conformité d'un flux binaire aux exigences du présent paragraphe fondées uniquement sur les informations contenues dans le flux binaire. Lorsque les informations du HRD ne sont pas présentes dans le flux binaire, comme c'est le cas pour tous les flux binaires de Type I autonomes, la conformité ne peut être vérifiée que lorsque les données du HRD sont fournies par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale.

Le HRD contient une mémoire tampon d'image codée (CPB), un processus de décodage instantané, une mémoire tampon d'image décodée (DPB), et le recadrage du résultat comme indiqué à la Figure C-2.



H.264_FC-2

Figure C-2 – Modèle de mémoire tampon de HRD

La taille du CPB (nombre de bits) est spécifiée par $CpbSize[SchedSelIdx]$ à l'Annexe E. La taille de DPB (nombre de tampons de trame) est spécifiée comme $Max(1, max_dec_frame_buffering)$ à l'Annexe E.

Le HRD fonctionne comme suit. Les données associées aux unités d'accès qui s'écoulent dans le CPB conformément à un schéma d'arrivée spécifié sont livrées par le HSS. Les données associées à chaque unité d'accès sont retirées et décodées instantanément par le processus de décodage instantané aux instants de retrait du CPB. Chaque image décodée est placée dans le DPB à son moment de retrait à moins qu'elle ne soit sortie à son moment de retrait du CPB et soit une image de non-référence. Lorsqu'une image est placée dans le DPB, elle est retirée du DPB au dernier moment de retrait du DPB ou au moment où elle est marquée comme "non utilisé comme référence".

Le fonctionnement du CPB est spécifié au § C.1. Le fonctionnement instantané du décodeur est spécifié aux § 8 et 9. Le fonctionnement du DPB est spécifié au § C.2. Le recadrage du résultat est spécifié au § C.2.2.

Les informations de HSS et HRD concernant le nombre de schémas de livraison énumérés et leurs débits binaires associés ainsi que les tailles de mémoires tampon sont spécifiées aux § E.1.1, E.1.2, E.2.1 et E.2.2. Le HRD est initialisé comme spécifié par le message SEI de période de mise en mémoire tampon comme spécifié aux § D.1.1 et D.2.1. Le moment de retrait des unités d'accès du CPB et le moment de sortie du DPB sont spécifiés au message SEI de temporisation d'image comme spécifié aux § D.1.2 et D.2.2. Toutes les informations de temporisation se rapportant à une unité d'accès spécifique doivent arriver avant le moment de retrait du CPB de l'unité d'accès.

Le HRD est utilisé pour vérifier la conformité des flux binaires et des décodeurs comme spécifié respectivement aux § C.3 et C.4.

NOTE – Alors que la conformité est garantie dans l'hypothèse où tous les débits de trame et d'horloge utilisés pour générer le flux binaire correspondent exactement aux valeurs signalées dans le flux binaire, dans un système réel, chacun d'eux peut différer de la valeur signalée ou spécifiée.

Toute l'arithmétique de la présente annexe est faite avec des valeurs réelles, de sorte qu'aucune erreur d'arrondi ne puisse se développer. Par exemple, le nombre de bits dans un CPB juste avant ou juste après le retrait d'une unité d'accès n'est pas nécessairement entier.

La variable t_c est déduite comme suit et est appelée un tic-tac d'horloge.

$$t_c = \text{num_units_in_tick} \div \text{time_scale} \quad (\text{C-1})$$

Ce qui suit est spécifié pour exprimer les contraintes de la présente annexe.

- Soit l'unité d'accès n la $n^{\text{ième}}$ unité d'accès dans l'ordre de décodage, la première unité d'accès étant l'unité d'accès 0.
- Soit l'image n l'image codée primaire ou l'image primaire décodée de l'unité d'accès n .

C.1 Fonctionnement de la mémoire tampon d'image codée (CPB)

Les spécifications du présent paragraphe s'appliquent indépendamment à chaque ensemble de paramètres de CPB présent et aux points de conformité de Type I et de Type II représentés sur la Figure C-1.

C.1.1 Moment de l'arrivée du flux binaire

Le HRD peut être initialisé à tout message SEI de période de mise en mémoire tampon. Avant l'initialisation, le CPB est vide.

NOTE – Après l'initialisation, le HRD n'est pas initialisé à nouveau par des messages SEI de période de mise en mémoire tampon ultérieurs.

Chaque unité d'accès est mentionnée comme unité d'accès n , où le numéro n identifie l'unité d'accès considérée. L'unité d'accès qui est associée au message SEI de période de mise en mémoire tampon qui initialise le CPB est mentionnée comme unité d'accès 0. La valeur de n est incrémentée de 1 pour chaque unité d'accès suivante dans l'ordre de décodage.

Le moment auquel le premier bit de l'unité d'accès n commence à entrer dans le CPB est désigné comme le moment d'arrivée initial $t_{ai}(n)$.

Le moment d'arrivée initial des unités d'accès est calculé comme suit:

- si l'unité d'accès est l'unité d'accès 0, $t_{ai}(0) = 0$,
- autrement (l'unité d'accès est l'unité d'accès n avec $n > 0$), on applique ce qui suit:
 - si $\text{cbr_flag}[\text{SchedSelIdx}]$ est égal à 1, le moment d'arrivée initial pour l'unité d'accès n est égal au moment d'arrivée final (qui est calculé ci-dessous) de l'unité d'accès $n - 1$, c'est-à-dire

$$t_{ai}(n) = t_{af}(n - 1) \quad (\text{C-2})$$

- autrement ($\text{cbr_flag}[\text{SchedSelIdx}]$ est égal à 0), le moment d'arrivée initial pour l'unité d'accès n est calculé par

$$t_{ai}(n) = \text{Max}(t_{af}(n - 1), t_{ai,earliest}(n)) \quad (\text{C-3})$$

où $t_{ai,earliest}(n)$ est calculé comme suit

- si l'unité d'accès n n'est pas la première unité d'accès d'une période de mise en mémoire tampon suivante, $t_{ai,earliest}(n)$ est calculé comme

$$t_{ai,earliest}(n) = t_{r,n}(n) - (\text{initial_cpb_removal_delay}[\text{SchedSelIdx}] + \text{initial_cpb_removal_delay_offset}[\text{SchedSelIdx}]) \div 90000 \quad (\text{C-4})$$

avec $t_{r,n}(n)$ qui est le moment de retrait nominal de l'unité d'accès n à partir du CPB comme spécifié au § C.4.2 et $initial_cpb_removal_delay[SchedSelIdx]$ et $initial_cpb_removal_delay_offset[SchedSelIdx]$ qui sont spécifiés dans le message SEI de période de mise en mémoire tampon précédente.

- Autrement (l'unité d'accès n suivante est la première unité d'accès d'une période de mise en mémoire tampon suivante), $t_{ai,earliest}(n)$ est calculé comme

$$t_{ai,earliest}(n) = t_{r,n}(n) - (initial_cpb_removal_delay[SchedSelIdx] \div 90000) \quad (C-5)$$

avec $initial_cpb_removal_delay[SchedSelIdx]$ qui est spécifié dans le message SEI de période de mise en mémoire tampon associé à l'unité d'accès n .

Le moment d'arrivée final pour l'unité d'accès n est calculé par

$$t_{af}(n) = t_{ai}(n) + b(n) \div BitRate[SchedSelIdx] \quad (C-6)$$

où $b(n)$ est la taille en bits de l'unité d'accès n , en comptant les bits des unités NAL VCL et des unités NAL de données de remplissage pour le point de conformité de Type I ou tous les bits de flux binaire de Type II pour le point de conformité de Type II, où les points de conformité de Type I et de Type II sont tels que représentés sur la Figure C-1.

Les valeurs de $SchedSelIdx$, $BitRate[SchedSelIdx]$, et $CpbSize[SchedSelIdx]$ sont soumises comme suit à des contraintes.

- Si l'unité d'accès n et l'unité d'accès $n - 1$ font partie de séquences vidéo codées différentes et que le contenu des ensembles actifs de paramètres de séquences des deux séquences vidéo codées diffère, le HSS choisit une valeur $SchedSelIdx1$ de $SchedSelIdx$ parmi les valeurs de $SchedSelIdx$ fournies pour la séquence vidéo codée qui contient l'unité d'accès n qui a pour résultat un $BitRate[SchedSelIdx1]$ ou $CpbSize[SchedSelIdx1]$ pour la seconde des deux séquences vidéo codées (qui contient l'unité d'accès $n - 1$) qui diffère de la valeur de $BitRate[SchedSelIdx0]$ ou $CpbSize[SchedSelIdx0]$ pour la valeur $SchedSelIdx0$ de $SchedSelIdx$ qui était utilisée pour la séquence vidéo codée contenant l'unité d'accès $n - 1$. La valeur de $BitRate[SchedSelIdx1]$ ou $CpbSize[SchedSelIdx1]$ peut différer de la valeur de $BitRate[SchedSelIdx0]$ ou $CpbSize[SchedSelIdx0]$ pour la valeur $SchedSelIdx0$ de $SchedSelIdx$ qui était utilisée pour la séquence vidéo codée contenant l'unité d'accès $n - 1$.
- Autrement, le HSS continue de fonctionner avec les valeurs précédentes de $SchedSelIdx$, $BitRate[SchedSelIdx]$ et $CpbSize[SchedSelIdx]$.

Lorsque le HSS choisit des valeurs de $BitRate[SchedSelIdx]$ ou $CpbSize[SchedSelIdx]$ qui diffèrent de celles de l'unité d'accès précédente, on applique ce qui suit.

- la variable $BitRate[SchedSelIdx]$ est mise en œuvre au temps $t_{ai}(n)$;
- la variable $CpbSize[SchedSelIdx]$ est mise en œuvre comme suit.
 - Si la nouvelle valeur de $CpbSize[SchedSelIdx]$ dépasse l'ancienne taille de CPB, elle est mise en œuvre au temps $t_{ai}(n)$.
 - Autrement, la nouvelle valeur de $CpbSize[SchedSelIdx]$ est mise en œuvre au temps $t_r(n)$.

C.1.2 Moment du retrait de l'image codée

Pour l'unité d'accès 0, le moment de retrait nominal de l'unité d'accès du CPB est spécifié par

$$t_{r,n}(0) = initial_cpb_removal_delay[SchedSelIdx] \div 90000 \quad (C-7)$$

Pour la première unité d'accès d'une période de mise en mémoire tampon qui n'initialise pas le HRD, le moment de retrait nominal de l'unité d'accès du CPB est spécifié par

$$t_{r,n}(n) = t_{r,n}(n_b) + t_c * cpb_removal_delay(n) \quad (C-8)$$

où $t_{r,n}(n_b)$ est le moment de retrait nominal de la première unité d'accès de la période précédente de mise en mémoire tampon et $cpb_removal_delay(n)$ est la valeur de $cpb_removal_delay$ spécifiée dans le message SEI de périodisation d'image associé à l'unité d'accès n .

Lorsqu'une unité d'accès n est la première unité d'accès d'une période de mise en mémoire tampon, n_b est mis égal à n au moment du retrait de l'unité d'accès n .

Le moment nominal de retrait $t_{r,n}(n)$ d'une unité d'accès n qui n'est pas la première unité d'accès d'une période de mise en mémoire tampon est donné par

$$t_{r,n}(n) = t_{r,n}(n_b) + t_c * cpb_removal_delay(n) \quad (C-9)$$

où $t_{r,n}(n_b)$ est le moment nominal de retrait de la première unité d'accès de la période de mise en mémoire tampon en cours et $cpb_removal_delay(n)$ est la valeur de $cpb_removal_delay$ spécifiée dans le message SEI de périodisation d'image associé à l'unité d'accès n .

Le moment de retrait d'une unité d'accès n est spécifié comme suit:

- Si $low_delay_hrd_flag$ est égal à 0 ou $t_{r,n}(n) \geq t_{af}(n)$, le moment de retrait de l'unité d'accès n est spécifié par

$$t_r(n) = t_{r,n}(n) \quad (C-10)$$

- Autrement ($low_delay_hrd_flag$ est égal à 1 et $t_{r,n}(n) < t_{af}(n)$), le moment de retrait de l'unité d'accès n est spécifié par

$$t_r(n) = t_{r,n}(n) + t_c * Ceil((t_{af}(n) - t_{r,n}(n)) \div t_c) \quad (C-11)$$

NOTE – Le dernier cas indique que la taille de l'unité d'accès n , $b(n)$, est si grande qu'elle empêche le retrait au moment de retrait nominal.

C.2 Fonctionnement de la mémoire tampon de l'image décodée (DPB)

La mémoire tampon de l'image décodée contient les mémoires tampon de trame. Chacune des mémoires tampon de trame peut contenir une trame décodée, une paire de champs complémentaires décodés ou un seul (non apparié) champ décodé qui sont marqués comme "utilisé pour référence" (images de référence) ou sont gardés pour une sortie future (images réordonnées ou retardées). Avant l'initialisation, le DPB est vide (le remplissage du DPB est mis à zéro). Les étapes suivantes des subdivisions du présent paragraphe surviennent toutes instantanément à $t_r(n)$ et selon la séquence indiquée.

C.2.1 Décodage des trous dans `frame_num` et mémorisation des trames "non existantes"

Le cas échéant, les trous dans `frame_num` sont détectés par le processus de décodage et les trames générées sont marquées et insérées dans le DPB comme spécifié ci-dessous.

Les trous dans `frame_num` sont détectés par le processus de décodage et les trames générées sont marquées comme spécifié au § 8.2.5.2.

Après le marquage de chaque trame générée, chaque image m marquée par le processus de "fenêtre glissante" comme "non utilisé comme référence" est retirée du DPB lorsque elle est aussi marquée comme "non existante" ou que son moment de sortie de DPB est inférieur ou égal au moment de retrait du CPB de l'image en cours n ; c'est-à-dire, $t_{o,dpb}(m) \leq t_r(n)$. Lorsqu'une trame ou le dernier champ d'une mémoire tampon de trame est retirée du DPB, le remplissage du DPB est décrémenté de un. La trame générée "non existante" est insérée dans le DPB et le remplissage du DPB est incrémenté de un.

C.2.2 Décodage d'image et sortie

L'image n est décodée et son temps de sortie du DPB $t_{o,dpb}(n)$ est calculé par

$$t_{o,dpb}(n) = t_r(n) + t_c * dpb_output_delay(n) \quad (C-12)$$

La sortie de l'image en cours est spécifiée comme suit.

- Si $t_{o,dpb}(n) = t_r(n)$, l'image en cours est sortie.
NOTE – Lorsque l'image en cours est une image de référence, elle sera mémorisée dans le DPB.
- Autrement ($t_{o,dpb}(n) > t_r(n)$), l'image en cours est sortie ultérieurement et sera mémorisée dans le DPB (comme spécifié au § C.2.4) et sera sortie au moment $t_{o,dpb}(n)$ à moins qu'il ne soit indiqué de ne pas la sortir au décodage ou qu'il y ait une inférence de `no_output_of_prior_pics_flag` égal à 1 au moment qui précède $t_{o,dpb}(n)$.

L'image de sortie doit être recadrée, en utilisant la mire spécifiée dans l'ensemble de paramètres de séquence pour la séquence.

Lorsque l'image n est une image qui est sortie et n'est pas la dernière image du flux binaire qui est sorti, la valeur de $\Delta t_{o,dpb}(n)$ est définie comme:

$$\Delta t_{o,dpb}(n) = t_{o,dpb}(n_n) - t_{o,dpb}(n) \quad (C-13)$$

où n_n indique l'image qui suit après l'image n dans l'ordre de sortie.

L'image décodée est temporairement mémorisée (mais pas dans le DPB).

C.2.3 Retrait des images du DPB avant une possible insertion de l'image en cours

Le retrait des images du DPB avant la possible insertion de l'image en cours s'effectue comme suit.

- Si l'image décodée est une image IDR, on applique ce qui suit.
 - Toutes les images de référence dans le DPB sont marquées comme "non utilisé comme référence" comme spécifié aux § 8.2.5.3 et 8.2.5.4.
 - Lorsque l'image IDR n'est pas la première image IDR décodée et que la valeur de `PicWidthInMbs` ou de `FrameHeightInMbs` ou de `max_dec_frame_buffering` déduite de l'ensemble actif de paramètres de séquence est différente respectivement de la valeur de `PicWidthInMbs` ou `FrameHeightInMbs` ou `max_dec_frame_buffering` déduite de l'ensemble de paramètres de séquence qui était actif pour la précédente séquence, `no_output_of_prior_pics_flag` est supposé être égal à 1 par le HRD, sans considération de la valeur réelle de `no_output_of_prior_pics_flag`.
NOTE – Les implémentations de décodeurs devraient essayer de traiter les changements de trame ou de taille de DPB plus harmonieusement que le HRD par rapport aux changements dans `PicWidthInMbs` ou `FrameHeightInMbs`.
- Lorsque `no_output_of_prior_pics_flag` est égal à 1 ou est supposé être égal à 1, toutes les mémoires tampon de trame dans le DPB sont vidées sans sortie des images qu'elles contiennent, et le remplissage de DPB est mis à 0.
- Autrement (l'image décodée n'est pas une image IDR), on applique ce qui suit.
 - Si l'en-tête de trame de l'image en cours inclut `memory_management_control_operation` égal à 5, toutes les images de référence dans le DPB sont marquées comme "non utilisé comme référence".
 - Autrement (l'en-tête de trame de l'image en cours n'inclut pas de `memory_management_control_operation` égal à 5), le processus de marquage d'image de référence décodée spécifié au § 8.2.5 est invoqué.

Toutes les images m dans le DPB, pour lesquelles toutes les conditions suivantes sont Vraies, sont retirées du DPB.

- L'image m est marquée comme "non utilisé comme référence" ou l'image m est une image de non-référence. Lorsqu'une image est une trame de référence, elle n'est considérée comme marquée comme "non utilisé comme référence" que lorsque ses deux champs ont été marqués comme "non utilisé comme référence".

- L'image m est marquée comme "non existante" ou son moment de sortie du DPB est inférieur ou égal au moment de retrait du CPB de l'image en cours n ; c'est-à-dire que $t_{o,dpb}(m) \leq t_r(n)$.

Lorsqu'une trame ou le dernier champ dans une mémoire tampon de trame est retiré du DPB, le remplissage du DPB est décrémenté de un.

C.2.4 Marquage et mémorisation de l'image décodée en cours

C.2.4.1 Marquage et mémorisation d'une image décodée de référence dans le DPB

Lorsque l'image en cours est une image de référence, elle est mémorisée dans le DPB comme suit.

- Si l'image décodée en cours est un second champ (dans l'ordre de décodage) d'une paire de champs de référence complémentaires, et que le premier champ de la paire est toujours dans le DPB, l'image décodée en cours est mémorisée dans la même mémoire tampon de trame que le premier champ de la paire.
- Autrement, l'image décodée en cours est mémorisée dans une mémoire tampon de trame vide, et le remplissage de DPB est incrémenté de un.

C.2.4.2 Mémorisation d'une image de non-référence dans le DPB

Lorsque l'image en cours est une image de non-référence et que l'image n en cours a $t_{o,dpb}(n) > t_r(n)$, elle est mémorisée dans le DPB comme suit.

- Si l'image décodée en cours est un second champ (dans l'ordre de décodage) d'une paire de champs de non-référence complémentaires, et que le premier champ de la paire est encore dans le DPB, l'image décodée en cours est mémorisée dans la même mémoire tampon de trame que le premier champ de la paire.
- Autrement, l'image décodée en cours est mémorisée dans une mémoire tampon de trame vide, et le remplissage de DPB est incrémenté de un.

C.3 Conformité de flux binaire

Un flux binaire de données codées conforme à la présente Recommandation | Norme internationale doit satisfaire aux exigences suivantes.

Le flux binaire est construit conformément à la syntaxe, à la sémantique, et aux contraintes spécifiées dans la présente Recommandation | Norme internationale hors de la présente annexe.

Le flux binaire est testé par le HRD comme spécifié ci-dessous:

pour les flux binaires de Type I, le nombre d'essais effectués est égal à $cpb_cnt_minus1 + 1$, où cpb_cnt_minus1 est l'élément syntaxique de `hrd_parameters()` suivant le `vcl_hrd_parameters_present_flag` ou est déterminé par l'application par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale. Un essai est effectué pour chaque combinaison de débit binaire et de taille de CPB spécifiée par les `hrd_parameters()` suivant le `vcl_hrd_parameters_present_flag`. Chacun de ces essais est effectué au niveau du point de conformité de Type I représenté sur la Figure C-1.

Pour les flux binaires de Type II, il y a deux ensembles d'essais. Le nombre d'essais du premier ensemble est égal à $cpb_cnt_minus1 + 1$ où cpb_cnt_minus1 est l'élément syntaxique de `hrd_parameters()` suivant le `vcl_hrd_parameters_present_flag` ou est déterminé par l'application par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale. Un essai est effectué pour chaque combinaison de débit binaire et de taille de CPB. Chacun de ces essais est effectué au niveau du point de conformité de Type I représenté sur la Figure C-1. Pour ces essais, seules les unités NAL VCL et de données de remplissage sont comptées pour le débit binaire d'entrée et de mémorisation du CPB.

Le nombre d'essais du second ensemble, pour les flux binaires de Type II, est égal à $\text{cpb_cnt_minus1} + 1$, où cpb_cnt_minus1 est l'élément syntaxique de $\text{hrd_parameters}()$ suivant le $\text{nal_hrd_parameters_present_flag}$ ou est déterminé par l'application par d'autres moyens non spécifiés dans la présente Recommandation | Norme internationale. Un essai est effectué pour chaque combinaison de débit binaire et de taille de CPB spécifiée par les $\text{hrd_parameters}()$ suivant le $\text{nal_hrd_parameters_present_flag}$. Chacun de ces essais est effectué au niveau du point de conformité de Type II représenté sur la Figure C-1. Pour ces essais, toutes les unités NAL (d'un flux d'unités NAL de Type II) ou tous les octets (d'un flux d'octets) sont comptés pour le débit binaire d'entrée et de mémorisation du CPB.

NOTE – Les paramètres NAL HRD établis par une valeur de SchedSelIdx pour les points de conformité de Type II représentés sur la Figure C-1 sont suffisants pour établir également la conformité VCL HRD pour le point de conformité de Type I représenté sur la Figure C-1 pour les mêmes valeurs de $\text{initial_cpb_removal_delay}[\text{SchedSelIdx}]$, $\text{BitRate}[\text{SchedSelIdx}]$ et $\text{CpbSize}[\text{SchedSelIdx}]$ pour le cas VBR ($\text{cbr_flag}[\text{SchedSelIdx}]$ égal à 0). Cela s'explique par le fait que le flux de données dans le point de conformité de Type I est un sous-ensemble du flux de données dans le point de conformité de Type II et par le fait que, pour le cas VBR, le CPB est autorisé à se vider et à rester vide jusqu'au moment où une nouvelle image devrait normalement commencer à arriver. Par exemple, lorsque les paramètres NAL HRD fournis pour le point de conformité de Type II entrent non seulement dans les limites de conformité de profil fixées pour ces paramètres au point j) du § A.3.1, mais aussi dans les limites de conformité de profil fixées pour les paramètres VCL HRD au point i) du § A.3.1, on peut également être assuré que les paramètres VCL HRD pour le point de conformité de Type I entreront dans les limites de conformité indiquées au point i) du § A.3.1.

Pour les flux binaires conformes, toutes les conditions suivantes doivent être satisfaites pour chacun des essais.

- Pour chaque unité d'accès n , avec $n > 0$, associée à un message SEI de période de mise en mémoire tampon, avec $\Delta t_{g,90}(n)$ spécifié par

$$\Delta t_{g,90}(n) = 90000 * (t_{r,n}(n) - t_{af}(n-1)) \quad (\text{C-14})$$

la valeur de $\text{initial_cpb_removal_delay}[\text{SchedSelIdx}]$ doit respecter les contraintes suivantes:

- Si $\text{cbr_flag}[\text{SchedSelIdx}]$ est égal à 0,

$$\text{initial_cpb_removal_delay}[\text{SchedSelIdx}] \leq \text{Ceil}(\Delta t_{g,90}(n)) \quad (\text{C-15})$$

- Autrement ($\text{cbr_flag}[\text{SchedSelIdx}]$ est égal à 1),

$$\text{Floor}(\Delta t_{g,90}(n)) \leq \text{initial_cpb_removal_delay}[\text{SchedSelIdx}] \leq \text{Ceil}(\Delta t_{g,90}(n)) \quad (\text{C-16})$$

NOTE – Le nombre exact de bits dans le CPB au moment de retrait de chaque image peut dépendre du message SEI de période de mise en mémoire tampon choisi pour initialiser le HRD. Les codeurs doivent en tenir compte pour faire en sorte que toutes les contraintes spécifiées soient respectées quel que soit le message SEI de période de mise en mémoire tampon choisi pour initialiser le HRD, car le HRD peut être initialisé à chacun des messages SEI de période de mise en mémoire tampon.

- Le débordement d'un CPB est spécifié comme l'état dans lequel le nombre total de bits dans le CPB est supérieur à la taille de celui-ci. Le CPB ne doit jamais déborder.
- Le sous-remplissage d'un CPB est spécifié comme l'état dans lequel $t_{r,n}(n)$ est inférieur à $t_{af}(n)$. Lorsque $\text{low_delay_hrd_flag}$ est égal à 0, le CPB ne doit jamais être sous-rempli.
- Le moment nominal de retrait des images du CPB (commençant à la seconde image dans l'ordre de décodage), doit satisfaire aux contraintes sur $t_{r,n}(n)$ et $t_r(n)$ exprimées aux § A.3.1 et A.3.2 pour le profil et niveau spécifié dans le flux binaire.
- Immédiatement après que toute image décodée est ajoutée au DPB, le remplissage du DPB doit être inférieur ou égal à la taille DPB telle qu'elle est spécifiée par les Annexes A, D, et E pour le profil et le niveau spécifiés dans le flux binaire.

- Toutes les images de référence doivent être présentes dans le DPB lorsqu'elles sont nécessaires pour la prédiction. Chaque image doit être présente dans le DPB à son moment de sortie du DPB à moins qu'elle ne soit pas mémorisée du tout dans le DPB, ou ne soit retirée du DPB avant son moment de sortie par un des processus spécifiés au § C.2.
- La valeur de $\Delta t_{o,dpb}(n)$ telle que donnée par l'équation C-13, qui est la différence entre le moment de sortie d'une image et celui de l'image la suivant immédiatement dans l'ordre de sortie, doit satisfaire à la contrainte exprimée au § A.3.1 pour le profil et le niveau spécifiés dans le flux binaire.

C.4 Conformité du décodeur

Un décodeur conforme à la présente Recommandation | Norme internationale doit satisfaire aux exigences suivantes.

Un décodeur prétendant à la conformité à un profil et niveau spécifiques doit être capable de décoder avec succès tous les flux binaires conformes spécifiés pour la conformité du décodeur au § C.3, pourvu que les ensembles de paramètres de séquence, les ensembles de paramètres d'image mentionnés dans les unités NAL VCL et les messages SEI appropriés de période de mise en mémoire tampon et de synchronisation d'image soient convoyés au décodeur, en temps utile, dans le flux binaire (par des unités NAL non-VCL), ou par des moyens externes non spécifiés par la présente Recommandation | Norme internationale.

Il y a deux types de conformité qui peuvent être revendiqués par un décodeur: une conformité de synchronisation de sortie et une conformité d'ordre de sortie.

Pour vérifier la conformité d'un décodeur, les essais de flux binaires conformes au profil et niveau revendiqué, comme spécifié au § C.3 sont fournis par un ordonnanceur fictif de flux (HSS, *hypothetical stream scheduler*) à la fois au HRD et au décodeur soumis à l'essai (DUT: *decoder under test*). Toutes les images sorties par le HRD doivent aussi être sorties par le DUT et, pour chaque image sortie par le HRD, les valeurs de tous les échantillons qui sont sortis par le DUT pour l'image correspondante doivent être égales aux valeurs des échantillons sortis par le HRD.

Pour la conformité de synchronisation de sortie du décodeur, le HSS fonctionne comme décrit ci-dessus, avec des schémas de livraison choisis uniquement parmi le sous-ensemble de valeurs de SchedSelIdx pour lequel le débit binaire et la taille de CPB sont soumis à des restrictions spécifiées à l'Annexe A pour le profil et niveau spécifié, ou avec des schémas de livraison "interpolés" tels que spécifiés ci-dessous pour lesquels le débit binaire et la taille de CPB sont soumises à des restrictions spécifiées à l'Annexe A. Le même schéma de livraison est utilisé pour le HRD et le DUT.

Lorsque les paramètres HRD et les messages SEI de période de mise en mémoire tampon sont présents avec `cpb_cnt_minus1` supérieur à 0, le décodeur doit être capable de décoder le flux binaire tel que livré du HSS en fonctionnant à l'aide du schéma de livraison "interpolé" spécifié comme ayant un débit binaire de pointe r , une taille de CPB $c(r)$, et un délai initial de retrait de CPB ($f(r) \div r$) comme suit

$$\alpha = (r - \text{BitRate}[\text{SchedSelIdx} - 1]) \div (\text{BitRate}[\text{SchedSelIdx}] - \text{BitRate}[\text{SchedSelIdx} - 1]), \quad (\text{C-17})$$

$$c(r) = \alpha * \text{CpbSize}[\text{SchedSelIdx}] + (1 - \alpha) * \text{CpbSize}[\text{SchedSelIdx} - 1], \quad (\text{C-18})$$

$$f(r) = \alpha * \text{initial_cpb_removal_delay}[\text{SchedSelIdx}] * \text{BitRate}[\text{SchedSelIdx}] + (1 - \alpha) * \text{initial_cpb_removal_delay}[\text{SchedSelIdx} - 1] * \text{BitRate}[\text{SchedSelIdx} - 1] \quad (\text{C-19})$$

pour tout `SchedSelIdx > 0` et r tels que $\text{BitRate}[\text{SchedSelIdx} - 1] \leq r \leq \text{BitRate}[\text{SchedSelIdx}]$ tel que r et $c(r)$ soient au sein des limites spécifiées à l'Annexe A pour le débit binaire et la taille de mémoire tampon maximum pour les profil et niveau spécifiés.

NOTE – `initial_cpb_removal_delay[SchedSelIdx]` peut être différent d'une période de mise en mémoire tampon à une autre et devoir être recalculé.

Pour la conformité de synchronisation de sortie du décodeur, on utilise un HRD comme décrit ci-dessus et la synchronisation (par rapport au moment de livraison du premier bit) de la sortie d'image est la même pour le HRD et le DUT jusqu'à l'écoulement d'un délai fixé.

Pour la conformité de l'ordre de sortie du décodeur, le HSS fournit le flux binaire au DUT "à la demande" du DUT, ce qui signifie que le HSS ne délivre les bits (dans l'ordre de décodage) que lorsque le DUT a besoin de plus de bits pour poursuivre son traitement. On utilise un HRD tel que décrit ci-dessus, et le HSS délivre le flux binaire au HRD selon un des schémas spécifiés dans le flux binaire ou par un schéma "interpolé" tel que le débit binaire et la taille du CPB suivent les limites spécifiées à l'Annexe A. L'ordre de sortie des images doit être le même pour le HRD et le DUT.

NOTE – Ceci signifie que pour cet essai, la mémoire tampon de l'image codée du DUT pourrait être aussi petite que la taille de la plus grande unité d'accès.

Pour la conformité de l'ordre de sortie du décodeur, la taille du CPB HRD est égale à $CpbSize[SchedSelIdx]$ pour le schéma choisi et la taille du DPB est égale à $MaxDpbSize$. Le moment de retrait du CPB pour le HRD est égal au moment final d'arrivée de bit et le décodage est immédiat. Le fonctionnement du DPB de ce HRD est décrit ci-dessous.

C.4.1 Fonctionnement de l'ordre de sortie du DPB

La mémoire tampon de l'image décodée contient des mémoires tampon de trame. Chacune des mémoires tampon de trame peut contenir une trame décodée, une paire de champs complémentaires décodés ou un seul (non apparié) champ décodé qui est marqué comme "utilisé pour référence" ou est gardé pour une sortie future (images réordonnées). A l'initialisation du HRD, le remplissage du DPB, mesuré en trames, est mis à 0. Les étapes suivantes surviennent toutes instantanément lorsqu'une unité d'accès est retirée du CPB, et dans l'ordre figurant sur la liste.

C.4.2 Décodage des trous dans `frame_num` et mémorisation d'images "non existantes"

Le cas échéant, les trous dans `frame_num` sont détectés par le processus de décodage et le nombre nécessaire de trames "non existante" sont déduites dans l'ordre spécifié par la génération des valeurs de `UnusedShortTermFrameNum` dans l'équation 7-10 et marquées comme spécifié au § 8.2.5.2. Chaque trame "non existante" est mémorisée dans le DPB comme suit:

- Lorsqu'il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage de DPB est égal à la taille de DPB), le processus de "bumping" spécifié au § C.4.5.3 est invoqué de façon répétée jusqu'à ce que l'on trouve une mémoire tampon de trame vide dans laquelle mémoriser la trame "non existante".
- La trame "non existante" est mémorisée dans une mémoire tampon de trame vide et est marquée comme "non nécessaire pour sortie" et le remplissage de DPB est incrémenté de un.

C.4.3 Décodage d'image

L'image codée primaire n est décodée et est mémorisée temporairement (mais pas dans le DPB).

C.4.4 Retrait d'images du DPB avant l'insertion possible de l'image en cours

Le retrait des images du DPB avant que ne soit possible l'insertion de l'image en cours s'effectue comme suit:

- Si l'image décodée est une image IDR on applique ce qui suit.
 - Toutes les images de référence dans le DPB sont marquées comme "non utilisé comme référence" comme spécifié au § 8.2.5.
 - Lorsque l'image IDR n'est pas la première image IDR décodée et que la valeur de `PicWidthInMbs` (*largeur d'image en macroblocs*) ou `FrameHeightInMbs` (*hauteur de trame en macroblocs*) ou `max_dec_frame_buffering` (*maximum de mémoire tampon de*

trame décodée) déduite de l'ensemble actif de paramètres de séquence est différent de la valeur de, respectivement, `PicWidthInMbs` ou `FrameHeightInMbs` ou `max_dec_frame_buffering`, déduite de l'ensemble de paramètres de séquence qui était actif pour la séquence précédente, `no_output_of_prior_pics_flag` (*fanion pas de sortie d'images précédentes*) est supposé être égal à 1 par le HRD, sans considération de la valeur réelle de `no_output_of_prior_pics_flag`.

NOTE – Les implémentations de décodeurs devraient essayer de traiter les changements de taille de trame ou de DPB plus harmonieusement que le HRD en ce qui concerne les changements dans `PicWidthInMbs` ou `FrameHeightInMbs`.

- Lorsque `no_output_of_prior_pics_flag` est égal à 1 ou est supposé être égal à 1, toutes les mémoires tampon de trame dans le DPB sont vidées sans sortie des images qu'elles contiennent, et le remplissage du DPB est mis à 0.
- Autrement (l'image décodée n'est pas une image IDR), le processus de marquage d'image de référence décodée est invoqué comme spécifié au § 8.2.5. Les mémoires tampon de trame contenant une trame ou une paire de champs complémentaires ou un champ non apparié marqués comme "non nécessaire pour sortie" et "non utilisé comme référence" sont vidées (sans sortie) et le remplissage de DPB est décrémenté du nombre de mémoires tampon de trame vidées.

Lorsque l'image en cours est une image IDR et que `no_output_of_prior_pics_flag` n'est pas égal à 1 et n'est pas supposé être égal à 1, ou que l'image en cours a `memory_management_control_operation` égal à 5, toutes les mémoires tampon de trame qui ne sont pas vides dans le DPB sont vidées en invoquant de façon répétée le processus de "bumping" spécifié au § C.4.5.3, et le remplissage du DPB est mis à 0.

C.4.5 Marquage et mémorisation de l'image décodée en cours

C.4.5.1 Mémorisation et marquage de l'image décodée de référence dans le DPB

Lorsque l'image en cours est une image de référence, elle est mémorisée dans le DPB comme suit.

- Si l'image décodée en cours est le second champ (dans l'ordre de décodage) d'une paire de champs de référence complémentaires, et que le premier champ de la paire est toujours dans le DPB, l'image en cours est mémorisée dans la même mémoire tampon de trame que le premier champ de la paire.
- Autrement, les opérations suivantes sont effectuées:
 - Lorsqu'il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage de DPB est égal à la taille de DPB), le processus de "bumping" spécifié au § C.4.5.3 est invoqué de façon répétée jusqu'à ce que l'on trouve une mémoire tampon de trame vide dans laquelle mémoriser l'image décodée en cours.
 - L'image décodée en cours est mémorisée par une mémoire tampon de trame vide et est marquée comme "nécessaire pour sortie", et le remplissage de DPB est incrémenté de un.

C.4.5.2 Mémorisation et marquage d'une image décodée de non-référence dans le DPB

Lorsque l'image en cours est une image de non-référence, les opérations suivantes sont effectuées.

- Si l'image décodée en cours est le second champ (dans l'ordre de décodage) d'une paire de champs de non-référence complémentaires, et que le premier champ de la paire est toujours dans le DPB, l'image en cours est mémorisée dans la même mémoire tampon de trame que le premier champ de la paire.
- Autrement, les opérations suivantes sont effectuées de façon répétée jusqu'à ce que l'image décodée en cours soit recadrée et sortie ou qu'elle soit mémorisée dans le DPB:

- s'il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage de DPB est égal à la taille de DPB), on applique ce qui suit.
 - Si l'image en cours n'a pas une valeur de PicOrderCnt() inférieure à celle de toutes les images du DPB qui sont marquées comme "nécessaire pour sortie", le processus de "bumping" spécifié au § C.4.5.3 est mis en œuvre.
 - Autrement (l'image en cours a une valeur de PicOrderCnt() inférieure à celle de toutes les images du DPB qui sont marquées comme "nécessaire pour sortie"), l'image en cours est recadrée, en utilisant la mire spécifiée dans l'ensemble de paramètres de séquence pour la séquence et l'image recadrée est sortie.
- Autrement (il y a une mémoire tampon de trame vide, c'est-à-dire que le remplissage de DPB est inférieur à la taille de DPB) l'image décodée en cours est mémorisée dans une mémoire tampon de trame vide et est marquée comme "nécessaire pour sortie", et le remplissage de DPB est incrémenté de un.

C.4.5.3 Processus de "bumping"

Le processus de "bumping" est invoqué dans les cas suivants:

- Il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage de DPB est égal à la taille de DPB) et une mémoire tampon de trame vide est nécessaire pour mémoriser une trame "non existante" déduite, comme spécifié au § C.4.2.
- L'image en cours est une image IDR et no_output_of_prior_pics_flag n'est pas égal à 1 et n'est pas supposée être égale à 1, comme indiqué au § C.4.4.
- L'image en cours a memory_management_control_operation égal à 5, comme spécifié au § C.4.4.
- Il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage de DPB est égal à la taille de DPB) et une mémoire tampon de trame vide est nécessaire pour mémoriser une image de référence (non IDR) décodée, comme spécifié au § C.4.5.1.
- Il n'y a pas de mémoire tampon de trame vide (c'est-à-dire que le remplissage de DPB est égal à la taille de DPB) et l'image en cours est une image de non-référence qui n'est pas le second champ d'une paire de champs de non-référence complémentaires et il y a des images dans le DPB qui sont marquées comme "nécessaire pour sortie" qui précèdent l'image de non-référence en cours dans l'ordre de sortie, comme spécifié au § C.4.5.2, ce qui rend nécessaire une mémoire tampon vide pour mémoriser l'image en cours.

Le processus "bumping" consiste en ce qui suit:

- l'image ou la paire de champs de référence complémentaires qui arrive en premier pour la sortie est sélectionnée comme suit.
 - On sélectionne la mémoire tampon de trame qui contient l'image ayant la plus faible valeur de PicOrderCnt() de toutes les images du DPB marquées comme "nécessaire pour sortie".
 - Si cette mémoire tampon de trame contient une paire de champs de non-référence complémentaires avec les deux champs marqués comme "nécessaire pour sortie" et que les deux champs ont le même PicOrderCnt(), le premier des deux champs dans l'ordre de décodage est considéré comme le premier pour la sortie.
 - Autrement, si cette mémoire tampon de trame contient une paire de champs de référence complémentaires avec les deux champs marqués comme "nécessaire pour sortie" et que les deux champs ont le même PicOrderCnt(), la paire de champs de référence complémentaires entière est considérée comme la première pour la sortie.
 - Autrement, l'image dans cette mémoire tampon de trame qui a la plus faible valeur de PicOrderCnt() est considérée comme la première pour la sortie.

- Si une image unique est considérée comme la première pour la sortie, cette image est recadrée, en utilisant la mire spécifiée dans l'ensemble de paramètres de séquence pour la séquence, l'image recadrée est sortie, et l'image est marquée comme "non nécessaire pour sortie".
- Autrement (une paire de champs de référence complémentaires est considérée la première pour la sortie), les deux champs de la paire de champs de référence complémentaires sont recadrés, en utilisant la mire spécifiée dans l'ensemble de paramètres de séquence pour la séquence, les deux champs de la paire de champs de référence complémentaires sont sortis ensemble, et les deux champs de la paire de champs de référence complémentaires sont marqués comme "non nécessaire pour sortie".
- La mémoire tampon de trame qui incluait l'image ou la paire de champs de référence complémentaires qui a été recadrée et sortie est vérifiée, et lorsque aucune des conditions suivantes est satisfaite, la mémoire tampon de trame est vidée et le remplissage de DPB est décrémenté de 1.
 - La mémoire tampon de trame contient un champ de non-référence non appareillé.
 - La mémoire tampon de trame contient une trame de non-référence.
 - La mémoire tampon de trame contient une paire de champs de non-référence complémentaires avec les deux champs marqués comme "non nécessaire pour sortie".
 - La mémoire tampon de trame contient un champ de référence non appareillé marqué comme "non utilisé comme référence".
 - La mémoire tampon de trame contient une trame de référence avec les deux champs marqués comme "non utilisé comme référence".
 - La mémoire tampon de trame contient une paire de champs de référence complémentaires avec les deux champs marqués comme "non utilisé comme référence" et "non nécessaire pour sortie".

Annexe D

Informations d'amélioration supplémentaires

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe spécifie la syntaxe et la sémantique pour les charges utiles de message SEI.

Les messages SEI aident aux processus se rapportant au décodage, à l'affichage et à d'autres fins. Cependant, les messages SEI ne sont pas nécessaires pour construire les échantillons luma ou chroma par le processus de décodage. Les décodeurs conformes ne sont pas tenus de traiter ces informations pour la conformité de l'ordre de sortie à la présente Recommandation | Norme internationale (voir l'Annexe C pour la spécification de la conformité). Certaines informations de messages SEI sont nécessaires pour vérifier la conformité du flux binaire et celle du délai de sortie du décodeur.

A l'Annexe D, les spécifications de présence des messages SEI sont aussi satisfaites lorsque ces messages (ou certains de leurs sous-ensembles) sont convoyés aux décodeurs (ou au HRD) par d'autres moyens non spécifiés par la présente Recommandation | Norme internationale. Lorsqu'ils sont présents dans le flux binaire, les messages SEI doivent respecter la syntaxe et la sémantique spécifiée aux § 7.3.2.3 et 7.4.2.3 et dans la présente annexe. Lorsque le contenu d'un message SEI est convoyé pour être appliqué par des moyens autres que la présence au sein du flux binaire, la représentation du contenu du message SEI n'est pas nécessaire pour l'utilisation de la même syntaxe que spécifiée dans la présente annexe. Pour les besoins du comptage des bits, seuls les bits appropriés qui sont réellement présents dans le flux binaire sont comptés.

D.1 Syntaxe de la charge utile SEI

	C	Descripteur
sei_payload(payloadType, payloadSize) {		
si(payloadType == 0)		
buffering_period(payloadSize)	5	
ou si(payloadType == 1)		
pic_timing(payloadSize)	5	
ou si(payloadType == 2)		
pan_scan_rect(payloadSize)	5	
ou si(payloadType == 3)		
filler_payload(payloadSize)	5	
ou si(payloadType == 4)		
user_data_registered_itu_t_t35(payloadSize)	5	
ou si(payloadType == 5)		
user_data_unregistered(payloadSize)	5	
ou si(payloadType == 6)		
recovery_point(payloadSize)	5	
ou si(payloadType == 7)		
dec_ref_pic_marking_repetition(payloadSize)	5	
ou si(payloadType == 8)		
spare_pic(payloadSize)	5	
ou si(payloadType == 9)		
scene_info(payloadSize)	5	
ou si(payloadType == 10)		
sub_seq_info(payloadSize)	5	
ou si(payloadType == 11)		
sub_seq_layer_characteristics(payloadSize)	5	
ou si(payloadType == 12)		
sub_seq_characteristics(payloadSize)	5	
ou si(payloadType == 13)		
full_frame_freeze(payloadSize)	5	
ou si(payloadType == 14)		
full_frame_freeze_release(payloadSize)	5	
ou si(payloadType == 15)		
full_frame_snapshot(payloadSize)	5	
ou si(payloadType == 16)		
progressive_refinement_segment_start(payloadSize)	5	
ou si(payloadType == 17)		
progressive_refinement_segment_end(payloadSize)	5	
ou si(payloadType == 18)		
motion_constrained_slice_group_set(payloadSize)	5	
ou		
reserved_sei_message(payloadSize)	5	
si(!byte_aligned()) {		
bit_equal_to_one /* égal à 1 */	5	f(1)
tandis que (!byte_aligned())		
bit_equal_to_zero /* égal à 0 */	5	f(1)
}		
}		

D.1.1 Syntaxe de message SEI de période de mise en mémoire tampon

	C	Descripteur
buffering_period(payloadSize) {		
seq_parameter_set_id	5	ue(v)
si(NalHrdBpPresentFlag) {		
pour(SchedSelIdx = 0; SchedSelIdx <= cpb_cnt_minus1; SchedSelIdx++) {		
initial_cpb_removal_delay [SchedSelIdx]	5	u(v)
initial_cpb_removal_delay_offset [SchedSelIdx]	5	u(v)
}		
}		
si(VclHrdBpPresentFlag) {		
pour(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 Syntaxe de message SEI de synchronisation d'image

	C	Descripteur
pic_timing(payloadSize) {		
si(CpbDpbDelaysPresentFlag) {		
cpb_removal_delay	5	u(v)
dpb_output_delay	5	u(v)
}		
si(pic_struct_present_flag) {		
pic_struct	5	u(4)
pour(i = 0; i < NumClockTS ; i++) {		
clock_timestamp_flag[i]	5	u(1)
si(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_droppeded_flag	5	u(1)
n_frames	5	u(8)
si(full_timestamp_flag) {		
seconds_value /* 0..59 */	5	u(6)
minutes_value /* 0..59 */	5	u(6)
hours_value /* 0..23 */	5	u(5)
} ou {		
seconds_flag	5	u(1)
si(seconds_flag) {		
seconds_value /* gamme 0..59 */	5	u(6)
minutes_flag	5	u(1)
si(minutes_flag) {		
minutes_value /* 0..59 */	5	u(6)
hours_flag	5	u(1)
si(hours_flag)		
hours_value /* 0..23 */	5	u(5)
}		
}		
}		
si(time_offset_length > 0)		
time_offset	5	i(v)
}		
}		
}		
}		

D.1.3 Syntaxe de message SEI de rectangle de balayage complet

	C	Descripteur
pan_scan_rect(payloadSize) {		
pan_scan_rect_id	5	ue(v)
pan_scan_rect_cancel_flag	5	u(1)
si(!pan_scan_rect_cancel_flag) {		
pan_scan_cnt_minus1	5	ue(v)
pour(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 Syntaxe de message SEI de charge utile de remplissage

	C	Descripteur
filler_payload(payloadSize) {		
pour(k = 0; k < payloadSize; k++)		
ff_byte /* égal à 0xFF */	5	f(8)
}		

D.1.5 Syntaxe de message SEI de données d'utilisateur enregistrées par la Rec. UIT-T T.35

	C	Descripteur
user_data_registered_itu_t_t35(payloadSize) {		
itu_t_t35_country_code	5	b(8)
si(itu_t_t35_country_code != 0xFF)		
i = 1		
ou {		
itu_t_t35_country_code_extension_byte	5	b(8)
i = 2		
}		
faire {		
itu_t_t35_payload_byte	5	b(8)
i++		
} tandis que(i < payloadSize)		
}		

D.1.6 Syntaxe de message SEI de données d'utilisateur non enregistrées

	C	Descripteur
user_data_unregistered(payloadSize) {		
uuid_iso_iec_11578	5	u(128)
pour(i = 16; i < payloadSize; i++)		
user_data_payload_byte	5	b(8)
}		

D.1.7 Syntaxe de message SEI de point de récupération

	C	Descripteur
recovery_point(payloadSize) {		
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 Syntaxe de message SEI de marquage d'image de référence décodée

	C	Descripteur
dec_ref_pic_marking_repetition(payloadSize) {		
original_idr_flag	5	u(1)
original_frame_num	5	ue(v)
si(!frame_mbs_only_flag) {		
original_field_pic_flag	5	u(1)
si(original_field_pic_flag)		
original_bottom_field_flag	5	u(1)
}		
dec_ref_pic_marking()	5	
}		

D.1.9 Syntaxe de message SEI d'image de rechange

	C	Descripteur
spare_pic(payloadSize) {		
target_frame_num	5	ue(v)
spare_field_flag	5	u(1)
si(spare_field_flag)		
target_bottom_field_flag	5	u(1)
num_spare_pics_minus1	5	ue(v)
pour(i = 0; i < num_spare_pics_minus1 + 1; i++) {		
delta_spare_frame_num[i]	5	ue(v)
si(spare_field_flag)		
spare_bottom_field_flag[i]	5	u(1)
spare_area_idc[i]	5	ue(v)
si(spare_area_idc[i] == 1)		
pour(j = 0; j < PicSizeInMapUnits; j++)		
spare_unit_flag[i][j]	5	u(1)
ou si(spare_area_idc[i] == 2) {		
mapUnitCnt = 0		
pour(j=0; mapUnitCnt < PicSizeInMapUnits; j++) {		
zero_run_length[i][j]	5	ue(v)
mapUnitCnt += zero_run_length[i][j] + 1		
}		
}		
}		
}		
}		

D.1.10 Syntaxe de message SEI d'informations de scène

	C	Descripteur
scene_info(payloadSize) {		
scene_info_present_flag	5	u(1)
si(scene_info_present_flag) {		
scene_id	5	ue(v)
scene_transition_type	5	ue(v)
si(scene_transition_type > 3)		
second_scene_id	5	ue(v)
}		
}		

D.1.11 Syntaxe de message SEI d'informations de sous-séquence

	C	Descripteur
sub_seq_info(payloadSize) {		
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)
si(sub_seq_frame_num_flag)		
sub_seq_frame_num	5	ue(v)
}		

D.1.12 Syntaxe de message SEI de caractéristiques de couche de sous-séquence

	C	Descripteur
sub_seq_layer_characteristics(payloadSize) {		
num_sub_seq_layers_minus1	5	ue(v)
pour(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 Syntaxe de message SEI de caractéristiques de sous-séquence

	C	Descripteur
sub_seq_characteristics(payloadSize) {		
sub_seq_layer_num	5	ue(v)
sub_seq_id	5	ue(v)
duration_flag	5	u(1)
si(duration_flag)		
sub_seq_duration	5	u(32)
average_rate_flag	5	u(1)
si(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)
pour(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 Syntaxe de message SEI de gel de trame entière

full_frame_freeze(payloadSize) {	C	Descripteur
full_frame_freeze_repetition_period	5	ue(v)
}		

D.1.15 Syntaxe de message SEI de libération de gel de trame entière

full_frame_freeze_release(payloadSize) {	C	Descripteur
}		

D.1.16 Syntaxe de message SEI de photographie de trame entière

full_frame_snapshot(payloadSize) {	C	Descripteur
snapshot_id	5	ue(v)
}		

D.1.17 Syntaxe de message SEI de début de segment de raffinement progressif

progressive_refinement_segment_start(payloadSize) {	C	Descripteur
progressive_refinement_id	5	ue(v)
num_refinement_steps_minus1	5	ue(v)
}		

D.1.18 Syntaxe de message SEI de fin de segment de raffinement progressif

progressive_refinement_segment_end(payloadSize) {	C	Descripteur
progressive_refinement_id	5	ue(v)
}		

D.1.19 Syntaxe de message SEI d'ensemble de groupes de tranches à mouvement restreint

motion_constrained_slice_group_set(payloadSize) {	C	Descripteur
num_slice_groups_in_set_minus1	5	ue(v)
pour(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)
si(pan_scan_rect_flag)		
pan_scan_rect_id	5	ue(v)
}		

D.1.20 Syntaxe de message SEI réservé

	C	Descripteur
reserved_sei_message(payloadSize) {		
pour(i = 0; i < payloadSize; i++)		
reserved_sei_message_payload_byte	5	b(8)
}		

D.2 Sémantique de charge utile SEI

D.2.1 Sémantique de message SEI de période de mise en mémoire tampon

Lorsque `NalHrdBpPresentFlag` ou `VclHrdBpPresentFlag` sont égaux à 1, un message SEI de période de mise en mémoire tampon peut être associé à toute unité d'accès dans le flux binaire, et un message SEI de période de mise en mémoire tampon doit être associé à chaque unité d'accès IDR et avec chaque unité d'accès associée à un message SEI de point de récupération.

NOTE – Pour certaines applications, la présence fréquente d'un message SEI de période de mise en mémoire tampon peut être souhaitable.

Un message SEI de période de mise en mémoire tampon est spécifié comme ensemble des unités d'accès entre deux instances du message SEI de période de mise en mémoire tampon dans l'ordre de décodage.

seq_parameter_set_id spécifie l'ensemble de paramètres de séquence qui contient la séquence d'attributs HRD. La valeur de `seq_parameter_set_id` doit être égale à la valeur de `seq_parameter_set_id` dans l'ensemble de paramètres d'image mentionné par l'image codée primaire associée au message SEI de période de mise en mémoire tampon. La valeur de `seq_parameter_set_id` doit être dans la gamme de 0 à 31 inclus.

initial_cpb_removal_delay[SchedSelIdx] spécifie le délai pour le `SchedSelIdx`^{ième} CPB entre le moment d'arrivée dans le CPB du premier bit des données codées associées à l'unité d'accès associée au message SEI de période de mise en mémoire tampon et le moment de retrait du CPB des données codées associées à la même unité d'accès, pour la première période de mise en mémoire tampon après l'initialisation du HRD. L'élément syntaxique a une longueur en bits donnée par `initial_cpb_removal_delay_length_minus1 + 1`. Elle est en unités d'horloge à 90 kHz. `initial_cpb_removal_delay[SchedSelIdx]` ne doit pas être égal à 0 et ne doit pas excéder $90000 * (\text{CpbSize}[\text{SchedSelIdx}] \div \text{BitRate}[\text{SchedSelIdx}])$, l'équivalent en temps de la taille du CPB en unités d'horloge à 90 kHz.

initial_cpb_removal_delay_offset[SchedSelIdx] est utilisé pour le `SchedSelIdx`^{ième} CPB en combinaison avec le `cpb_removal_delay` pour spécifier le moment initial de livraison des unités d'accès codées au CPB. Le `initial_cpb_removal_delay_offset[SchedSelIdx]` est en unités d'horloge à 90 kHz. L'élément syntaxique `initial_cpb_removal_delay_offset[SchedSelIdx]` est un code de longueur fixe dont la longueur en bits est donnée par `initial_cpb_removal_delay_length_minus1 + 1`. Cet élément syntaxique n'est pas utilisé par les décodeurs et n'est nécessaire que pour le mesureur de flux fictif (HSS) spécifié à l'Annexe C.

La somme de `initial_cpb_removal_delay[SchedSelIdx]` et `initial_cpb_removal_delay_offset[SchedSelIdx]` doit être constante pour chaque valeur de `SchedSelIdx` sur la séquence vidéo codée tout entière.

D.2.2 Sémantique de message SEI de synchronisation d'image

Lorsque `CpbDpbDelaysPresentFlag` est égal à 1, un message SEI de synchronisation d'image doit être associé à chaque unité d'accès dans le flux binaire.

cpb_removal_delay spécifie combien de tic-tac d'horloge (voir le § E.2.1) on doit attendre après le retrait du CPB de l'unité d'accès associée au plus récent message SEI de période de mise en

mémoire tampon avant de retirer de la mémoire tampon les données de l'unité d'accès associées au message SEI de synchronisation d'image. Cette valeur est aussi utilisée pour calculer le moment le plus tôt possible d'arrivée des données d'unité d'accès dans le CPB pour le HSS, comme spécifié à l'Annexe C. L'élément syntaxique est un code de longueur fixe dont la longueur en bits est donnée par $\text{cpb_removal_delay_length_minus1} + 1$. Le `cpb_removal_delay` est le reste d'un compteur $2^{(\text{cpb_removal_delay_length_minus1} + 1)}$.

La valeur de `cpb_removal_delay` pour la première image dans le flux binaire doit être égale à 0.

dpb_output_delay est utilisé pour calculer le temps de sortie de l'image du DPB. Il spécifie combien de tic-tac d'horloge on doit attendre avant le retrait d'une unité d'accès du CPB avant que l'image décodée puisse être sortie du DPB (voir le § C.2).

NOTE – Une image n'est pas retirée du DPB à son moment de sortie lorsqu'elle est toujours marquée comme "utilisé pour référence à court terme" ou "utilisé pour référence à long terme".

NOTE – Un seul `dpb_output_delay` est spécifié pour une image décodée.

La taille de l'élément syntaxique `dpb_output_delay` est donnée en bits par `dpb_output_delay_length_minus1 + 1`. Lorsque `max_dec_frame_buffering` est égal à 0, `dpb_output_delay` doit être égal à 0.

Le moment de sortie déduit du `dpb_output_delay` de toute image qui est sortie d'un décodeur conforme à la synchronisation de sortie, comme spécifié au § C.2, doit précéder le moment de sortie déduit du `dpb_output_delay` de toutes les images de toute séquence vidéo codée suivante dans l'ordre de décodage.

Le moment de sortie déduit du `dpb_output_delay` du second champ, dans l'ordre de décodage, d'une paire de champs de non-référence complémentaires doit excéder le moment de sortie déduit du `dpb_output_delay` du premier champ de la même paire de champs de non-référence complémentaires.

L'ordre de sortie d'image établi par les valeurs de cet élément syntaxique doit être le même ordre que celui établi par les valeurs de `PicOrderCnt()` comme spécifié par les § C.4.1 à C.4.5, excepté que lorsque les deux champs d'une paire de champs de référence complémentaires ont la même valeur de `PicOrderCnt()`, les deux champs ont des moments de sortie différents.

Pour les images qui ne sont pas sorties par le processus de "bumping" du § C.4.5 parce qu'elles précèdent, dans l'ordre de décodage, une image IDR avec `no_output_of_prior_pics_flag` égal à 1 ou supposé être égal à 1, les moments de sortie déduits de `dpb_output_delay` doivent être croissants avec des valeurs croissantes de `PicOrderCnt()` par rapport à toutes les images au sein de la même séquence vidéo codée à la suite de toute image ayant un `memory_management_control_operation` égal à 5.

pic_struct indique si une image devrait être affichée comme une trame ou un ou plusieurs champs, conformément au Tableau D-1. Doublage de trame (`pic_struct` égal à 7) indique que la trame devrait être affichée deux fois consécutives, et triplage de trame (`pic_struct` égal à 8) indique que la trame devrait être affichée trois fois consécutivement.

NOTE – Le doublage de trame peut faciliter l'affichage, par exemple, de 25p vidéo sur un affichage de 50p et 29,97p vidéo sur un affichage de 59,94p. L'utilisation du doublage de trame et du triplage de trame en combinaison sur chaque autre trame peut faciliter l'affichage de 23,98p vidéo sur un affichage de 59,94p.

Tableau D-1 – Interprétation de pic_struct

Valeur	Affichage d'image indiqué	Restrictions	NumClockTS
0	Trame	field_pic_flag doit être 0	1
1	Champ supérieur	field_pic_flag doit être 1, bottom_field_flag doit être 0	1
2	Champ inférieur	field_pic_flag doit être 1, bottom_field_flag doit être 1	1
3	Champ supérieur, champ inférieur, dans cet ordre	field_pic_flag doit être 0	2
4	Champ inférieur, champ supérieur, dans cet ordre	field_pic_flag doit être 0	2
5	Champ supérieur, champ inférieur, champ supérieur répété, dans cet ordre	field_pic_flag doit être 0	3
6	Champ inférieur, champ supérieur, champ inférieur répété dans cet ordre	field_pic_flag doit être 0	3
7	Doublage de trame	field_pic_flag doit être 0 fixed_frame_rate_flag doit être 1	2
8	Triplage de trame	field_pic_flag doit être 0 fixed_frame_rate_flag doit être 1	3
9..15	Réservé		

NumClockTS est déterminé par pic_struct comme spécifié au Tableau D-1. Il y a jusqu'à NumClockTS ensembles d'informations d'horodatage pour une image, comme spécifié par clock_timestamp_flag[i] pour chaque ensemble. Les ensembles d'informations d'horodatage s'appliquent au(x) champ(s) ou à la (aux) trame(s) associée(s) à l'image par pic_struct.

Le contenu des éléments syntaxiques d'horodatage indique un moment d'origine, de capture, ou d'affichage idéal de remplacement. Ce moment indiqué est calculé comme suit:

$$\text{clockTimestamp} = ((\text{hH} * 60 + \text{mM}) * 60 + \text{sS}) * \text{time_scale} + \text{nFrames} * (\text{num_units_in_tick} * (1 + \text{nuit_field_based_flag})) + \text{tOffset}, \quad (\text{D-1})$$

en unités de tic-tac d'horloge d'une horloge avec une fréquence d'horloge égale à time_scale Hz, par rapport à un point non spécifié dans le temps pour lequel clockTimestamp est égal à 0. L'ordre de sortie et la synchronisation de sortie du DPB ne sont pas affectés par la valeur de clockTimestamp. Lorsque deux, ou plus, trames avec pic_struct égal à 0 sont consécutives dans l'ordre de sortie et ont d'égales valeurs de clockTimestamp, l'indication est que les trames représentent le même contenu et que la dernière de ces trames dans l'ordre de sortie est la représentation préférée.

NOTE – Les indications de temps de clockTimestamp peuvent aider à l'affichage sur des appareils qui ont des taux de rafraîchissement autres que ceux qui sont en pleine correspondance avec les temps de sortie du DPB.

clock_timestamp_flag[i] égal à 1 indique qu'un certain nombre d'éléments syntaxiques d'horodatage sont présents et suivent immédiatement. clock_timestamp_flag[i] égal à 0 indique que les éléments syntaxiques d'horodatage associés ne sont pas présents. Lorsque NumClockTS est supérieur à 1 et que clock_timestamp_flag[i] est égal à 1 pour plus d'une valeur de i, la valeur de clockTimestamp doit être non décroissante avec une valeur de i croissante.

ct_type indique le type de balayage (entrelacé ou progressif) du matériau source comme suit:

deux champs d'une trame codée peuvent avoir différentes valeurs de ct_type.

Lorsque clockTimestamp est égal pour deux champs de parité opposée qui sont consécutifs dans l'ordre de sortie, tous deux avec ct_type égal à 0 (progressif) ou ct_type égal à 2 (inconnu), les deux champs sont indiqués comme provenant de la même trame progressive d'origine. Deux champs consécutifs dans l'ordre de sortie doivent avoir des valeurs de clockTimestamp différentes lorsque la valeur de ct_type pour l'un ou l'autre champ est 1 (entrelacé).

Tableau D-2 – Mappage de ct_type avec le balayage de l'image source

Valeur	Balayage de l'image d'origine
0	Progressif
1	Entrelacé
2	Inconnu
3	Réservé

nuit_field_based_flag: utilisé pour le calcul de clockTimestamp, comme spécifié à l'équation D-1.

counting_type: spécifie la méthode de réduction des valeurs de n_frames comme spécifié au Tableau D-3.

Tableau D-3 – Définition des valeurs de counting_type

Valeur	Interprétation
0	Pas de réduction des valeurs de comptage de n_frames et pas d'utilisation de time_offset
1	Pas de réduction des valeurs de comptage de n_frames
2	Réduction des valeurs zéro individuelles du comptage de n_frames
3	Réduction des valeurs individuelles MaxFPS-1 du comptage de n_frames
4	Réduction des deux comptes les plus faibles (valeur 0 et 1) de n_frames lorsque seconds_value est égal à 0 et que minutes_value n'est pas un multiple entier de 10
5	Réduction des valeurs individuelles non spécifiées de comptage de n_frames
6	Réduction des numéros non spécifiés des valeurs de comptage non spécifié de n_frames
7..31	Réservé

full_timestamp_flag égal à 1 spécifie que l'élément syntaxique n_frames est suivi par seconds_value, minutes_value, et hours_value. full_timestamp_flag égal à 0 spécifie que l'élément syntaxique n_frames est suivi par seconds_flag.

discontinuity_flag égal à 0 indique que la différence entre la valeur en cours de clockTimestamp et la valeur de clockTimestamp calculée d'après l'horodatage précédent dans l'ordre de sortie peut s'interpréter comme la différence de temps entre le moment d'origine ou de capture des trames ou champs associés. discontinuity_flag égal à 1 indique que la différence entre la valeur en cours de clockTimestamp et la valeur de clockTimestamp calculée d'après l'horodatage précédent dans l'ordre de sortie ne devrait pas être interprétée comme la différence de temps entre le moment d'origine ou de capture des trames ou champs associés. Lorsque discontinuity_flag est égal à 0, la valeur de clockTimestamp doit être supérieure ou égale à toutes les valeurs de clockTimestamp présentes pour la précédente image dans l'ordre de sortie du DPB.

cnt_dropped_flag spécifie qu'on saute une ou plusieurs valeurs de n_frames en utilisant la méthode de comptage spécifiée par counting_type.

n_frames spécifie la valeur de nFrames utilisée pour calculer clockTimestamp. n_frames doit être inférieur à

$$\text{MaxFPS} = \text{Ceil}(\text{time_scale} \div \text{num_units_in_tick}) \quad (\text{D-2})$$

NOTE – n_frames est un comptage fondé sur les trames. Pour les indications de synchronisation spécifiques des champs, time_offset devrait être utilisé pour indiquer un clockTimestamp distinct pour chaque champ.

Lorsque counting_type est égal à 2 et que cnt_dropped_flag est égal à 1, n_frames doit être égal à 1 et la valeur de n_frames pour l'image précédente dans l'ordre de sortie ne doit pas être égale à 0 à moins que discontinuity_flag ne soit égal à 1.

NOTE – Lorsque counting_type est égal à 2, on peut éviter d'être obligé d'avoir des valeurs de plus en plus grandes de tOffset dans l'équation D-1 lorsqu'on utilise des débits de trame fixes non entiers (par exemple, 12,5 trames par seconde avec time_scale égal à 25 et num_units_in_tick égal à 2 et nuit_field_based_flag égal à 0) en sautant occasionnellement la valeur n_frames égale à 0 lors du comptage (par exemple, en comptant n_frames de 0 à 12, puis en incrémentant seconds_value et en comptant n_frames de 1 à 12, puis en incrémentant seconds_value et en comptant n_frames de 0 à 12, etc.).

Lorsque counting_type est égal à 3 et que cnt_dropped_flag est égal à 1, n_frames doit être égal à 0 et la valeur de n_frames pour l'image précédente dans l'ordre de sortie ne doit pas être égale à MaxFPS – 1 à moins que discontinuity_flag ne soit égal à 1.

NOTE – Lorsque counting_type est égal à 3, on peut éviter d'être obligé d'avoir des valeurs de plus en plus grandes de tOffset dans l'équation D-1 lorsqu'on utilise des débits de trame fixes non entiers (par exemple, 12,5 trames par seconde avec time_scale égal à 25 et num_units_in_tick égal à 2 et nuit_field_based_flag égal à 0) en sautant occasionnellement la valeur n_frames égale à MaxFPS lors du comptage (par exemple, en comptant n_frames de 0 à 12, puis en incrémentant seconds_value et en comptant n_frames de 1 à 11, puis en incrémentant seconds_value et en comptant n_frames de 0 à 12, etc.).

Lorsque counting_type est égal à 4 et que cnt_dropped_flag est égal à 1, n_frames doit être égal à 2 et la valeur spécifiée de sS doit être zéro et la valeur spécifiée de mM ne doit pas être un multiple entier de dix et n_frames pour l'image précédente dans l'ordre de sortie ne doit pas être égal à 0 ou 1 à moins que discontinuity_flag ne soit égal à 1.

NOTE – Lorsque counting_type est égal à 4, le besoin d'avoir des valeurs de plus en plus grandes de tOffset dans l'équation D-1 lorsqu'on utilise des débits de trame fixes non entiers (par exemple, 30000÷1001 trames par seconde avec time_scale égal à 60000 et num_units_in_tick égal à 1001 et nuit_field_based_flag égal à 1) peut être réduit en sautant occasionnellement la valeur n_frames égale à MaxFPS lors du comptage (par exemple, en comptant n_frames de 0 à 29, puis en incrémentant seconds_value et en comptant n_frames de 0 à 29, etc., jusqu'à ce que seconds_value soit zéro et que minutes_value ne soit pas un multiple entier de dix, puis en comptant n_frames de 2 à 29, puis en incrémentant seconds_value et en comptant n_frames de 0 à 29, etc.). Cette méthode de comptage est bien connue dans l'industrie et est souvent mentionnée comme le comptage "réduction de trame NTSC".

Lorsque counting_type est égal à 5 ou 6 et que cnt_dropped_flag est égal à 1, n_frames ne doit pas être égal à 1 plus la valeur de n_frames pour l'image précédente dans l'ordre de sortie modulo MaxFPS à moins que discontinuity_flag ne soit égal à 1.

NOTE – Lorsque counting_type est égal à 5 ou 6, on peut éviter d'être obligé d'avoir des valeurs de plus en plus grandes de tOffset dans l'équation D-1 lorsqu'on utilise des débits de trame fixes non entiers en sautant occasionnellement certaines valeurs de n_frames lors du comptage. Les valeurs spécifiques de n_frames qui sont sautées ne sont pas spécifiées lorsque counting_type est égal à 5 ou 6.

seconds_flag égal à 1 spécifie que seconds_value et minutes_flag sont présents lorsque full_timestamp_flag est égal à 0. seconds_flag égal à 0 spécifie que seconds_value et minutes_flag ne sont pas présents.

seconds_value spécifie la valeur de sS utilisée pour calculer clockTimestamp. La valeur de seconds_value doit être dans la gamme de 0 à 59 inclus. Lorsque seconds_value n'est pas présent, le seconds_value précédant dans l'ordre de décodage doit être utilisé comme sS pour calculer clockTimestamp.

minutes_flag égal à 1 spécifie que minutes_value et hours_flag sont présents lorsque full_timestamp_flag est égal à 0 et seconds_flag est égal à 1. minutes_flag égal à 0 spécifie que minutes_value et hours_flag ne sont pas présents.

minutes_value spécifie la valeur de mM utilisée pour calculer clockTimestamp. La valeur de minutes_value doit être dans la gamme de 0 à 59 inclus. Lorsque minutes_value n'est pas présent, le minutes_value précédant dans l'ordre de décodage doit être utilisé comme mM pour calculer clockTimestamp.

hours_flag égal à 1 spécifie que hours_value est présent lorsque full_timestamp_flag est égal à 0 et que seconds_flag est égal à 1 et minutes_flag est égal à 1.

hours_value spécifie la valeur de hH utilisée pour calculer clockTimestamp. La valeur de hours_value doit être dans la gamme de 0 à 23 inclus. Lorsque hours_value n'est pas présent, le hours_value précédant dans l'ordre de décodage doit être utilisé comme hH pour calculer clockTimestamp.

time_offset spécifie la valeur de tOffset utilisée pour calculer clockTimestamp. Le nombre de bits utilisé pour représenter time_offset doit être égal à time_offset_length. Lorsque time_offset n'est pas présent, la valeur 0 doit être utilisée comme tOffset pour calculer clockTimestamp.

D.2.3 Sémantique de message SEI de rectangle de balayage complet

Les éléments syntaxiques de message SEI de rectangle de balayage complet spécifient les coordonnées d'un rectangle par rapport à la mire de l'ensemble de paramètres de séquence. Chaque coordonnée de ce rectangle est spécifiée en unités d'espacement d'un seizième d'échantillon par rapport à la grille d'échantillonnage luma.

pan_scan_rect_id contient un numéro d'identification qui peut être utilisé pour identifier l'objet du rectangle de balayage complet (par exemple, pour identifier le rectangle comme la zone à montrer d'un dispositif d'affichage particulier ou comme la zone qui contient un acteur particulier dans la scène). La valeur de pan_scan_rect_id doit être dans la gamme de 0 à $2^{32} - 1$ inclus.

Les valeurs de pan_scan_rect_id de 0 à 255 et de 512 à $2^{31} - 1$ peuvent être utilisées comme déterminé par l'application. Les valeurs de pan_scan_rect_id de 256 à 511 et de 2^{31} à $2^{32} - 1$ sont réservées pour être utilisées à l'avenir par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de pan_scan_rect_id dans la gamme de 256 à 511 ou dans la gamme de 2^{31} à $2^{32} - 1$ doivent l'ignorer (la retirer du flux binaire et la détruire).

pan_scan_rect_cancel_flag égal à 1 indique que le message SEI annule le maintien d'un message SEI précédant de rectangle de balayage complet. pan_scan_rect_cancel_flag égal à 0 indique que le message SEI n'annule pas le maintien d'un message SEI précédant de rectangle de balayage complet et que suivent des informations de rectangle de balayage complet.

pan_scan_cnt_minus1 spécifie le nombre de rectangles de balayage complet qui sont présents dans le message SEI. pan_scan_cnt_minus1 doit être dans la gamme de 0 à 2 inclus. pan_scan_cnt_minus1 égal à 0 indique qu'un seul rectangle de balayage complet est présent et qu'il s'applique à tous les champs de l'image décodée. pan_scan_cnt_minus1 doit être égal à 0 lorsque l'image en cours est un champ. pan_scan_cnt_minus1 égal à 1 indique que deux rectangles de balayage complet sont présents, dont le premier s'applique au premier champ de l'image dans l'ordre de sortie et dont le second s'applique au second champ de l'image dans l'ordre de sortie. pan_scan_cnt_minus1 égal à 2 indique que trois rectangles de balayage complet sont présents, dont le premier s'applique au premier champ de l'image dans l'ordre de sortie, le second s'applique au

second champ de l'image dans l'ordre de sortie, et le troisième s'applique à une répétition du premier champ comme troisième champ dans l'ordre de sortie.

pan_scan_rect_left_offset[i], **pan_scan_rect_right_offset[i]**, **pan_scan_rect_top_offset[i]**, et **pan_scan_rect_bottom_offset[i]** spécifient, comme quantités algébriques entières en unités d'espacement d'un seizième d'échantillon par rapport à la grille d'échantillonnage luma, la localisation du rectangle de balayage complet. Les valeurs de chacun de ces quatre éléments syntaxiques doivent être dans la gamme de -2^{31} à $2^{31} - 1$, inclus.

Le rectangle de balayage complet est spécifié, en unités d'espacement d'un seizième d'échantillon par rapport à une grille d'échantillonnage luma, comme la zone du rectangle avec les coordonnées comme suit:

- si **frame_mbs_only_flag** est égal à 1, le rectangle de balayage complet a des coordonnées horizontales de trame luma de $32 * \text{frame_crop_left_offset} + \text{pan_scan_rect_left_offset}[i]$ à $32 * (8 * \text{PicWidthInMbs} - \text{frame_crop_right_offset}) + \text{pan_scan_rect_right_offset}[i] - 1$ et avec des coordonnées verticales de $32 * \text{frame_crop_top_offset} + \text{pan_scan_rect_top_offset}[i]$ à $32 * (8 * \text{PicHeightInMbs} - \text{frame_crop_bottom_offset}) + \text{pan_scan_rect_bottom_offset}[i] - 1$, inclus. Dans ce cas, la valeur de $32 * \text{frame_crop_left_offset} + \text{pan_scan_rect_left_offset}[i]$ doit être inférieure ou égale à $32 * (8 * \text{PicWidthInMbs} - \text{frame_crop_right_offset}) + \text{pan_scan_rect_right_offset}[i] - 1$; et la valeur de $32 * \text{frame_crop_top_offset} + \text{pan_scan_rect_top_offset}[i]$ doit être inférieure ou égale à $32 * (8 * \text{PicHeightInMbs} - \text{frame_crop_bottom_offset}) + \text{pan_scan_rect_bottom_offset}[i] - 1$;
- autrement (**frame_mbs_only_flag** est égal à 0), le rectangle de balayage complet a des coordonnées horizontales de trame luma de $32 * \text{frame_crop_left_offset} + \text{pan_scan_rect_left_offset}[i]$ à $32 * (8 * \text{PicWidthInMbs} - \text{frame_crop_right_offset}) + \text{pan_scan_rect_right_offset}[i] - 1$ et avec des coordonnées verticales de $64 * \text{frame_crop_top_offset} + \text{pan_scan_rect_top_offset}[i]$ à $64 * (4 * \text{PicHeightInMbs} - \text{frame_crop_bottom_offset}) + \text{pan_scan_rect_bottom_offset}[i] - 1$, inclus. Dans ce cas, la valeur de $32 * \text{frame_crop_left_offset} + \text{pan_scan_rect_left_offset}[i]$ doit être inférieure ou égale à $32 * (8 * \text{PicWidthInMbs} - \text{frame_crop_right_offset}) + \text{pan_scan_rect_right_offset}[i] - 1$; et la valeur de $64 * \text{frame_crop_top_offset} + \text{pan_scan_rect_top_offset}[i]$ doit être inférieure ou égale à $64 * (4 * \text{PicHeightInMbs} - \text{frame_crop_bottom_offset}) + \text{pan_scan_rect_bottom_offset}[i] - 1$.

Lorsque la zone du rectangle de balayage complet inclut des échantillons en dehors de la mire, la région en dehors de la mire peut être remplie avec un contenu synthétisé (comme un contenu vidéo noir ou gris neutre) pour l'affichage.

pan_scan_rect_repetition_period indique si un autre message SEI de rectangle de balayage complet avec la même valeur de **pan_scan_rect_id** doit être présent dans le flux binaire et spécifie l'intervalle de compte d'ordre d'image au sein duquel il sera présent. La valeur de **pan_scan_rect_repetition_period** doit être dans la gamme de 0 à 16 384 inclus. Lorsque **pan_scan_cnt_minus1** est supérieur à 0, **pan_scan_rect_repetition_period** ne doit pas être supérieur à 1.

pan_scan_rect_repetition_period égal à 0 spécifie que les informations de rectangle de balayage complet s'appliquent uniquement à l'image décodée en cours.

pan_scan_rect_repetition_period égal à 1 spécifie que les informations de rectangle de balayage complet persistent dans l'ordre de sortie tant que l'une des conditions suivantes est Vraie.

- Une nouvelle séquence vidéo codée commence.
- Une image dans une unité d'accès contenant un message SEI de rectangle de balayage complet avec la même valeur de **pan_scan_rect_id** est sortie et elle a **PicOrderCnt()** plus grand que **PicOrderCnt(CurrPic)**.

pan_scan_rect_repetition_period égal à 0 ou égal à 1 indique qu'un autre message SEI de rectangle de balayage complet avec la même valeur de pan_scan_rect_id peut être ou n'être pas présente.

pan_scan_rect_repetition_period supérieur à 1 spécifie que les informations de rectangle de balayage complet persistent tant que l'une des conditions suivantes est Vraie.

- Une nouvelle séquence vidéo codée commence.
- Une image dans une unité d'accès contenant un message SEI de rectangle de balayage complet avec la même valeur de pan_scan_rect_id est sortie et elle a PicOrderCnt() supérieur à PicOrderCnt(CurrPic) + pan_scan_rect_repetition_period.

pan_scan_rect_repetition_period supérieur à 1 indique qu'un autre message SEI de rectangle de balayage complet avec la même valeur de pan_scan_rect_id doit être présent pour une image dans une unité d'accès qui est sortie en ayant PicOrderCnt() inférieur ou égal à PicOrderCnt(CurrPic) + pan_scan_rect_repetition_period; à moins qu'une nouvelle séquence vidéo codée ne commence sans sortie d'une telle image.

D.2.4 Sémantique de message SEI de charge utile de remplissage

Ce message contient une série d'octets de payloadSize d'une valeur de 0xFF, qui peut être ignorée.

ff_byte doit être un octet ayant la valeur 0xFF.

D.2.5 Sémantique de message SEI de données d'utilisateur enregistrées par la Rec. UIT-T T.35

Ce message contient des données d'utilisateur enregistrées comme spécifié par la Rec. UIT-T T.35, dont le contenu n'est pas spécifié par la présente Recommandation | Norme internationale.

itu_t_t35_country_code doit être un octet d'une valeur spécifiée comme code de pays par la Rec. UIT-T T.35, Annexe A.

itu_t_t35_country_code_extension_byte doit être un octet d'une valeur spécifiée comme code de pays par la Rec. UIT-T T.35, Annexe B.

itu_t_t35_payload_byte doit être un octet contenant des données enregistrées comme spécifié par la Rec. UIT-T T.35.

Le code de fournisseur de terminal et le code orienté fournisseur de terminal UIT-T T.35 doivent être contenus dans le ou les premiers octets de **itu_t_t35_payload_byte**, dans le format spécifié par l'Administration qui a fourni le code de fournisseur de terminal. Toutes les données restantes de **itu_t_t35_payload_byte** doivent être des données qui ont la syntaxe et la sémantique spécifiées par l'entité identifiée par le code de pays et les codes de fournisseur de terminal UIT-T T.35.

D.2.6 Sémantique de message SEI de données d'utilisateur non enregistrées

Ce message contient des données d'utilisateur non enregistrées identifiées par un UUID, dont le contenu n'est pas spécifié par la présente Recommandation | Norme internationale.

uuid_iso_iec_11578 doit avoir une valeur spécifiée comme un UUID conformément aux procédures de l'ISO/CEI 11578:1996, Annexe A.

user_data_payload_byte doit être un octet contenant des données ayant la syntaxe et la sémantique spécifiées par le générateur d'UUID.

D.2.7 Sémantique de message SEI de point de récupération

Le message SEI de point de récupération aide un décodeur à déterminer lorsque le processus de décodage va produire des images acceptables pour l'affichage après que le décodeur a initialisé un accès aléatoire, ou après que le codeur a indiqué une rupture de liaison dans la séquence. Lorsque le processus de décodage débute avec l'unité d'accès dans l'ordre de décodage associée au message SEI de point de récupération, toutes les images décodées au point de récupération ou à sa suite dans l'ordre de sortie spécifié dans ce message SEI sont indiquées comme étant correctes ou

approximativement correctes quant à leur contenu. Les images décodées produites par accès aléatoire au moment de l'image associée au message SEI de point de récupération, ou avant elle, n'ont pas besoin d'être correctes quant à leur contenu jusqu'au point de récupération indiqué, et l'opération du processus de décodage commençant à l'image associée au message SEI de point de récupération peut contenir des références aux images indisponibles dans la mémoire tampon d'image décodée.

De plus, en utilisant le fanion `broken_link_flag`, le message SEI de point de récupération peut indiquer au décodeur la localisation de certaines images dans le flux binaire qui peuvent produire à l'affichage de sérieuses distorsions visuelles, même lorsque le processus de décodage avait commencé à la localisation d'une unité d'accès IDR précédente dans l'ordre de décodage.

NOTE – Le fanion `broken_link_flag` peut être utilisé par les codeurs pour indiquer la localisation d'un point après lequel le processus de décodage pour le décodage de certaines images peut provoquer la référence à des images qui, bien que disponibles pour une utilisation dans le processus de décodage, ne sont pas les images qui étaient utilisées pour référence lorsque le flux binaire a été codé à l'origine (par exemple, à cause d'une opération de collage effectuée pendant la génération du flux binaire).

Le point de récupération est spécifié comme un compte en unités d'incrémentes `frame_num` suivant le `frame_num` de l'unité d'accès en cours à la position du message SEI.

NOTE – Lorsque des informations HRD sont présentes dans le flux binaire, un message SEI de période de mise en mémoire tampon devrait être associé à l'unité d'accès associée au message SEI de point de récupération afin d'établir l'initialisation du modèle de mémoire tampon HRD après un accès aléatoire.

recovery_frame_cnt spécifie le point de récupération des images sorties dans l'ordre de sortie. Toutes les images décodées dans l'ordre de sortie sont indiquées comme correctes ou approximativement correctes quant à leur contenu en commençant à la position d'ordre de sortie de l'image de référence qui a le `frame_num` égal au `frame_num` des unités NAL VCL pour l'unité d'accès en cours incrémentée de `recovery_frame_cnt` en arithmétique modulo `MaxFrameNum`. `recovery_frame_cnt` doit être dans la gamme de 0 à `MaxFrameNum - 1`, inclus.

exact_match_flag indique si les images décodées au point de récupération et à sa suite dans l'ordre de sortie, calculé en commençant le processus de décodage à l'unité d'accès associée au message SEI de point de récupération, doivent correspondre exactement aux images qui seraient produites en commençant le processus de décodage à la localisation d'une unité d'accès IDR précédente dans le flux d'unités NAL. La valeur 0 indique que la correspondance peut n'être pas exacte et la valeur 1 indique que la correspondance doit être exacte.

Lorsque le décodage débute à la localisation du message SEI de point de récupération, toutes les références à des images de référence indisponibles doivent être comprises comme des références à des images ne contenant que des macroblocs codés utilisant les modes d'intraprédiction de macrobloc et ayant des valeurs d'échantillon données par Y échantillons égaux à 128, Cb échantillons égaux à 128, et Cr échantillons égaux à 128 (gris moyen) pour les besoins de la détermination de la conformité de la valeur de `exact_match_flag`.

NOTE – Lorsqu'on effectue un accès aléatoire, les décodeurs devraient comprendre que toutes les références à des images de référence indisponibles sont des références à des images ne contenant que des macroblocs intra et qui ont des valeurs d'échantillon données par Y égal à 128, Cb égal à 128, et Cr égal à 128 (gris moyen), indépendamment de la valeur de `exact_match_flag`.

Lorsque `exact_match_flag` est égal à 0, la qualité de l'approximation au point de récupération est choisie par le processus de codage et n'est pas spécifiée par la présente Recommandation | Norme internationale.

broken_link_flag indique la présence ou l'absence d'une rupture de liaison dans le flux d'unité NAL à la localisation du message SEI de point de récupération et a la sémantique suivante:

- si `broken_link_flag` est égal à 1, les images produites en commençant le processus de décodage à la localisation d'une unité d'accès IDR précédente peuvent contenir des

distorsions visuelles indésirables à tel point que les images décodées à l'unité d'accès associée au message SEI de point de récupération et à sa suite dans l'ordre de décodage ne devraient pas être affichées jusqu'au point de récupération spécifié dans l'ordre de sortie;

- autrement (`broken_link_flag` est égal à 0), aucune indication n'est donnée concernant la présence potentielle de distorsions visuelles.

Indépendamment de la valeur du fanion `broken_link_flag`, les images à la suite du point de récupération spécifié dans l'ordre de sortie sont spécifiées comme correctes ou approximativement correctes quant à leur contenu.

NOTE – Lorsqu'un message SEI d'information de sous-séquence est présent en conjonction avec un message SEI de point de récupération dans lequel `broken_link_flag` est égal à 1 et lorsque `sub_seq_layer_num` est égal à 0, `sub_seq_id` devrait être différent du dernier `sub_seq_id` pour `sub_seq_layer_num` égal à 0 qui a été décodé avant la localisation du message SEI de point de récupération. Lorsque `broken_link_flag` est égal à 0, le `sub_seq_id` dans la couche 0 de sous-séquence devrait rester inchangé.

`changing_slice_group_idc` égal à 0 indique que les images décodées sont correctes ou approximativement correctes quant à leur contenu au point de récupération et à sa suite dans l'ordre de sortie lorsque tous les macroblocs des images codées primaires sont décodés au sein de la période de groupe de tranches changeant, c'est-à-dire, la période entre l'unité d'accès associée au message SEI de point de récupération (inclus) et le point de récupération spécifié (inclus) dans l'ordre de décodage. `changing_slice_group_idc` doit être égal à 0 lorsque `num_slice_groups_minus1` est égal à 0 dans toute image codée primaire au sein de la période de groupe de tranches changeant.

Lorsque `changing_slice_group_idc` est égal à 1 ou 2, `num_slice_groups_minus1` doit être égal à 1 et le type de mappage 3, 4 ou 5 de macrobloc à groupe de tranche doit être appliqué dans chaque image codée primaire dans la période de groupe de tranches changeant.

`changing_slice_group_idc` égal à 1 indique qu'au sein de la période de groupe de tranches changeant, aucune valeur d'échantillon en dehors des macroblocs décodés couverts par le groupe de tranches 0 n'est utilisée pour l'interprédiction de tout macrobloc au sein du groupe de tranches 0. De plus, `changing_slice_group_idc` égal à 1 indique que lorsque tous les macroblocs dans le groupe de tranches 0 au sein de la période de groupe de tranches changeant sont décodés, les images décodées seront correctes ou approximativement correctes quant à leur contenu au point de récupération spécifié et à sa suite dans l'ordre de sortie indépendamment du décodage éventuel d'un macrobloc dans le groupe de tranches 1 au sein de la période de groupe de tranches changeant.

`changing_slice_group_idc` égal à 2 indique qu'au sein de la période de groupe de tranches changeant, aucune valeur d'échantillon en dehors des macroblocs décodés couverts par le groupe de tranches 1 n'est utilisée pour l'interprédiction de tout macrobloc au sein du groupe de tranches 1. De plus, `changing_slice_group_idc` égal à 2 indique que lorsque tous les macroblocs dans le groupe de tranches 1 au sein de la période de groupe de tranches changeant sont décodés, les images décodées seront correctes ou approximativement correctes quant à leur contenu au point de récupération spécifié et à sa suite dans l'ordre de sortie indépendamment du décodage éventuel d'un macrobloc dans le groupe de tranches 0 au sein de la période de groupe de tranches changeant.

`changing_slice_group_idc` doit être dans la gamme de 0 à 2, inclus.

D.2.8 Sémantique de message SEI de répétition de marquage d'image de référence décodée

Le message SEI de répétition de marquage d'image de référence décodée est utilisé pour répéter la structure syntaxique de marquage d'image de référence décodée qui était localisée dans l'en-tête de tranche d'une image précédente dans la séquence dans l'ordre de décodage.

`original_idr_flag` doit être égal à 1 lorsque la structure syntaxique de marquage d'image de référence décodée survenait à l'origine dans une image IDR. `original_idr_flag` doit être égal à 0

lorsque la structure syntaxique de marquage d'image de référence décodée répétée ne survenait pas à l'origine dans une image IDR.

original_frame_num doit être égal au `frame_num` de l'image où survenait à l'origine la structure syntaxique de marquage d'image de référence décodée répétée. L'image indiquée par `original_frame_num` est l'image codée précédente qui a la valeur de `frame_num` spécifiée. La valeur de `original_frame_num` utilisée pour se référer à une image ayant un `memory_management_control_operation` égal à 5 doit être 0.

original_field_pic_flag doit être égal au `field_pic_flag` de l'image où survenait à l'origine la structure syntaxique de marquage d'image de référence décodée répétée.

original_bottom_field_flag doit être égal au `bottom_field_flag` de l'image où survenait à l'origine la structure syntaxique de marquage d'image de référence décodée répétée.

`dec_ref_pic_marking()` doit contenir une copie de la structure syntaxique de marquage d'image de référence décodée de l'image dont `frame_num` était `original_frame_num`. Le `nal_unit_type` utilisé pour la spécification de la structure syntaxique `dec_ref_pic_marking()` répétée doit être le `nal_unit_type` de ou des en-têtes de tranches de l'image dont `frame_num` était `original_frame_num` (c'est-à-dire que `nal_unit_type` comme utilisé au § 7.3.3.3 doit être considéré comme égal à 5 lorsque `original_idr_flag` est égal à 1 et ne doit pas être considéré comme égal à 5 lorsque `original_idr_flag` est égal à 0).

D.2.9 Sémantique de message SEI d'image de rechange

Ce message SEI indique que certaines unités de mappage de groupe de tranches, appelées unités de rechange de mappage de groupe de tranches, dans une ou plusieurs images de référence décodées, ressemblent aux unités de mappage de groupe de tranches colocalisées dans une image décodée spécifiée appelée l'image cible. Une unité de rechange de mappage de groupe de tranches peut être utilisée pour remplacer une unité de mappage de groupe de tranches colocalisées incorrectement décodée dans l'image cible. Une image décodée contenant des unités de rechange de mappage de groupe de tranches est appelée image de rechange.

Pour toutes les images de rechange identifiées dans un message SEI d'image de rechange, la valeur du `frame_mbs_only_flag` doit être égale à la valeur du `frame_mbs_only_flag` de l'image cible dans le même message SEI. Les images de rechange dans le message SEI sont soumises aux contraintes suivantes.

- Si l'image cible est un champ décodé, toutes les images de rechange identifiées dans le même message SEI doivent être des champs décodés.
- Autrement (l'image cible est une trame décodée), toutes les images de rechange identifiées dans le même message SEI doivent être des trames décodées.

Pour toute image de rechange identifiée dans un message SEI d'image de rechange, les valeurs de `pic_width_in_mbs_minus1` et `pic_height_in_map_units_minus1` doivent être respectivement égales aux valeurs de `pic_width_in_mbs_minus1` et `pic_height_in_map_units_minus1` de l'image cible dans le même message SEI. L'image associée (comme spécifié au § 7.4.1.2.3) à ce message doit apparaître après l'image cible, dans l'ordre de décodage.

target_frame_num indique le `frame_num` de l'image cible.

spare_field_flag égal à 0 indique que l'image cible et les images de rechange sont des trames décodées. **spare_field_flag** égal à 1 indique que l'image cible et les images de rechange sont des champs décodés.

target_bottom_field_flag égal à 0 indique que l'image cible est un champ supérieur. **target_bottom_field_flag** égal à 1 indique que l'image cible est un champ inférieur.

Une image cible est une image décodée de référence dont l'image codée primaire correspondante précède l'image en cours, dans l'ordre de décodage, et dans laquelle les valeurs de `frame_num`, `field_pic_flag` (lorsqu'il est présent) et `bottom_field_flag` (lorsqu'il est présent) sont respectivement égales à `target_frame_num`, `spare_field_flag` et `target_bottom_field_flag`.

num_spare_pics_minus1 indique le nombre d'images de rechange pour l'image cible spécifiée. Le nombre d'images de rechange est égal à `num_spare_pics_minus1 + 1`. La valeur de `num_spare_pics_minus1` doit être dans la gamme de 0 à 15, inclus.

delta_spare_frame_num[i] est utilisé pour identifier l'image de rechange qui contient le $i^{\text{ème}}$ ensemble d'unités de mappage de groupe de tranches de rechange, ci-après appelée la $i^{\text{ème}}$ image de rechange, comme spécifié ci-dessous. La valeur de `delta_spare_frame_num[i]` doit être dans la gamme de 0 à `MaxFrameNum - 1 - !spare_field_flag`, inclus.

Le `frame_num` de la $i^{\text{ème}}$ image de rechange, `spareFrameNum[i]`, est déduit comme suit pour toutes les valeurs de i de 0 à `num_spare_pics_minus1`, inclus:

```

candidateSpareFrameNum = target_frame_num - !spare_field_flag
pour( i = 0; i <= num_spare_pics_minus1; i++ ) {
    si( candidateSpareFrameNum < 0 )
        candidateSpareFrameNum = MaxFrameNum - 1
    spareFrameNum[ i ] = candidateSpareFrameNum - delta_spare_frame_num[ i ]
    si( spareFrameNum[ i ] < 0 )
        spareFrameNum[ i ] = MaxFrameNum + spareFrameNum[ i ]
    candidateSpareFrameNum = spareFrameNum[ i ] - !spare_field_flag
}

```

(D-3)

spare_bottom_field_flag[i] égal à 0 indique que la $i^{\text{ème}}$ image de rechange est un champ supérieur. `spare_bottom_field_flag[i]` égal à 1 indique que la $i^{\text{ème}}$ image de rechange est un champ inférieur.

L'image de rechange 0 est une image décodée de référence dont l'image codée primaire correspondante précède l'image cible, dans l'ordre de décodage, et dans laquelle les valeurs de `frame_num`, `field_pic_flag` (lorsqu'il est présent) et `bottom_field_flag` (lorsqu'il est présent) sont respectivement égales à `spareFrameNum[0]`, `spare_field_flag` et `spare_bottom_field_flag[0]`. La $i^{\text{ème}}$ image de rechange est une image décodée de référence dont l'image codée primaire correspondante précède la $(i - 1)^{\text{ème}}$ image de rechange, dans l'ordre de décodage, et dans laquelle les valeurs de `frame_num`, `field_pic_flag` (lorsqu'il est présent) et `bottom_field_flag` (lorsqu'il est présent) sont respectivement égales à `spareFrameNum[i]`, `spare_field_flag` et `spare_bottom_field_flag[i]`.

spare_area_idc[i] indique la méthode utilisée pour identifier les unités de mappage de groupe de tranches de rechange dans la $i^{\text{ème}}$ image de rechange. `spare_area_idc[i]` doit être dans la gamme de 0 à 2, inclus. `spare_area_idc[i]` égal à 0 indique que toutes les unités de mappage de groupe de tranches dans la $i^{\text{ème}}$ image de rechange sont des unités de rechange. `spare_area_idc[i]` égal à 1 indique que la valeur de l'élément syntaxique `spare_unit_flag[i][j]` est utilisée pour identifier les unités de mappage de groupe de tranches de rechange. `spare_area_idc[i]` égal à 2 indique que l'élément syntaxique `zero_run_length[i][j]` est utilisé pour calculer les valeurs de `spareUnitFlagInBoxOutOrder[i][j]`, comme décrit ci-dessous.

spare_unit_flag[i][j] égal à 0 indique que la $j^{\text{ème}}$ unité de mappage de groupe de tranches dans l'ordre de balayage de trame dans la $i^{\text{ème}}$ image de rechange est une unité de rechange. `spare_unit_flag[i][j]` égal à 1 indique que la $j^{\text{ème}}$ unité de mappage de groupe de tranches dans l'ordre de balayage de trame dans la $i^{\text{ème}}$ image de rechange n'est pas une unité de rechange.

zero_run_length[i][j] est utilisé pour calculer les valeurs de `spareUnitFlagInBoxOutOrder[i][j]` lorsque `spare_area_idc[i]` est égal à 2. Dans ce cas, les unités de mappage de groupe de tranches de rechange identifiées dans `spareUnitFlagInBoxOutOrder[i][j]` apparaissent dans l'ordre de sortie de boîte contraire des

aiguilles d'une montre, comme spécifié au § 8.2.2.4, pour chaque image de rechange. spareUnitFlagInBoxOutOrder[i][j] égal à 0 indique que la j^{ème} unité de mappage de groupe de tranches dans l'ordre de sortie de boîte contraire des aiguilles d'une montre dans la i^{ème} image de rechange est une unité de rechange. spareUnitFlagInBoxOutOrder[i][j] égal à 1 indique que la j^{ème} unité de mappage de groupe de tranches dans l'ordre de sortie de boîte contraire des aiguilles d'une montre dans la i^{ème} image de rechange n'est pas une unité de rechange.

Lorsque spare_area_idc[0] est égal à 2, spareUnitFlagInBoxOutOrder[0][j] est calculé comme suit:

```

pour( j = 0, loop = 0; j < PicSizeInMapUnits; loop++ ) {
    pour( k = 0; k < zero_run_length[ 0 ][ loop ]; k++ )
        spareUnitFlagInBoxOutOrder[ 0 ][ j++ ] = 0
        spareUnitFlagInBoxOutOrder[ 0 ][ j++ ] = 1
}

```

(D-4)

Lorsque spare_area_idc[i] est égal à 2 et que la valeur de i est supérieure à 0, spareUnitFlagInBoxOutOrder[i][j] est calculé comme suit:

```

pour( j = 0, loop = 0; j < PicSizeInMapUnits; loop++ ) {
    pour( k = 0; k < zero_run_length[ i ][ loop ]; k++ )
        spareUnitFlagInBoxOutOrder[ i ][ j ] = spareUnitFlagInBoxOutOrder[ i - 1 ][ j++ ]
        spareUnitFlagInBoxOutOrder[ i ][ j ] = !spareUnitFlagInBoxOutOrder[ i - 1 ][ j++ ]
}

```

(D-5)

D.2.10 Sémantique de message SEI d'informations de scène

Une scène et une transition de scène sont définies ci-après comme un ensemble d'images consécutives dans l'ordre de sortie.

NOTE – Les images décodées au sein d'une seule scène ont généralement un contenu similaire. Le message SEI d'informations de scène est utilisé pour étiqueter les images avec des identifiants de scène et pour indiquer les changements de scène. Le message spécifie comment ont été créées les images sources pour les images étiquetées. Le décodeur peut utiliser les informations pour choisir un algorithme approprié pour dissimuler les erreurs de transmission. Par exemple, un algorithme spécifique peut être utilisé pour dissimuler les erreurs de transmission qui sont survenues dans des images appartenant à une transition graduelle de scène. De plus, le message SEI d'informations de scène peut être utilisé d'une façon déterminée par l'application, comme l'indexation des scènes d'une séquence codée.

Un message SEI d'informations de scène étiquette toutes les images, dans l'ordre de décodage, depuis l'image codée primaire à laquelle le message est associé (inclusive), comme spécifié au § 7.4.1.2.3, jusqu'à l'image codée primaire à laquelle est associé (exclu) le prochain (s'il est présent) message SEI d'informations de scène dans l'ordre de décodage ou (autrement) jusqu'à la dernière unité d'accès dans le flux binaire (inclusive). Ces images sont désignées ici comme les images cibles.

scene_info_present_flag égal à 0 indique que la scène ou transition de scène à laquelle appartiennent les images cibles n'est pas spécifiée. scene_info_present_flag égal à 1 indique que les images cibles appartiennent à la même scène ou transition de scène.

scene_id identifie la scène à laquelle appartiennent les images cibles. Lorsque la valeur de scene_transition_type des images cibles est inférieure à 4, l'image précédente dans l'ordre de sortie est marquée avec une valeur de scene_transition_type inférieure à 4, et la valeur de scene_id est la même que la valeur de scene_id de l'image précédente dans l'ordre de sortie, ce qui indique que la scène source pour les images cibles et la scène pour l'image précédente (dans l'ordre de sortie) sont considérées par le codeur comme étant de la même scène. Lorsque la valeur de scene_transition_type des images cibles est supérieure à 3, et que l'image précédente dans l'ordre de sortie est marquée avec une valeur de scene_transition_type inférieure à 4, et que la valeur de scene_id est la même que la valeur de scene_id de l'image précédente dans l'ordre de sortie, cela indique qu'une des scènes sources pour les images cibles et la scène source pour l'image précédente (dans l'ordre de sortie) sont considérées par le codeur comme étant de la même scène. Lorsque la

valeur de `scene_id` n'est pas égale à la valeur de `scene_id` de l'image précédente dans l'ordre de sortie, cela indique que les images cibles et l'image précédente (dans l'ordre de sortie) sont considérées par le codeur comme étant de différentes scènes sources.

La valeur de `scene_id` doit être dans la gamme de 0 à $2^{32} - 1$ inclus. Les valeurs de `scene_id` dans la gamme de 0 à 255 inclus, et dans la gamme de 512 à $2^{31} - 1$ inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de `scene_id` dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à $2^{32} - 1$ inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de `scene_id` dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à $2^{32} - 1$ inclus, doivent l'ignorer (la retirer du flux binaire et la détruire).

`scene_transition_type` spécifie dans quel type de transition de scène (s'il en est) sont impliquées les images cibles. Les valeurs valides de `scene_transition_type` sont spécifiées au Tableau D-4.

Tableau D-4 – Valeurs de `scene_transition_type`

Valeur	Description
0	Pas de transition
1	Fondu vers le noir
2	Fondu à partir du noir
3	Transition non spécifiée à partir d'une ou vers une couleur constante
4	Dissoudre
5	Effacer
6	Mélange non spécifié de deux scènes

Lorsque `scene_transition_type` est supérieur à 3, les images cibles incluent à la fois des contenus venant de la scène étiquetée par son `scene_id` et la scène suivante, dans l'ordre de sortie, qui est étiquetée par `second_scene_id` (voir ci-dessous). Le terme "la scène en cours" est utilisé pour indiquer la scène étiquetée par `scene_id`. Le terme "la scène suivante" est utilisé pour indiquer la scène étiquetée par `second_scene_id`. Il n'est pas nécessaire pour toute image suivante, dans l'ordre de sortie, d'être étiquetée avec `scene_id` égal à `second_scene_id` du message SEI en cours.

Les types de transition de scène sont spécifiés comme suit.

"Pas de transition" spécifie que les images cibles ne sont pas impliquées dans une transition graduelle de scène.

NOTE – Lorsque deux images consécutives dans l'ordre de sortie ont `scene_transition_type` égal à 0 et des valeurs différentes de `scene_id`, une coupure de scène est intervenue entre les deux images.

"Fondu vers le noir" indique que les images cibles font partie d'une séquence d'images, dans l'ordre de sortie, impliquées dans une transition de scène en fondu vers le noir, c'est-à-dire que les échantillons luma de la scène s'approchent graduellement de zéro et que les échantillons chroma de la scène s'approchent graduellement de 128.

NOTE – Lorsque deux images sont étiquetées comme appartenant à la même transition de scène et que leur `scene_transition_type` est "Fondu vers le noir", la dernière, dans l'ordre de sortie, est plus sombre que la précédente.

"Fondu à partir du noir" indique que les images cibles font partie d'une séquence d'images, dans l'ordre de sortie, impliquées dans un fondu à partir d'une transition de scène noire, c'est-à-dire que les échantillons luma de la scène divergent graduellement de zéro et que les échantillons chroma de la scène peuvent diverger graduellement de 128.

NOTE – Lorsque deux images sont étiquetées comme appartenant à la même transition de scène et que leur `scene_transition_type` est "Fondu à partir du noir", la dernière, dans l'ordre de sortie, est plus claire que la précédente.

"Dissoudre" indique que les valeurs d'échantillon de chaque image cible (avant le codage) ont été générées en calculant une somme de valeurs pondérées d'échantillons colocalisés d'une image de la scène en cours et d'une image de la scène suivante. La pondération de la scène en cours décroît graduellement d'une prise en compte complète vers zéro, tandis que la pondération de la scène suivante s'accroît graduellement de zéro à une prise en compte complète. Lorsque deux images sont étiquetées comme appartenant à la même transition de scène et que leur `scene_transition_type` est "Dissoudre", la pondération de leur scène en cours par rapport à la dernière, dans l'ordre de sortie, est inférieure à la pondération de leur scène en cours pour la précédente, et la pondération de la scène suivante par rapport à la dernière, dans l'ordre de sortie, est supérieure à la pondération de la prochaine scène par rapport à la précédente.

"Effacer" indique que certaines des valeurs d'échantillon de chaque image cible (avant codage) ont été générées en copiant les valeurs d'échantillons colocalisés d'une image dans la scène en cours et que les valeurs d'échantillon restantes de chaque image cible (avant codage) ont été générées en copiant les valeurs d'échantillons colocalisés d'une image dans la scène suivante. Lorsque deux images sont étiquetées comme appartenant à la même transition de scène et que leur `scene_transition_type` est "Effacer", le nombre d'échantillons copiés de la scène suivante sur la dernière image dans l'ordre de sortie est supérieur au nombre d'échantillons copiés de la prochaine scène sur l'image précédente.

second_scene_id identifie la prochaine scène dans la transition de scène graduelle dans laquelle sont impliquées les images cibles. La valeur de `second_scene_id` ne doit pas être égale à la valeur de `scene_id`. La valeur de `second_scene_id` ne doit pas être égale à la valeur de `scene_id` dans l'image précédente dans l'ordre de sortie. Lorsque la prochaine image dans l'ordre de sortie est marquée avec une valeur de `scene_transition_type` inférieure à 4, et que la valeur de `second_scene_id` est la même que la valeur de `scene_id` de l'image suivante dans l'ordre de sortie, cela indique que le codeur considère que l'une des scènes sources pour les images cibles et la scène source pour l'image suivante (dans l'ordre de sortie) constituent la même scène. Lorsque la valeur de `second_scene_id` n'est pas égale à la valeur de `scene_id` ou de `second_scene_id` (s'il est présent) de l'image suivante dans l'ordre de sortie, cela indique que le codeur considère que les images cibles et l'image suivante (dans l'ordre de sortie) proviennent de scènes sources différentes.

Lorsque la valeur de `scene_id` d'une image est égale à la valeur de `scene_id` de l'image suivante dans l'ordre de sortie et que la valeur de `scene_transition_type` dans ces deux images est inférieure à 4, cela indique que le codeur considère que les deux images proviennent de la même scène source. Lorsque les valeurs de `scene_id`, `scene_transition_type` et `second_scene_id` (s'il est présent) d'une image sont (respectivement) égales aux valeurs de `scene_id`, `scene_transition_type` et `second_scene_id` de l'image suivante dans l'ordre de sortie et que la valeur de `scene_transition_type` est supérieure à 0, cela indique que le codeur considère que les deux images proviennent de la même transition graduelle de scène source.

La valeur de `second_scene_id` doit être dans la gamme de 0 à $2^{32}-1$ inclus. Les valeurs de `second_scene_id` dans la gamme de 0 à 255 inclus, et dans la gamme de 512 à $2^{31}-1$ inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de `second_scene_id` dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à $2^{32}-1$ inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de `second_scene_id` dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à $2^{32}-1$ inclus, doivent l'ignorer (la retirer du flux binaire et l'ignorer).

D.2.11 Sémantique de message SEI d'informations de sous-séquence

Le message SEI d'informations de sous-séquence est utilisé pour indiquer la position d'une image dans une hiérarchie de dépendance des données qui consiste en couches de sous-séquence et sous-séquences.

Une couche de sous-séquence contient un sous-ensemble des images codées dans une séquence. Les couches de sous-séquence sont numérotées avec des entiers non négatifs. Une couche ayant un plus grand numéro de couche est une couche plus élevée qu'une couche ayant un plus petit numéro de couche. Les couches sont ordonnées hiérarchiquement sur la base de leur dépendance les unes par rapport aux autres de sorte que toute image dans une couche ne doit pas être prédite à partir d'une image sur une couche supérieure.

NOTE – En d'autres termes, toute image de la couche 0 ne doit pas être prédite à partir d'une image quelconque de la couche 1 ou au-dessus, les images de la couche 1 peuvent être prédites à partir de la couche 0, les images de la couche 2 peuvent être prédites à partir des couches 0 et 1, etc.

NOTE – La qualité subjective est supposée croître avec le numéro des couches décodées.

Une sous-séquence est un ensemble d'images codées au sein d'une couche de sous-séquence. Une image doit résider dans une couche de sous-séquence et une seule. Aucune image dans une sous-séquence ne doit être prédite à partir d'une image dans une autre sous-séquence dans la même couche de sous-séquence ou d'une couche de sous-séquence supérieure. Une sous-séquence dans la couche 0 peut être décodée indépendamment de toute image qui n'appartient pas à la sous-séquence.

Le message SEI d'informations de sous-séquence concerne l'unité d'accès en cours. L'image codée primaire dans l'unité d'accès est désignée ici comme l'image en cours.

Le message SEI d'informations de sous-séquence ne doit pas être présent à moins que `gaps_in_frame_num_value_allowed_flag` dans l'ensemble de paramètres de séquence mentionné par l'image associée au message SEI de sous-séquence ne soit égal à 1.

sub_seq_layer_num spécifie le numéro de couche de sous-séquence de l'image en cours. Lorsque `sub_seq_layer_num` est supérieur à 0, les opérations de commande de gestion de mémoire ne doivent pas être utilisées dans un en-tête de tranche de l'image en cours. Lorsque l'image en cours réside dans une sous-séquence dont la première image dans l'ordre de décodage est une image IDR, la valeur de `sub_seq_layer_num` doit être égale à 0. Pour un champ de référence non apparié, la valeur de `sub_seq_layer_num` doit être égale à 0. `sub_seq_layer_num` doit être dans la gamme de 0 à 255 inclus.

sub_seq_id identifie la sous-séquence au sein d'une couche. Lorsque l'image en cours réside dans une sous-séquence dont la première image dans l'ordre de décodage est une image IDR, la valeur de `sub_seq_id` doit être la même que la valeur de `idr_pic_id` de l'image IDR. `sub_seq_id` doit être dans la gamme de 0 à 65535 inclus.

first_ref_pic_flag égal à 1 spécifie que l'image en cours est la première image de référence de la sous-séquence dans l'ordre de décodage. Lorsque l'image en cours n'est pas la première image de la sous-séquence dans l'ordre de décodage, le fanion `first_ref_pic_flag` doit être égal à 0.

leading_non_ref_pic_flag égal à 1 spécifie que l'image en cours est une image de non-référence précédant une image de référence dans l'ordre de décodage au sein de la sous-séquence ou que la sous-séquence ne contient pas d'images de référence. Lorsque l'image en cours est une image de référence ou que l'image en cours est une image de non-référence succédant au moins à une image de référence dans l'ordre de décodage au sein de la sous-séquence, le fanion `leading_non_ref_pic_flag` doit être égal à 0.

last_pic_flag égal à 1 indique que l'image en cours est la dernière image de la sous-séquence (dans l'ordre de décodage), y compris toutes les images de référence et de non-référence de la sous-séquence. Lorsque l'image en cours n'est pas la dernière image de la sous-séquence (dans l'ordre de décodage), `last_pic_flag` doit être égal à 0.

L'image en cours est allouée comme suit à une sous-séquence.

- Si une ou plusieurs des conditions suivantes sont Vraies, l'image en cours est la première image d'une sous-séquence dans l'ordre de décodage.
 - aucune image précédente dans l'ordre de décodage n'est étiquetée avec les mêmes valeurs de `sub_seq_id` et `sub_seq_layer_num` que l'image en cours;
 - la valeur de `leading_non_ref_pic_flag` est égale à 1 et la valeur de `leading_non_ref_pic_flag` est égale à 0 dans l'image précédente dans l'ordre de décodage ayant les mêmes valeurs de `sub_seq_id` et `sub_seq_layer_num` que l'image en cours;
 - la valeur de `first_ref_pic_flag` est égale à 1 et la valeur de `leading_non_ref_pic_flag` est égale à 0 dans l'image précédente dans l'ordre de décodage ayant les mêmes valeurs de `sub_seq_id` et `sub_seq_layer_num` que l'image en cours;
 - la valeur de `last_pic_flag` est égale à 1 dans l'image précédente dans l'ordre de décodage ayant les mêmes valeurs de `sub_seq_id` et `sub_seq_layer_num` que l'image en cours.
- Autrement, l'image en cours à la même sous-séquence que l'image précédente dans l'ordre de décodage ayant les mêmes valeurs de `sub_seq_id` et `sub_seq_layer_num` que l'image en cours.

`sub_seq_frame_num_flag` égal à 0 spécifie que `sub_seq_frame_num` n'est pas présent. `sub_seq_frame_num_flag` égal à 1 spécifie que `sub_seq_frame_num` est présent.

`sub_seq_frame_num` doit être égal à 0 pour la première image de référence de la sous-séquence et pour toute image de non-référence précédant la première image de référence de la sous-séquence dans l'ordre de décodage. `sub_seq_frame_num` est de plus soumis aux contraintes suivantes.

- Si l'image en cours n'est pas le second champ d'une paire de champs complémentaires, `sub_seq_frame_num` doit être incrémenté de 1, en opération modulo `MaxFrameNum`, par rapport à l'image de référence précédente, dans l'ordre de décodage, qui appartient à la sous-séquence.
- Autrement (l'image en cours est le second champ d'une paire de champs complémentaires), la valeur de `sub_seq_frame_num` doit être la même que la valeur de `sub_seq_frame_num` pour le premier champ de la paire de champs complémentaires.

`sub_seq_frame_num` doit être dans la gamme de 0 à `MaxFrameNum` – 1, inclus.

Lorsque l'image en cours est une image IDR, elle doit débiter par une nouvelle sous-séquence dans la couche de sous-séquence 0. Et donc, le `sub_seq_layer_num` doit être 0, le `sub_seq_id` doit être différent de la sous-séquence précédente dans la couche de sous-séquence 0, `first_ref_pic_flag` doit être 1, et `leading_non_ref_pic_flag` doit être égal à 0.

Lorsque le message SEI d'informations de sous-séquence est présent pour les deux champs codés d'une paire de champs complémentaires, les valeurs de `sub_seq_layer_num`, `sub_seq_id`, `leading_non_ref_pic_flag` et `sub_seq_frame_num`, lorsqu'ils sont présents, doivent être identiques pour ces deux images.

Lorsque le message SEI d'informations de sous-séquence n'est présent que pour un seul des champs codés d'une paire de champs complémentaires, les valeurs de `sub_seq_layer_num`, `sub_seq_id`, `leading_non_ref_pic_flag` et `sub_seq_frame_num`, lorsqu'ils sont présents, sont aussi applicables à l'autre champ codé de la paire de champs complémentaires.

D.2.12 Sémantique de message SEI de caractéristiques de couche de sous-séquence

Le message SEI de caractéristiques de couche de sous-séquence spécifie les caractéristiques des couches de sous-séquence.

num_sub_seq_layers_minus1 plus 1 spécifie le nombre de couches de sous-séquence dans la séquence. **num_sub_seq_layers_minus1** doit être dans la gamme de 0 à 255 inclus.

Une paire de **average_bit_rate** et **average_frame_rate** caractérise chaque couche de sous-séquence. La première paire de **average_bit_rate** et **average_frame_rate** spécifie les caractéristiques d'une couche de sous-séquence 0. Lorsqu'elle est présente, la seconde paire spécifie les caractéristiques des couches de sous-séquence 0 et 1 conjointement. Chaque paire dans l'ordre de décodage spécifie les caractéristiques pour une gamme de couches de sous-séquence depuis la couche numéro 0 jusqu'au numéro de couche spécifié par le compteur de boucles de couches. Les valeurs sont effectives depuis le point où elles sont décodées jusqu'à ce que soit décodée une mise à jour des valeurs.

accurate_statistics_flag égal à 1 indique que les valeurs de **average_bit_rate** et **average_frame_rate** sont arrondies à partir de valeurs correctes statistiquement. Le fanion **accurate_statistics_flag** égal à 0 indique que le **average_bit_rate** et le **average_frame_rate** sont estimés et peuvent dévier un peu des valeurs correctes.

Lorsque **accurate_statistics_flag** est égal à 0, la qualité de l'approximation utilisée dans le calcul des valeurs de **average_bit_rate** et de **average_frame_rate** est choisie par le processus de codage et n'est pas spécifié par la présente Recommandation | Norme internationale.

average_bit_rate indique le débit binaire moyen en unités de 1000 bits par seconde. Toutes les unités NAL dans la gamme de couches de sous-séquence spécifiées ci-dessus sont prises en compte dans le calcul. Le débit binaire moyen est calculé en fonction du temps de retrait de l'unité d'accès spécifié à l'Annexe C de la Recommandation | Norme internationale. Dans ce qui suit, **bTotal** est le nombre de bits dans toutes les unités NAL succédant à un message SEI de caractéristiques de couche de sous-séquence (y compris les bits des unités NAL de l'unité d'accès en cours) et précédant l'unité d'accès suivante (dans l'ordre de décodage) y compris un message SEI de caractéristiques de couche de sous-séquence (s'il est présent) ou la fin du flux (autrement). t_1 est le temps de retrait (en secondes) de l'unité d'accès en cours, et t_2 est le temps de retrait (en secondes) de la dernière unité d'accès (dans l'ordre de décodage) avant le prochain message SEI de caractéristiques de couche de sous-séquence (s'il est présent) ou la fin du flux (autrement).

Lorsque **accurate_statistics_flag** est égal à 1, les conditions suivantes doivent être respectées comme suit.

- Si t_1 n'est pas égal à t_2 , la condition suivante doit être Vraie

$$\text{average_bit_rate} == \text{Round}(\text{bTotal} \div ((t_2 - t_1) * 1000)) \quad (\text{D-6})$$

- Autrement (t_1 est égal à t_2), la condition suivante doit être Vraie

$$\text{average_bit_rate} == 0 \quad (\text{D-7})$$

average_frame_rate indique le débit de trame moyen en unités de trames/(256 secondes). Toutes les unités NAL dans la gamme de couches de sous-séquences spécifiée ci-dessus sont prises en compte dans le calcul. Dans ce qui suit, **fTotal** est le nombre de trames, de paires de champs complémentaires et de champs non appariés entre l'image en cours (incluse) et le prochain message SEI de caractéristiques de couche de sous-séquence (s'il est présent) ou la fin du flux (autrement). t_1 est le temps de retrait (en secondes) de l'unité d'accès en cours, et t_2 est le temps de retrait (en secondes) de la dernière unité d'accès (dans l'ordre de décodage) avant le prochain message SEI de caractéristiques de couche de sous-séquence (s'il est présent) ou la fin du flux (autrement).

Lorsque **accurate_statistics_flag** est égal à 1, les conditions suivantes doivent être respectées comme suit.

- Si t_1 n'est pas égal à t_2 , la condition suivante doit être Vraie

$$\text{average_frame_rate} == \text{Round}(\text{fTotal} * 256 \div (t_2 - t_1)) \quad (\text{D-8})$$

- Autrement (t_1 est égal à t_2), la condition suivante doit être Vraie

D.2.13 Sémantique de message SEI de caractéristiques de sous-séquence

Le message SEI de caractéristiques de sous-séquence indique les caractéristiques d'une sous-séquence. Il indique aussi la dépendance de l'interprédiction entre les sous-séquences. Ce message doit être contenu dans la première unité d'accès dans l'ordre de décodage de la sous-séquence à laquelle s'applique le message SEI de caractéristiques de sous-séquence. Cette sous-séquence est appelée ici la sous-séquence cible.

sub_seq_layer_num identifie le numéro de couche de sous-séquence de la sous-séquence cible. sub_seq_layer_num doit être dans la gamme de 0 à 255 inclus.

sub_seq_id identifie la sous-séquence cible. sub_seq_id doit être dans la gamme de 0 à 65535 inclus.

duration_flag égal à 0 indique que la durée de la sous-séquence cible n'est pas spécifiée.

sub_seq_duration spécifie la durée de la sous-séquence cible en tic-tac d'horloge d'une horloge à 90 kHz.

average_rate_flag égal à 0 indique que le débit binaire moyen et le débit de trame moyen de la sous-séquence cible ne sont pas spécifiés.

accurate_statistics_flag indique quel est le degré de confiance des valeurs de average_bit_rate et average_frame_rate. accurate_statistics_flag égal à 1, indique que average_bit_rate et average_frame_rate sont arrondis à partir de valeurs statistiquement correctes. accurate_statistics_flag égal à 0 indique que average_bit_rate et average_frame_rate sont estimés et peuvent dévier des valeurs statistiquement correctes.

average_bit_rate indique le débit binaire moyen en (1000 bits)/seconde de la sous-séquence cible. Toutes les unités NAL de la sous-séquence cible sont prises en compte dans le calcul. Le débit binaire moyen est calculé en fonction du temps de retrait de l'unité d'accès spécifié au § C.1.2. Dans ce qui suit, nB est le nombre de bits dans toutes les unités NAL dans la sous-séquence. t_1 est le temps de retrait (en secondes) de la première unité d'accès de la sous-séquence (dans l'ordre de décodage), et t_2 est le temps de retrait (en secondes) de la dernière unité d'accès de la sous-séquence (dans l'ordre de décodage).

Lorsque accurate_statistics_flag est égal à 1, les conditions suivantes doivent être satisfaites comme suit.

- Si t_1 n'est pas égal à t_2 , la condition suivante doit être Vraie

$$\text{average_bit_rate} == \text{Round}(nB \div ((t_2 - t_1) * 1000)) \quad (\text{D-10})$$

- Autrement (t_1 est égal à t_2), la condition suivante doit être Vraie

$$\text{average_bit_rate} == 0 \quad (\text{D-11})$$

average_frame_rate indique le débit de trame moyen en unités de trames/(256 secondes) de la sous-séquence cible. Toutes les unités NAL de la sous-séquence cible sont prises en compte dans le calcul. Le débit de trame moyen est calculé en fonction du temps de retrait d'unité d'accès spécifié au § C.1.2. Dans ce qui suit, fC est le nombre de trames, de paires de champs complémentaires et de champs non appariés dans la sous-séquence. t_1 est le temps de retrait (en secondes) de la première unité d'accès de la sous-séquence (dans l'ordre de décodage), et t_2 est le temps de retrait (en secondes) de la dernière unité d'accès de la sous-séquence (dans l'ordre de décodage).

Lorsque accurate_statistics_flag est égal à 1, les conditions suivantes doivent être satisfaites comme suit.

- Si t_1 n'est pas égal à t_2 , la condition suivante doit être Vraie

$$\text{average_frame_rate} == \text{Round}(fC * 256 \div (t_2 - t_1)) \quad (\text{D-12})$$

- Autrement (t_1 est égal à t_2), la condition suivante doit être Vraie

$$\text{average_frame_rate} == 0 \tag{D-13}$$

num_referenced_subseqs spécifie le nombre de sous-séquences qui contiennent des images qui sont utilisées comme images de référence pour l'interprédiction dans les images de la sous-séquence cible. **num_referenced_subseqs** doit être dans la gamme de 0 à 255 inclus.

ref_sub_seq_layer_num, **ref_sub_seq_id**, et **ref_sub_seq_direction** identifie la sous-séquence qui contient les images qui sont utilisées comme images de référence pour l'interprédiction dans les images de la sous-séquence cible. En fonction de **ref_sub_seq_direction**, on applique ce qui suit.

- Si **ref_sub_seq_direction** est égal à 0, un ensemble de sous-séquences candidates se compose des sous-séquences dont **sub_seq_id** est égal à **ref_sub_seq_id**, qui résident dans la couche de sous-séquence qui a **sub_seq_layer_num** égal à **ref_sub_seq_layer_num**, et dont la première image dans l'ordre de décodage précède la première image de la sous-séquence cible dans l'ordre de décodage.
- Autrement (**ref_sub_seq_direction** est égal à 1), un ensemble de sous-séquences candidates se compose des sous-séquences dont **sub_seq_id** est égal à **ref_sub_seq_id**, qui résident dans la couche de sous-séquence qui a **sub_seq_layer_num** égal à **ref_sub_seq_layer_num**, et dont la première image dans l'ordre de décodage succède à la première image de la sous-séquence cible dans l'ordre de décodage.

La sous-séquence utilisée comme référence pour la sous-séquence cible est la sous-séquence parmi l'ensemble de sous-séquences candidates dont la première image est la plus proche de la première image de la sous-séquence cible dans l'ordre de décodage.

D.2.14 Sémantique de message SEI de gel de trame entière

Le message SEI de gel de trame entière indique que le contenu de la trame vidéo entière avant affichage dans l'ordre de sortie devrait être gardé inchangé, sans mise à jour de l'affichage utilisant le contenu de l'image décodée en cours.

full_frame_freeze_repetition_period indique si un autre message SEI de gel de trame entière doit être présent dans le flux binaire et spécifie l'intervalle de compte d'ordre d'image au sein duquel un autre message SEI de gel de trame entière ou un message SEI de libération de gel de trame entière sera présent. La valeur de **full_frame_freeze_repetition_period** doit être dans la gamme de 0 à 16 384 inclus.

full_frame_freeze_repetition_period égal à 0 spécifie que le message SEI de gel de trame entière ne s'applique qu'à l'image décodée en cours.

full_frame_freeze_repetition_period égal à 1 spécifie que le message SEI de gel de trame entière persiste dans l'ordre de sortie tant qu'une des conditions suivantes est Vraie.

- Une nouvelle séquence vidéo codée commence.
- Une image dans une unité d'accès contenant un message SEI de libération de gel de trame entière est sortie avec **PicOrderCnt()** supérieur à **PicOrderCnt(CurrPic)**.

full_frame_freeze_repetition_period supérieur à 1 spécifie que le message SEI de gel de trame entière persiste tant qu'une des conditions suivantes est Vraie.

- Une nouvelle séquence vidéo codée commence.
- Une image dans une unité d'accès contenant un message SEI de libération de gel de trame entière ou un message SEI de libération de gel de trame entière est sortie avec **PicOrderCnt()** supérieur à **PicOrderCnt(CurrPic) + full_frame_freeze_repetition_period**.

full_frame_freeze_repetition_period supérieur à 1 indique qu'un autre message SEI de gel de trame entière ou un message SEI de libération de gel de trame entière doit être présent pour une image

dans une unité d'accès qui est sortie avec `PicOrderCnt()` inférieur ou égal à `PicOrderCnt(CurrPic) + full_frame_freeze_repetition_period`; à moins qu'une nouvelle séquence vidéo codée ne commence sans sortie d'une telle image.

D.2.15 Sémantique de message SEI de libération de gel de trame entière

Le message SEI de libération de gel de trame entière indique que la mise à jour de la trame vidéo affichée devrait repartir, en commençant avec le contenu de l'image décodée en cours et en continuant avec les images suivantes dans l'ordre de sortie. Le message SEI de libération de gel de trame entière annule les effets de tout message SEI de gel de trame entière envoyé avec les images qui précèdent l'image en cours dans l'ordre de sortie.

D.2.16 Sémantique de message SEI de photographie de trame entière

Le message SEI de photographie de trame entière indique que la trame en cours est étiquetée pour une utilisation déterminée par l'application en tant que photographie d'image fixe du contenu de la vidéo.

snapshot_id spécifie un numéro d'identification de photographie. **snapshot_id** doit être dans la gamme de 0 à $2^{32} - 1$ inclus.

Les valeurs de **snapshot_id** dans la gamme de 0 à 255 inclus, et dans la gamme de 512 à $2^{31} - 1$ inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de **snapshot_id** dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à $2^{32} - 1$ inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de **snapshot_id** dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à $2^{32} - 1$ inclus, doivent l'ignorer (la retirer du flux binaire et l'ignorer).

D.2.17 Sémantique de message SEI de début de segment de raffinement progressif

Le message SEI de début de segment de raffinement progressif spécifie le début d'un ensemble d'images codées consécutives qui est étiqueté comme étant l'image en cours suivie par une séquence d'une ou plusieurs images de raffinement de la qualité de l'image en cours, plutôt qu'une représentation d'une scène en mouvement continu.

L'ensemble étiqueté d'images codées consécutives doit continuer jusqu'à ce qu'une des conditions suivantes soit Vraie. Lorsqu'une des conditions ci-dessous devient Vraie, la prochaine tranche à décoder n'appartient pas à l'ensemble étiqueté d'images codées consécutives.

1. La prochaine tranche à décoder appartient à une image IDR.
2. **num_refinement_steps_minus1** est supérieur à 0 et le **frame_num** de la prochaine tranche à décoder est $(currFrameNum + num_refinement_steps_minus1 + 1) \% MaxFrameNum$, où **currFrameNum** est la valeur de **frame_num** de l'image dans l'unité d'accès contenant le message SEI.
3. **num_refinement_steps_minus1** est à 0 et un message SEI de fin de segment de raffinement progressif avec le même **progressive_refinement_id** que celui qui est dans ce message SEI est décodé.

L'ordre de décodage d'image au sein de l'ensemble étiqueté d'images consécutives devrait être le même que leur ordre de sortie. **progressive_refinement_id** spécifie un numéro d'identification pour l'opération de raffinement progressif. **progressive_refinement_id** doit être dans la gamme de 0 à $2^{32} - 1$ inclus.

Les valeurs de **progressive_refinement_id** dans la gamme de 0 à 255 inclus, et dans la gamme de 512 à $2^{31} - 1$ inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de **progressive_refinement_id** dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à $2^{32} - 1$ inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant

une valeur de `progressive_refinement_id` dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à $2^{32} - 1$ inclus, doivent l'ignorer (la retirer du flux binaire et l'ignorer).

num_refinement_steps_minus1 spécifie le nombre de trames de référence dans l'ensemble étiqueté d'images codées consécutives comme suit.

- Si `num_refinement_steps_minus1` est égal à 0, le nombre de trame de référence dans l'ensemble étiqueté d'images codées consécutives est inconnu.
- Autrement, le nombre de trame de référence dans l'ensemble étiqueté d'images codées consécutives est égal à `num_refinement_steps_minus1 + 1`.

`num_refinement_steps_minus1` doit être dans la gamme de 0 à `MaxFrameNum - 1`, inclus.

D.2.18 Sémantique de message SEI de fin de segment de raffinement progressif

Le message SEI de fin de segment de raffinement progressif spécifie la fin d'un ensemble d'images codées consécutives qui a été étiqueté en utilisant un message SEI de début de segment de raffinement progressif comme image initiale suivi d'une séquence d'une ou plusieurs images de raffinement de la qualité de l'image initiale, et se terminant avec l'image en cours.

progressive_refinement_id spécifie un numéro d'identification pour l'opération de raffinement progressif. `progressive_refinement_id` doit être dans la gamme de 0 à $2^{32} - 1$ inclus.

Le message SEI de fin de segment de raffinement progressif spécifie la fin de tout segment de raffinement progressif ayant commencé antérieurement en utilisant un message SEI de début de segment de raffinement progressif avec la même valeur de `progressive_refinement_id`.

Les valeurs de `progressive_refinement_id` dans la gamme de 0 à 255, inclus, et dans la gamme de 512 à $2^{31} - 1$ inclus, peuvent être utilisées comme déterminé par l'application. Les valeurs de `progressive_refinement_id` dans la gamme de 256 à 511 inclus, et dans la gamme de 2^{31} à $2^{32} - 1$ inclus, sont réservées pour une utilisation future par l'UIT-T | ISO/CEI. Les décodeurs rencontrant une valeur de `progressive_refinement_id` dans la gamme de 256 à 511 inclus, ou dans la gamme de 2^{31} à $2^{32} - 1$ inclus, doivent l'ignorer (la retirer du flux binaire et l'ignorer).

D.2.19 Sémantique de message SEI d'ensemble de groupes de tranches à mouvement restreint

Ce message SEI indique que l'interprédiction sur les frontières d'un groupe de tranches est soumise aux contraintes spécifiées ci-dessous. Lorsqu'il est présent, le message ne doit apparaître que lorsqu'il est associé, comme spécifié au § 7.4.1.2.3, avec une unité d'accès IDR.

L'ensemble d'image cible pour ce message SEI contient toutes les images codées primaires consécutives dans l'ordre de décodage débutant avec l'image IDR codée primaire associée (incluse) et se terminant avec l'image IDR codée primaire suivante (exclue) ou avec la toute dernière image codée primaire dans le flux binaire (inclus) dans l'ordre de décodage lorsqu'il n'y a pas d'image IDR codée primaire à la suite. L'ensemble de groupe de tranches est une collection d'un ou plusieurs groupes de tranches, identifiés par l'élément syntaxique `slice_group_id[i]`.

Ce message SEI indique que, pour chaque image dans l'ensemble d'image cible, le processus d'interprédiction est soumis aux contraintes suivantes: pour l'interprédiction de tout échantillon au sein de l'ensemble de groupe de tranches n'est utilisée aucune valeur d'échantillon en dehors de l'ensemble de groupe de tranches, et aucune valeur d'échantillon à une position d'échantillon fractionnaire qui soit calculée en utilisant une ou plusieurs valeurs d'échantillon en dehors de l'ensemble de groupe de tranches.

num_slice_groups_in_set_minus1 + 1 spécifie le nombre de groupe de tranches dans l'ensemble de groupe de tranches. La gamme admise de `num_slice_groups_in_set_minus1` est de 0 à `num_slice_groups_minus1`, inclus. La gamme admise de `num_slice_groups_minus1` est spécifiée à l'Annexe A.

slice_group_id[i] identifie le ou les groupes de tranches contenus au sein de l'ensemble de groupe de tranches. La gamme admise est de 0 à `num_slice_groups_in_set_minus1`, inclus. La taille de l'élément syntaxique `slice_group_id[i]` est de $\text{Ceil}(\text{Log}_2(\text{num_slice_groups_minus1} + 1))$ bits.

exact_sample_value_match_flag égal à 0 indique que, au sein de l'ensemble d'image cible, lorsque les macroblocs qui n'appartiennent pas à l'ensemble de groupe de tranches ne sont pas décodés, la valeur de chaque échantillon dans l'ensemble de groupe de tranches ne doit pas nécessairement être exactement la même que la valeur du même échantillon lorsque tous les macroblocs sont décodés. **exact_sample_value_match_flag** égal à 1 indique que, au sein de l'ensemble d'image cible, lorsque les macroblocs qui n'appartiennent pas à l'ensemble de groupe de tranches ne sont pas décodés, la valeur de chaque échantillon dans l'ensemble de groupe de tranches doit être exactement la même que la valeur du même échantillon lorsque sont décodés tous les macroblocs dans l'ensemble d'image cible.

NOTE – Lorsque `disable_deblocking_filter_idc` est égal à 2 dans toutes les tranches dans l'ensemble d'image cible, `exact_sample_value_match_flag` devrait être 1.

pan_scan_rect_flag égal à 0 spécifie que `pan_scan_rect_id` n'est pas présent. `pan_scan_rect_flag` égal à 1 spécifie que `pan_scan_rect_id` est présent.

pan_scan_rect_id indique que l'ensemble de groupe de tranches spécifié couvre au moins le rectangle de balayage complet identifié par `pan_scan_rect_id` au sein de l'ensemble d'image cible.

NOTE – Plusieurs messages SEI `motion_constrained_slice_group_set` peuvent être associés à la même image IDR. Par conséquent, plus d'un ensemble de groupe de tranches peut être actif au sein d'un ensemble d'image cible.

NOTE – La taille, forme et localisation des groupes de tranches dans l'ensemble de groupe de tranches peut changer au sein de l'ensemble d'image cible.

D.2.20 Sémantique de message SEI réservé

Ce message consiste en données réservées pour une utilisation future à compatibilité ascendante par l'UIT-T | ISO/CEI. Les codeurs conformes à la présente Recommandation | Norme internationale ne doivent pas envoyer de messages SEI réservés tant que l'utilisation de tels messages n'aura pas été spécifiée par l'UIT-T | ISO/CEI. Les décodeurs conformes à la présente Recommandation | Norme internationale qui rencontrent des messages SEI réservés doivent ignorer leur contenu sans effet sur le processus de décodage, excepté comme spécifié dans les futures Recommandations | Normes internationales spécifiées par l'UIT-T | ISO/CEI.

reserved_sei_message_payload_byte est un octet réservé pour utilisation future par l'UIT-T | ISO/CEI.

Annexe E

Informations d'utilisation vidéo (VUI)

(La présente annexe fait partie intégrante de la présente Recommandation | Norme internationale)

La présente annexe spécifie la syntaxe et la sémantique des paramètres VUI de l'ensemble de paramètres de séquences.

Les paramètres VUI ne sont pas nécessaires pour la construction des échantillons luma ou chroma par le processus de décodage. Les décodeurs conformes ne sont pas tenus de traiter ces informations pour être conformes à l'ordre de sortie de la présente Recommandation | Norme internationale (voir l'Annexe C pour la spécification de la conformité). Certains paramètres VUI sont nécessaires pour vérifier la conformité du flux binaire et pour la conformité du délai de sortie du décodeur.

A l'Annexe E, la spécification de présence des paramètres VUI est également satisfaite lorsque ces paramètres (ou un de leurs sous-ensembles) sont envoyés aux décodeurs (ou au HRD) par d'autres moyens non spécifiés par la présente Recommandation | Norme internationale. Lorsqu'ils sont présents dans le flux binaire, les paramètres VUI doivent suivre la syntaxe et la sémantique spécifiée aux § 7.3.2.1 et 7.4.2.1 et dans la présente annexe. Lorsque le contenu des paramètres VUI est envoyé pour l'application par des moyens autres que la présence au sein du flux binaire, la représentation du contenu des paramètres VUI ne doit pas nécessairement utiliser la syntaxe spécifiée dans la présente annexe. Pour les besoins du comptage des bits, seuls sont comptés les bits appropriés qui sont réellement présents dans le flux binaire.

E.1 Syntaxe VUI

E.1.1 Syntaxe des paramètres VUI

vui_parameters() {	C	Descripteur
aspect_ratio_info_present_flag	0	u(1)
si(aspect_ratio_info_present_flag) {		
aspect_ratio_idc	0	u(8)
si(aspect_ratio_idc == Extended_SAR) {		
sar_width	0	u(16)
sar_height	0	u(16)
}		
}		
overscan_info_present_flag	0	u(1)
si(overscan_info_present_flag)		
overscan_appropriate_flag	0	u(1)
video_signal_type_present_flag	0	u(1)
si(video_signal_type_present_flag) {		
video_format	0	u(3)
video_full_range_flag	0	u(1)
colour_description_present_flag	0	u(1)
si(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)
si(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)
si(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)
si(nal_hrd_parameters_present_flag)		
hrd_parameters()		
vcl_hrd_parameters_present_flag	0	u(1)
si(vcl_hrd_parameters_present_flag)		
hrd_parameters()		
si(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)
si(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 Syntaxe des paramètres HRD

hrd_parameters() {	C	Descripteur
cpb_cnt_minus1	0	ue(v)
bit_rate_scale	0	u(4)
cpb_size_scale	0	u(4)
pour(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)
hrd_parameters() {	C	Descriptor

E.2 Sémantique VUI

E.2.1 Sémantique des paramètres VUI

aspect_ratio_info_present_flag égal à 1 spécifie que **aspect_ratio_idc** est présent. **aspect_ratio_info_present_flag** égal à 0 spécifie que **aspect_ratio_idc** n'est pas présent.

aspect_ratio_idc spécifie la valeur du ratio d'aspect d'échantillon des échantillons luma. Le Tableau E-1 indique la signification du code. Lorsque **aspect_ratio_idc** indique **Extended_SAR**, le ratio d'aspect d'échantillon est représenté par **sar_width** et **sar_height**. Lorsque l'élément syntaxique **aspect_ratio_idc** n'est pas présent, la valeur de **aspect_ratio_idc** doit être supposée égale à 0.

Tableau E-1 – Signification de l'indicateur de ratio d'aspect d'échantillon

aspect_ratio_idc	Ratio d'aspect d'échantillon	(pour information) Exemples d'utilisations
0	Non spécifié	
1	1:1 ("carré")	Trame 1280x720 16:9 sans surbalayage Trame 1920x1080 16:9 sans surbalayage (recadré à partir du 1920x1088) Trame 640x480 4:3 sans surbalayage
2	12:11	Trame 720x576 4:3 avec surbalayage horizontal Trame 352x288 4:3 sans surbalayage
3	10:11	Trame 720x480 4:3 avec surbalayage horizontal Trame 352x240 4:3 sans surbalayage
4	16:11	Trame 720x576 16:9 avec surbalayage horizontal Trame 540x576 4:3 avec surbalayage horizontal
5	40:33	Trame 720x480 16:9 avec surbalayage horizontal Trame 540x480 4:3 avec surbalayage horizontal
6	24:11	Trame 352x576 4:3 sans surbalayage Trame 540x576 16:9 avec surbalayage horizontal
7	20:11	Trame 352x480 4:3 sans surbalayage Trame 480x480 16:9 avec surbalayage horizontal
8	32:11	Trame 352x576 16:9 sans surbalayage
9	80:33	Trame 352x480 16:9 sans surbalayage
10	18:11	Trame 480x576 4:3 avec surbalayage horizontal
11	15:11	Trame 480x480 4:3 avec surbalayage horizontal
12	64:33	Trame 540x576 16:9 avec surbalayage horizontal
13	160:99	Trame 540x480 16:9 avec surbalayage horizontal
14..254	Réservé	
255	Extended_SAR	

sar_width indique la taille horizontale du ratio d'aspect de l'échantillon (en unités arbitraires).

sar_height indique la taille verticale du ratio d'aspect de l'échantillon (dans les mêmes unités arbitraires que sar_width).

sar_width et sar_height doivent être premiers entre eux ou égaux à 0. Lorsque aspect_ratio_idc est égal à 0 ou que sar_width est égal à 0 ou que sar_height est égal à 0, le ratio d'aspect d'échantillon doit être considéré comme non spécifié par la présente Recommandation | Norme internationale.

overscan_info_present_flag égal à 1 spécifie que le fanion overscan_appropriate_flag est présent. Lorsque overscan_info_present_flag est égal à 0 ou n'est pas présent, la méthode d'affichage préférée pour le signal vidéo n'est pas spécifiée.

overscan_appropriate_flag égal à 1 indique que les sorties d'images décodées recadrées conviennent pour l'affichage en utilisant le surbalayage. overscan_appropriate_flag égal à 0 indique que les sorties d'images décodées recadrées contiennent des informations visuellement importantes dans toute la région jusqu'aux bordures de la mire de l'image, et que les sorties d'images codées recadrées ne devraient pas être affichées en utilisant le surbalayage. Au lieu de cela, elles devraient être affichées en utilisant soit une correspondance exacte entre la zone d'affichage et la mire, soit en utilisant le sous-balayage.

NOTE – Par exemple, `overscan_appropriate_flag` égal à 1 pourrait être utilisé pour de la programmation de télévision de loisirs, ou pour la vue de gens en direct dans une visioconférence, et `overscan_appropriate_flag` égal à 0 pourrait être utilisé pour capturer un écran d'ordinateur ou pour le contenu d'une caméra de sécurité.

video_signal_type_present_flag égal à 1 spécifie que `video_format`, `video_full_range_flag` et `colour_description_present_flag` sont présents. `video_signal_type_present_flag` égal à 0, spécifie que `video_format`, `video_full_range_flag` et `colour_description_present_flag` ne sont pas présents.

video_format indique la représentation des images comme spécifié au Tableau E-2, avant qu'elles ne soient codées en conformité avec la présente Recommandation | Norme internationale. Lorsque l'élément syntaxique `video_format` n'est pas présent, la valeur de `video_format` doit être supposée égale à 5.

Tableau E-2 – Signification de video_format

video_format	Signification
0	Composant
1	PAL
2	NTSC
3	SECAM
4	MAC
5	Format vidéo non spécifié
6	Réservé
7	Réservé

video_full_range_flag indique le niveau de noir et la gamme des signaux luma et chroma telle que calculée à partir des signaux de composants analogiques E'_Y , E'_{PB} , et E'_{PR} , comme suit.

– Si `video_full_range_flag` est égal à 0,

$$Y = \text{Round}(219 * E'_Y + 16) \tag{E-1}$$

$$Cb = \text{Round}(224 * E'_{PB} + 128) \tag{E-2}$$

$$Cr = \text{Round}(224 * E'_{PR} + 128) \tag{E-3}$$

– Autrement (`video_full_range_flag` est égal à 1),

$$Y = \text{Round}(255 * E'_Y) \tag{E-4}$$

$$Cb = \text{Round}(255 * E'_{PB} + 128) \tag{E-5}$$

$$Cr = \text{Round}(255 * E'_{PR} + 128) \tag{E-6}$$

Lorsque l'élément syntaxique `video_full_range_flag` n'est pas présent, la valeur de `video_full_range_flag` doit être supposée égale à 0.

colour_description_present_flag égal à 1 spécifie que `colour primaries`, `transfer_characteristics` et `matrix_coefficients` sont présents. `colour_description_present_flag` égal à 0 spécifie que `colour primaries`, `transfer_characteristics` et `matrix_coefficients` ne sont pas présents.

colour primaries indique les coordonnées chromatiques des primaires de la source, comme spécifié au Tableau E-3, dans les termes de la définition de la norme CEI 1931 de x et y comme spécifié par l'ISO/CEI 10527.

Tableau E-3 – Couleurs primaires

Valeur	Primaires		
0	Réservé		
1	Rec. UIT-R BT.709		
	primaire	x	y
	vert	0,300	0,600
	bleu	0,150	0,060
	rouge	0,640	0,330
	blanc D65	0,3127	0,3290
2	Non spécifié Les caractéristiques de l'image sont inconnues ou comme déterminé par l'application.		
3	Réservé		
4	Rec. UIT-R BT.470-2 Système M		
	primaire	x	y
	vert	0,21	0,71
	bleu	0,14	0,08
	rouge	0,67	0,33
	blanc C	0,310	0,316
5	Rec. UIT-R BT.470-2 Système B, G		
	primaire	x	y
	vert	0,29	0,60
	bleu	0,15	0,06
	rouge	0,64	0,33
	blanc D65	0,3127	0,3290
6	Société des ingénieurs d'image animée et de télévision 170M		
	primaire	x	y
	vert	0,310	0,595
	bleu	0,155	0,070
	rouge	0,630	0,340
	blanc D65	0,3127	0,3290
7	Société des ingénieurs d'image animée et de télévision 240M (1987)		
	primaire	x	y
	vert	0,310	0,595
	bleu	0,155	0,070
	rouge	0,630	0,340
	blanc D65	0,3127	0,3290

Tableau E-3 – Couleurs primaires

Valeur	Primaires
8	Film générique (filtres à couleur utilisant Illuminant C) primaire x y vert 0,243 0,692 (Wratten 58) bleu 0,145 0,049 (Wratten 47) rouge 0,681 0,319 (Wratten 25) blanc C 0,310 0,316
9-255	Réservé

Lorsque l'élément syntaxique colour_primaries n'est pas présent, la valeur de colour_primaries doit être supposée égale à 2 (le chromatisme n'est pas spécifié ou est déterminé par l'application).

transfer_characteristics indique les caractéristiques de transfert optoélectroniques de l'image source, comme spécifié au Tableau E-4, comme une fonction d'une entrée d'intensité optique linéaire L_c avec une gamme analogique de 0 à 1.

Tableau E-4 – Caractéristiques de transfert

Valeur	Caractéristiques de transfert
0	Réservé
1	Rec. UIT-R BT.709 $V = 1,099 L_c^{0,45} - 0,099$ pour $1 \geq L_c \geq 0,018$ $V = 4,500 L_c$ pour $0,018 > L_c$
2	Non spécifié Les caractéristiques de l'image sont inconnues ou sont déterminées par l'application.
3	Réservé
4	Rec. UIT-R BT.470-2 Système M Affichage supposé gamma 2.2
5	Rec. UIT-R BT.470-2 Système B, G Affichage supposé gamma 2.8
6	Société des ingénieurs d'image animée et de télévision 170M $V = 1,099 L_c^{0,45} - 0,099$ pour $1 \geq L_c \geq 0,018$ $V = 4,500 L_c$ pour $0,018 > L_c$
7	Société des ingénieurs d'image animée et de télévision 240M (1987) $V = 1,1115 L_c^{0,45} - 0,1115$ pour $L_c \geq 0,0228$ $V = 4,0 L_c$ pour $0,0228 > L_c$
8	Caractéristiques de transfert linéaires $V = L_c$
9	Caractéristiques de transfert logarithmiques (gamme 100:1) $V = 1,0 - \text{Log}_{10}(L_c) \div 2$ pour $1 \geq L_c \geq 0,01$ $V = 0,0$ pour $0,01 > L_c$

Tableau E-4 – Caractéristiques de transfert

Valeur	Caractéristiques de transfert
10	Caractéristiques de transfert logarithmiques (gamme 316,22777:1) $V = 1,0 - \text{Log}_{10}(L_c) \div 2,5$ pour $1 \geq L_c \geq 0,0031622777$ $V = 0,0$ pour $0,0031622777 > L_c$
11..255	Réservé

Lorsque l'élément syntaxique `transfer_characteristics` n'est pas présent, la valeur de `transfer_characteristics` doit être supposée égale à 2 (les caractéristiques de transfert ne sont pas spécifiées ou sont déterminées par l'application).

matrix_coefficients décrit les coefficients matriciels utilisés pour déduire les signaux luma et chroma d'après les primaires vertes, bleues et rouges, comme spécifié au Tableau E-5.

En utilisant les définitions suivantes:

E'_R , E'_G , et E'_B sont analogues avec des valeurs dans la gamme de 0 à 1.

Le blanc est spécifié comme ayant E'_R égal à 1, E'_G égal à 1, et E'_B égal à 1.

Ensuite:

$$E'_Y = K_R * E'_R + (1 - K_R - K_B) * E'_G + K_B * E'_B \quad (E-7)$$

$$E'_{PB} = 0,5 * (E'_B - E'_Y) \div (1 - K_B) \quad (E-8)$$

$$E'_{PR} = 0,5 * (E'_R - E'_Y) \div (1 - K_R) \quad (E-9)$$

NOTE – Ensuite E'_Y est analogue avec des valeurs dans la gamme de 0 à 1, E'_{PB} et E'_{PR} sont analogues avec des valeurs dans la gamme de -0,5 à 0,5, et le blanc est donné de façon équivalente par $E'_Y = 1$, $E'_{PB} = 0$, $E'_{PR} = 0$.

Tableau E-5 – Coefficients matriciels

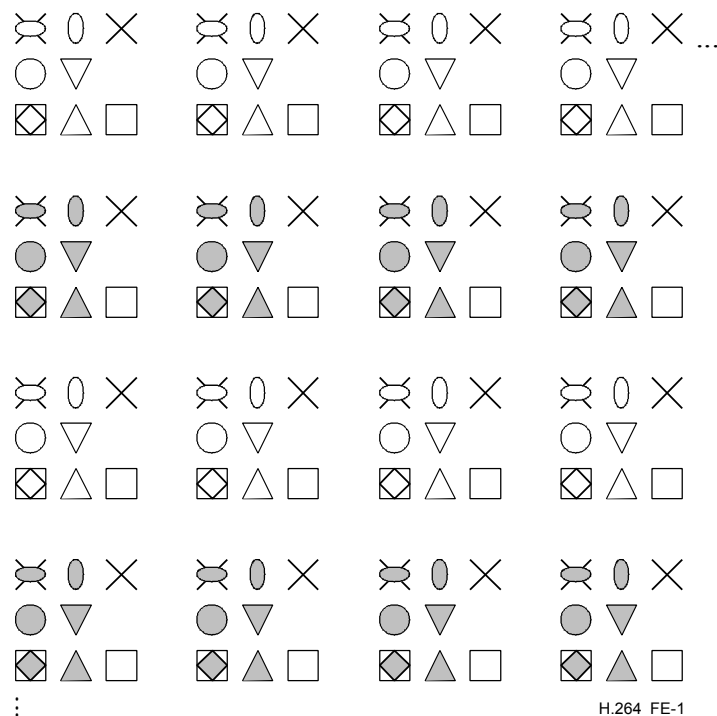
Valeur	Matrice
0	Réservé
1	Rec. UIT-R BT.709 $K_R = 0,2126$; $K_B = 0,0722$
2	Non spécifié Les caractéristiques de l'image sont inconnues ou sont déterminées par l'application.
3	Réservé
4	Commission fédérale des communications $K_R = 0,30$; $K_B = 0,11$
5	Rec. UIT-R BT.470-2 Système B, G: $K_R = 0,299$; $K_B = 0,114$
6	Société des ingénieurs d'image animée et de télévision 170M $K_R = 0,299$; $K_B = 0,114$
7	Société des ingénieurs d'image animée et de télévision 240M (1987) $K_R = 0,212$; $K_B = 0,087$
8-255	Réservé

Lorsque l'élément syntaxique `matrix_coefficients` n'est pas présent, la valeur de `matrix_coefficients` doit être supposée égale à 2.

`chroma_loc_info_present_flag` égal à 1 spécifie que `chroma_sample_loc_type_top_field` et `chroma_sample_loc_type_bottom_field` sont présents. `chroma_loc_info_present_flag` égal à 0 spécifie que `chroma_sample_loc_type_top_field` et `chroma_sample_loc_type_bottom_field` ne sont pas présents.

`chroma_sample_loc_type_top_field` et `chroma_sample_loc_type_bottom_field` spécifient la localisation des échantillons chroma pour le champ supérieur et le champ inférieur comme indiqué à la Figure E-1. La valeur de `chroma_sample_loc_type_top_field` et de `chroma_sample_loc_type_bottom_field` doit être dans la gamme de 0 à 5 inclus. Lorsque `chroma_sample_loc_type_top_field` et `chroma_sample_loc_type_bottom_field` ne sont pas présents, les valeurs de `chroma_sample_loc_type_top_field` et `chroma_sample_loc_type_bottom_field` doivent être supposées égales à 0.

NOTE – Lors du codage de matériau source progressif, `chroma_sample_loc_type_top_field` et `chroma_sample_loc_type_bottom_field` devraient avoir la même valeur.



Interprétation des symboles:

Indications de position des échantillons luma:

× champ supérieur d'échantillon luma □ champ inférieur d'échantillon luma

Indications de position d'un échantillon chroma, où un remplissage gris indique un type d'échantillon de champ inférieur et pas de remplissage indique un type de champ supérieur:

○ échantillon chroma de type 2 ○ échantillon chroma de type 3

○ échantillon chroma de type 0 ▽ échantillon chroma de type 1

◇ échantillon chroma de type 4 △ échantillon chroma de type 5

Figure E-1 – Localisation des échantillons chroma pour les champs supérieurs et inférieurs en fonction de `chroma_sample_loc_type_top_field` et `chroma_sample_loc_type_bottom_field`

`timing_info_present_flag` égal à 1 spécifie que `num_units_in_tick`, `time_scale` et `fixed_frame_rate_flag` sont présents dans le flux binaire. `timing_info_present_flag` égal à 0 spécifie

que `num_units_in_tick`, `time_scale` et `fixed_frame_rate_flag` ne sont pas présents dans le flux binaire.

num_units_in_tick est le nombre d'unités de temps d'une horloge fonctionnant à la fréquence `time_scale` Hz qui correspond à un incrément (appelé un tic-tac d'horloge) d'un compteur de tic-tac d'horloge. `num_units_in_tick` doit être supérieur à 0. Un tic-tac d'horloge est l'intervalle de temps minimal qui peut être représenté dans les données codées. Par exemple, lorsque la fréquence d'horloge d'un signal vidéo est $30000 \div 1001$ Hz, `time_scale` peut être 30000 et `num_units_in_tick` peut être 1001. Voir l'équation C-1.

time_scale est le nombre d'unités de temps qui passent en une seconde. Par exemple, un système de coordonnées temporelles qui mesure le temps en utilisant une horloge à 27 MHz a un `time_scale` de 27 000 000. `time_scale` doit être supérieur à 0.

fixed_frame_rate_flag égal à 1 indique que la distance en temps entre les temps de sortie du HRD de deux images quelconques consécutives dans l'ordre de sortie est soumise aux contraintes suivantes. `fixed_frame_rate_flag` égal à 0 indique qu'aucune contrainte semblable ne s'applique à la distance en temps entre les moments de sortie du HRD de deux images quelconques consécutives dans l'ordre de sortie.

Lorsque `fixed_frame_rate_flag` est égal à 1, pour tout `n` où `n` indique la $n^{\text{ième}}$ image dans l'ordre de sortie et où l'image `n` n'est pas la dernière image dans le flux binaire dans l'ordre de sortie, la valeur de $\Delta t_{fi,dpb}(n)$ est spécifiée par:

$$\Delta t_{fi,dpb}(n) = \Delta t_{o,dpb}(n) \div \text{DeltaTfiDivisor} \quad (\text{E-10})$$

où $\Delta t_{o,dpb}(n)$ est spécifié à l'équation C-13 et `DeltaTfiDivisor` est spécifié par le Tableau E-6 sur la base de la valeur de `pic_struct_present_flag`, `field_pic_flag`, et `pic_struct` pour l'image `n`. Les entrées marquées "-" dans le Tableau E-6 indiquent une absence de dépendance de `DeltaTfiDivisor` à l'élément syntaxique correspondant.

Lorsque `fixed_frame_rate_flag` est égal à 1, la valeur calculée pour $\Delta t_{fi,dpb}(n)$ doit être la même pour tout `n > 0`, et doit être égale à $\text{num_units_in_tick} \div \text{time_scale}$.

Tableau E-6 – Diviseur pour le calcul de $\Delta t_{fi,dpb}(n)$

<code>pic_struct_present_flag</code>	<code>field_pic_flag</code>	<code>pic_struct</code>	<code>DeltaTfiDivisor</code>
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 égal à 1 spécifie que les paramètres NAL HRD (relevant de la conformité de flux binaire de Type II) sont présents. `nal_hrd_parameters_present_flag` égal à 0 spécifie que les paramètres NAL HRD ne sont pas présents.

NOTE – Lorsque `nal_hrd_parameters_present_flag` est égal à 0, on ne peut pas vérifier la conformité du flux binaire sans fournir les paramètres NAL HRD, y compris les informations de paramètre HRD de séquence NAL et tous les messages SEI de période de mise en mémoire tampon et de synchronisation d'image, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.

Lorsque `nal_hrd_parameters_present_flag` est égal à 1, les paramètres NAL HRD (§ E.1.2 et E.2.2) suivent immédiatement le fanion.

La variable `NalHrdBpPresentFlag` est déduite comme suit.

- Si une des conditions suivantes est Vraie, la valeur de `NalHrdBpPresentFlag` doit être mise égale à 1.
 - `nal_hrd_parameters_present_flag` est présent dans le flux binaire et est égal à 1;
 - la nécessité de la présence de périodes de mise en mémoire tampon pour le fonctionnement NAL HRD dans le flux binaire dans les messages SEI de période de mise en mémoire tampon est déterminée par l'application, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.
- Autrement, la valeur de `NalHrdBpPresentFlag` doit être mise égale à 0.

`vcl_hrd_parameters_present_flag` égal à 1 spécifie que les paramètres VCL HRD (relevant de la conformité de tout flux binaire) sont présents. `vcl_hrd_parameters_present_flag` égal à 0 spécifie que les paramètres VCL HRD ne sont pas présents.

NOTE – Lorsque `vcl_hrd_parameters_present_flag` est égal à 0, la conformité du flux binaire ne peut pas être vérifiée sans la fourniture des paramètres VCL HRD et de tous les messages SEI de période de mise en mémoire tampon et de synchronisation d'image, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.

Lorsque `vcl_hrd_parameters_present_flag` est égal à 1, les paramètres VCL HRD (§ E.1.2 et E.2.2) suivent immédiatement le fanion.

La variable `VclHrdBpPresentFlag` est déduite comme suit.

- Si une des conditions suivantes est vraie, la valeur de `VclHrdBpPresentFlag` doit être mise égale à 1.
 - `vcl_hrd_parameters_present_flag` est présent dans le flux binaire et est égal à 1;
 - la nécessité de la présence de périodes de mise en mémoire tampon pour le fonctionnement VCL HRD dans le flux binaire dans les messages SEI de période de mise en mémoire tampon est déterminée par l'application, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.
- Autrement, la valeur de `VclHrdBpPresentFlag` doit être mise égale à 0.

La variable `CpbDpbDelaysPresentFlag` est déduite comme suit.

- Si une des conditions suivantes est vraie, la valeur de `CpbDpbDelaysPresentFlag` doit être mise égale à 1.
 - `nal_hrd_parameters_present_flag` est présent dans le flux binaire et est égal à 1;
 - `vcl_hrd_parameters_present_flag` est présent dans le flux binaire et est égal à 1;
 - la nécessité de la présence de délais de sortie CPB et DPB dans le flux binaire dans les messages SEI de synchronisation d'image est déterminée par l'application, par des moyens non spécifiés dans la présente Recommandation | Norme internationale.
- Autrement, la valeur de `CpbDpbDelaysPresentFlag` doit être mise égale à 0.

`low_delay_hrd_flag` spécifie le mode de fonctionnement du HRD comme spécifié à l'Annexe C. Lorsque `fixed_frame_rate_flag` est égal à 1, `low_delay_hrd_flag` doit être égal à 0.

NOTE – Lorsque `low_delay_hrd_flag` est égal à 1, "les grandes images", qui violent le temps nominal de retrait du CPB à cause du nombre de bits utilisés par une unité d'accès, sont permises. On supposera, sans l'exiger, que de telles "grandes images" ne surviennent qu'occasionnellement.

pic_struct_present_flag égal à 1 spécifie que les messages SEI de synchronisation d'image (§ D.2.2) sont présents et qu'ils incluent l'élément syntaxique `pic_struct`. `pic_struct_present_flag` égal à 0 spécifie que l'élément syntaxique `pic_struct` n'est pas présent dans les messages SEI de synchronisation d'image.

bitstream_restriction_flag égal à 1, spécifie que les paramètres de restriction de la séquence suivante du flux binaire sont présents. `bitstream_restriction_flag` égal à 0, spécifie que les paramètres de restriction de la séquence suivante du flux binaire sont absents.

motion_vectors_over_pic_boundaries_flag égal à 0 indique qu'aucun échantillon en dehors des limites de l'image et aucun échantillon à une position fractionnaire d'échantillon dont la valeur est calculée en utilisant un ou plusieurs échantillons en dehors des limites de l'image n'est utilisé pour l'interprédiction d'un échantillon. `motion_vectors_over_pic_boundaries_flag` égal à 1 indique qu'un ou plusieurs échantillons en dehors des limites de l'image peuvent être utilisés dans l'interprédiction. Lorsque l'élément syntaxique `motion_vectors_over_pic_boundaries_flag` n'est pas présent, la valeur de `motion_vectors_over_pic_boundaries_flag` doit être supposée égale à 1.

max_bytes_per_pic_denom indique un nombre d'octets qui n'est pas dépassé par la somme des tailles des unités NAL VCL associées à toute image codée dans la séquence.

Le nombre d'octets qui représentent une image dans le flux d'unités NAL est spécifié à cette fin comme le nombre total d'octets de données d'unité NAL VCL (c'est-à-dire le total des variables `NumBytesInNALunit` pour les unités NAL VCL) pour l'image. La valeur de `max_bytes_per_pic_denom` doit être dans la gamme de 0 à 16 inclus.

En fonction de `max_bytes_per_pic_denom` on applique ce qui suit.

- Si `max_bytes_per_pic_denom` est égal à 0, aucune limite n'est indiquée.
- Autrement (`max_bytes_per_pic_denom` n'est pas égal à 0), aucune image codée ne doit être représentée dans la séquence par plus que le nombre d'octets suivant:

$$(\text{PicSizeInMbs} * 256 * \text{ChromaFormatFactor}) \div \text{max_bytes_per_pic_denom} \quad (\text{E-11})$$

Lorsque l'élément syntaxique `max_bytes_per_pic_denom` n'est pas présent, la valeur de `max_bytes_per_pic_denom` doit être supposée égale à 2.

max_bits_per_mb_denom indique le nombre maximal de bits codés des données de `macroblock_layer()` pour tout macrobloc dans toute image de la séquence. La valeur de `max_bits_per_mb_denom` doit être dans la gamme de 0 à 16 inclus.

En fonction de `max_bits_per_mb_denom` on applique ce qui suit.

- Si `max_bits_per_mb_denom` est égal à 0, aucune limite n'est spécifiée.
- Autrement (`max_bits_per_mb_denom` n'est pas égal à 0), aucun `macroblock_layer()` codé ne doit être représenté dans le flux binaire par plus que le nombre de bits suivant:

$$(2048 * \text{ChromaFormatFactor} + 128) \div \text{max_bits_per_mb_denom} \quad (\text{E-12})$$

En fonction de `entropy_coding_mode_flag`, les bits des données de `macroblock_layer()` sont comptés comme suit.

- Si `entropy_coding_mode_flag` est égal à 0, le nombre de bits des données de `macroblock_layer()` est donné par le nombre de bits dans la structure syntaxique `macroblock_layer()` pour un macrobloc.
- Autrement (`entropy_coding_mode_flag` est égal à 1), le nombre de bits des données de `macroblock_layer()` pour un macrobloc est donné par le nombre de fois que `read_bits(1)` est

appelé aux § 9.3.3.2.2 et 9.3.3.2.3 lors de l'analyse grammaticale du macrobloc_layer() associé au macrobloc.

Lorsque l'élément syntaxique `max_bits_per_mb_denom` n'est pas présent, la valeur de `max_bits_per_mb_denom` doit être supposée égale à 1.

log2_max_mv_length_horizontal et **log2_max_mv_length_vertical** indiquent la valeur absolue maximale d'une composante, respectivement horizontale et verticale, de vecteur cinétique décodé, en unités de 1/4 d'échantillon luma, pour toutes les images dans la séquence. Une valeur de `n` atteste qu'aucune valeur de composante ne doit être en dehors de la gamme de -2^n à $2^n - 1$ inclus, en unités de déplacement d'un quart d'échantillon luma. La valeur de `log2_max_mv_length_horizontal` doit être dans la gamme de 0 à 16 inclus. La valeur de `log2_max_mv_length_vertical` doit être dans la gamme de 0 à 16 inclus. Lorsque `log2_max_mv_length_horizontal` n'est pas présent, les valeurs de `log2_max_mv_length_horizontal` et `log2_max_mv_length_vertical` doivent être supposées égales à 16.

NOTE – La valeur absolue maximal d'une composante verticale ou horizontale de vecteur cinétique décodé comporte aussi des limites de profil et de niveau, comme spécifié à l'Annexe A.

num_reorder_frames indique le nombre maximal de trames, paires de champs complémentaires, ou champs non appariés qui précède toute trame, paire de champs complémentaires, ou champ non apparié dans la séquence dans l'ordre de décodage, et la suit dans l'ordre de sortie. La valeur de `num_reorder_frames` doit être dans la gamme de 0 à `max_dec_frame_buffering`, inclus. Lorsque l'élément syntaxique `num_reorder_frames` n'est pas présent, la valeur de `num_reorder_frames` doit être supposée égale à `max_dec_frame_buffering`.

max_dec_frame_buffering spécifie la taille requise de la mémoire tampon d'image HRD décodée (DPB) en unités de mémoire tampon de trame. La séquence ne doit pas exiger une mémoire tampon d'image décodée d'une taille supérieure à $\text{Max}(1, \text{max_dec_frame_buffering})$ pour permettre la sortie des images décodées aux moments de sortie spécifiés par le `dpb_output_delay` des messages SEI de synchronisation de l'image. La valeur de `max_dec_frame_buffering` doit être dans la gamme de `num_ref_frames` à `MaxDpbSize` (comme spécifié au § A.3.1), inclus. Lorsque l'élément syntaxique `max_dec_frame_buffering` n'est pas présent, la valeur de `max_dec_frame_buffering` doit être supposée égale à `MaxDpbSize`.

E.2.2 Sémantique des paramètres HRD

cpb_cnt_minus1 plus 1 spécifie le nombre de spécifications CPB alternatives dans le flux binaire. La valeur de `cpb_cnt_minus1` doit être dans la gamme de 0 à 31 inclus. Lorsque `low_delay_hrd_flag` est égal à 1, `cpb_cnt_minus1` doit être égal à 0. Lorsque `cpb_cnt_minus1` n'est pas présent, il doit être supposé égal à 0.

bit_rate_scale (avec `bit_rate_value_minus1[SchedSelIdx]`) spécifie le débit binaire d'entrée maximal du $\text{SchedSelIdx}^{\text{ème}}$ CPB.

cpb_size_scale (avec `cpb_size_value_minus1[SchedSelIdx]`) spécifie la taille CPB du $\text{SchedSelIdx}^{\text{ème}}$ CPB.

bit_rate_value_minus1[SchedSelIdx] (avec `bit_rate_scale`) spécifie le débit binaire d'entrée maximal pour le $\text{SchedSelIdx}^{\text{ème}}$ CPB. `bit_rate_value_minus1[SchedSelIdx]` doit être dans la gamme de 0 à $2^{32} - 2$ inclus. Pour tout `SchedSelIdx > 0`, `bit_rate_value_minus1[SchedSelIdx]` doit être supérieur à `bit_rate_value_minus1[SchedSelIdx - 1]`. Le débit binaire en bits par seconde est donné par

$$\text{BitRate}[\text{SchedSelIdx}] = (\text{bit_rate_value_minus1}[\text{SchedSelIdx}] + 1) * 2^{(6 + \text{bit_rate_scale})} \quad (\text{E-13})$$

Lorsque l'élément syntaxique `bit_rate_value_minus1[SchedSelIdx]` n'est pas présent, `BitRate[SchedSelIdx]` doit être supposé égal à $1000 * \text{MaxBR}$ pour les paramètres VCL HRD.

Lorsque l'élément syntaxique `bit_rate_value_minus1[SchedSelIdx]` n'est pas présent, `BitRate[SchedSelIdx]` doit être supposé égal à $1200 * \text{MaxBR}$ pour les paramètres NAL HRD.

`cpb_size_value_minus1[SchedSelIdx]` est utilisé avec `cpb_size_scale` pour spécifier la taille du `SchedSelIdxème` CPB. `cpb_size_value_minus1[SchedSelIdx]` doit être dans la gamme de 0 à $2^{32} - 2$ inclus. Pour tout `SchedSelIdx` supérieur à 0, `cpb_size_value_minus1[SchedSelIdx]` doit être inférieur ou égal à `cpb_size_value_minus1[SchedSelIdx - 1]`.

La taille du CPB en bits est donnée par

$$\text{CpbSize[SchedSelIdx]} = (\text{cpb_size_value_minus1[SchedSelIdx]} + 1) * 2^{(4 + \text{cpb_size_scale})} \quad (\text{E-14})$$

Lorsque l'élément syntaxique `cpb_size_value_minus1[SchedSelIdx]` n'est pas présent, `CpbSize[SchedSelIdx]` doit être supposé égal à $1000 * \text{MaxCPB}$ pour les paramètres VCL HRD.

Lorsque l'élément syntaxique `cpb_size_value_minus1[SchedSelIdx]` n'est pas présent, `CpbSize[SchedSelIdx]` doit être supposé égal à $1200 * \text{MaxCPB}$ pour les paramètres NAL HRD.

Pour les paramètres VCL HRD, il doit y avoir au moins une valeur de `SchedSelIdx` pour laquelle `BitRate[SchedSelIdx] <= 1000*MaxBR` et `CpbSize[SchedSelIdx] <= 1000 * MaxCPB` (comme spécifié au § A.3.1).

Pour les paramètres NAL HRD, il doit y avoir au moins une valeur de `SchedSelIdx` pour laquelle `CpbSize[SchedSelIdx] <= 1200*MaxCPB` et `BitRate[SchedSelIdx] <= 1200*MaxBR`.

`cbr_flag[SchedSelIdx]` égal à 0 spécifie que pour faire décoder ce flux binaire par le HRD en utilisant la `SchedSelIdxème` spécification CPB, le programmeur factice de livraison de flux (HSS) fonctionne dans un mode de débit binaire intermittent. `cbr_flag[SchedSelIdx]` égal à 1 spécifie que le HSS fonctionne en mode de débit binaire constant (CBR). Lorsque l'élément syntaxique `cbr_flag[SchedSelIdx]` n'est pas présent, la valeur de `cbr_flag` doit être supposée égale à 0.

`initial_cpb_removal_delay_length_minus1` spécifie la longueur en bits des éléments syntaxiques `initial_cpb_removal_delay[SchedSelIdx]` et `initial_cpb_removal_delay_offset[SchedSelIdx]` du message SEI de période de mise en mémoire tampon. La longueur de `initial_cpb_removal_delay[SchedSelIdx]` et de `initial_cpb_removal_delay_offset[SchedSelIdx]` est `initial_cpb_removal_delay_length_minus1 + 1`. Lorsque l'élément syntaxique `initial_cpb_removal_delay_length_minus1` est présent dans plus d'une structure syntaxique `hrd_parameters()` au sein de la structure syntaxique des paramètres VUI, la valeur des paramètres `initial_cpb_removal_delay_length_minus1` doit être égale dans les deux structures syntaxiques `hrd_parameters()`. Lorsque l'élément syntaxique `initial_cpb_removal_delay_length_minus1` n'est pas présent, il doit être supposé égal à 23.

`cpb_removal_delay_length_minus1` spécifie la longueur en bits de l'élément syntaxique `cpb_removal_delay`. La longueur de l'élément syntaxique `cpb_removal_delay` du message SEI de synchronisation d'image est `cpb_removal_delay_length_minus1 + 1`. Lorsque l'élément syntaxique `cpb_removal_delay_length_minus1` est présent dans plus d'une structure syntaxique `hrd_parameters()` au sein de la structure syntaxique de paramètres VUI, la valeur des paramètres `cpb_removal_delay_length_minus1` doit être égale dans les deux structures syntaxiques `hrd_parameters()`. Lorsque l'élément syntaxique `cpb_removal_delay_length_minus1` n'est pas présent, il doit être supposé égal à 23.

`dpb_output_delay_length_minus1` spécifie la longueur en bits de l'élément syntaxique `dpb_output_delay`. La longueur de l'élément syntaxique `dpb_output_delay` du message SEI de synchronisation d'image est `dpb_output_delay_length_minus1 + 1`. Lorsque l'élément syntaxique `dpb_output_delay_length_minus1` est présent dans plus d'une structure syntaxique `hrd_parameters()` au sein de la structure syntaxique des paramètres VUI, la valeur des paramètres `dpb_output_delay_length_minus1` doit être égale dans les deux structures syntaxiques `hrd_parameters()`. Lorsque l'élément syntaxique `dpb_output_delay_length_minus1` n'est pas présent, il doit être supposé égal à 23.

time_offset_length supérieur à 0 spécifie la longueur en bits de l'élément syntaxique `time_offset`. `time_offset_length` égal à 0 spécifie que l'élément syntaxique `time_offset` n'est pas présent. Lorsque l'élément syntaxique `time_offset_length` est présent dans plus d'une structure syntaxique `hrd_paramètres()` de la structure syntaxique des paramètres VUI, la valeur des paramètres `time_offset_length` doit être égale dans les deux structures syntaxiques `hrd_parameters()`. Lorsque l'élément syntaxique `time_offset_length` n'est pas présent, il doit être supposé égal à 24.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série D	Principes généraux de tarification
Série E	Exploitation générale du réseau, service téléphonique, exploitation des services et facteurs humains
Série F	Services de télécommunication non téléphoniques
Série G	Systèmes et supports de transmission, systèmes et réseaux numériques
Série H	Systèmes audiovisuels et multimédias
Série I	Réseau numérique à intégration de services
Série J	Réseaux câblés et transmission des signaux radiophoniques, télévisuels et autres signaux multimédias
Série K	Protection contre les perturbations
Série L	Construction, installation et protection des câbles et autres éléments des installations extérieures
Série M	RGT et maintenance des réseaux: systèmes de transmission, circuits téléphoniques, télégraphie, télécopie et circuits loués internationaux
Série N	Maintenance: circuits internationaux de transmission radiophonique et télévisuelle
Série O	Spécifications des appareils de mesure
Série P	Qualité de transmission téléphonique, installations téléphoniques et réseaux locaux
Série Q	Commutation et signalisation
Série R	Transmission télégraphique
Série S	Equipements terminaux de télégraphie
Série T	Terminaux des services télématiques
Série U	Commutation télégraphique
Série V	Communications de données sur le réseau téléphonique
Série X	Réseaux de données et communication entre systèmes ouverts
Série Y	Infrastructure mondiale de l'information, protocole Internet et réseaux de nouvelle génération
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication