UNION INTERNATIONALE DES TÉLÉCOMMUNICATIONS

UIT-T T.88

SECTEUR DE LA NORMALISATION
DES TÉLÉCOMMUNICATIONS
DE L'UIT

(02/2000)

SÉRIE T: TERMINAUX DES SERVICES TÉLÉMATIQUES

Technologies de l'information – Codage avec ou sans perte des images au trait

Recommandation UIT-T T.88

(Antérieurement Recommandation du CCITT)

NORME INTERNATIONALE ISO/CEI 14492

RECOMMANDATION UIT-T T.88

TECHNOLOGIES DE L'INFORMATION – CODAGE AVEC OU SANS PERTE DES IMAGES AU TRAIT

Résumé

La présente Recommandation | Norme internationale, officieusement appelée "JBIG2", définit une métode de codage pour les images à deux niveaux (par exemple les imprimés en noir et blanc). Il s'agit d'images se composant d'un unique plan binaire rectangulaire dont chaque pixel ne prend qu'une de deux couleurs possibles. La présente Recommandation | Norme internationale, a été explicitement élaborée pour une compression d'images avec pertes, sans pertes et avec pertes convergeant vers zéro.

Source

La Recommandation T.88 de l'UIT-T, a été approuvée le 10 février 2000. Un texte identique est publié comme Norme Internationale ISO/CEI 14492.

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.

La Conférence mondiale de normalisation des télécommunications (CMNT), 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 la CMNT.

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.

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 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.

TABLE DES MATIÈRES

0 Introduction

La présente Recommandation | Norme internationale, officieusement appelée JBIG2, définit une méthode de codage pour les images à deux niveaux (par exemple imprimés en noir et blanc). Il s'agit d'images se composant d'un unique plan binaire rectangulaire dont chaque pixel ne prend qu'une des deux couleurs possibles. Les images polychromes doivent être traitées au moyen d'une norme de niveau supérieur comme la Recommandation UIT-T T.44, qui est en cours de rédaction par le Groupe mixte d'experts en images à deux niveaux (JBIG), "équipe collaborative" créée en 1988, qui rend compte à la fois au GT1 de l'ISO/CEI JTC 1/SC29 et à l'UIT-T.

La compression de ce type d'image est également traitée par des normes portant sur la télécopie, par exemple par les algorithmes de compression des Recommandations UIT-T T.4 (MH, MR), T.6 (MMR), T.82 (JBIG1) et T.85 (profil d'application du format JBIG1 pour la télécopie). En dehors de son application évidente à la télécopie, le format JBIG2 sera utile pour la mémorisation et l'archivage des documents, le codage des images sur la toile mondiale (WWW), la transmission de données sans fil, la gestion des impressions différées et même les téléconférences.

A la suite d'un processus qui s'est achevé en 1993, le groupe JBIG a établi une première norme de codage, officiellement désignée comme Recommandation UIT-T T.82 | Norme internationale ISO/CEI 11544, qui est officieusement connue sous l'appellation JBIG ou JBIG1. Ce dernier format est destiné à permettre un codage sans pertes et progressif (à pertes convergeant vers zéro). Bien que le format JBIG1 permette le codage avec pertes, les images avec pertes ainsi obtenues ont une qualité nettement inférieure aux images originales parce que le nombre de pixels contenus dans ces images ne peut pas dépasser un quart de ceux de l'image originale.

En revanche, le format JBIG2 a été explicitement élaboré pour une compression d'images avec pertes, sans pertes et avec pertes convergeant vers zéro. L'objectif théorique assigné au format JBIG2 était d'obtenir une meilleure performance de compression sans pertes que celle des normes existantes et de permettre une compression avec pertes à des taux beaucoup plus élevés que ceux des normes existantes sans dégradation de qualité visuelle ou presque. Par ailleurs, le format JBIG2 permet un codage qualitativement progressif (allant d'une qualité inférieure à une qualité supérieure ou sans pertes) et un codage quantitativement progressif (ajoutant différents types de données d'image, par exemple du texte d'abord puis des dégradés). Un codeur JBIG2 typique décompose l'image bitonale d'entrée en plusieurs régions et code séparément ces régions au moyen d'une méthode de codage différente. Une telle décomposition fondée sur le contenu est très souhaitable, en particulier dans les applications multimédias interactives. Le format JBIG2 peut également traiter une série d'images (document de plusieurs pages) d'une façon explicite.

Comme cela est normal dans une norme de compression d'image, la norme JBIG2 définit explicitement les exigences d'un flux binaire compatible et définit ensuite le comportement du décodeur. Elle ne définit pas explicitement un codeur normalisé mais elle est assez flexible pour autoriser des codeurs de conception évoluée. En fait, la conception du codeur sera un facteur déterminant pour différencier des réalisations JBIG2 concurrentes.

Bien que la présente Recommandation | Norme internationale soit rédigée en termes d'actions exécutées par des décodeurs pour interpréter un flux binaire, tout décodeur qui produit le résultat correct (tel que défini par ces actions) sera compatible, quelles que soient les actions qu'il exécute réellement.

Les Annexes A, B, C, D, E et F sont normatives et font donc partie intégrante de la présente Recommandation | Norme internationale. Les Annexes G et H sont informatives et ne font donc pas partie intégrante de la présente Recommandation | Norme internationale.

0.1 Interprétation et utilisation des prescriptions

Cette section est informative et conçue pour faciliter l'interprétation des prescriptions de la présente Recommandation | Norme internationale. Ces prescriptions sont rédigées de façon à être aussi générales que possible afin d'autoriser un niveau élevé de flexibilité lors de la réalisation. La rédaction des prescriptions ne donne donc pas de précisions sur les applications ou les réalisations. Dans cette section, une correspondance est établie entre la rédaction générale des prescriptions et l'usage prévu de la présente Recommandation | Norme internationale dans des applications typiques.

0.1.1 Objet du codage JBIG2

La présente norme JBIG2 est utilisée pour coder des documents à deux niveaux (bitonaux), contenant une ou plusieurs pages. Une page normale contient quelques données alphanumériques, c'est-à-dire quelques caractères de petites dimensions disposés en rangées horizontales ou verticales. Les caractères de la partie alphanumérique d'une page sont appelés *symboles* dans la norme JBIG2. Une page peut également contenir des *données de dégradé* c'est-à-dire des images en échelle de gris ou à plusieurs niveaux chromatiques (par exemple des photographies) qui ont été estompées afin de produire des images à deux niveaux. Les cellules matricielles périodiques dans la partie en dégradé de l'image sont appelées *structures* dans la présente norme JBIG2. Par ailleurs, une page peut contenir d'autres données comme du trait et du bruit. De telles données en noir et blanc, qui ne sont ni des lettres ni du gris, sont appelées *données génériques* dans la présente Recommandation | Norme internationale.

Le modèle d'image JBIG2 traite les données alphanumériques et de dégradé comme des cas particuliers. L'on s'attend qu'un décodeur JBIG2 subdivisera le contenu d'une page en une région alphanumérique contenant du texte numérisé, en une région de dégradé contenant des dégradés numérisés et en une région générique contenant le reste des données d'image numérisées, comme le trait. Dans certaines circonstances, il est préférable (en termes de qualité d'image ou de volume de données comprimées) de considérer les lettres ou les dégradés comme des données génériques; inversement, il est parfois préférable de considérer des données génériques comme appartenant à des cas particuliers précédents.

Un codeur a la possibilité de subdiviser une page donnée en un nombre quelconque de régions mais souvent trois régions seront suffisantes: l'une pour les symboles alphanumériques, l'autre pour les structures en dégradé et la troisième pour les données génériques restantes. Dans certains cas, tous les types de données peuvent ne pas être présents et la page peut se composer d'un nombre de régions inférieur à trois.

Les diverses régions peuvent se superposer sur la même page physique. La présente norme JBIG2 permet de spécifier la façon dont les régions en superposition se recombinent afin de former l'image de la page finale.

Une région alphanumérique se compose d'un certain nombre de symboles placés à des emplacements spécifiés d'un fond d'écran. Ces symboles correspondent habituellement à des caractères d'écriture individuels. La présente norme JBIG2 tire beaucoup de son efficacité du fait qu'elle utilise plusieurs fois des symboles individuels. Pour réutiliser un symbole, un codeur ou décodeur doit disposer d'un moyen rapide d'y faire référence. Dans la présente norme JBIG2, les symboles sont collationnés dans un ou plusieurs dictionnaires de symboles. Un dictionnaire de symboles est un ensemble de matrices de symboles alphanumériques, indexées de façon qu'une matrice de symbole puisse être citée en référence au moyen d'un numéro indiciel.

Une région de dégradé se compose d'un certain nombre de structures placées selon une grille régulière. Les structures correspondent habituellement à des valeurs d'échelle de gris. En réalité, la méthode de codage des indices structurels est conçue comme un codeur de niveaux de gris. L'on peut effectuer la compression de ces données en représentant les pixels binaires d'une cellule de grille donnée par un entier unique: l'indice de dégradé (qui est habituellement une valeur restituée en échelle de gris). Cette application convergente (de la structure d'une cellule vers une valeur d'échelle de gris) peut avoir pour effet que l'information de bordure, présente dans la phototrame originale, soit perdue par le codage du dégradé. C'est pourquoi un codage sans pertes ou presque sans pertes des dégradés sera souvent préférable en terme de qualité d'image (bien que les dimensions de celles-ci soient plus grandes) si ces dégradés sont analysés par codage des données génériques plutôt que par codage des données de dégradé.

0.1.2 Relation entre segments et documents

Un fichier JBIG2 contient les informations nécessaires pour décoder un document à deux niveaux. Un tel fichier se compose de *segments*. Une page type est codée au moyen de plusieurs segments. Dans un cas simple, il y a un segment d'information de page, un segment de dictionnaire de symboles, un segment de région alphanumérique, un segment de dictionnaire de structures, un segment de région de dégradé et un segment de fin de page. Le segment d'information de page donne des renseignements généraux sur la page, comme ses dimensions et sa résolution. Les segments de dictionnaire rassemblent les phototrames auxquelles il est fait référence dans les segments de région. Les segments régionaux décrivent l'aspect des régions de texte et de dégradé en faisant référence à des phototrames issues d'un dictionnaire et en spécifiant l'endroit où elles doivent apparaître sur la page. Le segment de fin de page indique la fin de la page.

0.1.3 Structure et utilisation des segments

Chaque segment contient un en-tête de segment, un en-tête de données et des données. L'en-tête de segment sert à acheminer des informations de référence de segment et, dans le cas de documents à pages multiples, des informations d'association de page. L'en-tête de données contient des informations utilisées pour décoder les données contenues dans le segment. Ces données décrivent une région d'image ou un dictionnaire, ou donnent d'autres informations.

Les segments sont numérotés en séquence. Un segment peut se rapporter à un segment de numéro inférieur ou *antérieur*. Un segment de région est toujours associé à une page spécifique du document. Un segment de dictionnaire peut être associé à une page spécifique du document ou être associé au document dans son ensemble.

Un segment de région peut faire référence à un ou plusieurs segments de dictionnaire antérieurs. L'objet d'une telle référence est de permettre au décodeur d'identifier dans un segment de dictionnaire des symboles qui sont présents dans l'image.

Un segment de région peut faire référence à un segment de région antérieur. L'objet d'une telle référence est de combiner l'image décrite par le segment antérieur avec la représentation actuelle de la page.

Un segment de dictionnaire peut faire référence à des segments de dictionnaire antérieurs. Les symboles ajoutés à un segment de dictionnaire peuvent être décrits directement ou en tant que raffinements de symboles déjà décrits, soit dans le même segment de dictionnaire soit dans des segments de dictionnaire antérieurs.

Un fichier JBIG2 peut être organisé de deux façons: en séquence ou aléatoirement. Dans l'organisation séquentielle, chaque en-tête de segment précède immédiatement l'en-tête de données et les données de ce segment, l'ordre séquentiel étant appliqué à tous les éléments. Dans l'organisation à accès aléatoire, tous les en-têtes de segment sont rassemblés au début du fichier et sont suivis des données (y compris les en-têtes de données) pour tous les segments, dans le même ordre. Cette deuxième organisation permet à un décodeur de déterminer toutes les dépendances de segment sans lire le fichier entier.

Une troisième façon d'encapsuler des données à codage JBIG2 consiste à les imbriquer dans un fichier non JBIG2, ce qui est parfois appelé *organisation imbriquée*. Dans ce cas, un format de fichier différent achemine les segments JBG2. L'en-tête de segment, l'en-tête de données et les données de chaque segment sont mémorisés ensemble mais le format du fichier d'imbrication peut conserver les segments selon un ordre quelconque et à un emplacement quelconque de sa propre structure.

0.1.4 Représentations internes

Les données décodées doivent toujours être mémorisées avant d'être imprimées ou affichées. Bien que la présente Recommandation | Norme internationale ne spécifie pas leur mode de stockage, leur modèle de décodage présuppose certaines structures de données, précisément des tampons et des dictionnaires. La Figure 1 décrit les principaux composants du décodeur et leurs tampons associés. Dans cette figure, les procédures de décodage sont indiquées en traits gras et les composants de mémoire en traits non gras. De même, les flèches en gras indiquent qu'une procédure de décodage donnée invoque une autre procédure de décodage. Par exemple, la procédure de décodage par dictionnaire de symboles invoque la procédure de décodage de région générique pour décoder les phototrames pour les symboles qu'elle définit. Les flèches maigres indiquent les flux de données: la procédure de décodage de la région alphanumérique lit les symboles dans la mémoire de symboles et les insère dans le tampon de page ou dans un tampon auxiliaire. Bien que cela ne soit pas représenté sur la Figure 1, le flux de données codées s'écoule vers les procédures de décodage et le bloc étiqueté "Tampons de page et tampons auxiliaires" produit les images finales des pages décodées.

Les ressources requises pour décoder un flux binaire JBIG2 donné dépendent de la complexité de ce flux. Certaines techniques comme le découpage en bandes peuvent servir à réduire les besoins en mémoire du décodeur. L'on estime qu'un décodeur à caractéristiques complètes peut nécessiter, afin de décoder la plupart des flux binaires, deux tampons de page complète plus environ la même quantité de mémoire pour le dictionnaire, plus environ 100 kbits de mémoire pour le contexte de codage arithmétique.

Un tampon est une représentation d'une phototrame. Il est destiné à conserver une grande quantité de données, normalement du volume d'une page. Un tampon peut contenir la description d'une région ou d'une page entière. Même si le tampon ne décrit qu'une région, il possède les informations qui y sont associées pour spécifier son emplacement dans la page. Le décodage d'un segment de région modifie le contenu d'un tampon.

Il existe un tampon spécial: le *tampon de page*. Il permet au décodeur d'y accumuler directement des données de page jusqu'à ce que celle-ci ait été complètement décodée; puis les données peuvent être envoyées vers un dispositif de sortie ou vers un fichier. Le décodage d'un segment de région *immédiate* modifie le contenu du tampon de page. La façon habituelle de préparer une page consiste à décoder un ou plusieurs segments de région immédiate dont chacun modifie le tampon de page. Le décodeur peut produire en sortie un tampon de page incomplet, soit dans le cadre d'une transmission en mode progressif soit en réponse à une commande de l'utilisateur. Ce type de sortie est facultatif et son contenu n'est pas spécifié par la présente Recommandation | Norme internationale.

Tous les autres tampons sont des tampons auxiliaires. L'on prévoit que le décodeur remplit un tampon auxiliaire puis l'utilise pour raffiner le tampon de page. Dans une application donnée, il sera souvent inutile d'avoir des tampons auxiliaires. Le décodage d'un segment de région *intermédiaire* modifie le contenu d'un tampon auxiliaire. Le décodeur peut utiliser des tampons auxiliaires pour produire des pages autres que celles qui se trouvent dans un tampon de page complète, soit dans le cadre d'une transmission en mode progressif soit en réponse à une commande de l'utilisateur. Ce type de sortie est facultatif et son contenu n'est pas spécifié par la présente Recommandation | Norme internationale.

Un dictionnaire de symboles se compose d'un ensemble de matrices indexées. Les matrices contenues dans un dictionnaire sont normalement petites, de la dimension de caractères alphanumériques environ. A la différence d'un tampon, une matrice de dictionnaire n'a pas d'information de mise en page associée.

0.1.5 Résultats du décodage

Le décodage d'un segment implique l'invocation d'une ou de plusieurs procédures de décodage, qui sont déterminées par le type de segment.

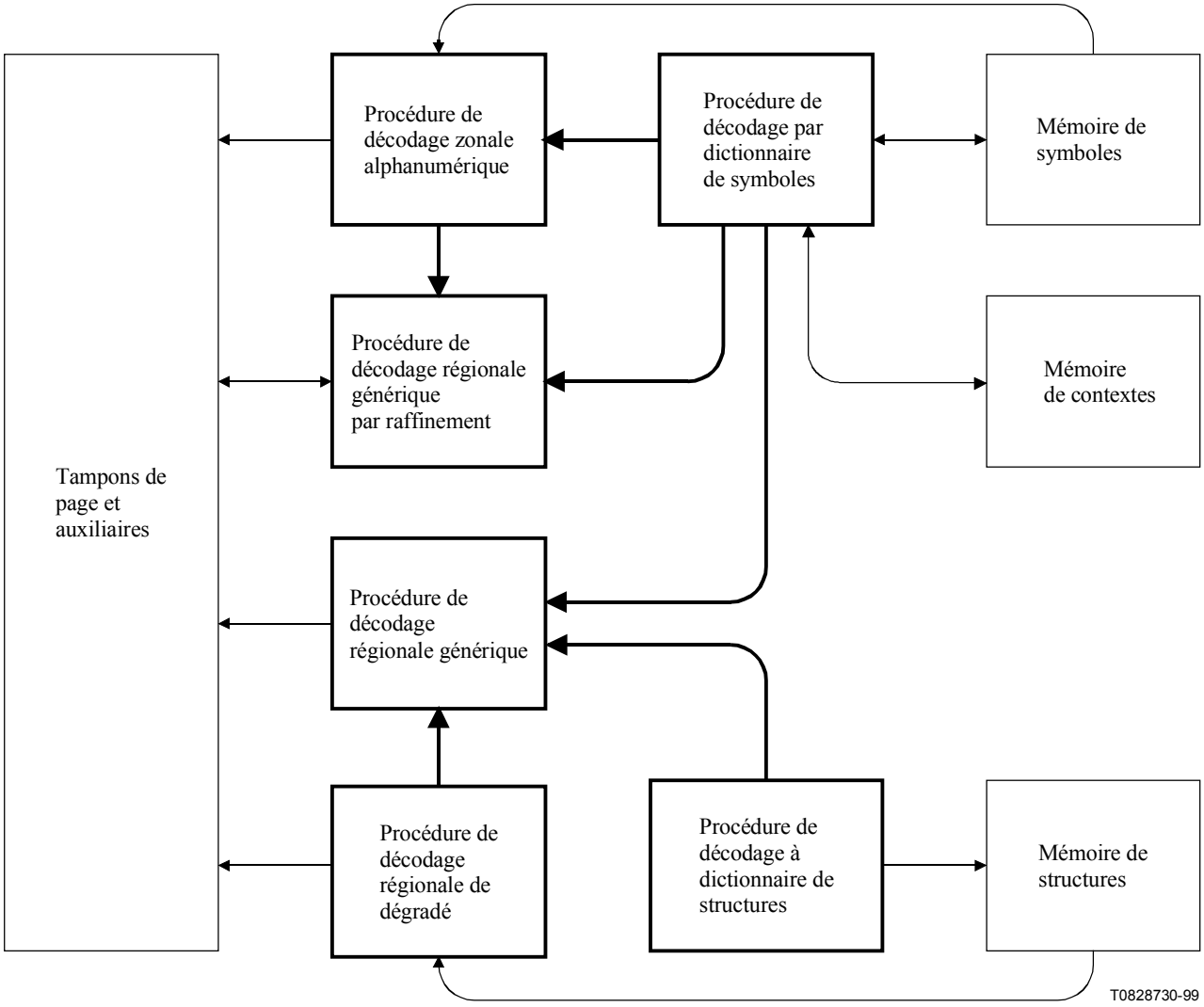


Figure 1 – Schéma fonctionnel des principaux composants du décodeur

Le résultat du décodage d'un segment de région est une phototrame mémorisée dans un tampon, qui peut être le tampon de page. Le décodage d'un segment de région peut remplir un nouveau tampon ou peut modifier un tampon existant. Dans les applications typiques, le fait de placer les données dans un tampon implique que les pixels passent de la couleur du fond à celle de l'avant-plan. Mais la présente Recommandation | Norme internationale spécifie d'autres moyens acceptables de modifier les pixels d'un tampon.

Une page normale sera décrite par un ou plusieurs segments de région immédiate, dont chacun produit une modification du tampon de page.

De même qu'il est possible de spécifier un nouveau symbole dans un dictionnaire en raffinant un symbole déjà spécifié, il est possible de spécifier un nouveau tampon en raffinant un tampon existant. Une région ne peut cependant être raffinée que par la procédure de décodage générique par raffinement. Un tel raffinement ne fait pas appel à la structure interne de la région contenue dans le tampon qui est raffiné. Une fois qu'un tampon a été raffiné, le tampon original n'est plus disponible.

Le résultat du décodage d'un segment de dictionnaire est un nouveau dictionnaire. Les symboles contenus dans ce dictionnaire pourront être placés ultérieurement dans un tampon par la procédure de décodage zonale alphanumérique.

0.1.6 Procédures de décodage

La *procédure de décodage régionale générique* remplit ou modifie un tampon directement, pixel par pixel si le codage arithmétique est utilisé ou par insertions de pixels d'avant-plan et d'arrière-plan si le codage MMR ou le codage de Huffman est utilisé. Dans le cas du codage arithmétique, le contexte de prédiction ne contient que les pixels déterminés par les données déjà décodées dans le segment en cours.

La *procédure de décodage régionale générique par raffinement* modifie un tampon pixel par pixel au moyen du codage arithmétique. Le contexte de prédiction utilise des pixels déterminés par des données déjà décodées dans le segment en cours ainsi que des pixels déjà présents dans le tampon de page ou dans un tampon auxiliaire.

La *procédure de décodage zonale alphanumérique* extrait les symboles d'un ou de plusieurs dictionnaires de symboles et les place dans un tampon. Cette procédure est invoquée au cours du décodage d'un segment de région alphanumérique, qui contient les informations de position et d'indice pour chaque symbole à placer dans le tampon. Les phototrames des symboles sont extraites des dictionnaires de symboles.

La *procédure de décodage par dictionnaire de symboles* crée un dictionnaire de symboles, c'est-à-dire un ensemble indexé de matrices symboliques. Dans un dictionnaire, une matrice peut être codée soit directement, soit en tant que raffinement d'un symbole déjà présent dans un dictionnaire ou comme une agrégation de deux ou plus de deux symboles déjà présents dans des dictionnaires. Cette procédure de décodage est invoquée au cours du décodage d'un segment de dictionnaire de symboles.

La *procédure de décodage régionale de dégradé* extrait des structures d'un dictionnaire de structures et les place dans un tampon. Cette procédure est invoquée au cours du décodage d'un segment de région de dégradé. Ce segment contient les informations de position pour toutes les structures à placer dans le tampon ainsi que des informations indicielles pour les structures elles-mêmes. Les structures, qui sont les matrices de dimensions fixes du dégradé, sont extraites des dictionnaires de dégradés.

La *procédure de décodage à dictionnaire de structures* crée un dictionnaire, c'est-à-dire un ensemble indexé de matrices de dimensions fixes (structures). Les matrices contenues dans le dictionnaire sont codées directement et conjointement. Cette procédure de décodage est invoquée au cours du décodage d'un segment de dictionnaire de structures.

La *procédure de décodage de commandes* décode les en-têtes de segment. Ceux-ci contiennent des informations relatives au type de segment qui déterminent la procédure de décodage qui doit être invoquée pour décoder le segment. Le type de segment détermine également l'endroit où seront placées les informations de sortie décodées. Les informations de référence du segment, également présentes dans l'en-tête de segment et décodées par la procédure de décodage de commandes, déterminent les autres segments qui doivent être utilisés pour décoder le segment en cours. La procédure de décodage de commandes s'applique à tous les éléments représentés sur la Figure 1 et n'y est donc pas représentée comme bloc distinct.

Le Tableau 1 résume les types de données à décoder, la procédure de décodage correspondante et les représentations finales des données décodées.

Tableau 1 – Entités participant au processus de décodage

Concept	Entité de flux JBIG2	Entité de décodage JBIG2	Représentation physique
Document	Fichier JBIG2	Décodeur JBIG2	Support ou dispositif de sortie
Page	Assemblage de segments	Entité implicite dans la procédure de décodage des commandes	Tampon de page
Région	Segment de région	Procédure de décodage zonale	Tampon de page ou tampon auxiliaire
Dictionnaire	Segment de dictionnaire	Procédure de décodage par dictionnaire	Liste de symboles
Caractère	Champ contenu dans un segment de dictionnaire de symboles	Procédure de décodage par dictionnaire de symboles	Matrice de symbole
Niveau de gris	Champ contenu dans un segment de dictionnaire de dégradés	Procédure de décodage à dictionnaire de structures	Structure

0.2 Codage avec pertes

La présente Recommandation | Norme internationale ne définit pas la façon de commander le codage avec pertes des images à deux niveaux. Elle définit plutôt la façon d'effectuer une reconstitution parfaite d'une phototrame que le codeur a choisi de coder. Si le codeur choisit de coder une matrice qui est différente de l'original, l'ensemble du processus devient un codage avec pertes. Les différentes méthodes de codage offrent différentes méthodes d'introduction rentable des pertes.

0.2.1 Codage des symboles

Le codage avec pertes des symboles offre un moyen naturel d'effectuer un codage avec pertes des régions alphanumériques. Le principe consiste à admettre de légères différences entre la matrice de symbole originale et celle qui est indexée dans le dictionnaire de symboles. Le gain de compression est obtenu par le fait qu'il n'est pas nécessaire de coder un gros dictionnaire puis d'avoir, en conséquence du plus petit dictionnaire, un codage économique des indices de symboles. Il appartient au codeur de décider dans quelle mesure deux phototrames sont pratiquement identiques ou pratiquement différentes. Cette technique a été décrite en premier dans la référence [1].

Le risque présenté par le codage des symboles avec pertes est d'obtenir des *erreurs de substitution* c'est-à-dire que le codeur remplace une matrice représentant un caractère donné par une matrice décrivant caractère différent, de sorte qu'un lecteur humain lira un caractère erroné. Le risque d'erreurs de substitution peut être diminué par l'utilisation de mesures associées des différences entre matrices ou par la vérification que les pixels critiques de la matrice indexée sont corrects. A cette fin, une méthode décrite en [5] consiste à indexer le symbole éventuellement erroné puis à appliquer le codage de raffinement à cette matrice de symbole. Le principe est de coder la forme de base du caractère de manière économique puis à corriger les pixels que le codeur estime altérer le sens du caractère.

Le processus d'introduction de pertes rentables dans des régions alphanumériques peut aussi prendre des formes plus simples comme l'élimination des "pattes de mouche" dans les documents ou le lissage des bordures des lettres. Le plus vraisemblablement, de telles modifications diminueront la longueur du code de la région sans dégrader l'apparence générale de celle-ci, avec même la possibilité d'une amélioration.

On trouvera dans [7] un certain nombre d'exemples d'exécution de ce type de codage de symboles avec pertes conformément à JBIG2.

NOTE – Bien que le texte "région alphanumérique" soit utilisé pour les régions de la page qui sont codées au moyen du codage de symboles, d'autres utilisations possibles de ce codage sont le codage du trait et d'autres données non alphanumériques.

0.2.2 Codage générique

Pour effectuer un codage presque sans pertes au moyen du codage générique, le codeur applique un processus préliminaire à une image originale puis code sans pertes d'image ainsi modifiée. Les difficultés consistent à s'assurer que les modifications produisent un code de longueur inférieure et que la qualité de l'image modifiée ne souffre pas trop des modifications apportées. Deux processus préliminaires possibles sont indiqués en [11]. Ils permutent des pixels qui, une fois permutés, diminuent notablement la longueur totale du code de la région sans que cette permutation dégrade sérieusement la qualité visuelle. Ces processus préliminaires assurent un codage presque sans pertes de dégradés périodiques, cela au prix d'un gain modeste en terme de compression des autres types de données. Les processus préliminaires ne sont bien adaptés ni aux images à diffusion d'erreur ni aux images estompées avec du bruit bleu car une compression perçue comme sans pertes ne sera pas obtenue à un taux nettement inférieur au taux sans pertes.

0.2.3 Codage des dégradés

Le codage des dégradés est la façon naturelle d'obtenir une compression très élevée pour des dégradés *périodiques*, comme des images estompées par grappes de points ordonnées. Contrairement au codage générique avec pertes qui a été décrit plus haut, le codage des dégradés ne vise pas à préserver la matrice originale, bien que cela soit possible dans certains cas. Pour obtenir une compression supplémentaire, il est possible d'introduire des pertes en n'insérant pas toutes les structures de l'image originale dans le dictionnaire, ce qui réduit à la fois le nombre de structures dégradées et le nombre d'éléments binaires nécessaires pour spécifier la structure qui est utilisée à un emplacement donné.

Dans le cas d'un codage avec pertes d'images à diffusion d'erreur et d'images estompées par bruit bleu, il est conseillé de faire appel au codage des dégradés avec une grille de petites dimensions. Une image reconstruite perdra les détails fins et pourra afficher du broutage tout en restant clairement reconnaissable. On peut réduire le broutage du côté décodeur par un processus postérieur, par exemple en utilisant d'autres structures de reconstruction que celles qui apparaissent dans le dictionnaire. Les images ayant subi une diffusion d'erreur peuvent également être codées sans pertes ou avec une perte contrôlée comme indiqué ci-dessus, au moyen du codage générique.

On pourra trouver en [12] de plus amples détails sur l'exécution de ce codage des dégradés.

0.2.4 Conséquences d'une segmentation inadéquate

Pour obtenir un codage optimal, aussi bien en termes de qualité que de volume de fichier, la forme correcte du codage doit être utilisée pour les régions appropriées des pages du document. Le présent sou-paragraphe décrit brièvement les conséquences d'erreurs lors de cette segmentation.

Le fait d'utiliser un codage avec pertes des symboles pour un document contenant à la fois des données alphanumériques et des données de dégradé se traduira par un faible taux de compression. Selon le codeur, la qualité des données de dégradé pourra être bonne ou mauvaise. Si l'on utilise la forme de codage avec pertes des symboles décrite en [5], la qualité visuelle ne sera probablement pas affectée.

L'utilisation du codage générique avec pertes (au moyen des processus préliminaires indiqués en [11]) pour un document contenant à la fois des données de symboles et des données de dégradés permettra habituellement d'obtenir une bonne qualité et un taux de compression modéré.

On pourra coder efficacement le trait et les régions de texte manuscrit au moyen du codage générique mais, selon le codeur, ces types de région pourront également être codés très efficacement par la procédure de codage des symboles.

NORME INTERNATIONALE

RECOMMANDATION UIT-T

TECHNOLOGIES DE L'INFORMATION – CODAGE AVEC OU SANS PERTE DES IMAGES AU TRAIT

1 Domaine d'application

La présente Recommandation | Norme internationale définit des méthodes de codage des images et des ensembles (documents comportant plusieurs pages) d'images à deux niveaux. Elle est particulièrement applicable aux images à deux niveaux contenant des données alphanumériques et estompées (en dégradé).

Les méthodes définies permettent un codage sans pertes (conservant les éléments binaires), un codage avec pertes et un codage progressif. En codage progressif, il y a des pertes sur la première image; les images suivantes peuvent être avec ou sans pertes.

La présente Recommandation | Norme internationale définit également les formats de fichier permettant de sauvegarder les données codées des images à deux niveaux.

2 Références normatives

Les Recommandations et Normes internationales 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. Toutes Recommandations et Normes sont sujettes à révision et les parties prenantes aux accords fondés sur la présente Recommandation | Norme internationale sont invitées 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 possèdent le registre des Normes internationales en vigueur. Le Bureau de la normalisation des télécommunications de l'UIT tient à jour une liste des Recommandations de l'UIT en vigueur.

– Recomandation CCITT T.6 (1988), *Schémas de codage et fonctions de commande de codage de la télécopie pour les télécopieurs du groupe 4*.

– ISO/CEI 8859-1:1988, *Technologies de l'information – Jeux de caractères graphiques codés sur un seul octet – Partie 1: Alphabet latin N°1*.

– ISO/CEI 10646-1:2000, *Technologies de l'information – Jeu universel de caractères codés à plusieurs octets (JUC) – Partie 1: Architecture et plan multilingue de base*.

3 Termes et définitions

Pour les besoins de la présente Recommandation | Norme internationale les termes suivants s'appliquent:

3.1 **pixels de gabarit adaptatif**: pixels particuliers d'un gabarit, dont l'emplacement n'est pas fixe.

3.2 **agrégation**: association ou fusion de plusieurs symboles individuels pour former un nouveau symbole.

3.3 **image au trait; image à deux niveaux**: matrice rectangulaire de bits.

3.4 **bit**: chiffre binaire de valeur **`0`** ou **`1`**.

3.5 **phototrame**: image à deux niveaux.

3.6 **tampon**: zone de stockage utilisée pour mémoriser une phototrame.

3.7 **octet**: huit bits de données.

3.8 **opérateur combinatoire**: opérateur utilisé pour combiner le contenu préalable d'une phototrame avec de nouvelles valeurs introduites dans cette phototrame.

3.9 **système de coordonnées**: système de numérotage à deux dimensions d'emplacements, ceux-ci étant désignés par deux nombres, le premier croissant de gauche à droite et le second croissant de haut en bas.

3.10 différence delta S: différence entre les coordonnées S de deux instances successives de symboles dans une bande non vide.

3.11 différence delta T: différence entre les coordonnées T de deux bandes non vides successives.

3.12 procédure de décodage: composante d'un décodeur qui décode un certain type de données.

3.12.1 procédure de décodage entière: procédure de décodage produisant en sortie, à chaque invocation, une valeur unique.

3.12.2 procédure de décodage arithmétique entière: procédure de décodage entière qui utilise le décodage entropique arithmétique.

3.12.3 procédure de décodage zonale: procédure de décodage dont la sortie est une phototrame.

3.12.4 procédure de décodage régionale générique: procédure de décodage zonale qui consiste à décoder les pixels individuellement ou par séquences.

3.12.5 procédure de décodage régionale générique par raffinement: procédure de décodage zonale qui consiste à modifier une phototrame de référence afin d'un générer une autre.

3.12.6 procédure de décodage d'échelle de gris: procédure de décodage dont la sortie est une image en échelle de gris.

3.12.7 procédure de décodage à dictionnaire de structures: procédure de décodage dont la sortie est une liste de structures.

3.12.8 procédure de décodage zonale de dégradé: procédure de décodage zonale qui consiste à insérer un ensemble de structures dans une phototrame, chaque structure étant placée selon une grille de dégradé.

3.12.9 procédure de décodage par table de Huffman: procédure de décodage qui produit en sortie une table de Huffmann.

3.12.10 procédure de décodage zonale alphanumérique: procédure de décodage zonale qui consiste à insérer un ensemble d'instances de symboles dans une phototrame.

3.12.11 procédure de décodage par dictionnaire de symboles: procédure de décodage dont la sortie est une liste de symboles.

3.13 décodeur: entité capable de décoder un flux binaire conforme à la présente Recommandation | Norme internationale.

3.14 dictionnaire: liste de phototrames.

3.14.1 dictionnaire de structures: liste de structures.

3.14.2 dictionnaire de symboles: liste de symboles.

3.15 fanion d'exportation: bit indiquant qu'un symbole est sur la liste d'exportation d'un dictionnaire de symboles.

3.16 liste d'exportation: liste des symboles d'un dictionnaire de symboles qui peut être utilisée par une référence à ce dictionnaire de symboles.

3.17 image en échelle de gris: matrice rectangulaire d'indices entiers non négatifs.

3.18 pixel d'échelle de gris: elément d'image en échelle de gris dont la valeur est un entier.

3.19 grille de dégradé: grille rectangulaire d'emplacements spécifiant où des structures doivent être placées.

3.20 classe de hauteur: ensemble de symboles contenus dans un dictionnaire de symboles dont les hauteurs sont toutes égales.

3.21 hauteur différentielle entre classes de hauteur: différence de hauteur entre deux classes de hauteur.

3.22 largeur différentielle d'une classe de hauteur: différence de largeur entre deux symboles d'une même classe de hauteur.

3.23 table de Huffman: ensemble de lignes de table spécifiant la façon dont des valeurs sont codées.

3.24 codage sans pertes: méthode de codage de données de façon que les données décodées soient identiques aux données originales.

3.25 codage avec pertes: méthode de codage de données de façon que les données décodées ne diffèrent des données originales que dans une mesure si possible non significative.

3.26 **numéro d'ordre**: valeur utilisée comme compteur.

3.27 **valeur hors bande**: valeur non numérique qui peut être produite à la place d'un entier.

3.28 **structure**: matrice produite par une procédure de décodage à dictionnaire de structures.

3.29 **pixel**: elément dont la valeur est **`0`** ou **`1`** dans une phototrame.

3.30 **longueur de préfixe**: longueur du préfixe de code de Huffman dans une ligne de table.

3.31 **longueur de plage**: nombre de bits de code additionnel dans une ligne de table.

3.32 **matrice de référence**: phototrame utilisée comme plan de référence au cours de la procédure de décodage régionale par raffinement.

3.33 **segment référencé**: autre segment requis pour décoder le segment en cours.

3.34 **région**: phototrame produite par une procédure de décodage de région.

3.35 **segment**: séquence composée d'un en-tête suivi de données.

3.36 **bande**: portion en pleine largeur ou en pleine hauteur du système de coordonnées d'une région alphanumérique.

3.36.1 **bande vide**: bande ne contenant aucun coin de référence d'instance de symbole.

3.36.2 **bande non vide**: bande contenant le coin de référence d'au moins une instance de symbole.

3.37 **dimension de bande**: nombre des pixels de la dimension non pleine d'une bande.

3.38 **symbole**: phototrame produite par une procédure de décodage par dictionnaire de symboles.

3.39 **identificateur de symbole**: entier utilisé comme numéro de symbole ou comme indice permettant d'extraire un symbole d'une table de symboles.

3.40 **instance de symbole**: symbole inséré, éventuellement avec raffinement, à un emplacement particulier d'une région alphanumérique.

3.41 **différence de hauteur de raffinement d'instance de symbole**: différence de hauteur entre la matrice de référence d'une instance de symbole et la phototrame produite par la procédure de décodage régionale générique par raffinement.

3.42 **différence de largeur de raffinement d'instance de symbole**: différence de largeur entre la matrice de référence d'une instance de symbole et la phototrame produite par la procédure de décodage régionale générique par raffinement.

3.43 **décalage X du raffinement d'une instance de symbole**: différence entre la coordonnée X du coin supérieur gauche d'une matrice de référence d'instance de symbole et la phototrame produite par la procédure de décodage régionale générique par raffinement.

3.44 **décalage Y du raffinement d'une instance de symbole**: différence entre la coordonnée Y du coin supérieur gauche d'une matrice de référence d'instance de symbole et la phototrame produite par la procédure de décodage régionale générique par raffinement.

3.45 **ligne de table**: spécification du codage d'une valeur isolée ou d'une étendue de valeurs sous la forme d'un préfixe de code de Huffman suivi d'un nombre fixe de bits de code additionnel.

3.46 **prédiction typique**: signalisation du fait qu'une rangée entière d'une région générique est identique à la rangée précédente.

3.47 **valeur**: entier ou indicateur hors bande qui est décodé.

4 Symboles et abréviations

NOTE – Compte tenu des règles de nomenclature ISO, le terme *symbole* sera utilisé localement pour désigner un nom de variable dans le contexte de l'article 4.

4.1 Abréviations

Pour les besoins de la présente Recommandation | Norme internationale, les abréviations suivantes sont utilisées.

AT Gabarit adaptatif (*adaptive template*)

EOFB Fin de bloc de télécopie (*end-of-facsimile-block*)

ID Identificateur

LPS Symbole moins probable, c'est-à-dire valeur binaire moins probable (*less probable symbol, i.e. less probable binary value*)

LSB Bit de plus faible poids (*least significant bit)*

MMR Modification du code READ modifié (*modified modified read*)

MPS Symbole plus probable, c'est-à-dire valeur binaire plus probable (*more probable symbol, i.e. more probable binary value*)

MSB Bit de plus fort poids (*most significant bit*)

OOB Hors bande (*out-of-band*)

READ Désignation d'adresse d'élément relatif (*relative element address designate)*

TPGD Prédiction typique pour le codage générique direct d'une phototrame (*typical prediction for generic direct bitmap coding*)

TPGR Prédiction typique pour le codage de raffinement générique d'une phototrame (*typical prediction for generic refinement bitmap coding)*

NOTE – Dans les abréviations LPS et MPS, le terme *symbole* ne se rapporte pas aux symboles (phototrames) décrits dans la présente Recommandation | Norme internationale. Les abréviations LPS et MPS sont malgré tout utilisées parce qu'elles correspondent à la terminologie généralement acceptée en codage arithmétique.

4.2 Définition des symboles

Les symboles utilisés dans la présente Recommandation | Norme internationale sont énumérés ci-dessous. Par convention, les paramètres relatifs à toutes les procédures de décodage utilisées dans la présente Recommandation | Norme internationale sont présentés en **caractères gras**.

A	intervalle de probabilité
a	nombre réel
ARR	Table
A_1, A_2, A_3, A_4	pixels de gabarit adaptatif dans la procédure de décodage régionale générique
B	octet actuel des données codées arithmétiquement
B1	octet de données codées arithmétiquement qui suit l'octet actuel
B_{HC}	matrice collective de classes de hauteur dans une procédure de décodage par dictionnaire de symboles
B_{HDC}	matrice collective de dictionnaires dans une procédure de décodage de dictionnaire de structures
B_P	matrice de structure dans une procédure de décodage de dictionnaire de structures
B_S	matrice de symbole dans une procédure de décodage par dictionnaire de symboles
BM	phototrame
BP	pointeur vers un octet B
BPST	valeur initiale du pointeur BP
C	valeur du flux binaire dans le registre de code
Chigh	16 bits de poids fort de la valeur C
Clow	16 bits de poids faible de la valeur C
CONTEXT	valeurs des pixels dans un gabarit utilisé dans la procédure de décodage générique ou générique par raffinement
CT	compteur à décalage des renormalisations
CURCODE	code de Huffman pour la ligne actuelle d'une table de Huffman

CUREXFLAG	fanion d'exportation actuel
CURLEN	longueur du préfixe de ligne actuelle d'une table de Huffman
CURRANGELOW	limite inférieure de l'étendue de la ligne actuelle d'une table de Huffman
CURS	coordonnée S actuelle dans une procédure de décodage de région alphanumérique
CURT	coordonnée T actuelle d'une instance de symbole par rapport à la coordonnée T actuelle d'une bande dans une procédure de décodage de région alphanumérique
CX	étiquette désignant un contexte de codage arithmétique
D	décision de codage arithmétique
DFS	différence de coordonnée S entre les premières instances de deux bandes
DT	nombre de bandes vides entre deux bandes non vides
DW	différence de largeur entre deux matrices de symbole dans une procédure de décodage par dictionnaire de symboles
EXFLAGS	table de fanions d'exportation
EXINDEX	indice pointant sur la table EXFLAGS
EXRUNLENGTH	longueur d'une séquence de valeurs identiques de fanion d'exportation
FIRSTS	première coordonnée S de la bande actuelle
FIRSTCODE	premier code attribué à une longueur de préfixe particulière dans une table de Huffman
$GBATX_1$	emplacement X du pixel 1 de gabarit adaptatif dans une procédure de décodage régionale générique
$GBATY_1$	emplacement Y du pixel 1 de gabarit adaptatif dans une procédure de décodage régionale générique
$GBATX_2$	emplacement X du pixel 2 de gabarit adaptatif dans une procédure de décodage régionale générique
$GBATY_2$	emplacement Y du pixel 2 de gabarit adaptatif dans une procédure de décodage régionale générique
$GBATX_3$	emplacement X du pixel 3 de gabarit adaptatif dans une procédure de décodage régionale générique
$GBATY_3$	emplacement Y du pixel 3 de gabarit adaptatif dans une procédure de décodage régionale générique
$GBATX_4$	emplacement X du pixel 4 de gabarit adaptatif dans une procédure de décodage régionale générique
$GBATY_4$	emplacement Y du pixel 4 de gabarit adaptatif dans une procédure de décodage régionale générique
GB	préfixe utilisé pour un grand nombre des variables associées à une procédure de décodage régionale (phototrame) générique
GBH	hauteur d'une région générique
GBREG	région produite par une procédure de décodage régionale générique
GBTEMPLATE	paramètre indiquant le nombre et la disposition des pixels d'un gabarit utilisé dans une procédure de décodage régionale générique
GBW	largeur d'une région générique
GI	table de valeurs d'échelle de gris
GR	préfixe utilisé pour un grand nombre des variables associées à une procédure de décodage régionale générique par raffinement
$GRATX_1$	emplacement X du pixel 1 de gabarit adaptatif dans une procédure de décodage régionale générique par raffinement
$GRATY_1$	emplacement Y du pixel 1 de gabarit adaptatif dans une procédure de décodage régionale générique par raffinement
$GRATX_2$	emplacement X du pixel 2 de gabarit adaptatif dans une procédure de décodage régionale générique par raffinement
$GRATY_2$	emplacement Y du pixel 2 de gabarit adaptatif dans une procédure de décodage régionale générique par raffinement

GRAY	valeur actuelle d'échelle de gris
GRAYMAX	valeur maximale d'échelle de gris pour laquelle une structure est indiquée dans une procédure de décodage de dictionnaire de structures
GRH	hauteur d'une région générique en cours de codage avec codage de raffinement
GRREFERENCE	matrice de référence dans une procédure de décodage régionale générique par raffinement
GRREFERENCEDX	décalage X de la matrice de référence par rapport à la phototrame en cours de décodage dans une procédure de décodage régionale générique par raffinement
GRREFERENCEDY	décalage Y de la matrice de référence par rapport à la phototrame en cours de décodage dans une procédure de décodage régionale générique par raffinement
GRREG	région produite par une procédure de décodage régionale générique par raffinement
GRTEMPLATE	paramètre indiquant le nombre et la disposition des pixels dans un gabarit utilisé pour décoder une région générique par codage de raffinement
GRW	largeur d'une région générique en cours de codage par codage de raffinement
GS	préfixe utilisé pour un grand nombre des variables associées à une procédure de décodage d'image en échelle de gris
GSBPP	nombre de bits par valeur d'échelle de gris dans une procédure de décodage d'image en échelle de gris
GSH	hauteur de l'image en échelle de gris dans une procédure de décodage d'image en échelle de gris
GSKIP	masque indiquant les valeurs d'échelle de gris à omettre
GSMMR	indication de l'utilisation du codage MMR dans une procédure de décodage d'image en échelle de gris
GSTEMPLATE	paramètre indiquant le nombre et la disposition des pixels dans un gabarit utilisé dans une procédure de décodage d'image en échelle de gris
GSUSESKIP	indication que certaines valeurs d'échelle de gris doivent être omises dans une procédure de décodage d'image en échelle de gris
GSVALS	image en échelle de gris décodée
GSW	largeur de l'image en échelle de gris dans une procédure de décodage d'image en échelle de gris
HB	préfixe utilisé pour un grand nombre des variables associées à une procédure de décodage de région (matrice) de dégradé
HBH	hauteur d'une région de dégradé
HBPP	nombre de bits par valeur dans une table de valeurs d'échelle de gris
HBW	largeur d'une région de dégradé
HCHEIGHT	hauteur de la classe de hauteur actuelle dans une procédure de décodage par dictionnaire de symboles
HCDH	différence de hauteur entre deux classes de hauteur dans une procédure de décodage par dictionnaire de symboles
HCFIRSTSYM	indice du premier symbole décodé dans une classe de hauteur
HCOMBOP	opérateur combinatoire utilisé dans une procédure de décodage zonale de dégradé
HD	préfixe utilisé pour un grand nombre des variables associées à une procédure de décodage à dictionnaire de structures
HDEFPIXEL	valeur par défaut d'un pixel dans une région de dégradé
HDMMR	indication de l'utilisation du code MMR dans une procédure de décodage d'un dictionnaire de structures
HDPATS	table de structures produite par une procédure de décodage d'un dictionnaire de symboles.
HDPH	hauteur des structures dans un dictionnaire de structures
HDPW	largeur des structures dans un dictionnaire de structures

HDTEMPLATE	identificateur de gabarit utilisé pour décoder des structures dans une procédure de décodage à dictionnaire de structures
HENABLESKIP	indication de l'omission de valeurs d'échelle de gris non requises dans une procédure de décodage zonale de dégradé
HGH	hauteur de l'image en échelle de gris dans une procédure de décodage zonale de dégradé
HGW	largeur de l'image en échelle de gris dans une procédure de décodage zonale de dégradé
HGX	décalage horizontal de la grille dans une procédure de décodage zonale de dégradé
HGY	décalage vertical de la grille dans une procédure de décodage zonale de dégradé
H_I	hauteur d'une matrice d'instance de symbole
HIGHPREFLEN	longueur du préfixe de la ligne de rangée supérieure dans une table de Huffman
HMMR	indication de l'utilisation du codage MMR dans une procédure de décodage zonale de dégradé
HNUMPATS	nombre de structures pouvant être utilisées dans une procédure de décodage zonale de dégradé
HO_I	hauteur de la phototrame originale d'une instance de symbole contenant des informations de raffinement
HPATS	matrice de structures utilisée dans une région de dégradé
HPH	hauteur de chaque structure dans une région de dégradé
HPW	largeur de chaque structure dans une région de dégradé
HRX	coordonnée horizontale d'un vecteur de grille de dégradé
HRY	coordonnée verticale d'un vecteur de grille de dégradé
HSKIP	masque indiquant des valeurs d'échelle de gris à omettre
HT	préfixe utilisé pour un grand nombre des variables associées à une procédure de décodage par table de Huffman
HTEMPLATE	paramètre indiquant le nombre et la disposition des pixels d'un gabarit utilisés dans une procédure de décodage zonale de dégradé
HTHIGH	1 + la valeur maximale qui est représentée par une ligne normale quelconque d'une table de Huffman
HTLOW	valeur minimale qui est représentée par une ligne normale quelconque d'une table de Huffman
HTOFFSET	décalage de plage d'une ligne de table lors du décodage au moyen d'une table de Huffman
HTOOB	indication de la possibilité qu'une table de Huffman produise la valeur OOB (hors bande)
HTPS	longueur du champ de préfixe codé dans une ligne de table de Huffman
HTREG	région produite par une procédure de décodage zonale de dégradé
HTRS	longueur du champ de plage codée dans une ligne de table de Huffman
HTVAL	valeur décodée au moyen d'une table de Huffman
I	table, indexée par CX, des indices des estimateurs adaptatifs de probabilité
I	indice de table
IAAI	procédure de décodage arithmétique d'entier utilisée pour décoder le nombre d'instances de symbole contenues dans une agrégation
IADH	procédure de décodage arithmétique d'entier utilisée pour décoder la différence de hauteur entre deux classes de hauteur
IADS	procédure de décodage arithmétique d'entier utilisée pour décoder la coordonnée S de la deuxième instance de symbole d'une bande et les instances suivantes
IADT	procédure de décodage arithmétique d'entier utilisée pour décoder la coordonnée T de la deuxième instance de symbole d'une bande et les instances suivantes

IADW	procédure de décodage arithmétique d'entier utilisée pour décoder la différence de largeur entre deux symboles d'une classe de hauteur
IAEX	procédure de décodage arithmétique d'entier utilisée pour décoder des fanions d'exportation
IAFS	procédure de décodage arithmétique d'entier utilisée pour décoder la coordonnée S de la première instance de symbole dans une bande
IAID	procédure de décodage arithmétique d'entier utilisée pour décoder les identificateurs de symboles d'instances de symboles.
IARDH	procédure de décodage arithmétique d'entier utilisée pour décoder la différence de hauteur de raffinements d'instance de symbole
IARDW	procédure de décodage arithmétique d'entier utilisée pour décoder la différence de largeur de raffinements d'instance de symbole
IARDX	procédure de décodage arithmétique d'entier utilisée pour décoder les valeurs de décalage X de raffinements d'instance de symbole
IARDY	procédure de décodage arithmétique d'entier utilisée pour décoder les valeurs de décalage Y de raffinements d'instance de symbole
IARI	procédure de décodage arithmétique d'entier utilisée pour décoder le bit R_I d'instances de symbole
IAIT	procédure de décodage arithmétique d'entier utilisée pour décoder la coordonnée T des instances de symbole dans une bande
IB_I	phototrame d'une instance de symbole
IBO_I	phototrame originale d'une instance de symbole contenant des informations de raffinement
ID_I	identificateur de symbole d'une instance de symbole
IDS	différence delta S pour une instance de symbole dans une procédure de décodage zonale alphanumérique
J	indice de table
K	numéro d'ordre d'un segment référencé
LENCOUNT	histogramme des longueurs de préfixe dans une table de Huffman
LENMAX	longueur maximale de préfixe dans une table de Huffman
LOGSBSTRIPS	logarithme de base 2 de la dimension de bande utilisée pour coder une région alphanumérique
LOWPREFLEN	longueur du préfixe de la ligne de rangée inférieure dans une table de Huffman
LTP	indication que la ligne actuelle est codée explicitement dans une procédure de décodage régionale générique ou dans une procédure de décodage régionale générique par raffinement
m_g	indice horizontal pour la valeur d'échelle de gris actuelle
MMR	indication que le codage MMR est utilisé dans une procédure de décodage régionale générique
MPS	table, indexée par CX, des valeurs binaires actuelles les plus probables
NINSTANCES	compteur d'instances de symbole
n_g	indice vertical de la valeur d'échelle de gris actuelle
NLPS	indice suivant pour une renormalisation par valeur LPS
NMPS	indice suivant pour une renormalisation par valeur MPS
NSYMSDECODED	nombre de symboles déjà décodés dans une procédure de décodage par dictionnaire de symboles
NTEMP	nombre de lignes d'une table de Huffman
OOB	valeur hors bande
P	page à laquelle un segment est associé
PREFLEN	table de longueurs de préfixe représentant les lignes d'une table de Huffman
Qe	estimateur de la probabilité d'une valeur LPS

r	fanion de rétention d'un segment
R	nombre de segments référencés par un autre segment
RANGELEN	table des longueurs de plage des lignes d'une table de Huffman
RANGELOW	table des limites inférieures de plage des lignes d'une table de Huffman
RA_1, RA_2	pixels de gabarit adaptatif dans la procédure de décodage régionale générique par raffinement
RDH_I	différence de hauteur d'une matrice de raffinement d'instance de symbole
RDW_I	différence de largeur d'une matrice de raffinement d'instance de symbole
RDX_I	décalage X d'un raffinement d'instance de symbole
RDY_I	décalage Y d'un raffinement d'instance de symbole
REFAGGNINST	nombre d'instances de symbole dans une agrégation
R_I	bit indiquant si des informations de raffinement sont présentes pour une instance de symbole
REFCORNER	coin d'une matrice d'instance de symbole qui doit être utilisé comme référence dans une procédure de décodage zonale alphanumérique
S	coordonnée du système de coordonnées utilisé dans une procédure de décodage zonale alphanumérique
S_I	coordonnée S d'une instance de symbole
SB	préfixe utilisé pour un grand nombre des variables associées à une procédure de décodage de région (matrice) de symbole
SBDSOFFSET	décalage des valeurs delta S codées dans une région alphanumérique
SBCOMBOP	opérateur combinatoire utilisé dans une procédure de décodage zonale alphanumérique
SBDEFPIXEL	valeur par défaut des pixels dans une région alphanumérique
SBH	hauteur d'une région alphanumérique
SBHUFF	indication d'utilisation du codage de Huffman dans une procédure de décodage zonale alphanumérique
SBHUFFDS	table de Huffman utilisée pour décoder la coordonnée S d'instances de symbole subséquentes dans une bande
SBHUFFDT	table de Huffman utilisée pour décoder les différences de coordonnée T entre bandes non vides
SBHUFFFS	table de Huffman utilisée pour décoder la coordonnée S de la première instance de symbole d'une bande
SBHUFFRDH	table de Huffman utilisée pour décoder la différence entre une hauteur de symbole et la hauteur d'une matrice d'instance de symbole à codage de raffinement
SBHUFFRDW	table de Huffman utilisée pour décoder la différence entre une largeur de symbole et la largeur d'une matrice d'instance de symbole à codage de raffinement
SBHUFFRDX	table de Huffman utilisée pour décoder la différence entre une coordonnée X d'instance de symbole et la coordonnée X d'une matrice à codage de raffinement
SBHUFFRDY	table de Huffman utilisée pour décoder la différence entre une coordonnée Y d'instance de symbole et la coordonnée Y d'une matrice à codage de raffinement
SBHUFFRSIZE	table de Huffman utilisée pour décoder la dimension d'une donnée de matrice de raffinement d'une instance de symbole
SBNUMINSTANCES	nombre d'instances de symbole dans une région alphanumérique
SBNUMSYMS	nombre de symboles pouvant être utilisés dans une région alphanumérique
$SBRATX_1$	emplacement X du pixel de gabarit adaptatif RA_1 dans une procédure de décodage zonale alphanumérique
$SBRATY_1$	emplacement Y du pixel de gabarit adaptatif RA_1 dans une procédure de décodage zonale alphanumérique

$SBRATX_2$	emplacement X du pixel de gabarit adaptatif RA_2 dans une procédure de décodage zonale alphanumérique
$SBRATY_2$	emplacement Y du pixel de gabarit adaptatif RA_2 dans une procédure de décodage zonale alphanumérique
SBREFINE	indication de l'utilisation du codage de raffinement dans une procédure de décodage zonale alphanumérique
SBREG	région produite par une procédure de décodage zonale alphanumérique
SBRTEMPLATE	identificateur de gabarit pour le codage de raffinement d'une phototrame dans une procédure de décodage zonale alphanumérique
SBSTRIPS	hauteur des bandes d'instances de symbole
SBSYMCODELEN	longueur des codes de symbole utilisés dans la procédure IAID
SBSYMCODES	table de codes de longueur variable désignant des symboles individuels
SBSYMS	table de symboles utilisés dans une région alphanumérique
SBW	largeur d'une région alphanumérique
SD	préfixe utilisé pour un grand nombre des variables associées à une procédure de décodage par dictionnaire de symboles
$SDATX_1$	emplacement X du pixel de gabarit adaptatif A_1 dans une procédure de décodage par dictionnaire de symboles
$SDATY_1$	emplacement Y du pixel de gabarit adaptatif A_1 dans une procédure de décodage par dictionnaire de symboles
$SDATX_2$	emplacement X du pixel de gabarit adaptatif A_2 dans une procédure de décodage par dictionnaire de symboles
$SDATY_2$	emplacement Y du pixel de gabarit adaptatif A_2 dans une procédure de décodage par dictionnaire de symboles
$SDATX_3$	emplacement X du pixel de gabarit adaptatif A_3 dans une procédure de décodage par dictionnaire de symboles
$SDATY_3$	emplacement Y du pixel de gabarit adaptatif A_3 dans une procédure de décodage par dictionnaire de symboles
$SDATX_4$	emplacement X du pixel de gabarit adaptatif A_4 dans une procédure de décodage par dictionnaire de symboles
$SDATY_4$	emplacement Y du pixel de gabarit adaptatif A_4 dans une procédure de décodage par dictionnaire de symboles
SDEXSYMS	symboles exportés d'un dictionnaire de symboles
SDHUFF	indication d'utilisation du codage de Huffman dans une procédure de décodage par dictionnaire de symboles
SDHUFFAGGINST	table de Huffman utilisée pour décoder le nombre d'instances de symbole contenues dans une agrégation, dans une procédure de décodage par dictionnaire de symboles
SDHUFFDH	table de Huffman utilisée pour décoder la différence de hauteur entre deux classes de hauteur dans une procédure de décodage par dictionnaire de symboles
SDHUFFDW	table de Huffman utilisée pour décoder la différence de largeur entre deux symboles dans une procédure de décodage par dictionnaire de symboles
SDHUFFBMSIZE	table de Huffman utilisée pour décoder la dimension d'une matrice collective de classes de hauteur, dans une procédure de décodage par dictionnaire de symboles
SDINSYMS	table de symboles utilisée comme paramètre dans une procédure de décodage par dictionnaire de symboles
SDNEWSYMS	symboles décodés dans un dictionnaire de symboles
SDNEWSYMWIDTHS	largeurs des symboles décodés dans un dictionnaire de symboles
SDNUMEXSYMS	nombre de symboles exportés d'un dictionnaire de symboles
SDNUMINSYMS	nombre de symboles produits dans un dictionnaire de symboles

SDNUMNEWSYMS	nombre de symboles produits dans un dictionnaire de symboles
SDREFAGG	indication de l'utilisation du codage de raffinement et du codage agrégé dans une procédure de décodage par dictionnaire de symboles
$SDRATX_1$	emplacement X du pixel de gabarit adaptatif RA_1 dans une procédure de décodage par dictionnaire de symboles
$SDRATY_1$	emplacement Y du pixel de gabarit adaptatif RA_1 dans une procédure de décodage par dictionnaire de symboles
$SDRATX_2$	emplacement X du pixel de gabarit adaptatif RA_2 dans une procédure de décodage par dictionnaire de symboles
$SDRATY_2$	emplacement Y du pixel de gabarit adaptatif RA_2 dans une procédure de décodage par dictionnaire de symboles
SDRTEMPLATE	identificateur de gabarit pour le codage de raffinement de matrices dans une procédure de décodage par dictionnaire de symboles
SDTEMPLATE	identificateur de gabarit pour le décodage de matrices de symbole dans une procédure de décodage par dictionnaire de symboles
SKIP	masque de pixels à omettre lors du décodage d'une région générique
SLTP	valeur binaire indiquant si la ligne actuelle a fait l'objet d'une prédiction typique et non la ligne précédente, ou inversement
STRIPT	coordonnée T numériquement minimale dans la bande actuelle
SWITCH	indication que les valeurs MPS et LPS sont commutées sur une renormalisation LPS
SYMWIDTH	largeur de la phototrame actuelle dans une procédure de décodage par dictionnaire de symboles
T	coordonnée du système de coordonnées utilisé dans une procédure de décodage zonale alphanumérique
TEMPC	registre temporaire dans le codeur MQ
T_I	coordonnée T d'une instance de symbole
TOTWIDTH	largeur totale des matrices dans une classe de hauteurs
TPGDON	indication d'utilisation de la prédiction typique dans une procédure de décodage régionale générique
TPGRON	indication d'utilisation de la prédiction typique dans une procédure de décodage régionale générique par raffinement
TPGRPIX	indication que le pixel actuel doit être décodé implicitement au moyen d'une prédiction TPGR
TPGRVAL	valeur du pixel actuel prédit par TPGR
TRANSPOSED	indication de la transposition des coordonnées d'instance de symbole dans une procédure de décodage zonale alphanumérique
USESKIP	indication que certains pixels doivent être omis dans le décodage d'une région générique
V1	valeur binaire
V2	valeur binaire
W_I	largeur d'une matrice d'instance de symbole
WO_I	largeur de la phototrame originale d'une instance de symbole contenant des informations de raffinement
x	coordonnée horizontale d'un emplacement dans une grille de dégradé
X	coordonnée horizontale d'un pixel dans une phototrame
y	coordonnée verticale d'un emplacement dans une grille de dégradé
Y	coordonnée verticale d'un pixel dans une phototrame.

4.3 Définition des opérateurs

Les opérateurs suivants sont définis:

OR — Si V1 et V2 sont deux valeurs binaires, alors V1 OR V2 est égale à **0** si V1 et V2 sont toutes deux **0**. V1 OR V2 est égale à **1** si soit V1 soit V2 est égale à **1**. Si V1 et V2 sont deux valeurs d'entier, elles sont le résultat d'une application bit par bit de l'opérateur OR.

AND — Si V1 et V2 sont deux valeurs binaires, alors V1 AND V2 sont égales à **0** si soit V1 soit V2 est **0**. V1 AND V2 sont égales à **1** si les deux valeurs V1 et V2 sont **1**. Si V1 et V2 sont deux valeurs d'entier, elles sont le résultat d'une application bit par bit de l'opérateur AND.

XOR — Si V1 et V2 sont deux valeurs binaires, alors V1 XOR V2 est égale à **0** si V1 et V2 sont égales. V1 XOR V2 est égale à **1** si V1 et V2 sont différentes. Si V1 et V2 sont deux valeurs d'entier, elles sont le résultat d'une application bit par bit de l'opérateur XOR.

XNOR — Si V1 et V2 sont deux valeurs binaires, alors V1 XNOR V2 est égale à **0** si V1 et V2 sont différentes. V1 XNOR V2 est égale à **1** si V1 et V2 sont égales.

REPLACE — Si V1 et V2 sont deux valeurs binaires, alors V1 REPLACE V2 est égale à V2.

NOT — Si V1 est une valeur binaire, alors NOT V1 est égale à **1** si V1 est **0** et égale à **0** si V1 est **1**.

min — Si x et y sont des nombres, alors l'expression min(x, y) est la plus petite de x et de y.

max — Si x et y sont des nombres, alors l'expression max(x, y) est la plus grande de x et de y.

$\lfloor\ \rfloor$ — Si a est un nombre, alors $\lfloor a \rfloor$ est le plus grand entier inférieur ou égal à a.

$\lceil\ \rceil$ — Si a est un nombre, alors $\lceil a \rceil$ est le plus petit entier supérieur ou égal à a.

<< — Si V1 et V2 sont deux entiers, alors V1 << V2 est la valeur obtenue par décalage de la valeur de V1 vers la gauche de V2 bits, les V2 bits à droite de la nouvelle valeur étant remplacés par **0**.

>> — Si V1 et V2 sont deux entiers, alors V1 >> V2 est la valeur obtenue par décalage de la valeur de V1 vers la droite de V2 bits, les V2 bits à gauche de la nouvelle valeur étant remplacés par **0**.

$>>_A$ — Si V1 et V2 sont deux entiers, alors V1 $>>_A$ V2 est la valeur obtenue par décalage de la valeur de V1 vers la droite de V2 bits, les V2 bits à gauche de la nouvelle valeur étant remplacés par **0** si V1 n'est pas négative et par **1** si V1 est négative.

5 Conventions

5.1 Conventions typographiques

Tous les noms de paramètre sont imprimés en **caractères gras**.

5.2 Notation binaire

Les deux valeurs binaires sont notées par **0** et **1**.

5.3 Notation hexadécimale

Le préfixe `0x` indique que la valeur qui suit doit être interprétée comme un nombre hexadécimal (base 16).

EXEMPLE – La valeur `0x6A` est égale à la valeur décimale 106.

5.4 Syntaxe des valeurs d'entier

5.4.1 Condensation du flux binaire

Les bits sont condensés en octets à partir du bit de poids fort. Si un décodeur doit lire une séquence de bits dans un flux binaire, il doit d'abord lire le bit de poids fort du premier octet, puis le bit de poids immédiatement inférieur et ainsi de suite jusqu'à l'octet suivant.

EXEMPLE – La séquence d'octets `0x2F 0x05 0xC1`, interprétée comme une séquence de bits, est la séquence

0 0 1 0 1 1 1 1 0 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1

5.4.2 Valeurs en octets multiples

Toutes les valeurs en octets multiples doivent être interprétées dans le mode du poids fort en premier: le premier octet de chaque valeur a le poids fort et le dernier octet le poids faible.

EXEMPLE – La séquence d'octets `0x01 0x5C 0x99 0xFA`, interprétée comme une valeur de quatre octets, représente la valeur `0x015C99FA`.

5.4.3 Numérotage des éléments binaires

Le bit de poids faible d'une valeur quelconque porte le numéro 0. Pour une valeur d'un octet, le bit de poids fort porte le numéro 7; pour une valeur de deux octets, le bit de poids fort porte le numéro 15; pour une valeur de quatre octets, le bit de poids fort porte le numéro 31.

5.4.4 Signe des valeurs

Sauf spécification contraire, toutes les valeurs à bits multiples doivent être traitées comme étant non signées. Lorsqu'une valeur doit être traitée comme un nombre signé, elle doit être interprétée sous la forme d'un complément à deux.

5.5 Notation et conventions relatives aux tables

Les tables sont numérotées à partir de zéro.

EXEMPLE – Une table ARR à deux dimensions, contenant douze éléments, possède les éléments

ARR[0], ARR[1], …, ARR[11]

5.6 Conventions relatives aux images et aux phototrames

Une phototrame est une table rectangulaire dont chaque élément a la valeur **0** ou **1**. Un élément d'une phototrame est appelé *pixel*.

NOTE 1 – Dans tout le texte de la présente Recommandation | Norme internationale, les pixels des phototrames sont traités comme ayant la valeur **0** ou **1**. La plupart des applications de la présente Recommandation | Norme internationale sélectionneront une certaine interprétation de ces deux valeurs. Une interprétation typique de ces pixels est que **0** représente le blanc, ou le fond, et que **1** représente le noir, ou l'avant-plan. Cela n'est toutefois pas une exigence de la présente Recommandation | Norme internationale et les applications peuvent faire d'autres interprétations de ces valeurs.

Les termes "gauche", "droit", "haut/supérieur", "bas/inférieur", "largeur" et "hauteur" sont souvent appliqués aux phototrames. Ces termes ne se rapportent à aucun aspect physique de la phototrame: si celle-ci est imprimée sur papier, elle peut l'être avec son bord "gauche" tangent à un bord quelconque du papier. Ces termes sont utilisés dans la présente Recommandation | Norme internationale pour désigner les quatre côtés d'une phototrame, comme indiqué dans la Figure 2.

Un pixel de phototrame est repéré par une paire de coordonnées X et Y, quelquefois écrite sous la forme de paire (X,Y). L'emplacement (0, 0) représente le pixel du coin supérieur gauche. La coordonnée X croît vers la droite et la coordonnée Y croît vers le bas.

Si *BM* est une phototrame, le pixel dont les coordonnées sont X et Y est repéré par l'expression *BM* [X,Y].

NOTE 2 – Ces conventions sont destinées à faciliter la description des opérations mettant en jeu des phototrames. Elles ne sont pas destinées à impliquer de quelconques caractéristiques physiques des images représentées par ces phototrames.

6 Procédures de décodage

6.1 Introduction aux procédures de décodage

La présente Recommandation | Norme internationale fait appel à un certain nombre de procédures de décodage pour différents types de données. Chacune de ces procédures de décodage produit en sortie une certaine sorte de données. La procédure de décodage régionale générique, la procédure de décodage régionale générique par raffinement, la procédure de décodage régionale de dégradé et la procédure de décodage régionale alphanumérique produisent toutes en sortie des régions. La procédure de décodage par dictionnaire de symboles produit en sortie une table de symboles. La procédure de décodage à dictionnaire de structures produit de même en sortie une table de phototrames représentant des cellules de dégradé.

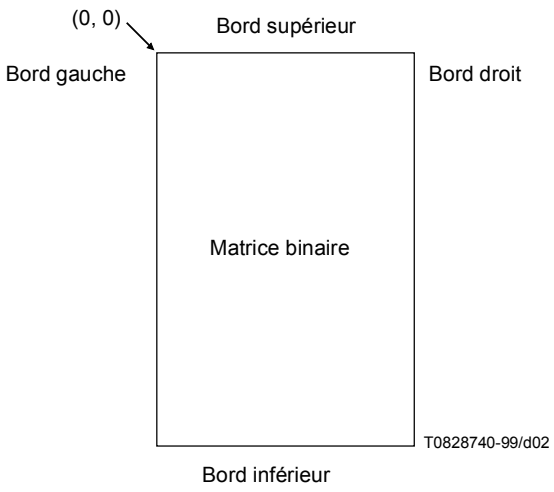


Figure 2 – Les quatre côtés d'une phototrame

Les diverses procédures de décodage régionale opèrent de différentes façons:

- la procédure de décodage régionale générique décode une phototrame en la traitant simplement comme une table de pixels binaires;
- la procédure de décodage régionale générique par raffinement décode une phototrame en la traitant comme une table de pixels binaires mais en codant chaque pixel par rapport à une matrice de référence;
- la procédure de décodage régionale alphanumérique décode une phototrame en y insérant une série de symboles et éventuellement en appliquant à chacun la procédure de décodage régionale générique par raffinement;
- la procédure de décodage régionale de dégradé décode une phototrame en y insérant une série de structures, aux emplacements spécifiés par une grille de dégradé.

Chaque procédure de décodage est spécifiée en termes de nombre de paramètres et de séquence d'opérations affectées par les valeurs de ces paramètres. Ceux-ci sont fournis à la procédure de décodage lors de chaque invocation, la même procédure de décodage pouvant être invoquée un grand nombre de fois au cours du décodage d'un flux binaire, avec différents paramètres à chaque fois.

Certains des paramètres de procédure de décodage ne sont pas utilisés dans certaines circonstances, habituellement en fonction des valeurs d'autres paramètres. Dans ces circonstances, aucune valeur n'a besoin d'être spécifiée pour ces paramètres inutilisés.

Dans le présent paragraphe, dans certains paragraphes suivants et dans les annexes normatives, des restrictions sont imposées au flux binaire soumis au décodage.

EXEMPLE 1 – Au 7.3, certains types de segment sont qualifiés comme suit: "Réservé; ne doit pas être utilisé".

EXEMPLE 2 – Au 7.4.2.1.1, si le champ **SDHUFF** est **0**, alors le champ **SDHUFFDH** doit contenir la valeur **0**.

Il y a lieu d'interpréter ces restrictions comme signifiant que le comportement d'un décodeur, confronté à un flux binaire qui ne respecte pas les restrictions, est indéfini et hors du domaine d'application de la présente Recommandation | Norme internationale.

NOTE – Cela signifie que si un décodeur rencontre un flux binaire qui ne répond pas aux restrictions, ce décodeur peut prendre une mesure quelconque: il peut abandonner et arrêter le décodage; il peut négliger l'erreur et tenter de continuer; il peut interpréter l'erreur et modifier son comportement (par exemple utiliser l'erreur pour essayer de faciliter la reprise sur d'autres erreurs); etc.

6.2 Procédure de décodage régionale générique

6.2.1 Description générale

Cette procédure de décodage sert à décoder une table rectangulaire de valeurs **0** ou **1** qui sont codées comme étant un pixel à chaque fois (c'est-à-dire que cette table est utilisée pour décoder une phototrame faisant appel au codage simple et générique). Cette procédure de décodage modifie également une table d'informations de probabilité pouvant être utilisées par d'autres invocations de cette procédure de décodage régionale générique.

La procédure de décodage régionale générique peut être fondée sur le codage séquentiel des pixels d'image utilisant le codage arithmétique spécifié dans l'Annexe E et un gabarit pour déterminer l'état du codage. Cette technique a été utilisée dans la Rec. UIT-T T.82 | ISO/CEI 11544 (JBIG). Ce type de décodage est décrit au 6.2.5.

En variante, pour améliorer la vitesse mais réduire le taux de compression, la procédure de décodage régionale générique peut être fondée sur le codage de Huffman de séries de pixels. Cette technique a été utilisée dans l'algorithme MMR (modification du code READ modifié) décrit dans la Recommandation UIT-T T.6 (G4). Ce type de décodage est décrit au 6.2.6.

6.2.2 Paramètres d'entrée

Les paramètres de cette procédure de décodage sont indiqués dans le Tableau 2.

Tableau 2 – Paramètres pour la procédure de décodage régionale générique

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
MMR	Entier	1	N	Indication d'utilisation du codage MMR
GBW	Entier	32	N	Largeur de la région
GBH	Entier	32	N	Hauteur de la région
GBTEMPLATE	Entier	2	N	Identificateur de gabarit[a)]
TPGDON	Entier	1	N	Indication d'utilisation de la prédiction typique[a)]
USESKIP	Entier	1	N	Indication que certains pixels doivent être omis dans le décodage[a)]
SKIP	Phototrame			Phototrame indiquant les pixels à omettre. **GBW**= largeur en pixels; **GBH** = hauteur en pixels[c)]
$\mathbf{GBATX_1}$	Entier	8	Y	Emplacement X du pixel A_1 de gabarit adaptatif[a)]
$\mathbf{GBATY_1}$	Entier	8	Y	Emplacement Y du pixel A_1 de gabarit adaptatif[a)]
$\mathbf{GBATX_2}$	Entier	8	Y	Emplacement X du pixel A_2 de gabarit adaptatif[b)]
$\mathbf{GBATY_2}$	Entier	8	Y	Emplacement Y du pixel A_2 de gabarit adaptatif[b)]
$\mathbf{GBATX_3}$	Entier	8	Y	Emplacement X du pixel A_3 de gabarit adaptatif[b)]
$\mathbf{GBATY_3}$	Entier	8	Y	Emplacement Y du pixel A_3 de gabarit adaptatif[b)]
$\mathbf{GBATX_4}$	Entier	8	Y	Emplacement X du pixel A_4 de gabarit adaptatif[b)]
$\mathbf{GBATY_4}$	Entier	8	Y	Emplacement Y du pixel A_4 de gabarit adaptatif[b)]

a) Paramètre inutilisé si **MMR** = **1**.
b) Paramètre inutilisé si **MMR** = **1** ou **GBTEMPLATE** ≠ **0**.
c) Paramètre inutilisé si **USESKIP** = **0** ou **MMR** = **1**.

6.2.3 Valeur de retour

La variable dont la valeur est le résultat de cette procédure de décodage est décrite dans le Tableau 3.

Tableau 3 – Valeur de retour issue de la procédure de décodage régionale générique

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
GBREG	Phototrame			Phototrame décodée de la région

6.2.4 Variables utilisées lors du décodage

Les variables utilisées dans cette procédure de décodage sont indiquées dans le Tableau 4.

Tableau 4 – Variables utilisées dans la procédure de décodage régionale générique

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
LTP	Entier	1	N	Indication que la ligne d'image actuelle est codée explicitement[a)]
SLTP	Entier	1	N	Indication que la valeur LTP de la ligne actuelle diffère de la valeur LTP de la ligne précédente[a)]
CONTEXT	Entier	16	N	Valeurs des pixels contenus dans le gabarit[a)]
a) Paramètre inutilisé si **MMR** = **1**.				

6.2.5 Décodage par gabarit et codage arithmétique

6.2.5.1 Description générale

Si le paramètre **MMR** a la valeur **0**, la procédure de décodage régionale générique est fondée sur le codage arithmétique avec un gabarit permettant de déterminer l'état du codage. Le reste du 6.2.5 décrit cette forme de décodage et ne s'applique que lorsque **MMR** = **0**.

6.2.5.2 Ordre de codage et conventions relatives aux bords

L'algorithme de codage effectue des passes dans la phototrame dans l'ordre de balayage d'un canevas matriciel c'est-à-dire par rangées de haut en bas et de gauche à droite dans chaque rangée. Le traitement d'un pixel cible actuel fera référence à certains pixels qui sont en relation spatiale fixe avec le pixel cible.

Près des bords de la matrice, ces références de voisinage peuvent ne pas se trouver dans la matrice proprement dite. La règle de résolution des références hors limites est la suivante:

- Tous les pixels se trouvant à l'extérieur de la matrice actuelle ont la valeur **0**.

6.2.5.3 Gabarits fixes

Un gabarit définit un voisinage autour d'un pixel à coder. Les valeurs des pixels se trouvant dans ce voisinage définissent un contexte. Chaque contexte possède son propre estimateur adaptatif de probabilité, qui est utilisé par le codeur arithmétique (voir Annexe E). Bien qu'un gabarit soit une structure géométrique de pixels, ceux-ci sont censés prendre des valeurs lorsque leur gabarit est aligné sur une partie particulière de l'image.

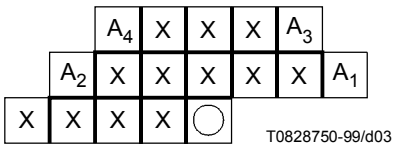


Figure 3 – Gabarit montrant les pixels AT à leurs coordonnées nominales lorsque GBTEMPLATE = 0

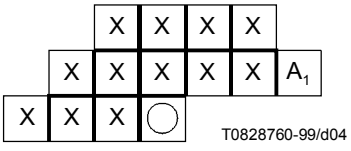


Figure 4 – Gabarit montrant les pixels AT à leurs coordonnées nominales lorsque GBTEMPLATE = 1

La Figure 3 montre le gabarit qui doit être utilisé lorsque **GBTEMPLATE** = 0. La Figure 4 montre le gabarit qui doit être utilisé lorsque **GBTEMPLATE** = 1. La Figure 5 montre le gabarit qui doit être utilisé lorsque **GBTEMPLATE** = 2. La Figure 6 montre le gabarit qui doit être utilisé lorsque **GBTEMPLATE** = 3. Dans chacune de ces figures, le pixel repéré par un cercle correspond au pixel à coder et ne fait pas partie du gabarit. Les pixels désignés par "X" correspondent aux pixels ordinaires du gabarit. Les pixels désignés par A_1-A_4 sont spéciaux dans le gabarit et sont désignés comme étant des pixels "adaptatifs" ou AT. Ces pixels ont la particularité que leur emplacement n'est pas fixe mais peut être modifié. On trouvera une description des pixels AT au 6.2.5.4. Les légendes A_1-A_4 indiquent les pixels 1 à 4. Les emplacements réels des pixels sont spécifiés sous la forme de paramètres pour cette procédure de décodage; les Figures 3 à 6 montrent l'emplacement nominal de ces pixels AT pour chaque gabarit.

Les valeurs des pixels contenus dans le gabarit doivent être combinées pour former un contexte. Chaque pixel du gabarit (y compris les pixels adaptatifs) doit correspondre à un bit spécifique du contexte, bien que les pixels du gabarit puissent être assignés à des bits du contexte dans un ordre quelconque. Comme il y a jusqu'à 16 pixels dans le gabarit, les contextes peuvent prendre jusqu'à 65536 valeurs différentes. Ce contexte doit être utilisé pour déterminer l'estimateur adaptatif de probabilité qui doit être utilisé par le codeur arithmétique pour coder le pixel à coder (voir Annexe E).

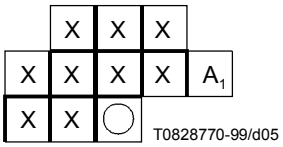


Figure 5 – Gabarit montrant les pixels AT à leurs coordonnées nominales lorsque GBTEMPLATE = 2

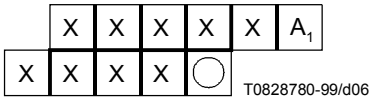


Figure 6 – Gabarit montrant les pixels AT à leurs coordonnées nominales lorsque GBTEMPLATE = 3

NOTE 1 – Une règle élémentaire consiste à utiliser de grands gabarits pour de grandes phototrames. Un dégradé périodique de pleines dimensions devra donc être codé avec le gabarit de 16 pixels tandis que de petites phototrames comme celles des symboles habituels devront être codées avec un des gabarits de 10 pixels. Dans certains cas, un gabarit intermédiaire est souhaité, pour des impératifs de performance ou de mémoire du décodeur: il convient alors d'utiliser un gabarit de 13 pixels. Il est également possible de produire d'autres gabarits en plaçant un ou plusieurs des pixels AT au-dessus d'un pixel de gabarit normal, pour en fixer la valeur.

NOTE 2 – Les gabarits de 10 pixels sont ceux qui sont utilisés dans la Rec. UIT-T T.82 | ISO/CEI 11544 (JBIG). La vitesse d'exécution logicielle est un peu plus grande avec le gabarit de deux lignes qu'avec l'un quelconque des gabarits de trois lignes. Pour la plupart des images, le gabarit de trois lignes donne un taux de compression plus élevé que le gabarit à deux lignes de 10 pixels.

6.2.5.4 Pixels de gabarit adaptatif

Lors du codage de l'image, les modifications du gabarit doivent être autorisées avec les restrictions décrites dans ce paragraphe.

Les pixels autorisés à changer sont appelés *pixels AT*. Leur emplacement nominal est indiqué dans les Figures 3, 4, 5 et 6 par "A_1", "A_2", "A_3" et "A_4". Noter que certains gabarits possèdent moins de quatre pixels AT. En général, un pixel AT peut être situé n'importe où dans le champ représenté sur la Figure 7, pixel actuel non compris. Il est donc possible d'utiliser une taille effective de gabarit de 15, 14, 13, 12 ou 9 pixels en faisant en sorte que l'emplacement déplacé du pixel AT chevauche un pixel normal du gabarit. Les emplacements réels des pixels AT pour toute invocation de cette procédure de décodage sont spécifiés sous la forme de paramètres pour la procédure de décodage. L'emplacement du pixel A_1 est donné par (**$GBATX_1$**, **$GBATY_1$**). Si **GBTEMPLATE** est 0, alors

- l'emplacement du pixel A_2 est donné par (**$GBATX_2$**, **$GBATY_2$**);
- l'emplacement du pixel A_3 est donné par (**$GBATX_3$**, **$GBATY_3$**);
- et l'emplacement du pixel A_4 est donné par (**$GBATX_4$**, **$GBATY_4$**).

NOTE 1 – Certains profils peuvent limiter les emplacements de pixel AT à une étendue plus étroite que celle qui est représentée sur la Figure 7.

NOTE 2 – Les indices des pixels AT sur la Figure 3 correspondent à la bonne qualité attendue. Si l'on ne déplace qu'un seul pixel AT à partir de son emplacement nominal, il est conseillé de déplacer le pixel A_4. Le pixel suivant à déplacer est A_3 et ainsi de suite.

NOTE 3 – Les emplacements nominaux des pixels AT sont indiqués dans le Tableau 5. Ces emplacements doivent être utilisés à moins que d'autres emplacements n'améliorent la performance en termes de compression. Certains profils peuvent limiter les emplacements de pixels AT à ces seuls emplacements nominaux.

NOTE 4 – Si un emplacement de pixel AT chevauche celui d'un pixel de gabarit normal, la valeur du pixel AT peut être négligée (car elle fait double emploi avec une autre valeur), ce qui permet de diminuer les besoins en mémoire du décodeur puisque toutes les valeurs CX ne pourront pas apparaître. Cependant, lorsque la prédiction TPGD est activée (**TPGDON** = **1**), le contexte utilisé pour coder la valeur SLTP est utilisée sans tenir compte du fait que des pixels AT chevauchent des pixels de gabarit normal. En d'autres termes, des contextes dans lesquels la valeur de pixel AT diffère de la valeur du pixel de gabarit normal peuvent encore apparaître, mais seulement pour le pseudo-pixel SlTP lorsque la prédiction TPGD est activée.

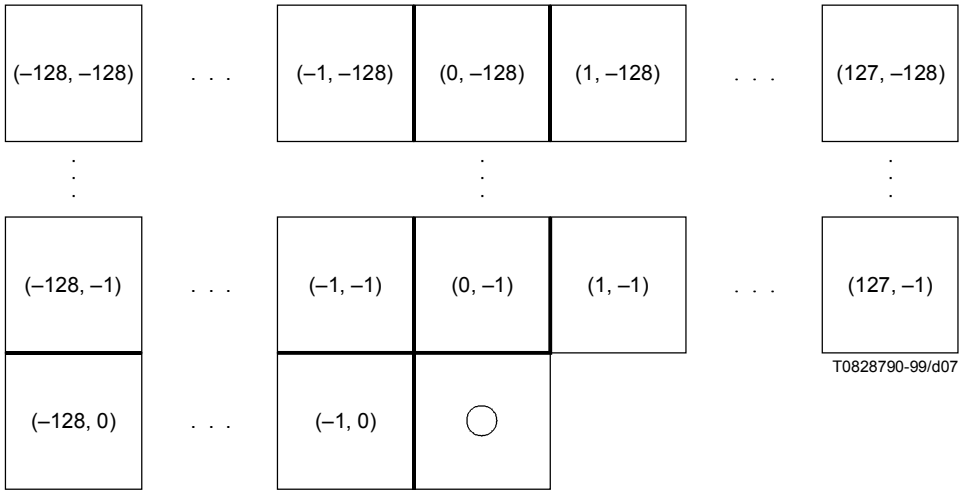


Figure 7 – Champ auquel les coordonnées de pixel sont restreintes

Tableau 5 – Valeurs nominales des coordonnées en pixels des gabarits adaptatifs

GBTEMPLATE	$GBATX_1$ $GBATY_1$	$GBATX_2$ $GBATY_2$	$GBATX_3$ $GBATY_3$	$GBATX_4$ $GBATY_4$
0	3 –1	–3 –1	2 –2	–2 –2
1	3 –1	NA NA	NA NA	NA NA
2	2 –1	NA NA	NA NA	NA NA
3	2 –1	NA NA	NA NA	NA NA
NOTE – "NA" signifie que le paramètre n'a pas de valeur nominale.				

6.2.5.5 Prédiction typique pour le codage direct de la région générique (TPGD)

La prédiction typique pour le codage direct de la région générique peut être activée ou désactivée par le paramètre **TPGDON**. Si la prédiction typique pour le codage direct de la région générique est activée (**TPGDON** = **1**), il faut décoder une valeur indiquant qu'une rangée est typique avant de décoder le premier pixel de cette rangée. Si la rangée est typique, chaque pixel de cette rangée est identique au pixel correspondant dans la rangée immédiatement supérieure, de sorte qu'aucun autre pixel de cette rangée n'a besoin d'être décodé. Si la rangée n'est pas typique, chacun de ses pixels doit être décodé.

NOTE – La valeur décodée avant le premier pixel de chaque rangée n'est utilisée dans aucun gabarit de pixel.

6.2.5.6 Pixels omis

Si le paramètre **USESKIP** est égal à **1**, le paramètre **SKIP** contient une matrice de dimensions **GBW** x **GBH** pixels, dont chaque pixel correspond à un pixel de la matrice en cours de décodage. Si le pixel de **SKIP** est égal à **1**, le pixel correspondant dans la matrice en cours de décodage est égal à **0** et n'est finalement pas décodé.

6.2.5.7 Décodage de la phototrame

Le décodage de la phototrame se déroule comme suit:

1) Poser:

$$LTP = \mathbf{0}$$

2) Créer une matrice GBREG de largeur **GBW** et de hauteur **GBH** pixels.

3) Décoder chaque rangée comme suit:

   a) si toutes les rangées **GBH** ont été décodées, le décodage est complet et l'on passe à l'étape 4);

   b) si **TPGDON** = **1**, on décode un bit au moyen du codeur d'entropie arithmétique, dont le contexte utilisé pour ce décodage varie en fonction du gabarit utilisé:

      - si **GBTEMPLATE** = 0, utiliser le contexte indiqué dans la Figure 8;
      - si **GBTEMPLATE** = 1, utiliser le contexte indiqué dans la Figure 9;
      - si **GBTEMPLATE** = 2, utiliser le contexte indiqué dans la Figure 10;
      - si **GBTEMPLATE** = 3, utiliser le contexte indiqué dans la Figure 11.

      Soit SLTP la valeur de ce bit. Posons:

$$LTP = LTP \text{ XOR } SLTP$$

      NOTE – Dans les Figures 8 à 11, le gabarit est représenté avec le(ou les) pixels AT à leur emplacement nominal. Les mêmes valeurs de pixel (**0** ou **1**) doivent être utilisées pour les pixels AT quel que soit leur emplacement réel. En d'autres termes, le déplacement des pixels AT n'affecte pas le contexte utilisé pour décoder SLTP.

   c) Si LTP = **1**, on pose l'égalité de chaque pixel de la rangée actuelle de la matrice GBREG avec le pixel correspondant de la rangée immédiatement supérieure.

   d) Si LTP = **0**, on décode de gauche à droite chaque pixel de la rangée actuelle de la matrice GBREG. La procédure est la suivante pour chaque pixel:

      i) si **USESKIP** = **1** et que le pixel de la matrice **SKIP** à l'emplacement correspondant au pixel actuel a la valeur **1**, mettre le pixel actuel à **0**;

      ii) si ce n'est pas le cas,

         - placer le gabarit indiqué par les paramètres **GBTEMPLATE**, $\mathbf{GBATX_1}$ à $\mathbf{GBATX_4}$ et $\mathbf{GBATY_1}$ à $\mathbf{GBATY_4}$ de façon que le pixel actuel soit aligné sur l'emplacement indiqué par un cercle dans la figure décrivant l'aspect du gabarit portant l'identificateur **GBTEMPLATE**;
         - former un entier CONTEXT en rassemblant les éléments d'image chevauchés par le gabarit (y compris les pixels AT) à l'emplacement actuel de ce gabarit. L'ordre de ce rassemblement n'est pas normalisé mais doit être cohérent et indépendant de l'emplacement des pixels AT;
         - décoder le pixel actuel en invoquant la procédure de décodage d'entropie arithmétique avec CX mis à la valeur formée par la concaténation de l'étiquette "GB" avec les valeurs des 10 ou 16 pixels rassemblées dans l'entier CONTEXT. Le résultat de cette invocation est la valeur du pixel actuel.

      EXEMPLE – Si le paramètre **GBTEMPLATE** = 2, les éléments d'image chevauchés par le gabarit sont comme indiqué sur la Figure 10 et les pixels sont rassemblés dans l'ordre de lecture (par rangées de haut en bas et de gauche à droite dans chaque rangée), puis CX est mis à la valeur "GB**0011100101**".

4) Une fois que toutes les rangées ont été décodées, le contenu actuel de la matrice GBREG est le résultat qui doit être obtenu par chaque décodeur, qu'il suive ou non cette séquence précise.

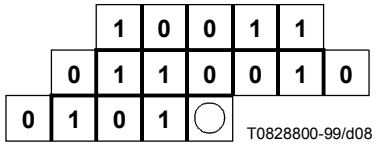


Figure 8 – Contexte réutilisé pour le codage de la valeur SLTP lorsque GBTEMPLATE = 0

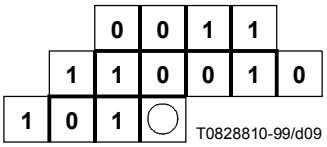


Figure 9 – Contexte réutilisé pour le codage de la valeur SLTP lorsque GBTEMPLATE = 1

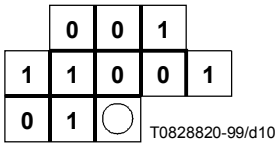


Figure 10 – Contexte réutilisé pour le codage de la valeur SLTP lorsque GBTEMPLATE = 2

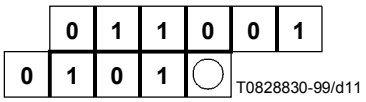


Figure 11 – Contexte réutilisé pour le codage de la valeur SLTP lorsque GBTEMPLATE = 3

6.2.6 Décodage au moyen du codage MMR

Si le paramètre **MMR** a la valeur **1**, la procédure de décodage de la région générique est identique à un décodage MMR (modification du codage READ modifié) décrit dans la Recommandation UIT-T T.6 avec les exceptions suivantes:

- Une invocation de la procédure de décodage de région générique avec **MMR** = **1** doit consommer un nombre entier d'octets, la série commençant et se terminant à une limite d'octet. Cela peut impliquer l'omission de certains bits dans le dernier octet lu.

- Le décodeur de la Recommandation UIT-T T.6 est spécifié comme produisant des pixels dont la valeur peut être soit "le noir" soit "le blanc". Dans le cadre de la présente Recommandation | Norme internationale, il faut interpréter comme suit le résultat de l'utilisation du décodeur MMR:

  – les pixels décodés par le décodeur MMR, ayant la valeur "noir", doivent être traités comme ayant la valeur **1**;

  – les pixels décodés par le décodeur MMR, ayant la valeur "blanc", doivent être traités comme ayant la valeur **0**.

- Si le nombre d'octets contenus dans la phototrame codée est connu d'avance, il est possible que le flux de données ne contienne pas de séquence EOFB (**000000000001000000000001**) à la fin des données à codage MMR. Cette procédure de décodage est invoquée lorsque le nombre d'octets est connu comme suit:

  – depuis la procédure de décodage à dictionnaire de structures;

  – depuis la procédure de décodage par dictionnaire de symboles;

  – dans le cadre du décodage d'une région générique dont la longueur de données est connue.

  Le nombre d'octets n'est pas connu lorsque cette procédure de décodage est invoquée depuis la procédure de décodage d'image en échelle de gris ou invoquée dans le cadre du décodage d'une région générique dont la longueur de données n'est pas connue. Dans ces cas-là, la séquence EOFB doit être présente.

NOTE 1 – Lorsque le nombre d'octets est connu, les moyens de le calculer sont les suivants:

– dans la procédure de décodage à dictionnaire de structures, le nombre d'octets est connu car toutes les données de segment, au-delà de l'en-tête de données de longueur fixe, constituent un seul bloc de données codées MMR, de sorte que la longueur des données MMR peut être calculée à partir de la longueur des données de segment;

– dans la procédure de décodage par dictionnaire de symboles, le nombre d'octets est connu à partir de BMSIZE;

– Dans la procédure de décodage régionale générique, le nombre d'octets est là encore calculé à partir de la longueur des données de segment.

La séquence EOFB peut être facultative car sa longueur est de trois octets tandis que le nombre d'octets est souvent connu au préalable ou peut être codé avec moins de trois octets. Ainsi, l'omission d'une séquence EOFB réduit la taille des données codées des phototrames; dans les dictionnaires de symboles, dans lesquels il existe souvent de nombreuses petites phototrames codées séparément, l'économie ainsi faite peut être conséquente.

NOTE 2 – Lorsque la séquence EOFB est facultative, un décodeur peut la traiter de plusieurs façons. Ces différentes méthodes utilisent le nombre d'octets connus et le fait que la séquence EOFB, si elle est présente, est comptée dans le nombre d'octets. Deux méthodes sont possibles:

– invoquer la procédure de décodage MMR et toujours vérifier la présence d'une séquence EOFB après le décodage de la phototrame. Toutefois, il faut faire en sorte que la procédure de décodage MMR n'examine pas plus d'octets que ceux qui sont censés faire partie du bloc de données comprimées MMR. Si la procédure de décodage MMR est à court de données pendant la vérification de la présence de la séquence EOFB, il ne s'agit pas d'une erreur mais d'une condition normale indiquant que la séquence EOFB n'était pas présente.

– invoquer la procédure de décodage MMR et ne jamais vérifier la présence de la séquence EOFB après le décodage de la phototrame, lorsque cette séquence est facultative. Lorsque la procédure de décodage MMR utilise moins d'octets que le nombre d'octets censés faire partie du bloc de données comprimées MMR, il ne s'agit pas d'une erreur mais d'une condition normale indiquant la présence de la séquence EOFB. Omettre les octets non utilisés;

- Les codes d'extension de la Recommandation T.6, y compris le mode sans compression, ne doivent pas être présents dans les données à codage MMR.

NOTE 3 – Le codage MMR offre un taux de compression inférieur à celui du codage arithmétique d'une matrice d'image. Le décodage d'une matrice d'image est plus rapide en MMR qu'en codage arithmétique.

6.3 Procédure de décodage régionale générique par raffinement

6.3.1 Description générale

Cette procédure de décodage est utilisée pour décoder une table rectangulaire de valeurs **0** ou **1** qui sont codées pixel par pixel. Une phototrame de référence est signalée à la procédure de décodage, qui l'utilise dans le cadre du processus de décodage. Cette matrice de référence est destinée à ressembler à la matrice en cours de décodage et cette similarité est utilisée pour augmenter la compression. Chaque pixel est décodé au moyen d'un contexte formé de pixels extraits de la matrice de référence ainsi que de pixels déjà décodés provenant de la matrice en cours de décodage.

6.3.2 Paramètres d'entrée

Les paramètres de cette procédure de décodage sont indiqués dans le Tableau 6.

Tableau 6 – Paramètres pour la procédure de décodage régionale générique par raffinement

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
GRW	Entier	32	N	Largeur de la région
GRH	Entier	32	N	Hauteur de la région
GRTEMPLATE	Entier	1	N	Identificateur de gabarit
GRREFERENCE	Phototrame			Phototrame de référence
GRREFERENCEDX	Entier	32	Y	Décalage X de la matrice de référence par rapport à la matrice en décodage
GRREFERENCEDY	Entier	32	Y	Décalage Y de la matrice de référence par rapport à la matrice en décodage
TPGRON	Entier	1	N	Indication d'utilisation de la prédiction typique pour le raffinement générique
$GRATX_1$	Entier	8	Y	Emplacement X du pixel RA_1 [a)] de gabarit adaptatif
$GRATY_1$	Entier	8	Y	Emplacement Y du pixel RA_1 [a)] de gabarit adaptatif.
$GRATX_2$	Entier	8	Y	Emplacement X du pixel RA_2 [a)] de gabarit adaptatif.
$GRATY_2$	Entier	8	Y	Emplacement Y du pixel RA_2 [a)] de gabarit adaptatif.

a) Paramètre inutilisé si **GRTEMPLATE** ≠ **0**.

6.3.3 Valeur de retour

La variable dont la valeur est le résultat de cette procédure de décodage est indiquée dans le Tableau 7.

Tableau 7 – Valeur de retour issue de la procédure de décodage régionale générique par raffinement

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
GRREG	Phototrame			Phototrame décodée de la région

6.3.4 Variables utilisées lors du décodage

Les variables utilisées par cette procédure de décodage sont indiquées dans le Tableau 8.

Tableau 8 – Variables utilisées dans la procédure de décodage régionale générique par raffinement

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
CONTEXT	Entier	13	N	Valeurs des pixels contenus dans le gabarit
LTP	Entier	1	N	Indication que la ligne d'image actuelle est décodée explicitement[a)]
SLTP	Entier	1	N	Indication que la valeur LTP de la ligne actuelle diffère de la valeur LTP de la ligne précédente[a)]
TPGRPIX	Entier	1	N	Indication que le pixel actuel doit être décodé implicitement au moyen d'une prédiction TPGR[a)]
TPGRVAL	Entier	1	N	Valeur du pixel actuel prédit par TPGR[a)]

a) Paramètre inutilisé si **TPGRON** = **0**.

6.3.5 Décodage par gabarit et codage arithmétique

6.3.5.1 Description générale

La procédure de décodage régionale générique par raffinement est fondée sur le codage arithmétique avec un gabarit permettant de déterminer l'état du codage. Le reste du 6.3.5 décrit cette forme de décodage.

6.3.5.2 Ordre de codage et conventions relatives aux bords

L'algorithme de codage effectue des passes dans la matrice de raffinement à décoder en fonction d'une matrice de référence, dans l'ordre de balayage d'un canevas matriciel, c'est-à-dire par rangées de haut en bas et de gauche à droite dans chaque rangée. Le traitement d'un pixel cible actuel fera référence à certains pixels qui sont en relation spatiale fixe avec le pixel cible. Certains de ces pixels sont extraits de la version de référence de la phototrame et d'autres sont extraits des pixels déjà codés de la matrice raffinée.

Près des bords de la matrice, ces références de voisinage peuvent ne pas se trouver dans la matrice proprement dite. La règle de résolution des références hors limites est la suivante:

- Tous les pixels se trouvant à l'extérieur de la matrice actuelle ou de la matrice de référence ont la valeur 0.

6.3.5.3 Gabarits fixes et gabarits adaptatifs

Un gabarit définit un voisinage autour d'un pixel à coder. Les valeurs des pixels se trouvant dans ce voisinage définissent un contexte. Chaque contexte possède son propre estimateur adaptatif de probabilité, qui est utilisé par le codeur arithmétique (voir Annexe E). Bien qu'un gabarit soit une structure géométrique de pixels, ceux-ci sont censés prendre des valeurs lorsque leur gabarit est aligné sur une partie particulière de l'image.

Figure 12 – Gabarit de raffinement de 13 pixels montrant les pixels AT à leurs coordonnées nominales

La Figure 12 montre le gabarit qui doit être utilisé lorsque **GRTEMPLATE** = **0**. La Figure 13 montre le gabarit qui doit être utilisé lorsque **GRTEMPLATE** = **1**. Dans chacune de ces figures, le groupe de gauche indique les pixels issus des pixels déjà codés dans la matrice raffinée, qui se trouvent dans le gabarit, tandis que le groupe de droite indique les pixels issus de la version de référence du gabarit, qui se trouvent dans celui-ci. Chaque groupe de chaque figure comporte un pixel repéré par un cercle, qui est le pixel à coder. Les pixels désignés par "X" correspondent aux pixels ordinaires du gabarit. Les pixels désignés par RA_1–RA_2 sont spéciaux dans le gabarit et sont désignés comme étant des pixels "adaptatifs" ou AT. Ces pixels ont la particularité que leur emplacement peut être modifié au cours du codage de l'image. On trouvera une description des pixels AT au 6.3.5.4. Les légendes RA_1–RA_2 indiquent l'emplacement nominal des pixels AT 1 à 2.

Le pixel AT RA_1 peut se situer n'importe où dans le champ représenté sur la Figure 7, pixel actuel non compris. Le pixel AT RA_2 peut se situer n'importe où dans l'étendue de (–128, –128) à (127, 127) de la matrice de référence.

Les pixels situés dans le groupe de gauche de chaque gabarit doivent être alignés sur les pixels déjà codés de la matrice en cours de décodage, le pixel marqué par un cercle se trouvant sur le pixel à décoder. Soit (X,Y) l'emplacement de ce pixel. Les pixels situés dans le groupe de droite de chaque gabarit doivent être alignés sur la matrice de référence **GRREFERENCE**, le pixel marqué par un cercle étant placé à l'emplacement (X – **GRREFERENCEDX**,Y – **GRREFERENCEDY**). Les valeurs des pixels contenus dans le gabarit doivent être combinées pour former un contexte.

Chaque pixel du gabarit (y compris les pixels adaptatifs) doit correspondre à un bit spécifique du contexte, bien que les pixels du gabarit puissent être assignés à des bits du contexte dans un ordre quelconque. Comme il y a jusqu'à 13 pixels dans le gabarit, les contextes peuvent prendre jusqu'à 8192 valeurs différentes. Ce contexte doit être utilisé pour déterminer l'estimateur adaptatif de probabilité qui doit être utilisé par le codeur arithmétique pour coder le pixel à coder (voir Annexe E).

Figure 13 – Gabarit de raffinement de 10 pixels

6.3.5.4 Pixels de gabarit adaptatif

Lors du codage de l'image, les modifications du gabarit doivent être autorisées avec les restrictions décrites dans ce paragraphe.

Les pixels autorisés à changer sont appelés *pixels AT*. Leur emplacement nominal est indiqué dans la Figure 12 par "RA_1" et "RA_2". Noter que certains gabarits possèdent moins de quatre pixels AT.

6.3.5.5 Prédiction typique pour le raffinement de la région générique (TPGR)

La prédiction typique pour le codage de raffinement de la région générique peut être activée ou désactivée par le paramètre **TPGRON**. Si la prédiction typique pour le codage direct de la région générique est activée (**TPGRON** = **1**), il faut décoder une valeur indiquant qu'une rangée est typique avant de décoder le premier pixel de cette rangée. Si la rangée n'est pas typique, chaque pixel de cette rangée a besoin d'être décodé explicitement. Si la rangée est typique, tous les pixels prévisibles typiquement peuvent être implicitement décodés au moyen de leur valeur prédite, le reste des pixels étant toujours décodé explicitement. Pour qu'un pixel soit prévisible typiquement, il doit répondre aux critères définis à l'étape 3 d) du 6.3.5.6.

NOTE – La valeur décodée avant le premier pixel de chaque rangée n'est utilisée dans aucun gabarit de pixel.

6.3.5.6 Décodage de la phototrame raffinée

Le décodage de la phototrame se déroule comme suit:

1) Poser LTP = **0**.

2) Créer une matrice GRREG de largeur **GRW** et de hauteur **GRH** pixels.

3) Décoder chaque rangée comme suit:

   a) si toutes les rangées **GRH** ont été décodées, le décodage est complet et l'on passe à l'étape 4).

   b) si **TPGRON** = **1**, on décode un bit au moyen du codeur d'entropie arithmétique, dont le contexte utilisé pour ce décodage varie en fonction du gabarit utilisé:

      - si **GRTEMPLATE** = **0**, utiliser le contexte indiqué dans la Figure 14;
      - si **GRTEMPLATE** = **1**, utiliser le contexte indiqué dans la Figure 15;

      Soit SLTP la valeur de ce bit décodé. Posons:

      $$LTP = LTP \text{ XOR } SLTP$$

   c) Si LTP = **0**, on décode explicitement de gauche à droite chaque pixel de la rangée actuelle de la matrice GRREG. La procédure est la suivante pour chaque pixel:

      i) placer le gabarit indiqué par les paramètres **GRTEMPLATE** (et $\mathbf{GRATX_1}$, $\mathbf{GRATY_1}$, $\mathbf{GRATX_2}$ et $\mathbf{GRATY_2}$ si **GRTEMPLATE** = **0**) de façon que le pixel actuel soit aligné sur l'emplacement indiqué par un cercle dans la figure décrivant l'aspect du gabarit portant l'identificateur **GRTEMPLATE**;

      ii) former un entier CONTEXT en rassemblant les valeurs des éléments d'image chevauchés par le gabarit (y compris les pixels AT) à l'emplacement actuel de ce gabarit. L'ordre de ce rassemblement n'est pas normalisé mais doit être cohérent et indépendant de l'emplacement des pixels AT;

iii) décoder le pixel actuel en invoquant la procédure de décodage d'entropie arithmétique avec CX mis à la valeur formée par la concaténation de l'étiquette "GR" avec les valeurs des 10 ou 13 pixels rassemblées dans l'entier CONTEXT. Le résultat de cette invocation est la valeur du pixel actuel.

EXEMPLE – Si le paramètre **GRTEMPLATE** = **1**, les éléments d'image chevauchés par le gabarit sont comme indiqué sur la Figure 15 et les pixels sont rassemblés dans l'ordre de lecture (par rangées de haut en bas et de gauche à droite dans chaque rangée, les pixels de la matrice GRREG étant pris en compte avant les pixels de la matrice GRREFERENCE), puis CX est mis à la valeur "GR**0000001000**".

d) Si LTP = **1**, décoder implicitement de gauche à droite certains pixels de la rangée actuelle de la matrice GRREG et décoder explicitement les autres pixels. La procédure est la suivante pour chaque pixel:

i) poser TPGRPIX = **1** si:

- **TPGRON** = **1** AND;
- une table de 3 × 3 pixels dans la matrice de référence (Figure 16), centrée sur l'emplacement correspondant au pixel actuel, contient des pixels qui ont tous la même valeur.

Lorsque TPGRPIX est mis à **1**, donner à TPGRVAL la valeur prédite du pixel actuel, qui est la valeur commune aux neuf pixels adjacents de la table de 3 × 3;

ii) si TPGRPIX = **1**, décoder implicitement le pixel actuel en lui donnant sa valeur prédite (TPGRVAL);

iii) si ce n'est pas le cas, décoder explicitement le pixel actuel par la méthode des étapes 3 c) i) à 3 c) iii) ci-dessus.

4) Une fois que toutes les rangées ont été décodées, le contenu actuel de la matrice GRREG est le résultat qui doit être obtenu par chaque décodeur, qu'il suive ou non cette séquence précise.

Figure 14 – Contexte réutilisé pour le codage de la valeur SLTP lorsque GRTEMPLATE = 0

Figure 15 – Contexte réutilisé pour le codage de la valeur SLTP lorsque GRTEMPLATE = 1

Figure 16 – Gabarit TPGR

6.4 Procédure de décodage zonale alphanumérique

6.4.1 Description générale

Cette procédure est utilisée pour décoder une phototrame par décodage d'un certain nombre d'instances de symbole. Une instance de symbole contient un emplacement et un identificateur de symbole, avec éventuellement une matrice de raffinement. Ces instances de symbole sont combinées pour former la matrice décodée.

NOTE – Cette procédure de décodage sera normalement utilisée pour décoder la partie alphanumérique d'une page. Les symboles sont normalement des caractères alphanumériques isolés extraits d'une police ou d'un jeu de caractères (alphabet).

6.4.2 Paramètres d'entrée

Les paramètres de cette procédure de décodage sont indiqués dans le Tableau 9.

NOTE – Les valeurs de certains de ces paramètres sont les suivantes dans une situation typique de décodage d'une phototrame contenant des caractères alphanumériques dans l'ordre de lecture normal de la langue anglaise avec valeur **1** du pixel d'avant-plan:

- **SBDEFPIXEL** = **0**
- **SBCOMPBOP** = OR
- **TRANSPOSED** = **0**
- **REFCORNER** = BOTTOMLEFT

Tableau 9 – Paramètres pour la procédure de décodage régionale alphanumérique

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
SBHUFF	Entier	1	N	Indication de l'utilisation du codage de Huffman.
SBREFINE	Entier	1	N	Indication de l'utilisation du codage de raffinement.
SBW	Entier	32	N	Largeur de la région.
SBH	Entier	32	N	Hauteur de la région.
SBNUMINSTANCES	Entier	32	N	Nombre d'instances de symbole dans cette région.
SBSTRIPS	Entier	4	N	Dimensions des bandes d'instances de symbole. Peut prendre les valeurs 1, 2, 4 ou 8.
SBNUMSYMS	Entier	32	N	Nombre de symboles pouvant être utilisés dans cette région.
SBSYMCODES	Table de codes de Huffman			Table contenant les symboles utilisés dans cette région alphanumérique. Contient des codes de symbole **SBNUMSYMS**[a)].
SBSYMCODELEN	Entier	6	N	Longueur des codes de symbole utilisés dans la procédure IAID[d)].
SBSYMS	Table de symboles			Table contenant les symboles utilisés dans cette région alphanumérique. Contient des codes de symbole **SBNUMSYMS**.
SBDEFPIXEL	Entier	1	N	Pixel par défaut pour cette phototrame.
SBCOMBOP	Opérateur			Opérateur combinatoire pour cette région alphanumérique. Peut prendre la valeur OR, AND, XOR ou XNOR.
TRANSPOSED	Entier	1	N	Indication de déroulement vertical des bandes.
REFCORNER	Coin			Coin de référence de chaque matrice d'instance de symbole. Peut prendre la valeur TOPLEFT, TOPRIGHT, BOTTOMLEFT ou BOTTOMRIGHT.
SBDSOFFSET	Entier	5	Y	Décalage de toutes les valeurs de delta S.
SBHUFFFS	Table de Huffman			Table de Huffman utilisée pour décoder la coordonnée S de la première instance de symbole dans chaque bande[a)].

Tableau 9 – Paramètres pour la procédure de décodage régionale alphanumérique *(fin)*

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
SBHUFFFDS	Table de Huffman			Table de Huffman utilisée pour décoder la coordonnée S des instances de symbole suivantes dans chaque bande[a)].
SBHUFFDT	Table de Huffman			Table de Huffman utilisée pour décoder la différence entre coordonnées T de bandes non vides[a)].
SBHUFFRDW	Table de Huffman			Table de Huffman utilisée pour décoder la différence entre une largeur de symbole et la largeur d'une matrice de raffinement codée[b)].
SBHUFFRDH	Table de Huffman			Table de Huffman utilisée pour décoder la différence entre une hauteur de symbole et la hauteur d'une matrice de raffinement codée[b)].
SBHUFFRDX	Table de Huffman			Table de Huffman utilisée pour décoder la différence entre une coordonnée X d'instance de symbole et la coordonnée X d'une matrice de raffinement codée[b)].
SBHUFFRDY	Table de Huffman			Table de Huffman utilisée pour décoder la différence entre une coordonnée Y d'instance de symbole et la coordonnée Y d'une matrice de raffinement codée[b)].
SBHUFFRSIZE	Table de Huffman			Table de Huffman utilisée pour décoder la dimension d'une donnée de matrice de raffinement d'une instance de symbole[b)].
SBRTEMPLATE	Entier	1	N	Identificateur de gabarit pour le codage de raffinement de matrices d'instance de symbole[c)].
$SBRATX_1$	Entier	8	Y	Emplacement X du pixel RA_1 de gabarit adaptatif[c)].
$SBRATY_1$	Entier	8	Y	Emplacement Y du pixel RA_1 de gabarit adaptatif[c)].
$SBRATX_2$	Entier	8	Y	Emplacement X du pixel RA_2 de gabarit adaptatif[c)].
$SBRATY_2$	Entier	8	Y	Emplacement Y du pixel RA_2 de gabarit adaptatif[c)].

a) Paramètre inutilisé si **SBHUFF** = **0**.
b) Paramètre inutilisé si **SBHUFF** = **0** ou **SBREFINE** = **0**.
c) Paramètre inutilisé si **SBREFINE** = **0**.
d) Paramètre inutilisé si **SBHUFF** = **1**.

6.4.3 Valeur de retour

La variable dont la valeur est le résultat de cette procédure de décodage est indiquée dans le Tableau 10.

Tableau 10 – Valeur de retour issue de la procédure de décodage zonale alphanumérique

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
SBREG	Phototrame			Matrice décodée de la région

6.4.4 Variables utilisées lors du décodage

Les variables utilisées par cette procédure de décodage sont indiquées dans le Tableau 11.

Tableau 11 – Variables utilisées dans la procédure de décodage zonale alphanumérique

Name	Type	Longueur (bits)	Signé?	Description et restrictions
STRIPT	Entier	32	Y	Coordonnée Y numériquement la plus petite dans la bande actuelle
FIRSTS	Entier	32	Y	Première coordonnée S de la bande actuelle.
NINSTANCES	Entier	32	N	Compteur d'instances de symbole.
DT	Entier	32	Y	Nombre de bandes vides entre deux bandes non vides.
DFS	Entier	32	Y	Différence de coordonnées S entre les premières instances de symbole de deux bandes.
CURS	Entier	32	Y	Coordonnée S actuelle.
CURT	Entier	3	N	Coordonnée T d'instance de symbole actuelle par rapport à la bande actuelle.
S_I	Entier	32	Y	Coordonnée S d'instance de symbole.
T_I	Entier	32	Y	Coordonnée T d'instance de symbole.
ID_I	Entier	32	N	Identificateur de symbole d'instance de symbole.
IB_I	Matrice de symbole			Matrice de symbole d'instance de symbole.
W_I	Entier	32	N	Largeur diférentielle d'une matrice de symbole d'instance de symbole.
H_I	Entier	32	N	Hauteur différencielle d'une matrice de symbole d'une instance de symbole.
IDS	Entier	32	Y	Différence de cordonnées S entre deux instances de symboles d'une même bande.
R_I	Entier	1	N	Indication du codage avec raffinement d'une matrice de symbole d'instance de symbole.
RDW_I	Entier	32	Y	Décalage de la largeur différentielle d'une matrice de raffinement d'instance de symbole[a)].
RDH_I	Entier	32	Y	Décalage de la hauteur différentielle d'une matrice de raffinement d'instance de symbole[a)].
RDX_I	Entier	32	Y	Décalage X d'une matrice de raffinement d'instance de symbole[a)].
RDY_I	Entier	32	Y	Décalage Y d'une matrice de raffinement d'instance de symbole[a)].
IBO_I	Matrice de symbole			Matrice de symbole originale d'une instance de symbole[a)].
WO_I	Entier	32	N	Largeur de la matrice IBO_I[a)].
HO_I	Entier	32	N	Hauteur de la matrice IBO_I[a)].
a) Paramètre inutilisé si **SBREFINE** = **0**.				

6.4.5 Décodage de la région alphanumérique

Une phototrame codée en symbole est représentée par un ensemble d'instances de symbole dont chacune code un emplacement, un identificateur de symbole et éventuellement des informations de raffinement. L'emplacement de chaque instance de symbole se compose d'une coordonnée S et d'une coordonnée T. Si **TRANSPOSED** = **0**, l'axe de coordonnée S correspond à l'axe X de la matrice et l'axe T correspond à l'axe Y de la matrice. Si **TRANSPOSED** = **1**, l'axe de coordonnée S correspond à l'axe Y de la matrice et l'axe T correspond à l'axe X de la phototrame.

NOTE 1 – La transposition des axes de coordonnée permet un codage efficace des textes écrits verticalement. Le coin de référence est variable parce que le codage le plus efficace est habituellement obtenu lorsque le coin de référence de chaque instance de symbole se trouve sur une ligne alphanumérique de base et lorsque les lignes alphanumériques de base peuvent s'écrire dans un sens quelconque.

Pour améliorer le taux de compression, les instances de symbole sont groupées en bandes en fonction de leur valeur T_I et de la valeur de **SBSTRIPS**. Les instances de symbole dont les valeurs T_I sont comprises entre 0 et **SBSTRIPS** – 1 sont groupées dans une même bande, les instances de symbole dont les valeurs T_I sont comprises entre **SBSTRIPS** et $2 \times$ **SBSTRIPS** – 1 sont groupées dans la bande suivante, et ainsi de suite. A l'intérieur de chaque bande, les instances de symbole sont codées dans l'ordre croissant des coordonnées S.

NOTE 2 – Normalement, les bandes apparaissent dans l'ordre strictement croissant des coordonnées T et les instances de symbole contenues dans chaque bande apparaissent dans l'ordre des coordonnées S non décroissantes. Il est cependant possible que des valeurs delta S ou delta T négatives apparaissent en cours de décodage, indiquant que les bandes et les instances de symbole peuvent apparaître dans un ordre quelconque.

La structure globale des données à décoder pour reconstruire la région alphanumérique est représentée sur la Figure 17. Le format de chaque bande est représenté sur la Figure 18. Lorsque **SBREFINE** = **0**, le format de chaque instance de symbole est conforme à la Figure 19. Lorsque **SBREFINE** = **1**, le format de chaque instance de symbole est conforme à la Figure 20.

NOTE 3 – Il se peut que le coin de référence de certaines instances de symbole se trouve au-dessus du sommet de la région. Pour coder ce coin, il faut faire en sorte qu'une bande se trouve également au-dessus du sommet de la région. La valeur initiale de STRIPT est la coordonnée permettant de localiser la première bande.

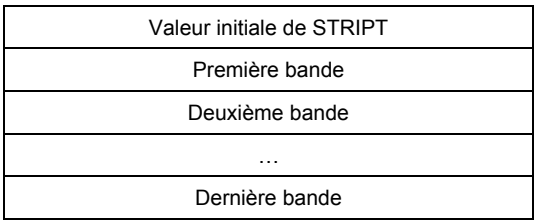


Figure 17 – Structure codée d'une région alphanumérique

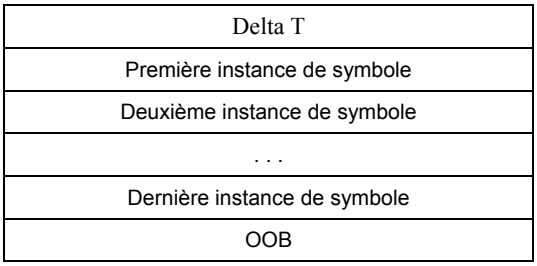


Figure 18 – Structure d'une bande

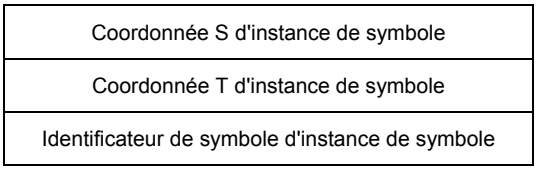


Figure 19 – Structure d'une instance de symbole lorsque SBREFINE = 0

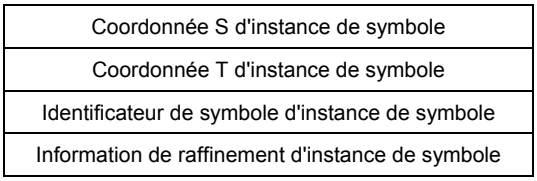


Figure 20 – Structure d'une instance de symbole lorsque SBREFINE = 1

Le résultat du décodage d'une région alphanumérique doit être la phototrame qui est produite par les étapes suivantes:

1) au moyen de la valeur **SBDEFPIXEL**, remplir une phototrame SBREG des dimensions indiquées par **SBW** et **SBH**;

2) décoder la valeur STRIPT initiale comme décrit au 6.4.6. Inverser la valeur décodée et attribuer cette valeur inversée à la variable STRIPT. Attribuer la valeur 0 à FIRSTS. Attribuer la valeur 0 à NINSTANCES;

3) décoder chaque bande comme suit:

   a) si NINSTANCES = **SBNUMINSTANCES**, il n'y a plus de bandes à décoder et le processus de décodage de la région alphanumérique est terminé; passer à l'étape 4);

   b) décoder la valeur delta T de la bande comme décrit au 6.4.6. Soit DT la valeur ainsi décodée. Posons:

$$\text{STRIPT} = \text{STRIPT} + \text{DT}$$

   c) décoder chaque instance de symbole de la bande comme suit:

      i) si l'instance de symbole actuelle est la première dans la bande, décoder la première coordonnée S d'instance de symbole comme décrit au 6.4.7. Soit DFS la valeur ainsi décodée. Posons:

$$\text{FIRSTS} = \text{FIRSTS} + \text{DFS}$$
$$\text{CURS} = \text{FIRSTS}$$

      ii) si ce n'est pas le cas (si l'instance de symbole actuelle n'est pas la première dans la bande), décoder la coordonnée S d'instance de symbole comme décrit au 6.4.8. Si le résultat de ce décodage est OOB, la dernière instance de symbole de la bande a été décodée; passer à l'étape 3 d). Sinon, soit IDS la valeur ainsi décodée. Posons:

$$\text{CURS} = \text{CURS} + \text{IDS} + \textbf{SBDSOFFSET}$$

NOTE 4 – L'usage prévu de **SBDSOFFSET** est d'annuler la valeur la plus souvent décodée au 6.4.8. Dans toutes les tables utilisées au 6.4.8, le code le plus court est celui de la valeur 0.

      iii) Décoder la coordonnée T de l'instance de symbole comme décrit au 6.4.9. Soit CURT la valeur ainsi décodée. Posons:

$$T_I = \text{STRIPT} + \text{CURT}$$

      iv) Décoder l'identificateur de symbole de l'instance de symbole comme décrit au 6.4.10. Soi ID_I la valeur ainsi décodée.

      v) Déterminer la phototrame IB_I de l'instance de symbole comme décrit au 6.4.11. La largeur et la hauteur de cette phototrame doivent être désignées respectivement par W_I et H_I.

      vi) Actualiser CURS comme suit:

- si **TRANSPOSED** = **0** et que **REFCORNER** = TOPRIGHT ou BOTTOMRIGHT, posons:

$$\text{CURS} = \text{CURS} + W_I - 1$$

- Si **TRANSPOSED** = **1** et que **REFCORNER** = BOTTOMLEFT ou BOTTOMRIGHT, posons:

$$\text{CURS} = \text{CURS} + H_I - 1$$

- sinon, ne pas modifier CURS à cette étape.

      vii) Posons:

$$S_I = \text{CURS}$$

      viii) Déterminer comme suit l'emplacement de la matrice d'instance de symbole par rapport à SBREG:

- si **TRANSPOSED** = **0**, alors:
  - si **REFCORNER** = TOPLEFT, le pixel gauche supérieur de la matrice d'instance de symbole IB_I doit être placé à SBREG[S_I, T_I];

– si **REFCORNER** = TOPRIGHT, le pixel droit supérieur de la matrice d'instance de symbole IB_I doit être placé à SBREG[S_I, T_I];

– si **REFCORNER** = BOTTOMLEFT, le pixel gauche inférieur de la matrice d'instance de symbole IB_I doit être placé à SBREG[S_I, T_I];

– si **REFCORNER** = BOTTOMRIGHT, le pixel droit inférieur de la matrice d'instance de symbole IB_I doit être placé à SBREG[S_I, T_I];

- si **TRANSPOSED** = **1**, alors

– si **REFCORNER** = TOPLEFT, le pixel gauche supérieur de la matrice d'instance de symbole IB_I doit être placé à SBREG[T_I, S_I];

– si **REFCORNER** = TOPRIGHT, le pixel droit supérieur de la matrice d'instance de symbole IB_I doit être placé à SBREG[T_I, S_I];

– si **REFCORNER** = BOTTOMLEFT, le pixel gauche inférieur de la matrice d'instance de symbole IB_I doit être placé à SBREG[T_I, S_I];

– si **REFCORNER** = BOTTOMRIGHT, le pixel droit inférieur de la matrice d'instance de symbole IB_I doit être placé à SBREG[T_I, S_I];

Si une partie quelconque de la matrice IB_I, placée à cet endroit, se trouve en dehors des limites de SBREG, ne pas tenir compte de cette partie de IB_I dans l'étape 3) c) ix).

ix) Insérer IB_I dans SBREG. Combiner chaque pixel de IB_I avec la valeur actuelle du pixel correspondant de SBREG, au moyen de l'opérateur combinatoire spécifié par **SBCOMBOP**. Ecrire le résultat de chaque combinaison sur ce pixel de SBREG.

x) Actualiser CURS comme suit:

- Si **TRANSPOSED** = **0** et que **REFCORNER** = TOPLEFT ou BOTTOMLEFT, posons:

$$\text{CURS} = \text{CURS} + W_I - 1$$

- Si **TRANSPOSED** = **1** et que **REFCORNER** = TOPLEFT ou TOPRIGHT, posons:

$$\text{CURS} = \text{CURS} + H_I - 1$$

- Sinon, ne pas modifier CURS à cette étape.

NOTE 5 – Les règles d'actualisation de CURS sont conçues de façon à permettre de coder l'intervalle entre instances de symbole adjacentes, plutôt que la distance entre leurs coins de référence; cela élimine une source de variation (largeur ou hauteur de matrice d'instance de symbole) et permet une meilleure compression.

xi) Posons:

$$\text{NINSTANCES} = \text{NINSTANCES} + 1$$

d) Lorsque la bande a été complètement décodée, passer au décodage de la bande suivante.

4) Une fois que toutes les bandes ont été décodées, le contenu actuel de SBREG est le résultat qui doit être obtenu par chaque décodeur, qu'il suive exactement cette séquence d'étapes ou non.

6.4.6 Différence delta T entre deux bandes

Si **SBHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SBHUFFDT** et multiplier la valeur ainsi obtenue par **SBSTRIPS**.

Si **SBHUFF**= **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IADT (voir Annexe A) et multiplier la valeur ainsi obtenue par **SBSTRIPS**.

6.4.7 Coordonnée S de la première instance de symbole

NOTE – La valeur de coordonnée S d'instance de symbole pour la première instance de symbole de chaque bande est codée différemment des instances symboliques suivantes de chaque bande, afin de tirer parti du fait que le début de chaque ligne est aligné sur le suivant.

Si **SBHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SBHUFFFS**.

Si **SBHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IAFS (voir Annexe A).

6.4.8 Coordonnée S de l'instance de symbole suivante

Si **SBHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SBHUFFDS**.

Si **SBHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IADS (voir Annexe A).

Dans un cas comme dans l'autre, il est possible que le résultat de ce décodage soit la valeur hors bande (OOB).

6.4.9 Coordonnée T d'une instance de symbole

Si **SBSTRIPS** = 1, la valeur décodée est toujours zéro. Sinon:

- si **SBHUFF** = **1**, décoder une valeur en lisant [$\log_2$ **SBSTRIPS**] directement dans le flux binaire;
- si **SBHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IAIT (voir Annexe A).

  NOTE – Si **SBSTRIPS** = 1, aucun bit n'est consommé et la procédure de décodage arithmétique d'entier IAIT n'est jamais invoquée.

6.4.10 Identificateur symbolique d'instance de symbole

Si **SBHUFF** = **1**, décoder une valeur en lisant un bit à la fois jusqu'à ce que la chaîne binaire résultante soit égale à l'une des entrées de **SBSYMCODES**. La valeur ainsi obtenue, ID_I, est l'indice pointant sur l'entrée qui est lue dans **SBSYMCODES**.

Si **SBHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IAID (voir Annexe A). Mettre ID_I à la valeur obtenue.

6.4.11 Matrice d'instance de symbole

Dans certains cas, la matrice d'instance de symbole IB_I est simplement la matrice du symbole désigné par ID_I. Dans d'autres cas, cependant, la matrice d'instance de symbole est la matrice qui a été modifiée par des informations additionnelles de raffinement. Le bit qui indique quelle option est vraie pour une instance de symbole est appelé R_I.

Si **SBREFINE** = **0**, mettre le bit R_I à **0**.

Si **SBREFINE** = **1**, décoder le bit R_I comme suit:

- si **SBHUFF** = **1**, lire un bit et mettre R_I à la valeur de ce bit;
- si **SBHUFF** = **0**, décoder un bit au moyen de la procédure de décodage arithmétique d'entier IARI et mettre R_I à la valeur de ce bit.

Si R_I = **0**, mettre la matrice d'instance de symbole IB_I à **SBSYMS**[ID_I].

Si R_I = **1**, déterminer la matrice d'instance de symbole comme suit:

1) décoder la largeur différentielle de raffinement d'instance de symbole comme décrit au 6.4.11.1. Soit RDW_I la valeur ainsi décodée;
2) définir la hauteur différentielle de raffinement d'instance de symbole comme décrit au 6.4.11.2. Soit RDH_I la valeur ainsi décodée;
3) décoder le décalage X de raffinement d'instance de symbole comme décrit au 6.4.11.3. Soit RDX_I la valeur ainsi décodée;
4) décoder le décalage Y de raffinement d'instance de symbole comme décrit au 6.4.11.4,. Soit RDY_I la valeur ainsi décodée;
5) si **SBHUFF** = **1**, alors:
   a) décoder la dimension de donnée de matrice de raffinement d'instance de symbole comme décrit au 6.4.11.5;
   b) omettre tous les bits restants dans le dernier octet lu;
6) soit IBO_I la matrice **SBSYMS**[ID_I]. Soit WO_I la largeur de la matrice IBO_I et HO_I sa hauteur. La matrice d'instance de symbole IB_I est le résultat de l'application de la procédure de décodage régionale générique par raffinement décrite au 6.3. Soit les paramètres de cette procédure de décodage comme indiqué au Tableau 12;
7) si **SBHUFF** = **1**, omettre tous les bits restant éventuellement dans le dernier octet lu. Le nombre total d'octets traités par la procédure de décodage de matrice de raffinement générique doit être égal à la valeur lue à l'étape 5) a).

6.4.11.1 Largeur différentielle du raffinement d'instance de symbole

Ce champ, ainsi que les champs suivants, indiquent les dimensions, l'emplacement et le contenu de la matrice de symbole raffinée car ces dimensions ne peuvent pas être les mêmes que celles de la matrice de symbole dont l'identificateur est donné dans cette instance de symbole; de même, le changement de dimensions de la matrice peut aller jusqu'au coin gauche supérieur et non seulement jusqu'au coin droit inférieur, de sorte qu'il faut indiquer un décalage ainsi qu'une dimension. Noter que les décalages sont exprimés par les lettres X et Y et non S et T.

Si **SBHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SBHUFFRDW**.

Si **SBHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IARDW (voir Annexe A).

Tableau 12 – Paramètres utilisés pour décoder une matrice d'instance de symbole utilisant le raffinement

Nom	Valeur
GRW	$WO_I + RDW_I$
GRH	$HO_I + RDH_I$
GRTEMPLATE	**SBRTEMPLATE**
GRREFERENCE	IBO_I
GRREFERENCEDX	$\lfloor RDW_I/2 \rfloor + RDX_I$
GRREFERENCEDY	$\lfloor RDH_I/2 \rfloor + RDY_I$
TPGRON	**0**
$\mathbf{GRATX_1}$	$\mathbf{SBRATX_1}$
$\mathbf{GRATY_1}$	$\mathbf{SBRATY_1}$
$\mathbf{GRATX_2}$	$\mathbf{SBRATX_2}$
$\mathbf{GRATY_2}$	$\mathbf{SBRATY_2}$

6.4.11.2 Hauteur différentielle du raffinement d'instance de symbole

Si **SBHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SBHUFFRDH**.

Si **SBHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IARDH (voir Annexe A).

6.4.11.3 Décalage X du raffinement d'instance de symbole

Si **SBHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SBHUFFRDX**.

Si **SBHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IARDX (voir Annexe A).

6.4.11.4 Décalage Y du raffinement d'instance de symbole

Si **SBHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SBHUFFRDY**.

Si **SBHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IARDY (voir Annexe A).

6.4.11.5 Dimensions des données matricielles du raffinement d'instance de symbole

Décoder une valeur au moyen de la table de Huffman spécifiée par **SBHUFFRSIZE**.

6.5 Procédure de décodage par dictionnaire de symboles

6.5.1 Description générale

Cette procédure de décodage est utilisée pour décoder un ensemble de symboles, qui pourront ensuite être utilisés par les procédures de décodage de région alphanumérique ou, parfois, par d'autres procédures de décodage par dictionnaire de symboles.

6.5.2 Paramètres d'entrée

Les paramètres de cette procédure de décodage sont indiqués dans le Tableau 13.

Tableau 13 – Paramètres pour la procédure de décodage par dictionnaire de symboles

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
SDHUFF	Entier	1	N	Indication d'utilisation du codage de Huffman.
SDREFAGG	Entier	1	N	Indication d'utilisation du codage de raffinement et du codage d'agrégation.
SDNUMINSYMS	Entier	32	N	Nombre de symboles utilisés à l'entrée de cette procédure de décodage par dictionnaire de symboles.
SDINSYMS	Table de symboles			Table contenant les symboles utilisés à l'entrée de cette procédure de décodage par dictionnaire de symboles. Ce paramètre contient les symboles **SDNUMINSYMS**.
SDNUMNEWSYMS	Entier	32	N	Nombre de symboles à définir dans ce dictionnaire de symboles.
SDNUMEXSYMS	Entier	32	N	Nombre de symboles à exporter à partir de ce dictionnaire de symboles.
SDHUFFDH	Table de Huffman			Table de Huffman utilisée pour décoder la différence de hauteur entre deux symboles[a)].
SDHUFFDW	Table de Huffman			Table de Huffman utilisée pour décoder la différence de largeur entre deux symboles[a)].
SDHUFFBMSIZE	Table de Huffman			Table de Huffman utilisée pour décoder les dimensions d'une matrice collective de classes de hauteur[a)].
SDHUFFAGGINST	Table de Huffman			Table de Huffman utilisée pour décoder le nombre d'instances de symbole dans une agrégation[b)].
SDTEMPLATE	Entier	2	N	Identificateur de gabarit utilisé pour décoder des matrices de symbole[c)].
$SDATX_1$	Entier	8	Y	Emplacement X du pixel A_1 de gabarit adaptatif[c)].
$SDATY_1$	Entier	8	Y	Emplacement Y du pixel A_1 de gabarit adaptatif[c)].
$SDATX_2$	Entier	8	Y	Emplacement X du pixel A_2 de gabarit adaptatif[c)].
$SDATY_2$	Entier	8	Y	Emplacement Y du pixel A_2 de gabarit adaptatif[c)].
$SDATX_3$	Entier	8	Y	Emplacement X du pixel A_3 de gabarit adaptatif[c)].
$SDATY_3$	Entier	8	Y	Emplacement Y du pixel A_3 de gabarit adaptatif[c)].
$SDATX_4$	Entier	8	Y	Emplacement X du pixel A_4 de gabarit adaptatif[c)].
$SDATY_4$	Entier	8	Y	Emplacement Y du pixel A_4 de gabarit adaptatif[c)].
SDRTEMPLATE	Entier	1	N	Identificateur de gabarit pour le codage de raffinement de phototrame[d)].
$SDRATX_1$	Entier	8	Y	Emplacement X du pixel RA_1 de gabarit adaptatif[d)].
$SDRATY_1$	Entier	8	Y	Emplacement Y du pixel RA_1 de gabarit adaptatif[d)].
$SDRATX_2$	Entier	8	Y	Emplacement X du pixel RA_2 de gabarit adaptatif[d)].
$SDRATY_2$	Entier	8	Y	Emplacement Y du pixel RA_2 de gabarit adaptatif[d)].

a) Paramètre inutilisé si **SDHUFF = 0**.
b) Paramètre inutilisé si **SDHUFF = 0** ou **SDREFAGG = 0**
c) Paramètre inutilisé si **SDHUFF = 1**.
d) Paramètre inutilisé si **SDREFAGG = 0**.

Le paramètre **SDREFAGG** détermine la façon dont sont codés les symboles contenus dans le dictionnaire de symboles actuel. Si **SDREFAGG = 0**, chaque matrice de symbole est codée par codage direct de phototrame. Si **SDREFAGG = 1**, chaque matrice de symbole est codée par raffinement ou par agrégation de matrices de symbole déjà définies, qui peuvent être extraites d'autres dictionnaires et fournies à l'entrée de cette procédure de décodage dans le paramètre **SDINSYMS** ou qui peuvent être définies dans le dictionnaire actuel.

6.5.3 Valeur de retour

La variable dont la valeur est le résultat de cette procédure de décodage est indiquée dans le Tableau 14.

Tableau 14 – Valeur de retour issue de la procédure de décodage par dictionnaire de symboles

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
SDEXSYMS	Table de symboles			Symboles exportés par ce dictionnaire de symboles. Contient les symboles **SDNUMEXSYMS**.

6.5.4 Variables utilisées lors du décodage

Les variables utilisées dans cette procédure de décodage sont indiquées dans le Tableau 15.

Tableau 15 – Variables utilisées dans la procédure de décodage par dictionnaire de symboles

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
SDNEWSYMS	Table de symboles			Symboles définis dans ce dictionnaire de symboles. Contient les symboles **SDNUMNEWSYMS**.
SDNEWSYMWIDTHS	Table d'entiers			Largeurs des symboles contenus dans SDNEWSYMS. Contient les entiers **SDNUMNEWSYMS**. Chaque entier est une valeur non signée de 32 bits.
HCHEIGHT	Entier	32	N	Hauteur de la classe de hauteur actuelle.
NSYMSDECODED	Entier	32	N	Nombre de symboles déjà décodés.
HCDH	Entier	32	Y	Différence de hauteur entre deux classes de hauteur.
SYMWIDTH	Entier	32	N	Largeur du symbole actuel.
TOTWIDTH	Entier	32	N	Largeur de la classe de hauteur actuelle.
HCFIRSTSYM	Entier	32	N	Indice du premier symbole dans la classe de hauteur actuelle.
DW	Entier	32	Y	Différence de largeur entre deux symboles.
B_S	Phototrame			Phototrame actuelle du symbole.
B_{HC}	Phototrame			Matrice collective actuelle des classes de hauteur.
I	Entier	32	N	Indice de table.
J	Entier	32	N	Indice de table.
REFAGGNINST	Entier	32	N	Nombre d'instances de symbole dans une agrégation.
EXFLAGS	Table d'entiers			Fanions d'exportation pour ce dictionnaire. Contient les valeurs **SDNUMINSYMS** + **SDNUMNEWSYMS**. Chaque valeur est un bit.
EXINDEX	Entier	32	N	Indice de table.
CUREXFLAG	Entier	1	N	Fanion d'exportation actuel.
EXRUNLENGTH	Entier	32	N	Longueur d'une séquence de valeurs de fanions d'exportation identiques.

6.5.5 Décodage par dictionnaire de symboles

La structure interne d'un dictionnaire de symboles est représentée dans la Figure 21. Les symboles définis dans le dictionnaire sont ordonnés en classes de hauteur. Une classe de hauteur contient un certain nombre de symboles dont les matrices ont la même hauteur.

NOTE 1 – Le plus souvent, les classes de hauteur apparaissent dans l'ordre strictement croissant des hauteurs, de la plus petite à la plus grande. Si **SDREFAGG** = **1**, un symbole peut cependant être codé comme un raffinement d'un plus grand symbole défini dans le même dictionnaire. Dans ce cas, la classe de hauteur pour ce symbole de base doit être décodée (et doit donc apparaître) avant la classe des plus petites hauteurs qui est codée pour raffiner ce symbole. C'est pourquoi les hauteurs différentielles (et les largeurs différentielles de symbole) d'une classe de hauteur peuvent avoir une valeur nulle, négative ou positive.

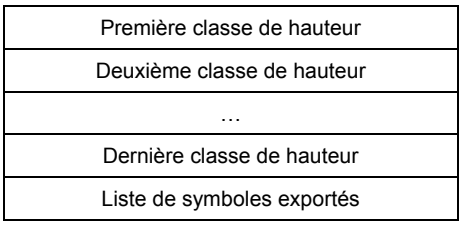


Figure 21 – Structure d'un dictionnaire de symboles

Si **SDHUFF** = **1** et si **SDREFAGG** = **0**, le format d'une classe de hauteur est conforme à la Figure 22. Sinon, le format d'une classe de hauteur est conforme à la Figure 23. Les champs mentionnés dans ces figures sont décrits complètement ci-dessous.

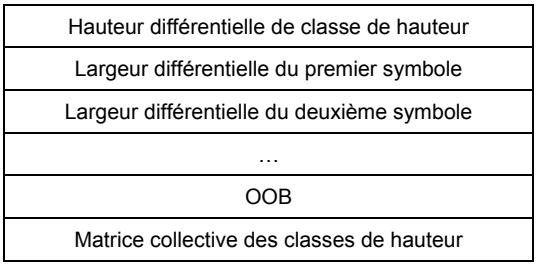


Figure 22 – Codage de classe de hauteur lorsque SDHUFF = 1 et que SDREFAGG = 0

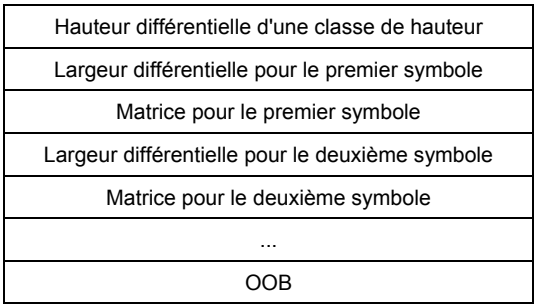


Figure 23 – Codage de classe de hauteur lorsque SDHUFF = 0 ou que SDREFAGG = 1

Le résultat du décodage d'un dictionnaire de symboles est une table SDEXSYMS contenant des matrices de type **SDNUMEXSYMS**. Cette table doit être produite par les étapes suivantes:

1) Créer une table SDNEWSYMS de phototrames, ayant **SDNUMNEWSYMS** entrées.
2) Si **SDHUFF** = **1** et **SDREFAGG** = **0**, créer une table SDNEWSYMWIDTHS d'entiers, ayant **SDNUMNEWSYMS** entrées.
3) Poser:

$$\text{HCHEIGHT} = 0$$
$$\text{NSYMSDECODED} = 0$$

4) Décoder chaque classe de hauteur comme suit:

a) si NSYMSDECODED = **SDNUMNEWSYMS**, alors tous les symboles contenus dans le dictionnaire ont été décodés; passer à l'étape 5);

b) décoder la hauteur différentielle de classe de hauteur comme décrit au 6.5.6. soit HCDH la valeur ainsi décodée. Poser:

$$\begin{aligned} \text{HCHEIGHT} &= \text{HCEIGHT} + \text{HCDH} \\ \text{SYMWIDTH} &= 0 \\ \text{TOTWIDTH} &= 0 \\ \text{HCFIRSTSYM} &= \text{NSYMSDECODED} \end{aligned}$$

c) décoder chaque symbole contenu dans la classe de hauteur comme suit:

i) décoder la largeur différentielle pour le symbole comme décrit au 6.5.7. Si le résultat de ce décodage est OOB, alors tous les symboles de cette classe de hauteur ont été décodés; passer à l'étape 4) d). Sinon, la valeur décodée est désignée par DW et l'on pose:

$$\begin{aligned} \text{SYMWIDTH} &= \text{SYMWIDTH} + \text{DW} \\ \text{TOTWIDTH} &= \text{TOTWIDTH} + \text{SYMWIDTH} \end{aligned}$$

ii) si **SDHUFF** = **0** ou si **SDREFAGG** = **1**, l'on décode la phototrame du symbole comme décrit au 6.5.8. Soit B_S la matrice décodée (dont la largeur est SYMWIDTH et la hauteur HCHEIGHT). Poser:

$$\text{SDNEWSYMS[NSYMSDECODED]} = B_S$$

iii) si **SDHUFF** = **1** et **SDREFAGG** = **0**, poser:

$$\text{SDNEWSYMWIDTHS[NSYMSDECODED]} = \text{SYMWIDTH}$$

iv) poser:

$$\text{NSYMSDECODED} = \text{NSYMSDECODED} + 1$$

d) si **SDHUFF** = **1** et **SDREFAGG** = **0**, décoder la matrice collective des classes de hauteur comme décrit au 6.5.9. Soit B_{HC} la matrice ainsi décodée, dont la largeur est TOTWIDTH et la hauteur HCHEIGHT. La matrice B_{HC} est subdivisée comme suit pour obtenir les symboles de SDNEWSYMS[HCFIRSTSYM] à SDNEWSYMS[NSYMSDECODED – 1].

B_{HC} contient les NSYMSDECODED – HCFIRSTSYM symboles concaténés de gauche à droite, sans intervalles. Pour chaque indice *I* entre HCFIRSTSYM et NSYMSDECODED – 1:

- la largeur de SDNEWSYMS[*I*] est la valeur de SDNEWSYMWIDTHS[*I*],
- la hauteur de SDNEWSYMS[*I*] est HCHEIGHT, et
- la matrice SDNEWSYMS[*I*] peut être obtenue par extraction des colonnes de la matrice B_{HC} depuis:

$$\sum_{J=\text{HCFIRSTSYM}}^{I-1} \text{SDNEWSYMWIDTHS}[J]$$

jusqu'à

$$\left(\sum_{J=\text{HCFIRSTSYM}}^{I} \text{SDNEWSYMWIDTHS}[J] \right) - 1$$

EXEMPLE – Les colonnes de 0 à SDNEWSYMWIDTHS[HCFIRSTSYM] – 1 de la matrice B_{HC} contiennent la matrice pour le premier symbole de la classe de hauteur, SDNEWSYMS[HCFIRSTSYM].

5) Déterminer quelles matrices de symboles sont exportées de ce dictionnaire de symboles, comme décrit au 6.5.10. Ces phototrames peuvent être extraites des symboles utilisés comme entrée dans la procédure de décodage du dictionnaire de symboles ainsi que des nouveaux symboles produits par la procédure de décodage.

NOTE 2 – Tous les nouveaux symboles n'ont pas besoin d'être exportés; cela permet au dictionnaire de définir un certain symbole, de l'utiliser par codage de raffinement/d'agrégation pour construire d'autres symboles, et de ne pas exporter finalement le symbole original. De même, puisque les symboles d'entrée peuvent être exportés, ce dictionnaire peut en effet copier des symboles issus d'autres dictionnaires.

6.5.6 Hauteur différentielle entre classes de hauteur

Si **SDHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SDHUFFDH**.

Si **SDHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IADH (voir Annexe A).

6.5.7 Largeur différentielle

Si **SDHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SDHUFFDW**.

Si **SDHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IADW (voir Annexe A).

Dans un cas comme dans l'autre, il est possible que le résultat de ce décodage soit la valeur hors bande (OOB).

6.5.8 Matrice de symbole

Ce champ n'est présent que si **SDHUFF** = **0** ou **SDREFAGG** = **1**. Il prend l'une des deux formes suivantes, selon la valeur de **SDREFAGG**.

6.5.8.1 Matrice de symbole à codage direct

Si **SDREFAGG** = **0**, décoder la matrice de symbole au moyen d'une procédure de décodage de région générique comme décrit au 6.2. Régler les paramètres de cette procédure de décodage comme indiqué dans le Tableau 16.

Tableau 16 – Paramètres utilisés pour décoder une matrice d'instance de symbole utilisant le raffinement

Nom	Valeur
MMR	**0**
GBW	SYMWIDTH
GBH	HCHEIGHT
GBTEMPLATE	**SDTEMPLATE**
TPGDON	**0**
USESKIP	**0**
$\mathbf{GBATX_1}$	$\mathbf{SDATX_1}$
$\mathbf{GBATY_1}$	$\mathbf{SDATY_1}$
$\mathbf{GBATX_2}$	$\mathbf{SDATX_2}$
$\mathbf{GBATY_2}$	$\mathbf{SDATY_2}$
$\mathbf{GBATX_3}$	$\mathbf{SDATX_3}$
$\mathbf{GBATY_3}$	$\mathbf{SDATY_3}$
$\mathbf{GBATX_4}$	$\mathbf{SDATX_4}$
$\mathbf{GBATY_4}$	$\mathbf{SDATY_4}$

6.5.8.2 Matrice de symbole à codage raffiné/agrégé

Si **SDREFAGG** = **1**, la matrice de symbole est codée par raffinement et agrégation d'autres symboles déjà définis. Décoder cette matrice comme suit:

1) décoder le nombre d'instances contenues dans l'agrégation, comme spécifié au 6.5.8.2.1. Soit REFAGGNINST la valeur ainsi décodée;

2) si REFAGGNINST a une valeur supérieure à 1, décoder la matrice proprement dite au moyen d'une procédure de décodage de région alphanumérique comme décrit au 6.4. Régler les paramètres de cette procédure de décodage comme indiqué dans le Tableau 17;

3) si REFAGGNINST = 1, décoder la phototrame binaire comme décrit au 6.5.8.2.2.

Tableau 17 – Paramètres utilisés pour décoder une matrice de symbole au moyen du décodage raffiné/agrégé

Nom	Valeur
SBHUFF	**SDHUFF**
SBREFINE	**1**
SBW	SYMWIDTH
SBH	HCHEIGHT
SBNUMINSTANCES	REFAGGNINST
SBSTRIPS	1
SBNUMSYMS	**SDNUMINSYMS** + NSYMSDECODED
SBSYMCODES	Voir 6.5.8.2.3[a)].
SBSYMCODELEN	Voir 6.5.8.2.3[b)].
SBSYMS	Voir 6.5.8.2.4.
SBDEFPIXEL	**0**
SBCOMBOP	OR
TRANSPOSED	**0**
REFCORNER	TOPLEFT
SBDSOFFSET	0
SBHUFFFS	Tableau B.6[a)]
SBHUFFDS	Tableau B.8[a)]
SBHUFFDT	Tableau B.11[a)]
SBHUFFRDW	Tableau B.15[a)]
SBHUFFRDH	Tableau B.15[a)]
SBHUFFRDX	Tableau B.15[a)]
SBHUFFRDY	Tableau B.15[a)]
SBHUFFRSIZE	Tableau B.1[a)]
SBRTEMPLATE	**SDRTEMPLATE**
$SBRATX_1$	**$SDRATX_1$**
$SBRATY_1$	**$SDRATY_1$**
$SBRATX_2$	**$SDRATX_2$**
$SBRATY_2$	**$SDRATY_2$**

a) Si **SDHUFF** = **0**, ce paramètre n'a pas de valeur.
b) Si **SDHUFF** = **1**, ce paramètre n'a pas de valeur.

6.5.8.2.1 Nombre d'instances symboliques dans une agrégation

Si **SDHUFF** = **1**, décoder une valeur au moyen de la table de Huffman spécifiée par **SDHUFFAGGINST**.

Si **SDHUFF** = **0**, décoder une valeur au moyen de la procédure de décodage arithmétique d'entier IAAI (voir Annexe A).

6.5.8.2.2 Décodage d'une matrice lorsque REFAGGNINST = 1

Si une matrice de symbole est codée par raffinement/agrégation et qu'il n'y ait qu'un seul symbole dans l'agrégation, cette matrice est décodée comme suit. Il s'agit essentiellement de la méthode suivie par la procédure de décodage de région de symbole, sauf que, lorsqu'une valeur est connue, elle n'est pas décodée.

1) Poser **SBHUFF** = **SDHUFF**.

2) Décoder un identificateur de symbole comme décrit au 6.4.10 au moyen des valeurs de **SBSYMCODES** et **SBSYMCODELEN** décrites au 6.5.8.2.3. Soit ID_I la valeur ainsi décodée.

3) Décoder le décalage X du raffinement d'instance comme décrit au 6.4.11.3. Si **SDHUFF** = **1**, utiliser le Tableau B.15 pour **SBHUFFRDX**. Soit RDX_I la valeur ainsi décodée.

4) Décoder le décalage Y du raffinement d'instance comme décrit au 6.4.11.4. Si **SDHUFF** = **1**, utiliser le Tableau B.15 pour **SBHUFFRDX**. Soit RDY_I la valeur ainsi décodée.

5) Si **SDHUFF** = **1**, alors:

   a) décoder la dimension de donnée de matrice de raffinement d'instance de symbole comme décrit au 6.4.11.5, au moyen du Tableau B.1 pour **SBHUFFRSIZE**;

   b) omettre tous les bits restant éventuellement dans le dernier octet lu.

6) Soit IBO_I la matrice **SBSYMS[*ID*$_I$]** où SBSYMS est conforme au 6.5.8.2.4. La matrice de symbole est le résultat de l'application de la procédure de décodage régionale générique par raffinement décrite au 6.3. Régler les paramètres de cette procédure de décodage comme indiqué dans le Tableau 18.

7) Si **SBHUFF** = **1**, omettre tous les bits restant éventuellement dans le dernier octet lu. Le nombre total d'octets traités par la procédure de décodage régionale générique par raffinement doit être égal à la valeur lue à l'étape 5) a).

Tableau 18 – Paramètres utilisés pour décoder une matrice d'instance de symbole lorsque REFAGGNINST = 1

Nom	Valeur
GRW	SYMWIDTH
GRH	HCHEIGHT
GRTEMPLATE	**SDRTEMPLATE**
GRREFERENCE	IBO_I
GRREFERENCEDX	RDX_I
GRREFERENCEDY	RDY_I
TPGRON	**0**
GRATX$_1$	**SDRATX$_1$**
GRATY$_1$	**SDRATY$_1$**
GRATX$_2$	**SDRATX$_2$**
GRATY$_2$	**SDRATY$_2$**

6.5.8.2.3 Réglage des paramètres SBSYMCODES et SBSYMCODELEN

Lorsque **SDHUFF** = **1**, l'on attribue le paramètre **SBSYMCODES** à une table de codes **SBNUMSYMS**, dans laquelle chaque code a une longueur égale à:

$$\max(\lceil \log_2 (\mathbf{SDNUMINSYMS} + \mathbf{SDNUMNEWSYMS}) \rceil, 1)$$

et dans laquelle le code **SBSYMCODES**[*I*] est *I* (pour *I* compris entre 0 et **SBNUMSYMS** – 1).

NOTE – Ce réglage donne aux codes une longueur égale, attribuée à partir de zéro. Les longueurs de code sont calculées à partir du nombre maximal de symboles disponibles dans le dictionnaire de symboles actuel: tous les symboles importés et tous les symboles définis ici. Un certain gaspillage se produit lors du choix de cette longueur de code et de l'attribution de ces codes. Grâce à cette méthode cependant, ni les longueurs de code ni les codes réellement attribués à chaque symbole ne changent pendant le processus de décodage du dictionnaire de symboles.

De même, lorsque **SDHUFF** = **0**, il y a lieu de régler **SBSYMCODELEN** à la valeur suivante:

$$\lceil \log_2 (\mathbf{SDNUMINSYMS} + \mathbf{SDNUMNEWSYMS}) \rceil$$

de façon que la longueur des chaînes binaires décodées par la méthode IAID ne change pas pendant le décodage du dictionnaire de symboles.

6.5.8.2.4 Réglage du paramètre SBSYMS

L'on attribue le paramètre **SBSYMS** à une table de symboles **SDNUMINSYMS** + NSYMSDECODED, formée par concaténation de la table **SDINSYMS** et des premières entrées NSYMSDECODED de la table SDNEWSYMS.

6.5.9 Matrice collective des classes de hauteur

Ce champ n'est présent que si **SDHUFF** = **1** et **SDREFAGG** = **0**.

Ce champ contient les phototrames de tous les symboles contenus dans la classe de hauteur, concaténées de gauche à droite et codées MMR. Ce champ est précédé d'un décompte de sa longueur en octets.

Ce champ est décodé comme suit:

1) Lire la longueur en octets au moyen de la table **SDHUFFBMSIZE**. Soit BMSIZE la valeur ainsi décodée.

2) Omettre les bits restant éventuellement dans le dernier octet lu.

3) Si BMSIZE = 0, la matrice est mémorisée sans compression et la longueur réelle (en octets) est égale à:

$$\text{HCHEIGHT} \times \left\lceil \frac{\text{TOTWIDTH}}{8} \right\rceil$$

Décoder la matrice en lisant ce nombre d'octets et en les traitant comme HCHEIGHT rangées de TOTWIDTH pixels, chaque rangée étant bourrée avec de 0 à 7 bits **0** jusqu'à une limite d'octet.

4) Sinon, décoder la phototrame au moyen d'une procédure de décodage de matrice générique comme décrit au 6.2. Régler les paramètres de cette procédure de décodage comme indiqué dans le Tableau 19.

Tableau 19 – Paramètres utilisés pour décoder une matrice collective de classes de hauteur

Nom	Valeur
MMR	**1**
GBW	TOTWIDTH
GBH	HCHEIGHT

5) Omettre tous les bits restant éventuellement dans le dernier octet lu.

NOTE – BMSIZE est utilisé pour déterminer le nombre d'octets des données codées MMR et permet ainsi au codeur d'omettre la séquence EOFB (voir 6.2.6); il en découle généralement une réduction de la taille des données codées. Le codeur peut également transmettre la phototrame non comprimée lorsque le codage MMR entraînerait une extension.

6.5.10 Symboles exportés

Les symboles qui peuvent être exportés à partir d'un dictionnaire donné comprennent tous les symboles qui sont insérés dans le dictionnaire, plus tous les symboles définis dans ce dictionnaire.

La table des symboles exportés du dictionnaire est produite par décodage d'un bit pour chacun de ces symboles. Ces bits forment une table EXFLAGS de valeurs binaires **SDNUMINSYMS** + **SDNUMNEWSYMS**, dont chacune correspond à un symbole d'entrée ou à un symbole récemment défini. Un bit **1** pour un symbole indique que ce symbole est exporté. Exactement **SDNUMEXSYMS** symboles doivent être exportés du dictionnaire. L'ordre des symboles exportés est celui de la concaténation de la table **SDINSYMS** avec la table SDNEWSYMS.

La procédure suivante produit cette table de symboles exportés:

1) Poser:

$$\text{EXINDEX} = 0$$
$$\text{CUREXFLAG} = 0$$

2) Décoder une valeur au moyen du Tableau B.1 si **SDHUFF** = **1** ou au moyen de la procédure de décodage arithmétique d'entier IAEX si **SDHUFF** = **0**. Soi EXRUNLENGTH la valeur ainsi décodée.

3) Mettre de EXFLAGS[EXINDEX] à EXFLAGS[EXINDEX + EXRUNLENGTH – 1] à CUREXFLAG. Si EXRUNLENGTH = 0, aucune valeur n'est modifiée par cette étape.

4) Poser:

$$\begin{aligned} \text{EXINDEX} &= \text{EXINDEX} + \text{EXRUNLENGTH} \\ \text{CUREXFLAG} &= \text{NOT(CUREXFLAG)} \end{aligned}$$

5) Répéter les étapes 2 à 4 jusqu'à ce que EXINDEX = **SDNUMINSYMS** + **SDNUMNEWSYMS**.

6) La table EXFLAGS contient maintenant le bit **1** pour chaque symbole qui est exporté du dictionnaire et le bit **0** pour chaque symbole qui n'est pas exporté.

7) Poser:

$$\begin{aligned} I &= 0 \\ J &= 0 \end{aligned}$$

8) Pour chaque valeur de *I* entre 0 et **SDNUMINSYMS** + **SDNUMNEWSYMS** – 1, si EXFLAGS[*I*] = **1**, exécuter les opérations suivantes:

   a) si $I <$ **SDNUMINSYMS**, poser:

$$\begin{aligned} \text{SDEXSYMS}[J] &= \textbf{SDINSYMS}[\boldsymbol{I}] \\ J &= J + 1 \end{aligned}$$

   b) si $I \geq$ **SDNUMINSYMS**, poser:

$$\begin{aligned} \text{SDEXSYMS}[J] &= \text{SDNEWSYMS}[I - \textbf{SDNUMINSYMS}] \\ J &= J + 1 \end{aligned}$$

NOTE – La plupart des dictionnaires exporteront exactement les nouveaux symboles qu'ils définissent; ils n'exporteront aucun des symboles contenus dans **SDINSYMS**. Dans ce cas, les premières valeurs **SDNUMINSYMS** de la table EXFLAGS sont **0** et les autres valeurs **SDNUMNEWSYMS** sont **1**.

6.6 Procédure de décodage régionale de dégradé

6.6.1 Description générale

Cette procédure de décodage sert à décoder une phototrame en décodant une table de valeurs qui sont utilisées pour dessiner des structures dans une grille de dégradé. Ces structures sont combinées pour former la matrice décodée.

NOTE – Cette forme de codage convient pour transmettre efficacement une phototrame contenant des données d'image à dégradé *périodique*, comme les données estompées par grappes de points ordonnées. D'autres formes de données d'image à dégradé, comme les données à diffusion d'erreur, peuvent être converties en cette forme par détramage ou peuvent être codées en une forme ressemblant mieux à l'original au moyen du codage de matrice générique.

6.6.2 Paramètres d'entrée

Les paramètres de cette procédure de décodage sont indiqués dans le Tableau 20.

6.6.3 Valeur de retour

La variable dont la valeur est le résultat de cette procédure de décodage est indiquée dans le Tableau 21.

6.6.4 Variables utilisées lors du décodage

Les variables utilisées par cette procédure de décodage sont indiquées dans le Tableau 22.

Tableau 20 – Paramètres pour la procédure de décodage zonale de dégradé

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
HBW	Entier	32	N	Largeur de la région.
HBH	Entier	32	N	Hauteur de la région.
HMMR	Entier	1	N	Indication d'utilisation du codage MMR.
HTEMPLATE	Entier	2	N	Identificateur de gabarit[a)].
HNUMPATS	Entier	32	N	Nombre de structures pouvant être utilisées dans cette région.
HPATS	Table de structures			Table contenant les structures pouvant être utilisées dans cette région. Contient les structures **HNUMPATS**.
HDEFPIXEL	Entier	1	N	Pixel par défaut pour cette phototrame.
HCOMBOP	Opérateur			Opérateur combinatoire utilisé dans cette région de dégradé. Peut prendre les valeurs OR, AND, XOR, XNOR et REPLACE.
HENABLESKIP	Entier	1	N	Indication d'omission des valeurs d'échelle de gris inutiles[a)].
HGW	Entier	32	N	Largeur de l'image en échelle de gris.
HGH	Entier	32	N	Hauteur de l'image en échelle de gris.
HGX	Entier	32	Y	256 fois le décalage horizontal de l'origine de la grille.
HGY	Entier	32	Y	256 fois le décalage vertical de l'origine de la grille.
HRX	Entier	16	N	256 fois la coordonnée horizontale du vecteur de grille.
HRY	Entier	16	N	256 fois la coordonnée verticale du vecteur de grille.
HPW	Entier	8	N	Largeur de chaque structure.
HPH	Entier	8	N	Hauteur de chaque structure.
a) Paramètre inutilisé si **HMMR** = **1**.				

Tableau 21 – Valeur de retour issue de la procédure de décodage zonale de dégradé

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
HTREG	Phototrame			Phototrame décodée de la région.

Tableau 22 – Variables utilisées dans la procédure de décodage zonale de dégradé

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
n_g	Entier	32	N	Indice horizontal pour la valeur actuelle d'échelle de gris.
m_g	Entier	32	N	Indice vertical pour la valeur actuelle d'échelle de gris.
x	Entier	32	Y	Coordonnée horizontale pour la structure correspondant à la valeur actuelle d'échelle de gris.
y	Entier	32	Y	Coordonnée verticale pour la structure correspondant à la valeur actuelle d'échelle de gris.
HSKIP	Phototrame			Masque indiquant les valeurs d'échelle de gris à omettre. HSKIP mesure **HGW** × **HGH** pixels[a)]
HBPP	Entier	32	N	Nombre de bits par valeur dans la table des valeurs d'échelle de gris.
GI	Tableau			Table de valeurs d'échelle de gris. GI est une table de dimensions **HGW** × **HGH** dont chaque entrée est un entier non signé de HBPP bits.
a) Paramètre inutilisé si **HENABLESKIP** = **0**.				

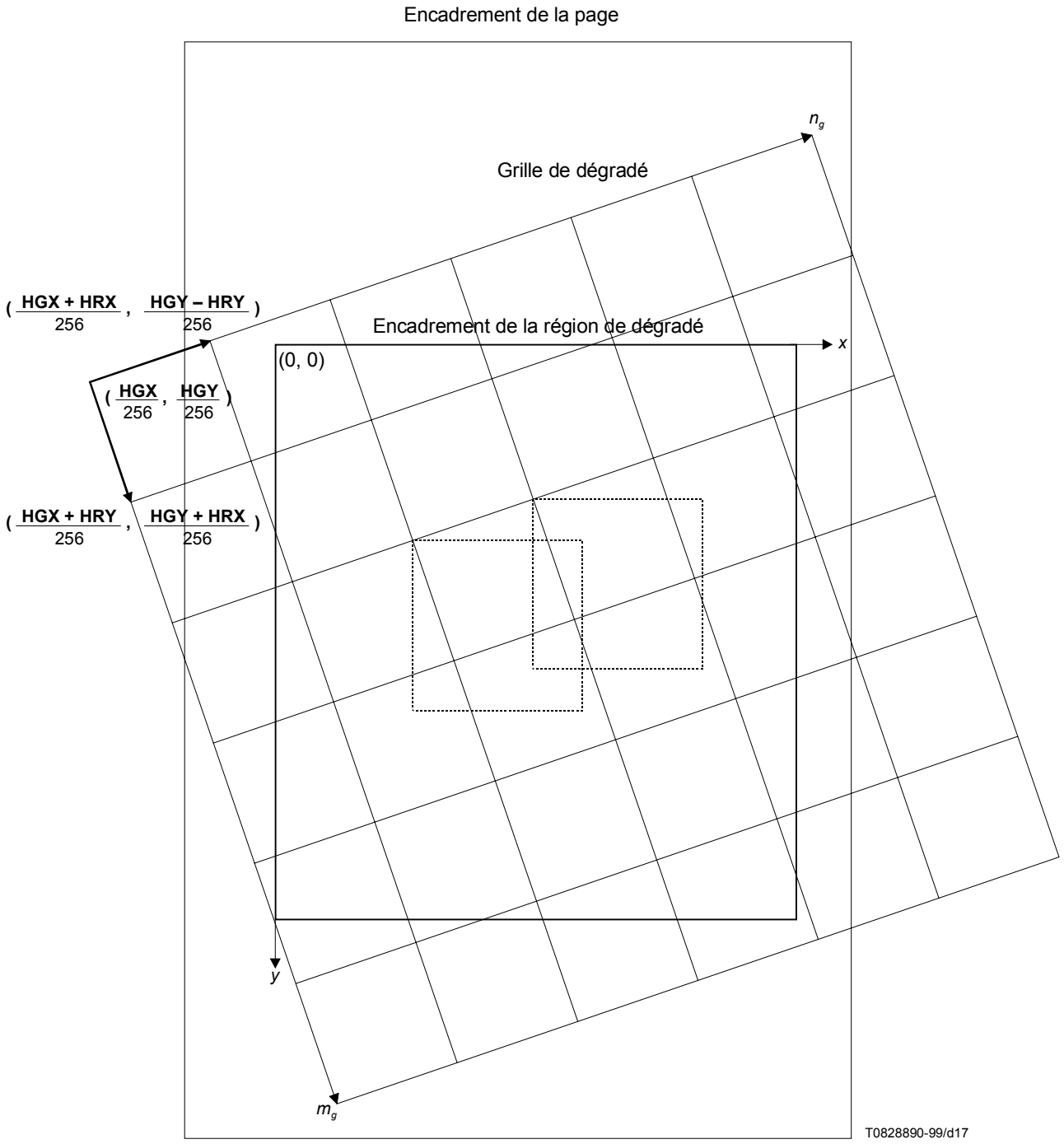


Figure 24 – Spécification des systèmes de données et des paramètres de grille

6.6.5 Décodage zonale de dégradé

Une phototrame codée en dégradés est représentée par un ensemble d'instances de structure. Chaque instance code une structure, dont l'emplacement n'est pas codé explicitement mais indiqué par une grille qui est globale par rapport à l'ensemble de la phototrame de dégradé. L'origine de la grille de dégradé est spécifiée par les paramètres **HGX** et **HGY**. Le pas de la grille est défini par les paramètres **HRX** et **HRY** (voir Figure 24). Les paramètres **HGX**, **HGY**, **HRX** et **HRY** sont normalisés par multiplication par 256, ce qui implique que l'origine et le pas de grille sont des multiples de 8 éléments binaires.

NOTE 1 – On notera que les paramètres **HRX** et **HRY** sont des valeurs non signées, c'est-à-dire toujours supérieures ou égales à zéro. Autrement dit, le vecteur de grille est toujours inscrit dans un seul quadrant. Malgré cette restriction, toute grille de dégradé peut être codée par un ajustement approprié des paramètres **HGX** et de **HGY**, qui doivent être réglés de façon que l'origine de la grille soit sur le coin de gauche. Il s'agit du coin supérieur gauche si la grille est alignée sur l'axe ou légèrement inclinée à gauche d'une grille alignée sur l'axe (comme indiqué sur la Figure 24). Il s'agit du coin inférieur gauche si la grille est légèrement inclinée à droite d'une grille alignée sur l'axe.

Les structures possibles sont énumérées dans un dictionnaire. L'identité d'une structure est spécifiée par un indice qui représentera habituellement la valeur d'échelle de gris de la structure.

NOTE 2 – Le terme *valeur d'échelle de gris* est utilisé pour l'indice afin d'expliquer le principe de la compression. La présente spécification ne prescrit pas que l'indice doive vraiment correspondre à la valeur d'échelle de gris.

Le résultat du décodage d'une matrice de dégradé est la phototrame qui est produite par les étapes suivantes:

1) utiliser la valeur **HDEFPIXEL** pour remplir une matrice HTREG, de dimensions données par **HBW** et **HBH**;

2) si **HENABLESKIP** = **1**, calculer une phototrame HSKIP comme indiqué au 6.6.5.1;

3) mettre le paramètre HBPP à la valeur $\lceil \log_2 (\mathbf{HNUMPATS}) \rceil$.

4) Décoder une image GI de dimensions **HGW** × **HGH** avec HBPP bits par pixel au moyen de la procédure de décodage d'image en échelle de gris décrite dans l'Annexe C. Régler les paramètres de cette procédure de décodage comme indiqué dans le Tableau 23.

Tableau 23 – Paramètres utilisés pour décoder une table de valeurs d'échelle de gris dans la région de dégradé

Nom	Valeur
GSMMR	**HMMR**
GSW	**HGW**
GSH	**HGH**
GSBPP	HBPP
GSUSESKIP	**HENABLESKIP**
GSKIP	HSKIP[a)]
GSTEMPLATE	**HTEMPLATE**[b)]

a) Ce paramètre n'a pas de valeur si **HENABLESKIP** = **0**.
b) Ce paramètre n'a pas de valeur si **HMMR** = **1**.

Soit l'image GI résultant de l'invocation de cette procédure de décodage;

5) insérer en séquence dans la matrice HTREG les structures correspondant aux valeurs contenues dans l'image GI par la procédure décrite au 6.6.5.2. La procédure de restitution est décrite dans la Figure 24. Les bordures des deux structures sont marquées par des cadres en pointillé;

6) une fois que toutes les structures ont été placées sur la phototrame, le contenu actuel de la matrice codée en dégradé est le résultat qui doit être obtenu par tout décodeur, qu'il suive ou non cette séquence exacte d'étapes.

NOTE 3 – Si **HGX** = 0, **HGY** = 0, **HRX** = **HPW** × 256 et **HRY** = 0, alors la grille est simple: elle est alignée sur l'axe, la direction de base est horizontale et le pas de grille est égal à la dimension des structures. Dans ce cas, il est possible d'optimiser le processus de dessin car aucune des structures ne peut se superposer à une autre.

6.6.5.1 Calcul de la valeur HSKIP

La matrice HSKIP contient le bit **1** à l'emplacement d'un pixel si le dessin d'une structure à cet emplacement de la grille de dégradé n'a pas d'incidence sur d'autres pixels de la matrice HTREG. Le calcul est le suivant:

1) pour chaque valeur de m_g comprise entre 0 et **HGH** – 1, à partir de 0, exécuter les opérations suivantes:

   a) pour chaque valeur de n_g comprise entre 0 et **HGW** – 1, à partir de 0, exécuter les opérations suivantes:

      i) poser:

$$x = (\mathbf{HGX} + m_g \times \mathbf{HRY} + n_g \times \mathbf{HRX}) >>_A 8$$
$$y = (\mathbf{HGY} + m_g \times \mathbf{HRX} - n_g \times \mathbf{HRY}) >>_A 8$$

      ii) si ((x + **HPW** ≤ 0) OR (x ≥ **HBW**) OR (y + **HPH** ≤ 0) OR (y ≥ **HBH**)) alors poser:

$$\mathrm{HSKIP}[n_g, m_g] = \mathbf{1}$$

      Sinon, poser:

$$\mathrm{HSKIP}[n_g, m_g] = \mathbf{0}$$

6.6.5.2 Restitution des structures

Dessiner les structures dans la matrice HTREG selon la procédure suivante:

1) Pour chaque valeur de m_g comprise entre 0 et **HGH** – 1, à partir de 0, exécuter les opérations suivantes:

   a) pour chaque valeur de n_g comprise entre 0 et **HGW** – 1, à partir de 0, exécuter les opérations suivantes:

      i) poser:

$$x = (\mathbf{HGX} + m_g \times \mathbf{HRY} + n_g \times \mathbf{HRX}) >> A^8$$
$$y = (\mathbf{HGY} + m_g \times \mathbf{HRX} - n_g \times \mathbf{HRY}) >> A^8$$

      ii) dessiner la structure **HPATS**[GI[n_g,m_g]] dans la matrice HTREG de façon que son pixel supérieur gauche soit à l'emplacement (x, y) dans cette matrice.

      Une structure est dessinée comme suit dans la matrice HTREG. Chaque pixel de la structure doit être combiné avec la valeur actuelle du pixel correspondant dans la phototrame de dégradé, au moyen de l'opérateur combinatoire spécifié par **HCOMBOP**. Les résultats de chaque combinaison doivent être écrits sur ce pixel de la phototrame codée en dégradé.

      Si une partie quelconque d'une structure décodée, placée à un emplacement (x, y), se trouve à l'extérieur de la matrice de dégradé actuelle, cette partie de structure doit être négligée dans le processus de combinaison de la structure avec la phototrame.

NOTE – L'image en échelle de gris peut être utilisée par le décodeur pour obtenir une bonne restitution du dégradé sur un dispositif de sortie à niveaux multiples ayant une résolution spatiale limitée, comme un écran d'ordinateur. L'utilisation de l'image en échelle de gris à cette fin est hors du domaine d'application de la présente Recommandation | Norme internationale.

L'image en échelle de gris est codée dans le plan binaire, de façon que le décodeur la reçoive progressivement. Par conséquent, le codeur peut restituer une image en dégradé en utilisant comme indices les valeurs d'échelle de gris quantifiées. De telles images à dégradé intermédiaire ne doivent pas influencer la matrice codée en dégradé finale.

6.7 Procédure de décodage du dictionnaire de structures

6.7.1 Description générale

Cette procédure de décodage est utilisée pour décoder un ensemble de structures de dimensions fixes; ces structures pourront ensuite être utilisées par les procédures de décodage de région de dégradé.

6.7.2 Paramètres d'entrée

Les paramètres de cette procédure de décodage sont indiqués dans le Tableau 24.

Tableau 24 – Paramètres pour la procédure de décodage zonale de dégradé

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
HDMMR	Entier	1	No	Indication d'utilisation du codage MMR.
HDPW	Entier	32	No	Largeur de chaque structure.
HDPH	Entier	32	No	Hauteur de chaque structure.
GRAYMAX	Entier	32	No	Valeur maximale d'échelle de gris pour laquelle une structure est indiquée.
HDTEMPLATE	Entier	2	No	Gabarit utilisé pour coder les structures[a)].
[a)] Paramètre inutilisé si **HDMMR** = **1**.				

6.7.3 Valeur de retour

La variable dont la valeur est le résultat de cette procédure de décodage est indiquée dans le Tableau 25.

Tableau 25 – Valeur de retour pour la procédure de décodage à dictionnaire de structures

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
HDPATS	Table de structures			Structures exportées par ce dictionnaire de structures. Contient **GRAYMAX** + 1 structure.

6.7.4 Variables utilisées lors du décodage

Les variables utilisées par cette procédure de décodage sont indiquées dans le Tableau 26.

Tableau 26 – Variables utilisées dans la procédure de décodage à dictionnaire de structures

Nom	Type	Longueur (bits)	Signé?	Description et restrictions
GRAY	Entier	32	N	Indice d'échelle de gris.
B_{HDC}	Phototrame			Matrice collective du dictionnaire.
B_P	Phototrame			Matrice de dimensions **HDPW** × **HDPH**.

6.7.5 Décodage du dictionnaire de structures

Le résultat du décodage d'un dictionnaire de structures est un ensemble de structures allant de **HDPATS[0]** à **HDPATS[GRAYMAX]**. Ces structures doivent être produites selon les étapes suivantes:

1) créer une phototrame B_{HDC} de hauteur **HDPH** et de largeur (**GRAYMAX** + 1) × **HDPW**. Cette phototrame contient toutes les structures concaténées de gauche à droite.

2) décoder la matrice collective par une procédure de décodage régionale générique comme celle décrite au 6.2. Régler les paramètres de cette procédure de décodage comme indiqué dans le Tableau 27.

Tableau 27 – Paramètres utilisés pour décoder une matrice collective à dictionnaire de structures

Nom	Valeur
MMR	**HDMMR**
GBW	(**GRAYMAX** + 1) × **HDPW**
GBH	**HDPH**
GBTEMPLATE	**HDTEMPLATE**[a]
TPGDON	**0**[a]
USESKIP	**0**
$\mathbf{GBATX_1}$	**–HDPW**[a]
$\mathbf{GBATY_1}$	0[a]
$\mathbf{GBATX_2}$	–3[b]
$\mathbf{GBATY_2}$	–1[b]
$\mathbf{GBATX_3}$	2[b]
$\mathbf{GBATY_3}$	–2[b]
$\mathbf{GBATX_4}$	–2[b]
$\mathbf{GBATY_4}$	–2[b]

[a] Ce paramètre n'a pas de valeur si **HDMMR** = **1**.
[b] Ce paramètre n'a pas de valeur si **HDMMR** = **1** ou si **HDTEMPLATE** ≠ 0.

3) Poser:

$$\text{GRAY} = 0$$

4) Tant que GRAY ≤ **GRAYMAX**:

   a) Soit B_P la sous-image de B_{HDC} constituée des **HPH** lignes et des colonnes **HDPW** × GRAY à **HDPW** × (GRAY + 1) – 1. Poser:

$$\textbf{HDPATS}[\text{GRAY}] = B_P$$

   b) Incrémenter:

$$\text{GRAY} = \text{GRAY} + 1$$

7 Procédure de décodage des commandes

7.1 Description générale

Cette procédure de décodage commande l'invocation de toutes les autres procédures de décodage. Le flux binaire codé se compose d'un ensemble de segments dont chacun contient une partie des données nécessaires pour le décodage. Il existe plusieurs types de segment.

Un segment se compose de deux parties: une partie d'en-tête de segment et une partie de données de segment. Tous les types de segment utilisent un format commun pour l'en-tête de segment, mais différents formats pour les données de segment.

Certains segments donnent des informations sur la structure du document: début de page, fin de page, etc. Certains segments codent des régions, qui sont à leur tour utilisées pour produire l'image décodée d'une certaine page. Certains segments ("segments de dictionnaire") ne codent pas de régions mais plutôt des ressources pouvant être utilisées par les segments qui codent les régions.

Un segment peut être associé à une certaine page ou n'être associé à aucune page. Un segment peut faire référence à d'autres segments qui le précèdent. Un segment comporte également des bits de rétention concernant le segment auquel il fait référence et le concernant lui-même. Ces bits indiquent le moment où le décodeur peut rejeter les données créées par le décodage d'un segment.

EXEMPLE – Un segment de région alphanumérique peut utiliser des symboles définis dans de précédents segments de dictionnaire de symboles. Cette utilisation est indiquée par le fait que l'en-tête du segment de région alphanumérique fait référence à ces segments de dictionnaire de symboles.

Le format des en-têtes de segment est décrit au 7.2. Les types de segment sont définis au 7.3. La syntaxe de chaque type de segment est définie au 7.4.

Certaines références sont faites, ci-dessous, à des segments "précédents" ou "suivants" (et à d'autres indications impliquant un ordonnancement des segments). Ces termes sont définis par rapport à l'ordre imposé aux segments par leur numéro de segment: un segment précède tous les segments dont le numéro de segment est plus grand que celui de ce segment. Dans les organisations séquentielles et à accès aléatoire (voir D.1 et D.2), les segments doivent apparaître dans le fichier par ordre croissant de leurs numéros de segment. Toutefois, dans l'organisation imbriquée (voir D.3), cela n'est pas le cas parce que les segments JBIG2 sont encapsulés dans un autre format de fichier.

NOTE – Il est possible qu'il y ait des lacunes dans le numérotage des segments. Un fichier JBIG2 peut contenir des segments numérotés 2, 3, 4, 8 et 10. Cela peut se produire en raison d'une révision: des numéros de segments qui pouvaient être contigus à l'origine ont été supprimés à un moment donné de la vie du fichier et les segments restants n'ont pas été renumérotés.

Une partie en-tête de segment commence et finit toujours à une limite d'octet.

Une partie données de segment commence et finit toujours à une limite d'octet. D'éventuels bits inutilisés dans l'octet final d'un segment doivent contenir **0** et ne doivent pas être examinés par le décodeur.

La partie en-tête et la partie données de segment n'ont pas besoin d'apparaître en séquence continue dans le flux binaire à décoder. Voir à l'Annexe D une organisation dans laquelle la partie en-tête de segment peut être mémorisée à une certaine distance de la partie données de ce segment.

Ce paragraphe contient des Figures qui décrivent diverses parties des données codées, comme les Figures 25 et 31. Les conventions utilisées dans ces figures sont les suivantes:

- le premier octet rencontré dans le flux binaire est à l'extrémité gauche;
- les champs dont la longueur est fixe et qui sont toujours présents sont indiqués par des traits maigres;

- les champs dont la longueur n'est pas fixe ou qui ne sont pas toujours présents, ou dont les structures sont entièrement décrites ailleurs, sont décrits par des traits gras;
- certaines figures (comme la Figure 25) sont subdivisées en champs dont chacun a une longueur égale à un nombre entier d'octets. Dans ces figures, des graduations partant du haut de la figure indiquent les limites d'octet. Les champs sont séparés par des lignes délimitant des cellules sur la hauteur de la figure;
- les autres figures sont subdivisées en champs dont chacun a une longueur égale à un nombre entier d'octets. Dans ces figures, des graduations courtes, partant du bas de la figure, montrent les limites de bit. Les champs sont séparés par des graduations plus longues partant également du bas de la figure. Chaque numéro de bit est indiqué sous la figure.

7.2 Syntaxe d'en-tête de segment

7.2.1 Champs d'en-tête de segment

Un en-tête de segment contient les champs indiqués dans la Figure 25 et décrits ci-dessous.

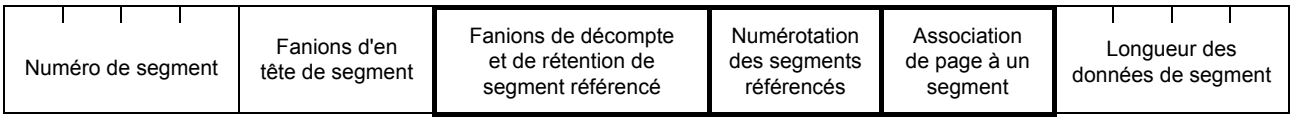


Figure 25 – Structure d'un en-tête de segment

Numéro de segment – Voir 7.2.2.

Fanions d'en-tête de segment – Voir 7.2.3.

Fanions de décompte et de rétention de segment référencé – Voir 7.2.4.

Numérotation des segments référencés – Voir 7.2.5.

Association de page à un segment – Voir 7.2.6.

Longueur des données de segment – Voir 7.2.7.

7.2.2 Numéro de segment

Ce champ de quatre octets contient le numéro du segment. Les numéros de segment valides sont compris entre 0 et 4294967295 (`0xFFFFFFFF`) inclus. Ainsi qu'il est mentionné précédemment, il est possible qu'il existe des écarts dans la numérotation des segments.

7.2.3 Fanions d'en-tête de segment

Il s'agit d'un champ d'un octet. Les bits qui sont définis sont indiqués dans la Figure 26 et sont décrits ci-dessous.

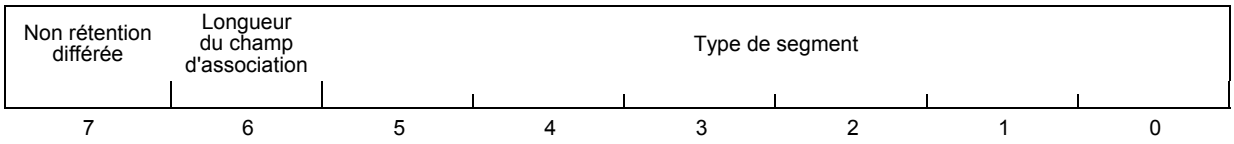


Figure 26 – Fanions d'un en-tête de segment

Bits 0 à 5 Type de segment: voir 7.3.

Bit 6 Longueur du champ d'association de page: voir 7.2.6.

Bit 7 Non-rétention différée. Si ce bit a la valeur **1**, ce segment est étiqueté comme n'étant retenu que par lui-même et par ses segments d'extension associés. Il est alors étiqueté comme n'étant pas retenu par les derniers segments d'extension associés. Un segment d'extension est dit *associé* lorsqu'il ne se rapporte qu'à un même segment. Les seuls segments (éventuellement) compris entre le segment associé et le segment référencé sont d'autres segments d'extension qui ne se réfèrent également qu'à ce segment référencé.

NOTE – L'objet de ce bit est d'indiquer au décodeur que le segment n'est référencé que par un petit nombre de segments d'extension. Le décodeur peut entreprendre des actions assez longues lorsque des segments sont étiquetés comme étant retenus; mais si cette rétention n'est faite qu'au profit des segments d'extension associés à ce segment, ces actions peuvent être inutiles et cette connaissance par avance est un avantage.

7.2.4 Fanions de décompte et de rétention de segment référencé

Ce champ contient un ou plusieurs octets indiquant combien d'autres segments sont référencés par le segment actuel et quels segments contiennent des données qui seront nécessaires après le segment actuel.

NOTE – L'on peut réduire les besoins en mémoire du décodeur en indiquant à celui-ci le moment où il est autorisé à ne plus tenir compte des données représentées par un segment antérieur.

Le nombre d'octets contenus dans ce champ dépend du nombre d'octets référencés par le segment actuel. Si celui-ci fait référence à quatre ou moins de quatre segments, ce champ a une longueur de 1 octet. S'il fait référence à plus de quatre segments, ce champ a une longueur de $4 + \lceil (R + 1)/8 \rceil$ octets, où R est le nombre de segments auquel le segment actuel fait référence.

EXEMPLE – Si le segment actuel fait référence à un nombre compris entre 5 et 7 d'autres segments, le champ a une longueur de 5 octets; s'il fait référence à un nombre compris entre 8 et 15 d'autres segments, le champ a une longueur de 6 octets.

Les trois bits de poids fort du premier octet dans ce champ déterminent la longueur de celui-ci. Si la valeur de ce sous-champ de trois bits est comprise entre 0 et 4, le champ a une longueur de 1 octet. Si la valeur de ce sous-champ de trois bits est égale à 7, le champ a une longueur d'au moins 5 octets. Ce sous-champ de 3 bits ne doit pas contenir les valeurs de 5 et 6.

Si le champ a une longueur de 1 octet, celui-ci est formaté comme indiqué dans la Figure 27 et comme décrit ci-dessous.

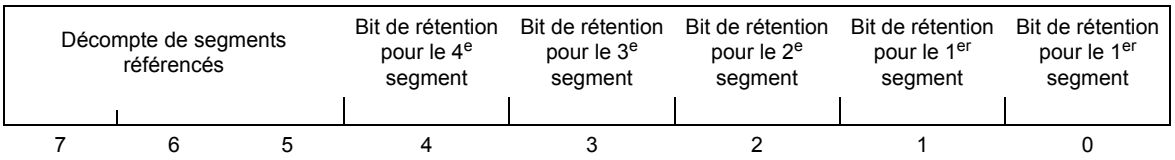


Figure 27 – Fanions de décompte et de rétention de segment référencé – forme courte

Bit 0 Bit de rétention pour le segment actuel.

Bit 1 Bit de rétention pour le premier segment référencé. Si le segment actuel ne fait référence à aucun autre segment, ce champ doit contenir **0**.

Bit 2 Bit de rétention pour le deuxième segment référencé. Si le segment actuel fait référence à moins de deux autres segments, ce champ doit contenir **0**.

Bit 3 Bit de rétention pour le troisième segment référencé. Si le segment actuel fait référence à moins de trois autres segments, ce champ doit contenir **0**.

Bit 4 Bit de rétention pour le quatrième segment référencé. Si le segment actuel fait référence à moins de quatre autres segments, ce champ doit contenir **0**.

Bits 5 à 7 Décompte de segments référencés. Ce champ peut prendre des valeurs comprises entre zéro et quatre. Il spécifie le nombre de segments auxquels le segment actuel fait référence.

Si le format de ce champ est du type long (au moins 5 octets), ce champ se compose d'un champ initial de quatre octets, suivi d'une succession de champs de 1 octet. Le champ initial de 4 octets est formaté comme suit:

Bits 0 à 28 Décompte des segments référencés, spécifiant le nombre de segments auxquels le segment actuel fait référence.

Bits 29 à 31 Indication du format de type long. Ce champ doit contenir la valeur 7.

Le premier champ de 1 octet suivant le champ initial de 4 octets est formaté comme suit:

Bit 0 Bit de rétention pour le segment actuel.

Bit 1 Bit de rétention pour le premier segment référencé.

Bit 2 Bit de rétention pour le deuxième segment référencé.

Bit 3 Bit de rétention pour le troisième segment référencé.

Bit 4 Bit de rétention pour le quatrième segment référencé.

Bit 5 Bit de rétention pour le cinquième segment référencé. Si le segment actuel fait référence à moins de 5 autres segments, ce champ doit contenir **0**.

Bit 6 Bit de rétention pour le sixième segment référencé. Si le segment actuel fait référence à moins de 6 autres segments, ce champ doit contenir **0**.

Bit 7 Bit de rétention pour le septième segment référencé. Si le segment actuel fait référence à moins de 7 autres segments, ce champ doit contenir **0**.

Le deuxième champ de 1 octet, s'il est présent, contient des bits de rétention pour les 8^e au 15^e segments référencés; les bits correspondant à tous les segments se trouvant au-delà du décompte de segments effectivement référencés doivent être **0**. Les champs de 1 octet suivants sont formatés de la même façon.

Si le bit de rétention pour la valeur du segment actuel est **0**, aucun segment ne peut faire référence à ce segment.

Si le bit de rétention pour la valeur du premier segment référencé est **0**, aucun segment après celui-ci ne peut faire référence au premier segment référencé par le segment actuel (c'est-à-dire que celui-ci est le dernier segment qui fait référence à cet autre segment). Les autres valeurs de bit de rétention ont des significations similaires: si le bit de rétention pour la valeur du K^e segment référencé est **0**, aucun segment après celui-ci ne peut faire référence au K^e segment référencé par le segment actuel.

7.2.5 Numérotation des segments référencés

Ce champ contient les numéros des segments auxquels le segment actuel fait éventuellement référence. Le nombre de valeurs contenues dans ce champ est déterminé par le champ de fanions de décompte de segments référencés et de rétention. Chaque valeur est le numéro d'un segment auquel le segment actuel fait référence. Si un segment fait référence à d'autres segments, il ne doit faire référence qu'aux segments dont le numéro est inférieur. Lorsque le numéro du segment actuel est inférieur ou égal à 256, chaque numéro de segment référencé a une longueur de 1 octet. Sinon, lorsque le numéro du segment actuel est inférieur ou égal à 65536, chaque numéro de segment référencé a une longueur de 2 octets. Sinon, chaque numéro de segment référencé a une longueur de 4 octets.

7.2.6 Association de page à un segment

Ce champ code le numéro de la page à laquelle le segment actuel appartient. La première page doit toujours être numérotée "1". Ce champ peut contenir une valeur 0, qui indique que le segment actuel n'est associé à aucune page.

Un segment qui possède une association page-segment différente de zéro ne peut être référencé que par des segments ayant la même valeur d'association page-segment que lui.

Ce champ a une longueur de 1 octet si le bit-fanion de longueur du champ d'association de page au segment actuel est **0** et il a une longueur de 4 octets si ce bit-fanion est **1**.

NOTE – Ce champ est de type court car la plupart des documents ont moins de 256 pages et les valeurs de 0 à 255 peuvent être codées sur un seul octet. Le champ d'association de page peut également n'avoir qu'une longueur de 1 octet pour les segments non associés.

7.2.7 Longueur des données de segment

Ce champ de 4 octets contient la longueur (en octets) de la partie données du segment.

Si le type du segment est "région générique immédiate", le champ de longueur peut contenir la valeur `0xFFFFFFFF`. Cette valeur vise à indiquer que la longueur de la partie données du segment n'est pas connue au moment où l'en-tête de segment est écrit (par exemple dans une application à flux continu comme la télécopie). Dans ce cas, la vraie longueur de la partie données de segment doit être déterminée par examen des données: si le segment utilise le codage arithmétique à l'échelle du gabarit, la partie données de ce segment se termine par la séquence des deux octets `0xFF 0xAC`, suivie d'un numérotage de rangée sur quatre octets. Si le segment utilise le codage MMR, sa partie données se termine par la séquence de deux octets `0x00 0x00`, suivie d'un numérotage de rangée sur quatre octets. La forme de codage utilisée par le segment peut être déterminée par l'examen du 18^e octet de sa partie données et les séquences finales peuvent apparaître à un endroit quelconque après ce 18^e octet.

NOTE – En présence d'une liste d'en-têtes de segment dans l'organisation à accès aléatoire (voir Figure D.2), un décodeur peut construire une carte du reste du fichier s'il connaît la longueur des données associées à chaque segment, ce qui lui permet d'effectuer un accès aléatoire.

7.2.8 Exemple d'en-tête de segment

EXEMPLE 1 – Un en-tête de segment composé de la séquence d'octets:

```
0x00 0x00 0x00 0x20 0x86 0x6B 0x02 0x1E 0x05 0x04
```

est analysé comme suit:

0x00 0x00 0x00 0x20 Le numéro de ce segment est 0x00000020 ou le nombre décimal 32.

0x86 Le type de ce segment est 6. Son champ d'association de page a une longueur de 1 octet. Il n'est retenu que par ses seuls segments d'extension associés.

0x6B Ce segment fait référence à trois autres segments. Il est référencé par un autre segment. Il s'agit de la dernière référence au deuxième des trois segments auxquels il fait référence.

0x02 0x1E 0x05 Les trois segments auxquels il fait référence ont les numéros 2, 30 et 5.

0x04 Ce segment est associé à la page numéro 4.

EXEMPLE 2 – Un en-tête de segment composé de la séquence d'octets hexadécimaux suivante:

```
00  00  02  34  40  E0  00  00  09  02  FD  01  00  00  02  00
1E  00  05  02  00  02  01  02  02  02  03  02  04  00  00  04
01
```

est analysé comme suit:

00 00 02 34 Le numéro de ce segment est 0x00000234 soit le nombre décimal 564.

40 Le type de ce segment est 0. Son champ d'association de page a une longueur de 4 octets.

E0 00 00 09 Le champ de comptage de segment référencé par ce segment est de type long. Ce segment fait référence à 9 autres segments.

02 FD Ce segment est référencé par un autre segment. Il s'agit de la dernière référence au 1er et au 8e des 9 segments auxquels il fait référence.

01 00 ... 02 04 Les 9 segments auxquels il fait référence sont chacun identifiés par 2 octets car le numéro de ce segment est compris entre 256 et 65535. Les segments auxquels il fait référence ont les numéros décimaux 256, 2, 30, 5, 512, 513, 514, 515 et 516.

00 00 04 01 Ce segment est associé à la page numéro 1025.

7.3 Types de segment

Chaque segment est d'un certain type. Ce type spécifie la sorte des données associées au segment. Il limite le nombre des autres segments auxquels il peut faire référence et par lesquels il peut être référencé. Ces restrictions sont détaillées au 7.3.1.

Le type de segment est un nombre compris entre 0 et 63, inclus. Toutes les valeurs ne sont pas autorisées. La liste des valeurs autorisées des types de segment, leurs noms exacts et le paragraphe définissant leur format sont les suivants:

0 Dictionnaire de symboles – voir 7.4.2.

4 Région alphanumérique intermédiaire – voir 7.4.3.

6 Région alphanumérique immédiate – voir 7.4.3.

7 Région alphanumérique immédiate sans pertes – voir 7.4.3.

16 Dictionnaire de structures – voir 7.4.4.

20 Région de dégradé intermédiaire – voir 7.4.5.

22 Région de dégradé immédiate – voir 7.4.5.

23 Région de dégradé immédiate sans pertes – voir 7.4.5.

36 Région générique intermédiaire – voir 7.4.6.

38 Région générique immédiate – voir 7.4.6.

39 Région générique immédiate sans pertes – voir 7.4.6.

40 Région générique par raffinement intermédiaire – voir 7.4.7.

42 Région générique par raffinement immédiate – voir 7.4.7.

43 Région générique par raffinement immédiate sans pertes – voir 7.4.7.

48 Informations de page – voir 7.4.8.

49 Fin de page – voir 7.4.9.

50 Fin de bande – voir 7.4.10.

51 Fin de fichier – voir 7.4.11.

52 Profils – voir 7.4.12.

53 Tables – voir 7.4.13.

62 Extension – voir 7.4.14.

Tous les autres types de segment sont réservés et ne doivent pas être utilisés.

NOTE – Ces numéros de type de segment sont attribués conformément aux règles suivantes. Les deux bits de poids fort (bits 4 et 5) de ce numéro spécifient le type primaire du segment et les quatre bits de poids faible (bits 0 à 3) spécifient le type secondaire du segment.

Les types primaires sont les suivants:

0 Données de matrice de symbole

1 Données de matrice de dégradé

2 Données de matrice générique

3 Métadonnées

Les types primaires 0 à 2 sont collectivement désignés par l'expression types de région.

Pour les types de région, l'interprétation des 4 bits de poids faible est la suivante:

Bit 0 Si ce bit est **1**, il indique que le segment rend sans pertes une région de la page.

Bit 1 Si ce bit est **1**, il indique que le segment peut être inséré directement dans la matrice de page. Si ce bit est **0**, il indique que le segment est de type intermédiaire. Voir 8.2.

Bits 2 et 3 Ces deux bits définissent un sous-type du type primaire:

0 Dictionnaire
1 Région directe
2 Région de raffinement

Pour les métadonnées l'interprétation des quatre bits de poids faible est la suivante:

0 Information de page

1 Fin de page

2 Fin de bande

3 Fin de fichier

4 Profils

5 Tables

6-13 Champs réservés

14 Extension

15 Champ réservé

On désigne par le terme collectif de *segments de région* les segments de type "région alphanumérique intermédiaire", "région alphanumérique immédiate", "région alphanumérique immédiate sans pertes", "région de dégradé intermédiaire", "région de dégradé immédiate", "région de dégradé immédiate sans perte", région générique intermédiaire", "région générique immédiate", "région générique immédiate sans pertes", "région générique par raffinement intermédiaire", "région générique par raffinement immédiate" et "région générique par raffinement, immédiate sans pertes".

On désigne par le terme collectif de *segments de région directe* les segments de type "région alphanumérique intermédiaire", "région alphanumérique immédiate", "région alphanumérique immédiate sans pertes", "région de dégradé intermédiaire", "région de dégradé immédiate", "région de dégradé immédiate sans perte", "région générique intermédiaire", "région générique immédiate" et "région générique immédiate sans pertes".

On désigne par le terme collectif de *segments de région intermédiaire* les segments de type "région alphanumérique intermédiaire", "région de dégradé intermédiaire", "région générique intermédiaire" et "région générique par raffinement intermédiaire".

On désigne par le terme collectif de *segments de région immédiate* les segments de type "région alphanumérique immédiate", "région alphanumérique immédiate sans pertes", "région de dégradé immédiate", "région de dégradé immédiate sans perte", "région générique immédiate", "région générique immédiate sans pertes", "région générique par raffinement immédiate" et "région générique par raffinement immédiate sans pertes".

On désigne par le terme collectif de *segments de région de raffinement* les segments de type "région générique par raffinement intermédiaire", "région générique par raffinement immédiate" et "région générique par raffinement immédiate sans pertes".

7.3.1 Règles de référenciation aux segments

Les règles de référenciation aux segments sont les suivantes:

- un segment de région intermédiaire ne peut être référencé que par un seul autre segment qui n'est pas d'extension; il peut être référencé par un nombre quelconque de segments d'extension;
- un segment de type "dictionnaire de symboles" (type 0) peut faire référence à un nombre quelconque de segments de type "dictionnaire de symboles" et à un maximum de 4 segments de type "table";
- un segment de type "région alphanumérique intermédiaire", "région alphanumérique immédiate" ou "région alphanumérique immédiate sans pertes" (type 4, 6 ou 7) peut faire référence à un nombre quelconque de segments de type "dictionnaire de symboles" et à un maximum de 8 segments de type "table";
- un segment de type "dictionnaire de structures" (type 16) ne doit faire référence à aucun autre segment;
- un segment de type "région de dégradé intermédiaire", "région de dégradé immédiate" ou "région de dégradé immédiate sans pertes" (type 20, 22 ou 23) doit faire référence à exactement un seul segment, qui doit être du type "dictionnaire de symboles";
- un segment de type "région générique intermédiaire", "région générique immédiate" ou "région générique immédiate sans pertes" (type 36, 38 ou 39) ne doit faire référence à aucun autre segment;
- un segment de type "région générique par raffinement intermédiaire" (type 40) doit faire référence à exactement un seul segment, qui doit être un segment de région intermédiaire;
- un segment de type "région générique par raffinement immédiate" ou "région générique par raffinement immédiate sans pertes" (type 42 ou 43) peut faire référence soit à zéro soit à exactement un seul autre segment. S'il fait référence à un seul autre segment, celui-ci doit être de type "région intermédiaire";
- un segment de type "information de page" (type 48) ne doit faire référence à aucun autre segment;
- un segment de type "fin de page" (type 49) ne doit faire référence à aucun autre segment;
- un segment de type "fin de bande" (type 50) ne doit faire référence à aucun autre segment;
- un segment de type "fin de fichier" (type 51) ne doit faire référence à aucun autre segment;
- un segment de type "profils" (type 52) ne doit faire référence à aucun autre segment;
- un segment de type "tables" (type 53) ne doit faire référence à aucun autre segment;
- un segment de type "extension" (type 62) peut faire référence à un nombre quelconque de segments d'un type quelconque, à moins que le type de segment d'extension n'impose des restrictions.

7.3.2 Règles d'association aux pages

Chaque segment de région doit être associé à une certaine page (c'est-à-dire avoir un champ d'association de page non nul). Les segments de types "information de page", "fin de page" et "fin de bande" doivent toujours être associés à une page. Un segment de type "fin de fichier" ne doit jamais être associé à une page. Les segments des autres types peuvent être associés à une page ou ne pas l'être.

Si un segment n'est associé à aucune page, il ne doit faire référence à aucun segment associé à une page quelconque.

Si un segment est associé à une page, il peut faire référence à des segments qui ne sont pas associés à une page et à des segments qui sont associés à la même page. Il ne doit pas faire référence à un segment associé à une page différente.

7.4 Syntaxes des segments

Ce paragraphe décrit en détail la syntaxe de la partie données de segment pour chaque type de segment, ainsi que la façon dont elle doit être décodée.

7.4.1 Champ d'information de segment de région

Chaque partie données d'un segment de région commence par un champ d'information de segment de région, dont le format est spécifié ci-dessous. Un champ d'information de segment de région contient les sous-champs suivants, comme indiqué sur la Figure 28 et décrit ci-dessous.

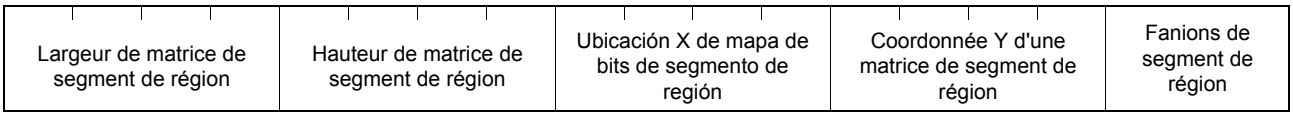


Figure 28 – Structure d'en-tête de données de segment de région

Largeur de matrice de segment de région – Voir 7.4.1.1.

Hauteur de matrice de segment de région – Voir 7.4.1.2.

Coordonnée X d'une matrice de segment de région – Voir 7.4.1.3.

Coordonnée Y d'une matrice de segment de région – Voir 7.4.1.4.

Fanions de segment de région – Voir 7.4.1.5.

7.4.1.1 Largeur de matrice de segment de région

Ce champ de 4 octets indique en pixels la largeur de la phototrame codée dans ce segment.

7.4.1.2 Hauteur de matrice de segment de région

Ce champ de 4 octets indique en pixels la hauteur de la phototrame codée dans ce segment.

7.4.1.3 Coordonnée X d'une matrice de segment de région

Ce champ de 4 octets indique en pixels le décalage horizontal de la phototrame codée dans ce segment par rapport à la matrice de page.

7.4.1.4 Coordonnée Y d'une matrice de segment de région

Ce champ de 4 octets indique en pixels le décalage vertical de la phototrame codée dans ce segment par rapport à la matrice de page.

7.4.1.5 Fanions de segment de région

Ce champ de 1 octet est formaté comme indiqué dans la Figure 29 et comme décrit ci-dessous.

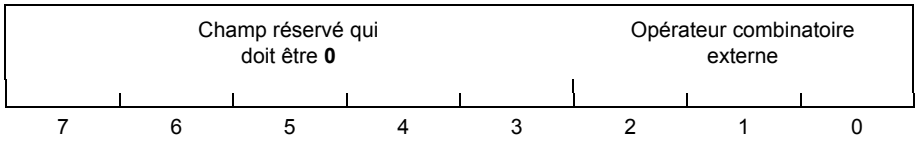


Figure 29 – Structure du champ des fanions d'un segment de région

Bits 0 à 2 Opérateur combinatoire externe. Ce champ de 3 bits peut prendre les valeurs suivantes, qui représentent un des 5 opérateurs combinatoires possibles:

0 OR
1 AND
2 XOR
3 XNOR
4 REPLACE

NOTE 1 – Ces opérateurs décrivent la façon dont la matrice du segment doit être combinée avec celle de la page. L'opérateur REPLACE est destiné à être utilisé par les régions raffinement, qui remplacent les régions qu'elles raffinent. Les opérateurs tels que AND peuvent servir pour le masquage lorsqu'une partie de la matrice de page, qui contient déjà des données, doit être effacée afin qu'une autre phototrame puisse y être écrite (par exemple pour écrire une matrice à travers un masque).

NOTE 2 – Les segments de région intermédiaire ne sont jamais combinés directement avec la page, de sorte qu'on n'utilise pas leurs coordonnées et leurs opérateurs combinatoires externes. Ces valeurs peuvent toutefois rester utiles: si un décodeur souhaite dessiner une version de la page avant que tous les segments aient été décodés (pour un balayage progressif), ce décodeur peut vouloir restituer les segments intermédiaires, Pour faciliter la production par le décodeur d'une séquence utile de raffinements progressifs de la page, il peut être utile de régler les cordonnées et les opérateurs combinatoires externes en fonction du mode de combinaison avec la page du raffinement final de ce segment intermédiaire.

Bits 3 à 7 Champ réservé; doivent être mis à **0**.

En d'autres termes, ce champ d'information de segment de région décrit les dimensions et l'emplacement de la matrice codée dans ce segment.

EXEMPLE – Si les valeurs de dimensions et d'emplacement sont (dans l'ordre) 100, 200, 50 et 75, ce segment décrit une phototrame de 100 pixels de large et de 200 pixels de haut, dont le coin supérieur gauche est à 50 pixels à droite et à 75 pixels au-dessous du coin supérieur gauche de la page.

7.4.2 Syntaxe du segment de dictionnaire de symboles

7.4.2.1 En-tête de données de segment de dictionnaire de symboles

Une partie données de segment de dictionnaire de symboles commence par un en-tête de données de segment de dictionnaire de symboles, contenant les champs indiqués dans la Figure 30 et décrits ci-dessous.

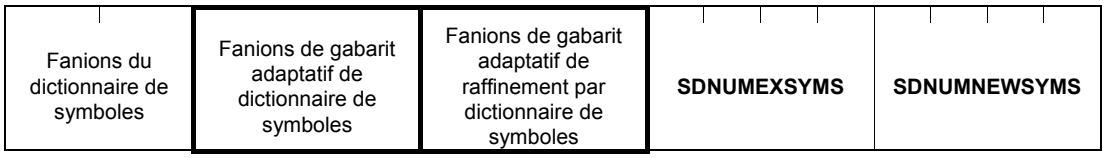


Figure 30 – Structure d'en-tête de données de segment de dictionnaire de symboles

Fanions du dictionnaire de symboles – Voir 7.4.2.1.1.

Fanions de gabarit adaptatif de dictionnaire de symboles – Voir 7.4.2.1.2.

Fanions de gabarit adaptatif de raffinement du dictionnaire de symboles – Voir 7.4.2.1.3.

SDNUMEXSYMS – Voir 7.4.2.1.4.

SDNUMNEWSYMS – Voir 7.4.2.1.5.

7.4.2.1.1 Fanions du dictionnaire de symboles

Ce champ de 2 octets est formaté comme indiqué sur la Figure 31 et comme décrit ci-dessous.

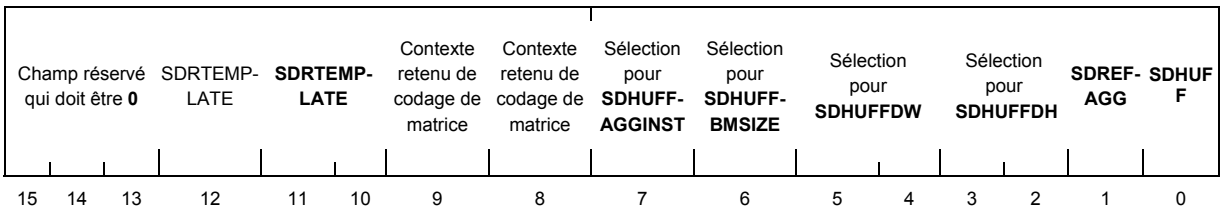


Figure 31 – Structure du champ des fanions AT d'un dictionnaire de symboles

Bit 0 **SDHUFF**

Si ce bit est **1**, le segment utilise la variante de codage de Huffman. S'il est **0**, le segment utilise la variante de codage arithmétique. Le réglage de ce fanion détermine comment les données de ce segment sont codées. Il peut également modifier l'ordre dans lequel certaines des données sont codées.

Bit 1 **SDREFAGG**

Si ce bit est **0**, aucun codage de raffinement ou d'agrégation n'est utilisé dans ce segment. S'il est **1**, chaque matrice de symbole est codée par raffinement ou agrégation.

Bits 2 et 3 Sélection de **SDHUFFDH**. Ce champ de 2 bits peut prendre une des trois valeurs suivantes pour indiquer la table qui doit être utilisée pour **SDHUFFDH**.

0 Table B.4
1 Table B.5
3 Table fournie par l'utilisateur.

La valeur 2 n'est pas permise.

Si **SDHUFF = 0**, ce champ doit contenir la valeur **0**.

Bits 4 et 5 Sélection de **SDHUFFDW**. Ce champ de 2 bits peut prendre une des trois valeurs suivantes pour indiquer la table qui doit être utilisée pour **SDHUFFDW**.

0 Table B.2
1 Table B.3
3 Table fournie par l'utilisateur

La valeur 2 n'est pas permise.

Si **SDHUFF** = **0**, ce champ doit contenir la valeur 0.

Bit 6 Sélection de **SDHUFFBMSIZE**.

Si ce champ est **0**, la table B.1 est utilisée pour **SDHUFFBMSIZE**. S'il est **1**, une table fournie par l'utilisateur est utilisée pour **SDHUFFBMSIZE**.

Si **SDHUFF = 0**, ce champ doit contenir la valeur **0**.

Bit 7 Sélection de **SDHUFFAGGINST**.

Si ce champ est **0**, la table B.1 est utilisée pour **SDHUFFAGGINST**. S'il est **1**, une table fournie par l'utilisateur est utilisée pour **SDHUFFAGGINST**.

Si **SDHUFF = 0** ou si **SDREFAGG = 0**, ce champ doit contenir la valeur **0**.

Bit 8 Contexte utilisé pour le codage de matrice.

Si **SDHUFF = 1** et si **SDREFAGG = 0**, ce champ doit contenir la valeur **0**.

Bit 9 Contexte retenu pour le codage de matrice.

Si **SDHUFF = 1** et si **SDREFAGG = 0**, ce champ doit contenir la valeur **0**.

Bits 10 et 11 **SDTEMPLATE**.

Ce champ commande le gabarit utilisé pour décoder les matrices de symbole si **SDHUFF = 0**. Si **SDHUFF = 1**, ce champ doit contenir la valeur **0**.

Bit 12 **SDRTEMPLATE**.

Ce champ commande le gabarit utilisé pour décoder les matrices de symbole si **SDREFAGG = 1**. Si **SDREFAGG = 0**, ce champ doit contenir la valeur **0**.

Bits 13 à 15 Champ réservé qui doit être **0**.

7.4.2.1.2 Fanions de gabarit adaptatif de dictionnaire de symboles

Ce champ n'est présent que si **SDHUFF = 0**. Si **SDTEMPLATE = 0**, ce champ est codé sur 8 octets formatés comme indiqué sur la Figure 32 et comme décrit ci-dessous.

$SDATX_1$	$SDATY_1$	$SDATX_2$	$SDATY_2$	$SDATX_3$	$SDATY_3$	$SDATX_4$	$SDATY_4$

Figure 32 – Structure du champ des fanions AT d'un dictionnaire de symboles lorsque SDTEMPLATE = 0

Octet 0 **$SDATX_1$**

Octet 1 **$SDATY_1$**

Octet 2 **$SDATX_2$**

Octet 3 **$SDATY_2$**

Octet 4 **$SDATX_3$**

Octet 5 **$SDATY_3$**

Octet 6 **$SDATX_4$**

Octet 7 **$SDATY_4$**

Si **SDTEMPLATE** = 1, 2 ou 3, ce champ est codé sur 2 octets comme indiqué sur la Figure 33 et comme décrit ci-dessous.

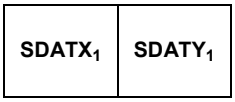


Figure 33 – Structure du champ des fanions AT d'un dictionnaire de symboles lorsque SDTEMPLATE n'est pas égal à 0

Octet 0 **$SDATX_1$**

Octet 1 **$SDATY_1$**

Si **SDTEMPLATE** = 1, 2 ou 3, les valeurs de **$SDATX_2$** à **$SDATX_4$** et de **$SDATY_2$** à **$SDATY_4$** sont toutes égales à 0.

Les champs des coordonnées X et Y du gabarit adaptatif contiennent des valeurs signées qui sont permises conformément à la Figure 7.

7.4.2.1.3 Fanions de gabarit adaptatif de raffinement du dictionnaire de symboles

Ce champ n'est présent que si **SDREFAGG = 1** et **SDRTEMPLATE = 0**. Il est codé sur 4 octets, formatés comme indiqué sur la Figure 34 et comme décrit ci-dessous.

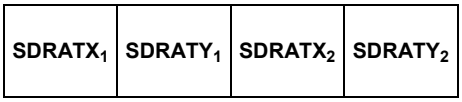


Figure 34 – Structure du champ des fanions AT de raffinement d'un dictionnaire de symboles

Octet 0 **$SDRATX_1$**

Octet 1 **$SDRATY_1$**

Octet 2 **$SDRATX_2$**

Octet 3 **$SDRATY_2$**

Les champs des coordonnées X et Y du gabarit adaptatif contiennent des valeurs signées qui sont permises conformément au 6.3.5.3.

7.4.2.1.4 Nombre de symboles exportés (SDNUMEXSYMS)

Ce champ de 4 octets contient le nombre de symboles exportés du dictionnaire actuel.

Pour le décodeur, il est très utile de pouvoir trouver facilement le nombre de symboles présents, par exemple pour attribuer une séquence de structures avant de commencer le décodage du dictionnaire.

7.4.2.1.5 Nombre de symboles nouveaux (SDNUMNEWSYMS)

Ce champ de 4 octets contient le nombre de symboles définis dans le dictionnaire actuel.

NOTE – **SDNUMEXSYMS** et **SDNUMNEWSYMS** sont souvent, mais pas toujours, de valeur égale. Par exemple, si un dictionnaire réexporte certains des symboles qu'il a importés de dictionnaires auxquels il fait référence, ce dictionnaire copie effectivement ces symboles. Ceux-ci sont décrits par **SDNUMEXSYMS** mais non par **SDNUMNEWSYMS**. Une autre différence peut également provenir de la possibilité qu'un dictionnaire définisse certains symboles qu'il n'exporte pas.

7.4.2.1.6 Sélection des tables de Huffman pour le segment de dictionnaire de symboles

Régler les valeurs des paramètres **SDHUFFDH**, **SDHUFFDW**, **SDHUFFBMSIZE** et **SDHUFFAGGINST** en fonction des champs de sélection indiqués au 7.4.2.1.1 et des segments de table référencés par le segment actuel. Plus précisément, parmi ces 4 tables de Huffman, certaines peuvent être spécifiées pour utiliser des tables normalisées et d'autres pour utiliser des tables fournies par l'utilisateur. Le numéro spécifié pour utiliser une table fournie par l'utilisateur doit être égal au numéro des segments de table auxquels le segment actuel fait référence. Ces segments de table sont mis en correspondance avec les tables de Huffman utilisant des tables fournies par l'utilisateur conformément à l'ordre dans lequel les segments de table sont référencés et à l'ordre suivant:

1) **SDHUFFDH**
2) **SDHUFFDW**
3) **SDHUFFBMSIZE**
4) **SDHUFFAGGINST**

Si une table spécifiée par l'utilisateur est utilisée pour **SDHUFFDW**, cette table doit pouvoir coder la valeur hors bande (OOB). Si une table spécifiée par l'utilisateur est utilisée pour **SDHUFFDH**, **SDHUFFBMSIZE** ou **SDHUFFAGGINST**, cette table ne doit pas pouvoir coder la valeur hors bande (OOB).

EXEMPLE – Si **SDHUFFDH** et **SDHUFFAGGINST** sont spécifiés pour utiliser des tables fournies par l'utilisateur et si **SDHUFFDW** et **SDHUFFBMSIZE** sont spécifiés pour utiliser des tables normalisées (Tableaux B.2 et B.1 respectivement), ce segment doit faire référence à exactement deux segments de table; le segment de table auquel il est fait référence en premier est utilisé pour **SDHUFFDH** et le segment de table auquel il est fait référence en second est utilisé pour **SDHUFFAGGINST**.

7.4.2.2 Décodage d'un segment de dictionnaire de symboles

Un segment de dictionnaire de symboles est décodé selon les étapes suivantes:

1) interpréter son en-tête comme décrit au 7.4.2.1;
2) décoder (ou extraire les résultats de décodage) tous segments de dictionnaire et de table éventuellement référencés;
3) si le bit "contexte de codage de matrice utilisé" contenu dans l'en-tête est **1**, alors, comme décrit au E.3.8, donner aux statistiques de codage arithmétique, pour les procédures de décodage régionale générique et de région générique par raffinement, les valeurs que ces statistiques contenaient à la fin du décodage du dernier segment de dictionnaire de symboles référencé. Cet en-tête de données de segment de dictionnaire de symboles doit avoir le bit "contexte de codage de matrice retenu" égal à **1**. Les valeurs des paramètres **SDHUFF**, **SDREFAGG**, **SDTEMPLATE**, **SDRTEMPLATE** et tous les emplacements AT (aussi bien directs que de raffinement) pour ce dictionnaire de symboles doivent correspondre aux valeurs analogues extraites du dictionnaire de symboles dont les valeurs de contexte sont en cours d'utilisation;
4) si le bit "contexte de codage de matrice utilisé" a la valeur **0** dans l'en-tête, alors, comme décrit au E.3.7, remettre à zéro toutes les statistiques de codage arithmétique pour les procédures de décodage régionale générique et régionale générique par raffinement;
5) remettre à zéro les statistiques de codage arithmétique pour tous les contextes de tous les codeurs arithmétiques d'entier;
6) invoquer la procédure de décodage par dictionnaire de symboles décrite au 6.5, après réglage des paramètres de la procédure de décodage par dictionnaire de symboles comme indiqué dans le Tableau 28;
7) si le bit "contexte de codage de matrice retenu" a la valeur **1** dans l'en-tête, alors, comme décrit au E.3.8, préserver le contenu actuel des statistiques de codage arithmétique pour les procédures de décodage régionale générique et régionale générique par raffinement.

NOTE – L'étape 3) est destinée à réduire les frais de codage des dictionnaires de symboles. Un effet secondaire du décodage d'un dictionnaire de symboles est que les statistiques de codage arithmétique utilisées pour les phototrames de codage "apprennent" les statistiques probables des symboles contenus dans ce dictionnaire de symboles. Ces deux étapes [3) et 7)] permettent une certaine réutilisation limitée de ces statistiques: les statistiques apprises lors du décodage du dictionnaire de symboles qui est le dernier dictionnaire de ce type à être référencé sont utilisées comme point de départ pour le décodage de ce dictionnaire de symboles.

L'étape 7) est explicitement présente parce que les statistiques de codage arithmétique de dictionnaire de symboles ne seront pas toutes utilisées par un autre dictionnaire. Le fait de savoir qu'elles ne seront pas utilisées permet au décodeur de les éliminer, ce qui réduit le taux d'utilisation de la mémoire.

Tableau 28 – Paramètres utilisés pour décoder un segment de dictionnaire de symboles

Nom	Valeur
SDHUFF	Comme indiqué au 7.4.2.1.1
SDREFAGG	Comme indiqué au 7.4.2.1.1
SDNUMINSYMS	Nombre total de symboles exportés à partir de tous les segments de dictionnaire de symboles référencés par le segment actuel.
SDINSYMS	Concaténation des tables de symboles exportés à partir de tous les segments de dictionnaire de symboles référencés par le segment actuel, dans l'ordre de leur référenciation.
SDNUMNEWSYMS	Comme indiqué au 7.4.2.1.5.
SDNUMEXSYMS	Comme indiqué au 7.4.2.1.4
SDHUFFDH	Voir 7.4.2.1.6
SDHUFFDW	Voir 7.4.2.1.6
SDHUFFBMSIZE	Voir 7.4.2.1.6
SDHUFFAGGINST	Voir 7.4.2.1.6
SDTEMPLATE	Voir 7.4.2.1.1
$\mathbf{SDATX_1}$	Voir 7.4.2.1.2
$\mathbf{SDATY_1}$	Voir 7.4.2.1.2
$\mathbf{SDATX_2}$	Voir 7.4.2.1.2
$\mathbf{SDATY_2}$	Voir 7.4.2.1.2
$\mathbf{SDATX_3}$	Voir 7.4.2.1.2
$\mathbf{SDATY_3}$	Voir 7.4.2.1.2
$\mathbf{SDATX_4}$	Voir 7.4.2.1.2
$\mathbf{SDATY_4}$	Voir 7.4.2.1.2
SDRTEMPLATE	Voir 7.4.2.1.1
$\mathbf{SDRATX_1}$	Voir 7.4.2.1.3
$\mathbf{SDRATY_1}$	Voir 7.4.2.1.3
$\mathbf{SDRATX_2}$	Voir 7.4.2.1.3
$\mathbf{SDRATY_2}$	Voir 7.4.2.1.3

7.4.3 Syntaxe d'un segment de région alphanumérique

Les parties données des trois types de segment de région alphanumérique ("région alphanumérique intermédiaire", "région alphanumérique immédiate" et "région alphanumérique immédiate sans pertes") sont codées de façon identique mais sont traitées différemment (voir 8.2). La syntaxe de ces parties données des types de segment est spécifiée ci-après.

7.4.3.1 En-tête de données de segment de région alphanumérique

La partie données d'un segment de région alphanumérique commence par un en-tête de données de segment de région alphanumérique. Cet en-tête contient les champs indiqués dans la Figure 35 et décrits ci-dessous.

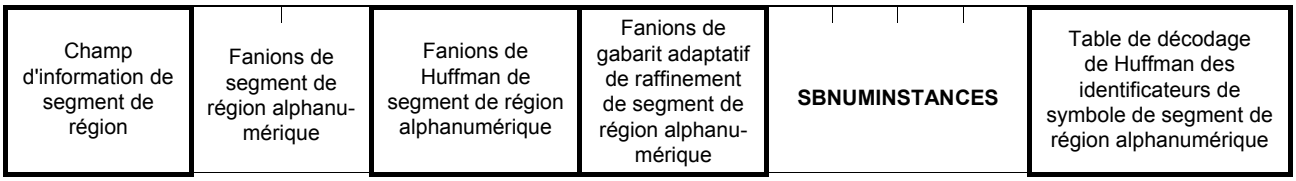


Figure 35 – Structure d'en-tête de données de segment de région alphanumérique

Champ d'information de segment de région – Voir 7.4.1.

Fanions de segment de région alphanumérique – Voir 7.4.3.1.1.

Fanions de Huffman de segment de région alphanumérique – Voir 7.4.3.1.2.

Fanions de gabarit adaptatif de raffinement de segment de région alphanumérique – Voir 7.4.3.1.3.

SBNUMINSTANCES – Voir 7.4.3.1.4.

Table de décodage de Huffman des identificateurs de symbole de segment de région alphanumérique – Voir 7.4.3.1.5.

7.4.3.1.1 Fanions de segment de région alphanumérique

Ce champ de deux octets est formaté comme indiqué sur la Figure 36 et comme décrit ci-dessous.

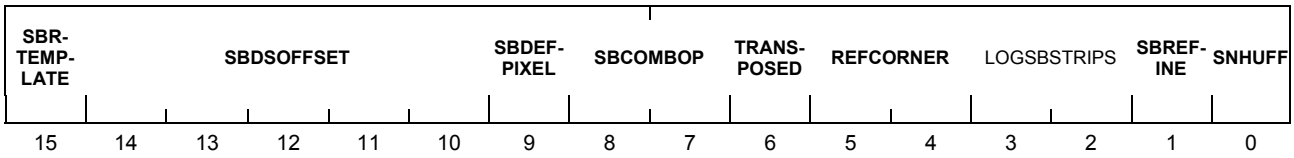


Figure 36 – Structure du champ des fanions de région alphanumérique

Bit 0 **SBHUFF**

Si ce bit est **1**, le segment utilise la variante de codage de Huffman. S'il est **0**, le segment utilise la variante de codage arithmétique. La position de ce fanion détermine la façon dont les données de ce segment sont codées.

Bit 1 **SBREFINE**

Si ce bit est **0**, le segment ne contient pas de raffinements d'instance de symbole. S'il est **1**, le segment peut contenir des raffinements d'instance de symbole.

Bits 2 et 3 LOGSBSTRIPS

Ce champ de 2 bits code le logarithme de base 2 de la dimension de bande à coder. Les dimensions de bande 1, 2, 4 et 8 peuvent donc être codées.

Bits 4 et 5 **REFCORNER**. Les 4 valeurs possibles de ce champ de 2 bits sont les suivantes:

0 BOTTOMLEFT
1 TOPLEFT
2 BOTTOMRIGHT
3 TOPRIGHT

NOTE – La meilleure compression est habituellement obtenue lorsque le point de référence de chaque symbole se trouve sur la ligne de base du texte. Etant donné que le texte peut s'écrire dans un quelconque de 8 sens, il faut une certaine flexibilité pour déterminer le coin qui sera utilisé comme point de référence d'un symbole donné.

Bit 6 **TRANSPOSED**

Si ce bit = **1**, le sens primaire du codage est de haut en bas. S'il est = **0**, le sens primaire du codage est de gauche à droite, ce qui permet au texte de s'écrire de haut en bas de la page.

Bits 7 et 8 **SBCOMBOP**. Ce champ a 4 valeurs possibles, qui représentent 1 opérateur combinatoire possible sur 4:

0 OR
1 AND
2 XOR
3 XNOR

Bit 9 **SBDEFPIXEL**

Ce bit contient la valeur initiale pour chaque pixel contenu dans la région alphanumérique, avant que d'éventuels symboles soient dessinés.

Bits 10 à 14 **SBDSOFFSET**

Ce champ de cinq bits signé contient la valeur de **SBDSOFFSET** – voir 6.4.8.

Bit 15 **SBRTEMPLATE**

Ce champ commande le gabarit utilisé pour décoder les raffinements d'instance de symbole si **SBREFINE** = **1**. Si **SBREFINE** = **0**, ce champ doit contenir la valeur **0**.

7.4.3.1.2 Fanions de Huffman de segment de région alphanumérique

Ce champ n'est présent que si **SBHUFF** = **1**.

Ce champ de 2 octets est formaté comme indiqué dans la Figure 37 et comme décrit ci-dessous.

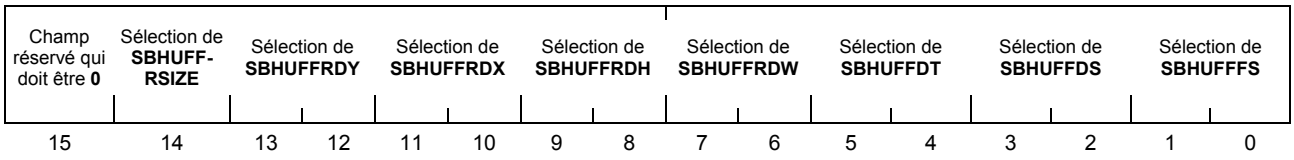


Figure 37 – Structure du champ des fanions de Huffman de région alphanumérique

Bits 0 et 1 Sélection de **SBHUFFFS.** Ce champ de 2 bits peut prendre une des trois valeurs suivantes, qui indiquent la table à utiliser pour **SBHUFFFS**.

0 Table B.6
1 Table B.7
3 Table fournie par l'utilisateur.

La valeur 2 n'est pas permise.

Bits 2 et 3 Sélection de **SBHUFFDS**. Ce champ de 2 bits peut prendre une des quatre valeurs suivantes, qui indiquent la table à utiliser pour **SBHUFFDS**.

0 Table B.8
1 Table B.9
2 Table B.10
3 Table fournie par l'utilisateur.

Bits 4 et 5 Sélection de **SBHUFFDT.** Ce champ de 2 bits peut prendre une des quatre valeurs suivantes, qui indiquent la table à utiliser pour **SBHUFFDT**.

0 Table B.11
1 Table B.12
2 Table B.13
3 Table fournie par l'utilisateur.

Bits 6 et 7 Sélection de **SBHUFFRDW**. Ce champ de 2 bits peut prendre une des trois valeurs suivantes, qui indiquent la table à utiliser pour **SBHUFFRDW**.

0 Table B.14
1 Table B.15
3 Table fournie par l'utilisateur.

La valeur 2 n'est pas permise. Si **SBREFINE** a la valeur **0**, ce champ doit contenir la valeur 0.

Bits 8 et 9 Sélection de **SBHUFFRDH**. Ce champ de 2 bits peut prendre une des trois valeurs suivantes, qui indiquent la table à utiliser pour **SBHUFFRDH**.

0 Table B.14
1 Table B.15
3 Table fournie par l'utilisateur.

La valeur 2 n'est pas permise. Si **SBREFINE** a la valeur **0**, ce champ doit contenir la valeur 0.

Bits 10 et 11 Sélection de **SBHUFFRDX.** Ce champ de 2 bits peut prendre une des trois valeurs suivantes, qui indiquent la table à utiliser pour **SBHUFFRDX**.

0 Table B.14

1 Table B.15

3 Table fournie par l'utilisateur.

La valeur 2 n'est pas permise. Si **SBREFINE** a la valeur **0**, ce champ doit contenir la valeur 0.

Bits 12 et 13 Sélection de **SBHUFFRDY**. Ce champ de 2 bits peut prendre une des trois valeurs suivantes, qui indiquent la table à utiliser pour **SBHUFFRDY**.

0 Table B.14

1 Table B.15

3 Table fournie par l'utilisateur.

La valeur 2 n'est pas permise. Si **SBREFINE** a la valeur **0**, ce champ doit contenir la valeur 0.

Bit 14 Sélection de **SBHUFFRSIZE.** Si ce champ est **0**, la Table B.1 est utilisée pour **SBHUFFRSIZE**. Si ce champ est **1**, une table fournie par l'utilisateur est utilisée pour **SBHUFFRSIZE**. Si **SBREFINE** a la valeur **0**, ce champ doit contenir la valeur **0**.

Bit 15 Réservé; doit avoir la valeur **0**.

7.4.3.1.3 Fanions de gabarit adaptatif de raffinement de région alphanumérique

Ce champ n'est présent que si **SBREFINE = 1** et **SBRTEMPLATE = 0**. Il est codé sur 4 octets, qui sont formatés comme indiqué sur la Figure 38 et comme décrit ci-dessous.

$SBRATX_1$	$SBRATY_1$	$SBRATX_2$	$SBRATY_2$

Figure 38 – Structure du champ des fanions AT de raffinement de région alphanumérique

Octet 0 $SBRATX_1$

Octet 1 $SBRATY_1$

Octet 2 $SBRATX_2$

Octet 3 $SBRATY_2$

Les champs des coordonnées X et Y de gabarit AT sont des valeurs signées qui peuvent être l'une de celles qui sont permises selon le 6.3.5.3.

7.4.3.1.4 Nombre d'instances de symbole (SBNUMINSTANCES)

Ce champ de 4 octets contient le nombre d'instances de symbole codées dans ce segment.

7.4.3.1.5 Table de Huffman pour le décodage des identificateurs de symbole d'un segment de région alphanumérique

Ce champ contient une version codée des codes de Huffman utilisés pour décoder les identificateurs d'instance de symbole dans la procédure de décodage de région alphanumérique. Il est décodé comme spécifié au 7.4.3.1.7. Il n'est présent que si **SBHUFF = 1**.

7.3.4.1.6 Sélection des tables de Huffman pour le segment de région alphanumérique

Cette sélection règle des valeurs des paramètres **SBHUFFFS**, **SBHUFFDS**, **SBHUFFDT**, **SBHUFFRDW**, **SBHUFFRDH**, **SBHUFFRDX**, **SBHUFFRDY** et **SBHUFFRSIZE** conformément aux champs de sélection indiqués au 7.4.3.1.2 et aux segments de tables référencés par ce segment. Plus précisément, sur ces 8 tables de Huffman, certaines peuvent être spécifiées de façon à utiliser une table normalisée tandis que d'autres peuvent être spécifiées de façon à utiliser une table fournie par l'utilisateur. Le numéro spécifié pour utiliser une table fournie par l'utilisateur doit être égal au numéro des segments de table référencés par ce segment. Ces segments de table sont mis en correspondance avec les tables de Huffman utilisant des tables fournies par l'utilisateur selon l'ordre dans lequel ces segments de table sont référencés et selon l'ordre suivant:

1) **SBHUFFFS**

2) **SBHUFFDS**

3) **SBHUFFDT**

4) **SBHUFFRDW**

5) **SBHUFFRDH**

6) **SBHUFFRDX**

7) **SBHUFFRDY**

8) **SBHUFFRSIZE**

Si une table spécifiée par l'utilisateur est utilisée pour **SBHUFFDS**, cette table doit permettre de coder la valeur hors bande (OOB). Si une table spécifiée par l'utilisateur est utilisée pour le paramètre **SBHUFFFS**, **SBHUFFDT**, **SBHUFFRDW**, **SBHUFFRDH**, **SBHUFFRDX**, **SBHUFFRDY** ou **SBHUFFRSIZE**, cette table ne doit pas permettre de coder la valeur hors bande (OOB).

7.4.3.1.7 Décodage de table de Huffman pour le décodage des identificateurs de symbole

Cette table est codée sous la forme des longueurs des codes d'identification de symbole **SBNUMSYMS**. Les codes effectivement contenus dans **SBSYMCODES** sont attribués au moyen de l'algorithme indiqué en B.3 à partir de ces longueurs de codes d'identification de symbole.

Les longueurs des codes d'identification de symbole sont elles-mêmes codées en longueurs de séquence et les séquences sont codées en valeurs de Huffman. Cette méthode est très proche du format à codage "zlib" qui est décrit dans le commentaire RFC 1951, bien qu'il n'y ait pas identité. Le codage est fondé sur les séquences indiquées dans le Tableau 29.

Le décodage d'une table de Huffman d'identificateurs de symbole se déroule comme suit:

1) lire les longueurs codées des séquences allant de RUNCODE0 à RUNCODE34; chaque longueur est mémorisée sous la forme d'une valeur de 4 éléments binaires;

2) etant donné les longueurs, attribuer des codes de Huffman aux séquences allant de RUNCODE0 à RUNCODE34 au moyen de l'algorithme de B.3;

3) au moyen de ces attributions de codes, lire un code de Huffman qui désigne une des séquences allant de RUNCODE0 à RUNCODE34. S'il s'agit de RUNCODE32, lire deux bits supplémentaires. S'il s'agit de RUNCODE33, lire trois bits supplémentaires. S'il s'agit de RUNCODE34, lire sept bits supplémentaires;

4) interpréter le code RUNCODE et les (éventuels) bits supplémentaires conformément au Tableau 29 afin d'obtenir les longueurs des codes d'identification de symbole pour un ou plusieurs symboles;

5) répéter les étapes 3) et 4) jusqu'à ce que les longueurs de code d'identification de symbole aient été déterminées pour tous les symboles **SBNUMSYMS**;

6) omettre les bits restants du dernier octet lu, de façon que la procédure de décodage de région alphanumérique proprement dite commence à une limite d'octet;

7) attribuer un code de Huffman à chaque symbole en appliquant l'algorithme de B.3 aux longueurs de code d'identification de symbole venant d'être décodées. Le résultat est la table de Huffman des identificateurs de symbole **SBSYMCODES**.

Tableau 29 – Signification des séquences codées

RUNCODE0	Longueur du code d'identification de symbole = 0
RUNCODE1	Longueur du code d'identification de symbole = 1
RUNCODE2	Longueur du code d'identification de symbole = 2
RUNCODE3	Longueur du code d'identification de symbole = 3
RUNCODE4	Longueur du code d'identification de symbole = 4
RUNCODE5	Longueur du code d'identification de symbole = 5
RUNCODE6	Longueur du code d'identification de symbole = 6
RUNCODE7	Longueur du code d'identification de symbole = 7
RUNCODE8	Longueur du code d'identification de symbole = 8
RUNCODE9	Longueur du code d'identification de symbole = 9
RUNCODE10	Longueur du code d'identification de symbole = 10
RUNCODE11	Longueur du code d'identification de symbole = 11
RUNCODE12	Longueur du code d'identification de symbole = 12
RUNCODE13	Longueur du code d'identification de symbole = 13
RUNCODE14	Longueur du code d'identification de symbole = 14
RUNCODE15	Longueur du code d'identification de symbole = 15
RUNCODE16	Longueur du code d'identification de symbole = 16
RUNCODE17	Longueur du code d'identification de symbole = 17
RUNCODE18	Longueur du code d'identification de symbole = 18
RUNCODE19	Longueur du code d'identification de symbole = 19
RUNCODE20	Longueur du code d'identification de symbole = 20
RUNCODE21	Longueur du code d'identification de symbole = 21
RUNCODE22	Longueur du code d'identification de symbole = 22
RUNCODE23	Longueur du code d'identification de symbole = 23
RUNCODE24	Longueur du code d'identification de symbole = 24
RUNCODE25	Longueur du code d'identification de symbole = 25
RUNCODE26	Longueur du code d'identification de symbole = 26
RUNCODE27	Longueur du code d'identification de symbole = 27
RUNCODE28	Longueur du code d'identification de symbole = 28
RUNCODE29	Longueur du code d'identification de symbole = 29
RUNCODE30	Longueur du code d'identification de symbole = 30
RUNCODE31	Longueur du code d'identification de symbole = 31
RUNCODE32	Copier de 3 à 6 fois la précédente longueur de code d'identification de symbole. Les deux bits suivants, plus 3, indiquent cette longueur de répétition.
RUNCODE33	Copier de 3 à 10 fois une longueur de code d'identification de symbole égale à 0. Les trois bits suivants, plus 3, indiquent cette longueur de répétition.
RUNCODE34	Répéter de 11 à 138 fois une longueur de code d'identification de symbole égale à 0. Les sept bits suivants, plus 11, indiquent cette longueur de répétition.

EXEMPLE 1 – Supposons que le nombre **SBNUMSYMS** soit 32 et que les longueurs de code d'identification de symbole soient les suivantes pour ces 32 symboles, dans l'ordre:

0	0	0	9	6	6	6	6	3	4	4	4	4	4	4	0
7	9	8	7	5	5	5	5	5	5	3	6	7	4	7	7

longueurs de code d'identification de symbole peuvent être transmises sous la forme de la séquence d'octets hexadécimaux suivante:

```
0x50  0x03  0x35  0x32  0x53  0x00  0x00  0x00  0x00
0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x35  0x0F
0x8B  0x30  0x9E  0xB8  0x5F  0x1D  0xD2  0x83  0x00
```

L'interprétation de cette séquence d'octets peut être subdivisée en trois étapes comme suit:

1) les 17 premiers octets plus les 4 premiers bits du 18e octet attribuent comme suit les longueurs de code aux 35 séquences codées:

RUNCODE0	5	RUNCODE1	0	RUNCODE2	0
RUNCODE3	3	RUNCODE4	3	RUNCODE5	5
RUNCODE6	3	RUNCODE7	2	RUNCODE8	5
RUNCODE9	3	RUNCODE10	0	RUNCODE11	0
RUNCODE12	0	RUNCODE13	0	RUNCODE14	0
RUNCODE15	0	RUNCODE16	0	RUNCODE17	0
RUNCODE18	0	RUNCODE19	0	RUNCODE20	0
RUNCODE21	0	RUNCODE22	0	RUNCODE23	0
RUNCODE24	0	RUNCODE25	0	RUNCODE26	0
RUNCODE27	0	RUNCODE28	0	RUNCODE29	0
RUNCODE30	0	RUNCODE31	0	RUNCODE32	3
RUNCODE33	5	RUNCODE34	0		

Il est rappelé que l'on attribue une longueur de code d'identification de symbole égale à zéro aux séquences codées qui ne sont pas utilisées.

2) L'algorithme de B.3 attribue les codes de Huffman suivants aux séquences codées (celles qui n'ont pas reçu de code de Huffman sont omises):

RUNCODE0	**11100**	RUNCODE3	**010**	RUNCODE4	**011**
RUNCODE5	**11101**	RUNCODE6	**100**	RUNCODE7	**00**
RUNCODE8	**11110**	RUNCODE9	**101**	RUNCODE32	**110**
RUNCODE33	**11111**				

3) La partie restante de la séquence d'octets est la suivante:

```
0xF  0x8B  0x30  0x9E  0xB8  0x5F  0x1D  0xD2  0x83  0x00
```

où la moitié du premier octet a déjà été consommée. Le décodage de cette séquence au moyen de ces codes de Huffman donne les résultats suivants:

11111 000 RUNCODE33(0) – c'est-à-dire la séquence codée RUNCODE33 suivie par 3 bits contenant la valeur 0, qui indique une séquence de 3 longueurs égales à zéro
101 RUNCODE9
100 RUNCODE6
110 00 RUNCODE32(0) – c'est-à-dire la séquence RUNCODE32 suivie de 2 bits contenant la valeur 0
010 RUNCODE3
011 RUNCODE4
110 10 RUNCODE32(2)
11100 RUNCODE0
00 RUNCODE7
101 RUNCODE9
11110 RUNCODE8
00 RUNCODE7
11101 RUNCODE5
110 10 RUNCODE32(2)
010 RUNCODE3
100 RUNCODE6
00 RUNCODE7
011 RUNCODE4
00 RUNCODE7
00 RUNCODE7
0000 Quatre bits de bourrage pour compléter le dernier octet

4) Après l'interprétation des séquences codées conformément au Tableau 29, l'on décode la séquence souhaitée des longueurs de code d'identification de symbole.

EXEMPLE 2 – Cet exemple décrit comment un codeur peut produire une table de Huffman des identificateurs de symbole codés. La table des identificateurs de symboles est identique à celle de l'exemple précédent.

Supposons qu'une région alphanumérique fasse référence à un dictionnaire contenant 32 symboles dont chacun est utilisé comme suit:

0	0	0	1	8	8	8	8	64	32	32	32	32	32	32	0
4	1	2	4	16	16	16	16	16	16	64	8	4	32	4	4

Par exemple, les premier, deuxième et troisième symboles du dictionnaire de symboles ne sont pas utilisés du tout, le quatrième symbole est utilisé une seule fois, le cinquième symbole est utilisé huit fois, et ainsi de suite.

Le Tableau 30 montre donc, de droite à gauche, la progression du codage.

Tableau 30 – Exemple de codage de table de Huffman pour identificateurs de symboles

Symbole	Comptage d'utilisation	Longueur du code d'identification de symbole	Séquence de longueurs	Séquences RUNCODEs
Symbole #1	0	0	Séquence de longueurs 3 de la longueur 0	RUNCODE33(0)
Symbole #2	0	0		
Symbole #3	0	0		
Symbole #4	1	9	Séquence de longueurs 1 de la longueur 9	RUNCODE9
Symbole #5	8	6	Séquence de longueurs 4 de la longueur 6	RUNCODE6
Symbole #6	8	6		RUNCODE32(0)
Symbole #7	8	6		
Symbole #8	8	6		
Symbole #9	64	3	Séquence de longueurs 1 de la longueur 3	RUNCODE3
Symbole #10	32	4	Séquence de longueurs 6 de la longueur 4	RUNCODE4
Symbole #11	32	4		RUNCODE32(2)
Symbole #12	32	4		
Symbole #13	32	4		
Symbole #14	32	4		
Symbole #15	32	4		
Symbole #16	0	0	Séquence de longueurs 1 de la longueur 0	RUNCODE0
Symbole #17	4	7	Séquence de longueurs 1 de la longueur 7	RUNCODE7
Symbole #18	1	9	Séquence de longueurs 1 de la longueur 9	RUNCODE9
Symbole #19	2	8	Séquence de longueurs 1 de la longueur 8	RUNCODE8
Symbole #20	4	7	Séquence de longueurs 1 de la longueur 7	RUNCODE7
Symbole #21	16	5	Séquence de longueurs 6 de la longueur 5	RUNCODE5
Symbole #22	16	5		RUNCODE32(2)
Symbole #23	16	5		
Symbole #24	16	5		
Symbole #25	16	5		
Symbole #26	16	5		
Symbole #27	64	3	Séquence de longueurs 1 de la longueur 3	RUNCODE3
Symbole #28	8	6	Séquence de longueurs 1 de la longueur 6	RUNCODE6
Symbole #29	4	7	Séquence de longueurs 1 de la longueur 7	RUNCODE7
Symbole #30	32	4	Séquence de longueurs 1 de la longueur 4	RUNCODE4
Symbole #31	4	7	Séquence de longueurs 2 de la longueur 7	RUNCODE7
Symbole #32	4	7		RUNCODE7

Au moyen d'un algorithme normal d'arborescence de Huffman, les longueurs de code indiquées dans la colonne "Longueur du code d'identification de symbole" sont attribuées aux symboles (où une longueur de code d'identification de symbole égale à 0 représente un symbole "inutilisé"). Ensuite, ces longueurs de code sont regroupées en séquences de longueurs, comme indiqué dans la colonne "Séquences de longueurs". Ensuite, chaque séquence de longueur est exprimée sous la forme d'une ou de plusieurs séquences codées (RUNCODE) dont chacune peut être complétée de bits supplémentaires. Par exemple, la séquence codée RUNCODE32(2) représente la séquence RUNCODE32 suivie de 2 bits codant la valeur "2" pour indiquer "copier 5 fois le précédent code d'identification de symbole".

Une fois ce décodage effectué, l'on compte le nombre de fois où chaque séquence RUNCODE est utilisée. Ces décomptes sont les suivants (les séquences inutilisées ne sont pas indiquées):

RUNCODE0	1	RUNCODE3	2	RUNCODE4	2
RUNCODE5	1	RUNCODE6	2	RUNCODE7	5
RUNCODE8	1	RUNCODE9	2	RUNCODE32	3
RUNCODE33	1				

Ces décomptes sont ensuite convertis en longueurs de code au moyen d'un algorithme normal d'arborescence de Huffman:

RUNCODE0	5	RUNCODE3	3	RUNCODE4	3
RUNCODE5	5	RUNCODE6	3	RUNCODE7	2
RUNCODE8	5	RUNCODE9	3	RUNCODE32	3
RUNCODE33	5				

L'algorithme de B.3 attribue les codes de Huffman suivants aux séquences codées:

RUNCODE0	**11100**	RUNCODE3	**010**	RUNCODE4	**011**
RUNCODE5	**11101**	RUNCODE6	**100**	RUNCODE7	**00**
RUNCODE8	**11110**	RUNCODE9	**101**	RUNCODE32	**110**
RUNCODE33	**11111**				

et ces codes de Huffman sont à leur tour utilisés pour coder comme suit la colonne "RUNCODE" du Tableau 30:

11111 000 RUNCODE33(0)
101 RUNCODE9
100 RUNCODE6
110 00 RUNCODE32(0)
010 RUNCODE3
011 RUNCODE4
110 10 RUNCODE32(2)
11100 RUNCODE0
00 RUNCODE7
101 RUNCODE9
11110 RUNCODE8
00 RUNCODE7
11101 RUNCODE5
110 10 RUNCODE32(2)
010 RUNCODE3
100 RUNCODE6
00 RUNCODE7
011 RUNCODE4
00 RUNCODE7
00 RUNCODE7

Le codeur émet ensuite les longueurs codées des séquences RUNCODE, suivies de la liste des séquences RUNCODE plus 4 bits de bourrage pour compléter le dernier octet, ce qui donne la séquence d'octets suivante:

```
0x50  0x03  0x35  0x32  0x53  0x00  0x00  0x00  0x00
0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x35  0x0F
0x8B  0x30  0x9E  0xB8  0x5F  0x1D  0xD2  0x83  0x00
```

7.4.3.2 Décodage d'un segment de région alphanumérique

Un segment de région alphanumérique est décodé selon les étapes suivantes:

1) interpréter son en-tête, comme décrit au 7.4.3.1;

2) effectuer le décodage (ou l'extraction des résultats du décodage) d'éventuels segments référencés de dictionnaire de symboles et de tables;

3) comme décrit au E.3.7, remettre à zéro toutes les statistiques de codage arithmétique;

4) invoquer la procédure de décodage de région alphanumérique décrite au 6.4, après réglage des paramètres de la procédure de décodage de région alphanumérique comme indiqué au Tableau 31.

Tableau 31 – Paramètres utilisés pour décoder un segment de région alphanumérique

Nom	Valeur
SBHUFF	Comme indiqué au 7.4.3.1.1
SBREFINE	Comme indiqué au 7.4.3.1.1
SBDEFPIXEL	Comme indiqué au 7.4.3.1.1
SBCOMBOP	Comme indiqué au 7.4.3.1.1
TRANSPOSED	Comme indiqué au 7.4.3.1.1
REFCORNER	Comme indiqué au 7.4.3.1.1
SBDSOFFSET	Comme indiqué au 7.4.3.1.1
SBW	Comme spécifié par la largeur de matrice de segment de région indiquée dans l'en-tête de données de ce segment de région.
SBH	Comme spécifié par la hauteur de matrice de segment de région indiquée dans l'en-tête de données de ce segment de région.
SBNUMINSTANCES	Comme indiqué au 7.4.3.1.4
SBSTRIPS	$2^{\text{LOGSBSTRIPS}}$
SBNUMSYMS	Nombre total de symboles exportés dans tous les segments de dictionnaire de symboles référencés par ce segment.
SBSYMCODES	Comme indiqué au 7.4.3.1.7
SBSYMCODELEN	$\lceil \log_2 \textbf{SBNUMSYMS} \rceil$
SBSYMS	Concaténation des tables de symboles exportés à partir de tous les segments de dictionnaire de symboles référencés par ce segment, dans l'ordre de leur référenciation.
SBHUFFFS	Voir 7.4.3.1.6
SBHUFFDS	Voir 7.4.3.1.6
SBHUFFDT	Voir 7.4.3.1.6
SBHUFFRDW	Voir 7.4.3.1.6
SBHUFFRDH	Voir 7.4.3.1.6
SBHUFFRDX	Voir 7.4.3.1.6
SBHUFFRDY	Voir 7.4.3.1.6
SBHUFFRSIZE	Voir 7.4.3.1.6
SBRTEMPLATE	Comme indiqué au 7.4.3.1.1
$\textbf{SBRATX}_1$	Voir 7.4.3.1.3
$\textbf{SBRATY}_1$	Voir 7.4.3.1.3
$\textbf{SBRATX}_2$	Voir 7.4.3.1.3
$\textbf{SBRATY}_2$	Voir 7.4.3.1.3

7.4.4 Syntaxe d'un segment de dictionnaire de structures

7.4.4.1 En-tête de données de segment de dictionnaire de structures

Une partie données de segment de dictionnaire de structures commence par un en-tête de données de segment de dictionnaire de structures, formaté comme indiqué dans la Figure 39 et comme décrit ci-dessous.

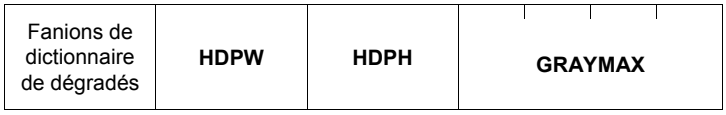


Figure 39 – Structure d'en-tête de dictionnaire de structures

Fanions de dictionnaire de dégradés – Voir 7.4.4.1.1.

HDPW – Voir 7.4.4.1.2.

HDPH – Voir 7.4.4.1.3.

GRAYMAX – Voir 7.4.4.1.4.

7.4.4.1.1 Fanions du dictionnaire de structures

Ce champ de 1 octet est formaté comme indiqué dans la Figure 40 et comme décrit ci-dessous:

Bit 0 **HDMMR**

Si ce bit est **1**, le segment utilise la variante de codage MMR. S'il est **0**, le segment utilise la variante de codage arithmétique.

Bits 1 et 2 **HDTEMPLATE**

Ce champ commande le gabarit utilisé pour décoder les structures si **HDMMR** = **0**. Si **HDMMR** = **1**, ce champ doit contenir la valeur 0.

Bits 3 à 7 Champ réservé qui doit être **0**.

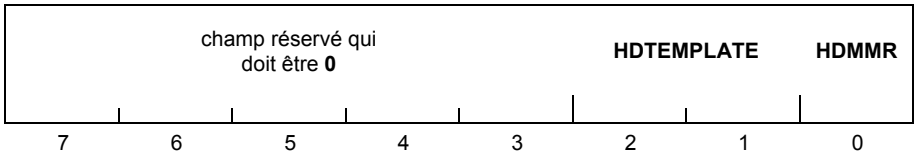


Figure 40 – Structure du champ des fanions de dictionnaire de structures

7.4.4.1.2 Largeur des structures dans le dictionnaire de structures (HDPW)

Ce champ de 1 octet contient la largeur des structures définies dans ce dictionnaire de structures. Sa valeur doit être supérieure à zéro.

7.4.4.1.3 Hauteur des structures dans le dictionnaire de structures (HDPH)

Ce champ de 1 octet contient la hauteur des structures définies dans ce dictionnaire de symboles. Sa valeur doit être supérieure à zéro.

7.4.4.1.4 Plus grande valeur d'échelle de gris (GRAYMAX)

Ce champ de 4 octets contient le nombre de structures définies dans ce dictionnaire de structures moins 1.

7.4.4.2 Décodage d'un segment de dictionnaire de structures

Un segment de dictionnaire de structures est décodé selon les étapes suivantes:

1) interpréter son en-tête comme décrit au 7.4.4.1.;
2) remettre à zéro toutes les statistiques de codage arithmétique, comme décrit au E.3.7;
3) invoquer la procédure de décodage de dictionnaire de structures décrite au 6.7 après avoir réglé comme indiqué dans le Tableau 32 les paramètres de la procédure de décodage du dictionnaire de structures.

Tableau 32 – Paramètres utilisés pour décoder un segment de dictionnaire de structures

Nom	Valeur
HDMMR	Comme indiqué au 7.4.4.1.1
HDTEMPLATE	Como se muestra en 7.4.4.1.1
HDPW	Comme indiqué au 7.4.4.1.2
HDPH	Comme indiqué au 7.4.4.1.3
GRAYMAX	Comme indiqué au 7.4.4.1.4

7.4.5 Syntaxe d'un segment de région de dégradé

Les parties données des trois types de segment de région de dégradé ("région de dégradé intermédiaire", "région de dégradé immédiate" et "région de dégradé immédiate sans pertes") sont codées de façon identique mais sont traitées différemment; voir 8.2. La syntaxe de ces parties données des types de segment est spécifiée ci-après.

7.4.5.1 En-tête de données de segment de région de dégradé

La partie données d'un segment de région de dégradé commence par un en-tête de données de segment de région de dégradé. Cet en-tête contient les champs indiqués dans la Figure 41 et décrits ci-dessous.

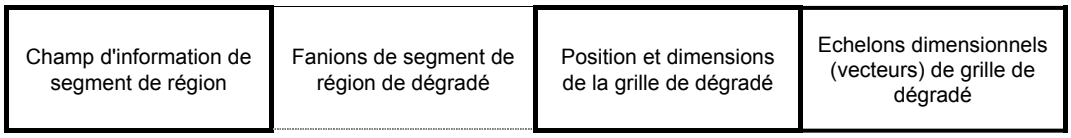


Figure 41 – Structure d'en-tête de données de segment de région de dégradé

Champ d'information de segment de région – Voir 7.4.1.

Fanions de segment de région de dégradé – Voir 7.4.5.1.1.

Position et dimensions de la grille de dégradé – Voir 7.4.5.1.2

Vecteur de grille de dégradé – Voir 7.4.5.1.3.

7.4.5.1.1 Fanions de segment de région de dégradé

Ce champ de 1 octet est formaté comme indiqué sur la Figure 42 et comme décrit ci-dessous.

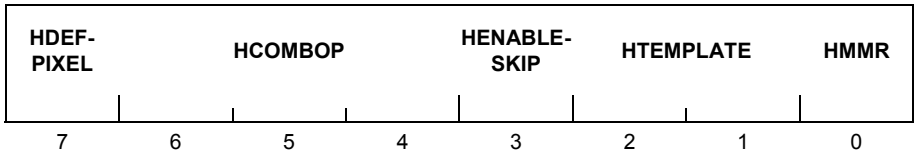


Figure 42 – Structure du champ des fanions de segment de région de dégradé

Bit 0 **HMMR**

Si ce bit est **1**, le segment utilise la variante de codage MMR. S'il est **0**, le segment utilise la variante de codage arithmétique.

Bits 1 et 2 **HTEMPLATE**

Ce champ commande le gabarit utilisé pour décoder les plans binaires à valeurs d'échelle de gris en dégradé si **HMMR = 0**. Si **HMMR = 1**, ce champ doit contenir la valeur **0**.

Bit 3 **HENABLESKIP**

Ce champ commande l'omission des valeurs d'échelle de gris qui ne contribuent pas au contenu de la région au cours du décodage. Si **HMMR = 1**, ce champ doit contenir la valeur **0**.

Bits 4 à 6 **HCOMBOP**

Ce champ a 5 valeurs possibles, représentant l'un des 5 opérateurs combinatoires suivants:

0 OR
1 AND
2 XOR
3 XNOR
4 REPLACE

Bit 7 **HDEFPIXEL**

Ce bit contient la valeur initiale pour chaque pixel dans la région de dégradé, avant que d'éventuelles structures soient dessinées.

7.4.5.1.2 Position et dimensions de la grille de dégradé

Ce champ décrit l'emplacement et la taille de la grille de valeurs d'échelle de gris. La Figure 24 décrit ces valeurs. Ce champ est formaté comme indiqué dans la Figure 43 et comme décrit ci-dessous.

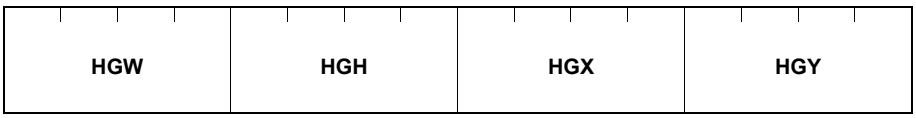


Figure 43 – Structure du champ de position et de dimensions de grille de dégradé

HGW – Voir 7.4.5.1.2.1.

HGH – Voir 7.4.5.1.2.2.

HGX – Voir 7.4.5.1.2.3.

HGY – Voir 7.4.5.1.2.4.

7.4.5.1.2.1 Largeur de l'image en échelle de gris (HGW)

Ce champ de 4 octets contient la largeur de la série tabulaire des valeurs d'échelle de gris.

7.4.5.1.2.2 Hauteur de l'image en échelle de gris (HGH)

Ce champ de 4 octets contient la hauteur de la série tabulaire des valeurs d'échelle de gris.

7.4.5.1.2.3 Décalage horizontal de la grille (HGX)

Ce champ signé de 4 octets contient 256 fois le décalage horizontal de l'origine de la grille de dégradé.

7.4.5.1.2.4 Décalage vertical de la grille (HGY)

Ce champ signé de 4 octets contient 256 fois le décalage vertical de l'origine de la grille de dégradé.

7.4.5.1.3 Vecteur de grille de dégradé

Ce champ décrit le vecteur utilisé pour dessiner la grille de valeurs d'échelle de gris. La Figure 24 décrit ces valeurs. Ce champ est formaté comme indiqué dans la Figure 44 et comme décrit ci-dessous.

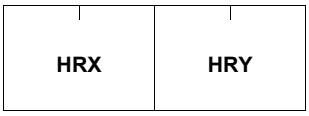


Figure 44 – Structure du champ de vecteur de grille de dégradé

HRX – Voir 7.4.5.1.3.1.

HRY – Voir 7.4.5.1.3.2.

7.4.5.1.3.1 Coordonnée horizontale du vecteur de grille de dégradé (HRX)

Ce champ non signé de 2 octets contient 256 fois la coordonnée horizontale du vecteur de grille de dégradé.

7.4.5.1.3.2 Coordonnée verticale du vecteur de grille de dégradé (HRY)

Ce champ non signé de 2 octets contient 256 fois la coordonnée verticale du vecteur de grille de dégradé.

7.4.5.2 Décodage d'un segment de région de dégradé

Un segment de région de dégradé est décodé selon les étapes suivantes:

1) interpréter son en-tête, comme décrit au 7.4.5.1;
2) effectuer le décodage (ou l'extraction des résultats du décodage) du segment référencé de dictionnaire de structures;
3) comme décrit au E.3.7, remettre à zéro toutes les statistiques de codage arithmétique;
4) invoquer la procédure de décodage de région de dégradé décrite au 6.6, après réglage des paramètres de la procédure de décodage de région de dégradé comme indiqué au Tableau 33.

Tableau 33 – Paramètres utilisés pour décoder un segment de région de dégradé

Nom	Valeur
HBW	Comme spécifié par la largeur de matrice de segment de région indiquée dans l'en-tête de données de ce segment de région.
HBH	Comme spécifié par la hauteur de matrice de segment de région indiquée dans l'en-tête de données de ce segment de région.
HMMR	Comme indiqué au 7.4.5.1.1
HTEMPLATE	Comme indiqué au 7.4.5.1.1
HENABLESKIP	Comme indiqué au 7.4.5.1.1
HCOMBOP	Comme indiqué au 7.4.5.1.1
HDEFPIXEL	Comme indiqué au 7.4.5.1.1
HGW	Comme indiqué au 7.4.5.1.2.1
HGH	Comme indiqué au 7.4.5.1.2.2
HGX	Comme indiqué au 7.4.5.1.2.3
HGY	Comme indiqué au 7.4.5.1.2.4
HRX	Comme indiqué au 7.4.5.1.3.1
HRY	Comme indiqué au 7.4.5.1.3.2
HNUMPATS	Nombre de structures contenues dans le segment de dictionnaire de structures référencé par ce segment.
HPATS	Structures contenues dans le segment de dictionnaire de structures référencé par ce segment.
HPW	Largeur en pixels de chacune des structures contenues dans **HPATS**.
HPH	Hauteur en pixels de chacune des structures contenues dans **HPATS**.

7.4.6 Syntaxe d'un segment de région générique

Les parties données des trois types de segment de région générique ("région générique intermédiaire", "région générique immédiate" et "région générique immédiate sans pertes") sont codées identiquement mais traitées différemment; voir 8.2. La syntaxe des parties données de ces types de segment est spécifiée ci-après.

7.4.6.1 En-tête de données d'un segment de région générique

La partie données d'un segment de région générique commence par un en-tête de données de segment de région générique. Cet en-tête contient les champs indiqués dans la Figure 45 et décrits ci-dessous.

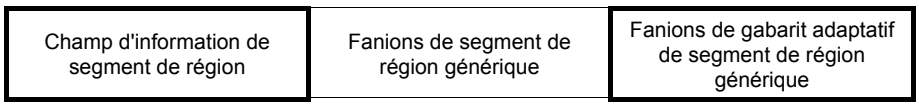


Figure 45 – Structure d'en-tête de données de segment de région générique

Champ d'information de segment de région – Voir 7.4.1.

Fanions de segment de région générique – Voir 7.4.6.2.

Fanions de gabarit adaptatif d'un segment de région générique – Voir 7.4.6.3.

7.4.6.2 Fanions d'un segment de région générique

Ce champ de 1 octet est formaté comme indiqué dans la Figure 46 et comme décrit ci-dessous.

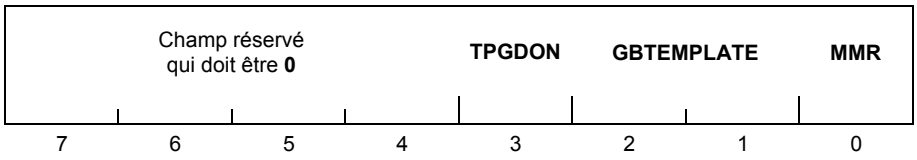


Figure 46 – Structure du champ des fanions de segment de région générique

Bit 0 **MMR**

Bits 1 et 2 **GBTEMPLATE**

Ce champ spécifie le gabarit utilisé pour le codage arithmétique fondé sur un gabarit. Si **MMR** = **1**, ce champ doit contenir la valeur zéro.

Bit 3 **TPGDON**

Ce champ spécifie l'utilisation de la prédiction typique pour le codage générique direct.

Bits 4 à 7 Champ réservé qui doit être 0.

7.4.6.3 Fanions de gabarit adaptatif d'un segment de région générique

Ce champ n'est présent que si **MMR** = **0**. Si **GBTEMPLATE** est 0, c'est un champ de 8 octets, formaté comme indiqué dans la Figure 47 et comme décrit ci-dessous.

$GBATX_1$	$GBATY_1$	$GBATX_2$	$GBATY_2$	$GBATX_3$	$GBATY_3$	$GBATX_4$	$GBATY_4$

Figure 47 – Structure du champ des fanions AT de région générique lorsque GBTEMPLATE = 0

Octet 0 **$GBATX_1$**

Octet 1 **$GBATY_1$**

Octet 2 **$GBATX_2$**

Octet 3 **$GBATY_2$**

Octet 4 **$GBATX_3$**

Octet 5 **$GBATY_3$**

Octet 6 **$GBATX_4$**

Octet 7 **$GBATY_4$**

Si **GBTEMPLATE** = 1, 2 ou 3, c'est un champ de 2 octets formaté comme indiqué dans la Figure 48 et comme décrit ci-dessous. Si **GBTEMPLATE** = 1, 2 ou 3, les valeurs de **$GBATX_2$** à **$GBATX_4$** et de **$GBATY_2$** à **$GBTAY_4$** sont toutes égales à zéro.

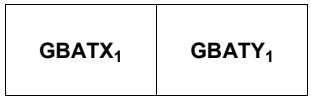


Figure 48 – Structure du champ des fanions AT de région générique lorsque GBTEMPLATE n'est pas égal à 0

Octet 0 **$GBATX_1$**

Octet 1 **$GBATY_1$**

Les champs des coordonnées X et Y de gabarit adaptatif sont des valeurs signées qui peuvent prendre les valeurs permises de la Figure 7.

7.4.6.4 Décodage d'un segment de région générique

Un segment de région générique est décodé selon les étapes suivantes:

1) interpréter son en-tête comme décrit au 7.4.6.1;
2) remettre à zéro toutes les statistiques de codage arithmétique, comme décrit au E.3.7;
3) invoquer la procédure de décodage de région générique décrite au 6.2, après réglage comme indiqué au Tableau 34 des paramètres de la procédure de décodage de région générique.

Tableau 34 – Paramètres utilisés pour décoder un segment de région générique

Nom	Valeur
MMR	Comme indiqué au 7.4.6.2
GBTEMPLATE	Comme indiqué au 7.4.6.2
TPGDON	Comme indiqué au 7.4.6.2
USESKIP	**0**
GBW	Comme spécifié par la largeur de matrice de segment de région indiquée dans l'en-tête de données de ce segment de région.
GBH	Comme spécifié par la hauteur de matrice de segment de région indiquée dans l'en-tête de données de ce segment de région.
$GBATX_1$	Voir 7.4.6.3
$GBATY_1$	Voir 7.4.6.3
$GBATX_2$	Voir 7.4.6.3
$GBATY_2$	Voir 7.4.6.3
$GBATX_3$	Voir 7.4.6.3
$GBATY_3$	Voir 7.4.6.3
$GBATX_4$	Voir 7.4.6.3
$GBATY_4$	Voir 7.4.6.3

Comme noté au 7.2.7, il existe un cas particulier lorsqu'un segment de région générique immédiate peut avoir une longueur inconnue. Dans ce cas, il est également possible que le segment contienne moins de rangées de données matricielles qu'indiqué dans le champ d'information du segment de région.

Afin que le décodeur décode correctement le segment, il doit lire le champ de quatre octets contenant le nombre de rangées, qui est mémorisé dans les quatre derniers octets de la partie données du segment. Ces quatre octets peuvent être détectés sans connaissance préalable de la longueur de la partie données: si **MMR** = **1**, ils sont précédés de la séquence des deux octets suivants: `0x00 0x00`; si **MMR** = **0**, ils sont précédés par la séquence des deux octets suivants: `0xFF 0xAC`. Le champ de nombre de rangées contient le nombre réel de rangées contenu dans ce segment; ce nombre ne doit pas être supérieur à la valeur de hauteur de matrice de segment de région indiquée dans le champ d'information du segment de région.

NOTE – La séquence `0x00 0x00` ne peut pas apparaître dans les données à codage MMR. La séquence `0xFF 0xAC` ne peut apparaître qu'à la fin de données à codage arithmétique. Ces séquences ne peuvent donc apparaître fortuitement dans les données qui sont décodées pour produire le contenu de la région générique.

7.4.7 Syntaxe d'une région générique par raffinement

Les parties données des trois types de segment de région générique par raffinement ("région de raffinement générique intermédiaire", "région générique par raffinement immédiate" et "région générique par raffinement immédiate sans pertes") sont codées identiquement mais sont traitées différemment; voir 8.2. La syntaxe de ces types de données est spécifiée ci-après.

7.4.7.1 En-tête de données d'un segment de région générique par raffinement

La partie données d'un segment de région générique par raffinement commence par l'en-tête de segment correspondant, qui contient les champs indiqués dans la Figure 49 et décrits ci-dessous.

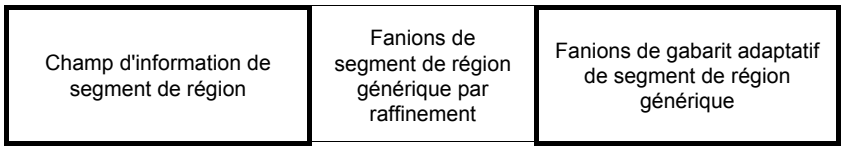


Figure 49 – Structure d'en-tête de données de segment de région générique par raffinement

Champ d'information de segment de région – Voir 7.4.1

Fanions de segment de région générique par raffinement – Voir 7.4.7.2

Fanions de gabarit adaptatif de segment de région générique – Voir 7.4.7.3

7.4.7.2 Fanions d'un segment de région générique par raffinement

Ce champ de 1 octet est formaté comme indiqué dans la Figure 50 et comme décrit ci-dessous.

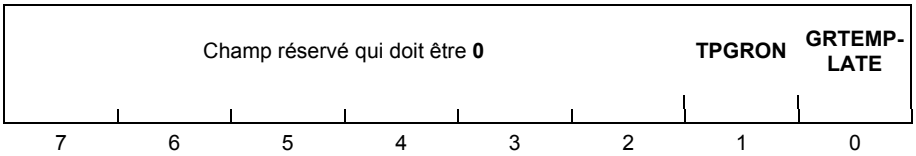


Figure 50 – Structure du champ des fanions de segment de région générique par raffinement

Bit 0 **GRTEMPLATE**

Ce champ spécifie le gabarit utilisé pour le codage arithmétique à gabarit.

Bit 1 **TPGRON**

Ce champ spécifie si la prédiction typique est utilisée pour le raffinement générique.

Bits 2 à 7 Champ réservé qui doit être zéro.

7.4.7.3 Fanions de gabarit adaptatif d'un segment de région générique

Ce champ n'est présent que si **GRTEMPLATE** = **0**. C'est un champ de 4 octets, formaté comme indiqué dans la Figure 51 et comme décrit ci-dessous.

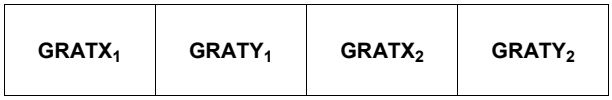


Figure 51 – Structure du champ des fanions AT de région générique par raffinement

Octet 0 **$GRATX_1$**

Octet 1 **$GRATY_1$**

Octet 2 **$GRATX_2$**

Octet 3 **$GRATY_2$**

Les champs des coordonnées X et Y de gabarit adaptatif sont des valeurs signées qui peuvent prendre les valeurs permises selon le 6.3.5.3.

7.4.7.4 Sélection de la matrice de référence

Si ce segment fait référence à un autre segment de région, le paramètre **GRREFERENCE** de matrice de référence est mis aux valeurs du contenu actuel du tampon auxiliaire associé au segment de région auquel ce segment fait référence.

Si ce segment ne fait pas référence à un autre segment de région, le paramètre **GRREFERENCE** est mis aux valeurs d'une matrice contenant le contenu actuel du tampon de page (voir § 8), avec limitation à la zone du tampon de page qui est spécifiée par le champ d'information de ce segment de région.

7.4.7.5 Décodage d'un segment de région générique par raffinement

Un segment de région générique par raffinement est décodé selon les étapes suivantes:

1) Interpréter son en-tête comme décrit au 7.4.7.1. Si ce segment ne fait pas référence à un autre segment de région, son opérateur combinatoire externe doit être REPLACE. S'il fait effectivement référence à un autre segment de région, les dimensions, l'emplacement et l'opérateur externe de sa matrice de région doivent être égaux à ceux de cet autre segment de région.

   NOTE – L'exigence de correspondance des emplacements et des opérateurs combinatoires externes a pour objet d'aider les décodeurs qui souhaitent produire des images d'une page qui n'est que partiellement décodée: elle garantit que l'emplacement final et l'opérateur combinatoire externe seront connus pour tous les segments intermédiaires. Ces images de page partiellement décodée sont hors du domaine d'application de la présente Recommandation | Norme internationale.

2) Remettre à zéro toutes les statistiques de codage arithmétique comme décrit au E.3.7.

3) Déterminer le tampon associé au segment de région auquel ce segment fait référence.

4) Invoquer la procédure de décodage de région générique par raffinement décrite au 6.3, avec réglage comme indiqué dans le Tableau 35 des paramètres de cette procédure.

Tableau 35 – Paramètres utilisés pour décoder un segment de région générique par raffinement

Nombre	Valeur
GRTEMPLATE	Comme indiqué au 7.4.6.2
TPGRON	Comme indiqué au 7.4.6.2
GRW	Comme spécifié par la largeur de matrice de segment de région indiquée dans l'en-tête de données de ce segment de région
GRH	Comme spécifié par la hauteur de matrice de segment de région indiquée dans l'en-tête de données de ce segment de région
GRREFERENCE	Voir 7.4.7.4
GRREFERENCEDX	0
GRREFERENCEDY	0
$GRATX_1$	Voir 7.4.7.3
$GRATX_2$	Voir 7.4.7.3
$GRATY_1$	Voir 7.4.7.3
$GRATY_2$	Voir 7.4.7.3

7.4.8 Syntaxe d'un segment d'informations de page

Un segment d'informations de page décrit une page. Il contient les champs indiqués dans la Figure 52 et décrits ci-après.

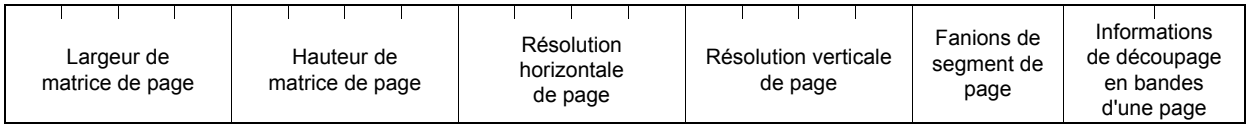


Figure 52 – **Structure de segment d'informations de page**

Largeur de matrice de page – Voir 7.4.8.1.

Hauteur de matrice de page – Voir 7.4.8.2.

Résolution horizontale d'une page – Voir 7.4.8.3.

Résolution verticale d'une page – Voir 7.4.8.4.

Fanions de segment de page – Voir 7.4.8.5.

Informations de découpage en bandes d'une page – Voir 7.4.8.6.

Le premier segment qui est associé à une page quelconque doit toujours être un segment d'informations de page.

7.4.8.1 Largeur d'une matrice de page

Ce champ de 4 octets contient la largeur en pixels de la matrice de page.

7.4.8.2 Hauteur d'une matrice de page

Ce champ de 4 octets contient la hauteur en pixels de la matrice de page. Dans certains cas, cette valeur ne peut pas être connue au moment de l'écriture du segment d'informations de page. Dans ce cas, ce champ doit contenir la séquence `0xFFFFFFFF` et la hauteur de page réelle peut être communiquée ultérieurement, une fois connue.

7.4.8.3 Résolution horizontale d'une page

Ce champ de 4 octets contient la résolution du support original de page, mesurée en pixels par mètre dans la direction horizontale. Si cette valeur est inconnue, ce champ doit contenir la séquence `0x00000000`.

7.4.8.4 Résolution verticale d'une page

Ce champ de 4 octets contient la résolution du support original de page, mesurée en pixels par mètre dans la direction horizontale. Si cette valeur est inconnue, ce champ doit contenir la séquence `0x00000000`.

7.4.8.5 Fanions d'un segment de page

Il s'agit d'un champ de 1 octet qui est formaté comme indiqué dans la Figure 53 et décrit ci-dessous.

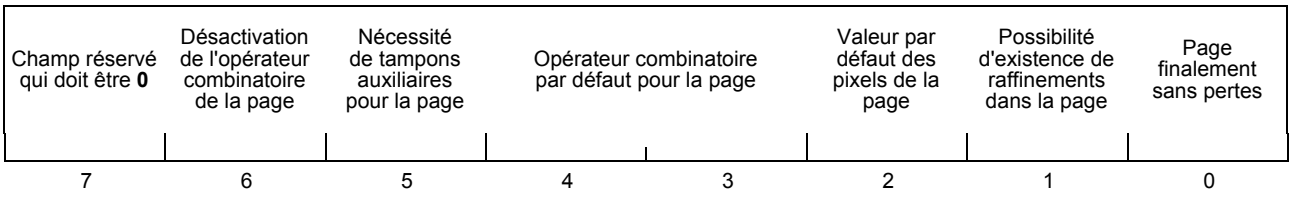


Figure 53 – Structure du champ des fanions de segment de page

Bit 0 Page finalement sans pertes. Si ce bit est **0**, le fichier ne contient pas de représentation sans pertes (de précodage) de la page. Si ce bit est **1**, le fichier contient assez d'informations pour reconstruire la page originale.

Bit1 Possibilité d'existence de raffinements dans la page. Si ce bit est **0**, aucun segment de région de raffinement ne peut être associé à la page. S'il est **1**, de tels segments peuvent être associés à la page.

Bit 2 Valeur par défaut des pixels de la page. Ce fanion contient la valeur initiale pour chaque pixel de la page, avant le décodage ou le dessin d'éventuels segments de région.

Bits 3 et 4 Opérateur combinatoire par défaut de la page. Ce champ a 4 valeurs possibles, représentant un des quatre opérateurs combinatoires possibles suivants:

0 OR
1 AND
2 XOR
3 XNOR

Cet opérateur sert à fusionner des segments de région superposés ainsi qu'à combiner des segments de région avec la valeur par défaut des pixels de la page.

Bit 5 Nécessité de tampons auxiliaires pour la page. Si ce bit est **0**, aucun segment de région nécessitant un tampon auxiliaire ne peut être associé à la page. Si ce bit est **1**, de tels segments peuvent être associés à la page.

Bit 6 Désactivation de l'opérateur combinatoire de la page. Si ce bit est **0**, chaque segment de région directe associé à cette page doit utiliser l'opérateur combinatoire par défaut de cette page. Si ce bit est **1**, les segments de région directe, associés à la page, peuvent utiliser tout opérateur combinatoire

NOTE 1 – Tous les segments de région, sauf ceux de région de raffinement, sont des segments de région directe. Compte tenu des exigences du 7.4.7.5 limitant les opérateurs combinatoires externes des segments de région de raffinement, si ce bit est **0**, les segments de région de raffinement associés à cette page, ne faisant référence à aucun segment de région, doivent avoir l'opérateur combinatoire externe REPLACE et tous les autres segments de région associés à cette page doivent avoir l'opérateur combinatoire externe qui est spécifié par le champ "opérateur combinatoire par défaut pour la page".

NOTE 2 – Si tous les segments de région directe associés à une page utilisent le même opérateur combinatoire, il est possible de les réordonner dans une certaine mesure (il n'est pas possible de modifier l'ordre relatif d'un segment de raffinement quelconque). Si certains segments utilisent des opérateurs combinatoires différents, le décodeur n'est pas en mesure d'effectuer un quelconque réordonnancement de ce type. Par ailleurs, le décodeur ne peut pas, à partir des en-têtes de segment, déterminer si de tels opérateurs combinatoires non initiaux sont utilisés dans la page, de sorte que ce fanion indique si le réordonnancement est possible si le décodeur souhaite l'effectuer.

Bit 7 Champ réservé qui doit être **0**.

7.4.8.6 Informations de découpage en bandes d'une page

Ce champ de 2 octets est formaté comme indiqué dans la Figure 54 et comme décrit ci-dessous.

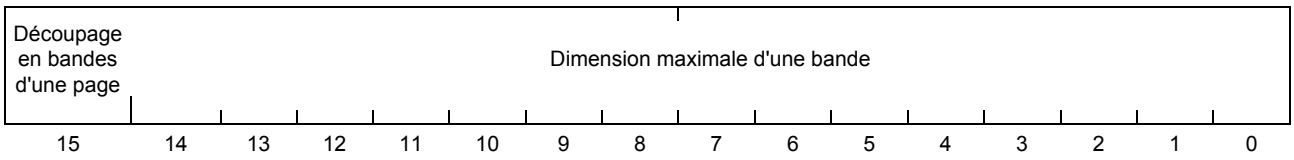


Figure 54 – Structure du champ des informations de découpage de page

Bits 0 à 14 Dimension maximale d'une bande

Bit 15 Découpage en bandes d'une page

Si le fanion de découpage de page est **1**, la page peut être associée à des segments de fin de bande. Dans ce cas, la dimension maximale de chaque bande (la distance entre une rangée finale de segment de fin de bande et la rangée finale du précédent segment de fin de bande, ou 0 dans le cas du premier segment de fin de bande) ne doit pas être supérieure à la dimension maximale de bande de la page.

Si la hauteur de matrice de page est inconnue (ce qui est indiqué par une hauteur de matrice de page égale à `0xFFFFFFFF`), le fanion de découpage de bande doit être **1**.

7.4.9 Syntaxe de segment de fin de page

Un segment de fin de page n'a pas de données associées. Son champ de longueur de données de segment doit être 0.

Le dernier segment qui est associé à une page quelconque doit être un segment de fin de page. Exactement une fin de segment de page doit être associée à chaque page.

Si une hauteur de page était inconnue à l'origine, il doit y avoir au moins un segment de fin de bande associé à la page. Dans ce cas, la rangée finale de cette dernière bande est la dernière rangée de la matrice de page et aucun segment de région ne peut apparaître entre le dernier segment de fin de bande et le segment de fin de page.

7.4.10 Syntaxe de segment de fin de bande

Un segment de fin de bande indique que le codeur a fini de coder une partie de la page avec laquelle le segment est associé et qu'il n'y reviendra pas. Il spécifie la coordonnée Y d'une rangée de la page. Aucun segment suivant la fin de bande ne peut modifier une partie quelconque de la matrice de page se trouvant sur cette rangée ou au-dessus d'elle. Par ailleurs, aucun segment précédant la fin de bande ne peut modifier une partie quelconque de la matrice de page se trouvant sous cette rangée. Celle-ci est appelée *rangée finale* de la bande.

NOTE 1 – Dans certains cas, le décodeur ne peut disposer que d'une quantité limitée de mémoire tampon pour la matrice de page, inférieure aux dimensions de cette page. Le décodeur a besoin d'être informé du moment où il est en mesure d'émettre le contenu du tampon actuel et de libérer celui-ci pour la bande suivante de la page.

La rangée finale spécifiée par un segment de fin de bande doit toujours se trouver au-dessous d'une éventuelle rangée finale précédente pour cette page.

Une page dont la hauteur était inconnue initialement doit toujours contenir au moins un segment de fin de bande.

NOTE 2 – Un segment de fin de bande est utilisé dans ce cas pour communiquer les dimensions de la page.

Les données d'un segment de fin de bande se composent d'une valeur codée sur 4 octets, spécifiant la coordonnée Y de la rangée finale.

NOTE 3 – Si une limite de bande coupe une ligne de texte en deux parties, une partie supérieure et une partie inférieure, les caractères à cheval sur cette limite sont également séparés. Une manière de traiter ces caractères consiste à les coder avec une région générique ou à ajouter les moitiés supérieures et inférieures à un dictionnaire de symboles et à les utiliser comme d'autres symboles.

Cependant, si le codeur dispose d'un espace tampon supérieur à celui du décodeur, il est possible d'appliquer une méthode de codage plus efficace: le codeur peut (temporairement) ne pas tenir compte de la limite de bande et générer une liste d'instances de symboles ainsi qu'un dictionnaire de symboles. Il peut ensuite coder une région alphanumérique dans chaque bande; la première région alphanumérique contient toutes les instances de symboles qui affectent la partie de la page située au-dessus de la limite de bande; la deuxième région alphanumérique contient toutes les instances de symboles qui affectent la partie située au-dessous de la limite de bande.

Autrement dit, certaines instances de symboles apparaissent deux fois, utilisant à chaque fois le même symbole du dictionnaire de symboles. La première apparition permet de coder la moitié supérieure du caractère, la moitié inférieure étant coupée selon les règles de dessin appliquées dans la procédure de décodage des régions alphanumériques. De même, la deuxième apparition permet de coder la moitié inférieure du caractère: le caractère entier est codé, mais la moitié supérieure est coupée. Ainsi, cette méthode de codage permet de réduire la quantité de données de dictionnaire de symboles requise.

7.4.11 Syntaxe de segment de fin de fichier

Si un fichier contient un segment de fin de fichier, ce doit être le dernier segment.

Un segment de fin de fichier n'a pas de données associées. Son champ de longueur de données associées doit être zéro.

7.4.12 Syntaxe de segment de profils

Un segment de profils contient une liste des profils avec lesquels un flux de données JBIG2 est en conformité. Si certains segments de profils sont présents, le premier segment du flux de données doit être un segment de profils qui ne doit être associé à aucune page. Les profils de la présente Recommandation | Norme internationale sont énumérés dans l'Annexe F.

Un segment de profils commence par un champ de 4 octets contenant le nombre de profils énumérés. Ce champ est suivi par autant de champs de 4 octets dont chacun contient un numéro d'identification de profil. Le flux de données doit toujours être en conformité avec chacun des profils énumérés.

Plusieurs segments de profils peuvent être présents. Dans ce cas, chacun (sauf le premier) doit être associé à une page. Aucune page ne peut être associée à plus d'un seul segment de profil. De même, chaque segment de profils qui suit le premier segment doit être plus restrictif que celui-ci. En d'autres termes, il doit énumérer tous les numéros d'identification de profil qui sont indiqués dans le premier segment, et éventuellement d'autres numéros. Les segments qui constituent chaque page doivent, collectivement, être en conformité avec chacun des profils énumérés dans tout segment de profils associé à cette page.

NOTE – Le segment de profils global permet à un décodeur de constater rapidement qu'il ne peut pas décoder un certain flux de données. Pour faciliter le transfert de pages d'un fichier à un autre, chaque page doit pouvoir contenir un segment de profils éventuellement différent (bien que plus restrictif).

7.4.13 Syntaxe de segment de table de codage

Une syntaxe de segment de table de codage est décrite dans l'Annexe B.

7.4.14 Syntaxe de segment d'extension

Les données d'un segment d'extension commencent par un en-tête d'extension:

Type d'extension: ce champ de 4 octets contient l'identificateur du type des données présentes dans le segment d'extension.

Les trois bits de poids fort de ce champ ont la signification spéciale suivante:

Bit 29 Champ réservé. De futures révisions de la présente Recommandation | Norme internationale pourront définir des types d'extension qui pourront eux-mêmes être enregistrés par d'autres parties. Celles-ci pourront n'enregistrer que les types d'extension dont ce fanion est égal à 0. Tous les types d'extension ayant ce bit 29 égal à **1** sont réservés pour de futures révisions de la présente Recommandation | Norme internationale.

Bit 30 Dépendant. Si ce fanion est **1**, le codage des données dans le segment d'extension dépend du codage exact des données dans les segments auxquels le segment d'extension fait référence. Tout programme de manipulation de fichier modifiant ces segments référencés doit modifier en conséquence ces données de segment d'extension. S'il ne comprend pas le segment d'extension (parce qu'il ne reconnaît pas son type d'extension) et si ce n'est pas un segment d'extension nécessaire, il y a lieu de supprimer ce segment.

EXEMPLE – Un segment d'extension contenant un code CRC du segment auquel il fait référence doit être étiqueté comme étant dépendant.

Bit 31 Nécessaire. Si ce fanion est **1**, un décodeur qui ne sait pas comment analyser les extensions de ce type de segment d'extension ne sera pas en mesure de décoder correctement le fichier pour produire les images de page décodée prévues.

NOTE – Cette indication est destinée à faciliter de futures extensions du format JBIG2, comme des améliorations de codage. Si ce fanion est **1**, un décodeur qui ne comprend pas l'extension sait qu'il a rencontré des données nécessaires pour le décodage correct de la page qu'il ne peut pas traiter. Par exemple, un segment d'extension contenant une région codée par une nouvelle méthode donnée sera étiqueté comme étant "nécessaire" car sans cette région l'image de page sera incomplète. Un autre exemple pourrait être celui d'un segment d'extension contenant une série de couleurs à appliquer aux symboles au fur et à mesure de leur dessin sur la page.

Si le bit "nécessaire" est **1**, le bit "réservé" doit aussi être **1**.

Le reste des données de segment d'extension suit immédiatement le champ de type d'extension; il est formaté d'une façon propre à chaque type d'extension.

7.4.15 Types d'extension définis

Les types d'extension suivants sont définis actuellement.

`0x20000000` Commentaire codé sur un seul octet: voir 7.4.15.1.

`0x20000002` Commentaire codé sur plusieurs octets: voir 7.4.15.2.

7.4.15.1 Commentaire codé sur un seul octet

Un segment d'extension de type commentaire codé sur un seul octet contient des informations alphanumériques sur un autre segment, sur une autre page ou sur le flux binaire dans son ensemble. S'il ne fait référence à aucun autre segment mais qu'il soit associé à une certaine page, il contient donc un ensemble de commentaires applicables à cette page. S'il fait référence à d'autres segments, il contient un certain ensemble de commentaires applicables à ces segments.

Un segment de commentaire codé sur un seul octet contient un certain nombre de paires (nom-valeur). Chaque élément de chaque paire est une chaîne de caractères terminée par un octet `0x00`. La dernière paire est suivie d'un octet `0x00` additionnel. Les autres octets doivent être interprétés selon l'ISO/CEI 8859-1:1998.

NOTE – Un segment d'extension de commentaire codé sur un seul octet peut contenir tout caractère ISO/CEI 8859-1 valide, y compris ceux dont la valeur est supérieure à 127.

EXEMPLE – Le commentaire contenant les paires suivantes:

```
Titre     Histoire illustrée des Fausses dents
Auteur    Le Grand Fromage
```

est mémorisé sous la forme de la séquence d'octets suivante. Ces octets sont représentés en nombres hexadécimaux avec leurs équivalents imprimables, un point "." indiquant un octet non imprimable. Noter le type d'extension de 4 octets au début des données de segment:

```
20 00 00 00 54 69 74 6C 65 00 41 6E 20 49 6C 6C  ...Titre.Histoire illustrée
75 73 74 72 61 74 65 64 20 48 69 73 74 6F 72 79 des Fausses Dents
20 6F 66 20 46 61 6C 73 65 20 54 65 65 74 68 00  Auteur.
41 75 74 68 6F 72 00 54 68 65 20 42 69 67 20 43 Le Grand Fromage...
68 65 65 73 65 00 00
```

7.4.15.2 Commentaire codé sur plusieurs octets

Un segment d'extension de commentaire codé sur plusieurs octets est formaté de la même manière qu'un segment d'extension de commentaire codé sur un seul octet sauf que les caractères individuels occupent chacun deux octets dans le codage ISO/CEI 10646-1:1993 (UCS-2). Chaque élément de chaque paire dans le commentaire se termine par une séquence `0x0000` et la paire finale est suivie d'une séquence `0x0000` additionnelle.

8 Mise en page

8.1 Modèle du décodeur

Ce paragraphe décrit le résultat qu'un décodeur conforme à la présente Recommandation | Norme internationale doit produire lors du décodage d'une page. A cette fin, il spécifie une série d'étapes qui aboutissent au résultat recherché. Un décodeur conforme n'a pas besoin de suivre exactement ces étapes mais il doit produire le même résultat que s'il les avait suivies.

Nous ne décrirons ici que les étapes suivies pour décoder une page isolée. Un décodeur conforme peut opérer sur plusieurs pages à la fois, du moment qu'il fournit le résultat correct pour chaque page.

Dans la description suivante, l'on partira de l'hypothèse simplificatrice que le décodeur possède un seul tampon de page, des tampons auxiliaires à utiliser pour décoder cette page et une mémoire additionnelle pour un dictionnaire. Les décodeurs possédant d'autres éléments sont autorisés à condition qu'ils produisent le même tampon de page que ce décodeur théorique.

A la fin du processus de décodage, le tampon de page contient le résultat du décodage de la page.

Chaque tampon auxiliaire est associé à un emplacement qui est celui du pixel supérieur gauche de ce tampon auxiliaire, par rapport au pixel supérieur gauche du tampon de page. Certains segments de région nécessitent l'emploi de tampons auxiliaires; d'autres peuvent être décodés directement dans le tampon de page. Voir 8.2 pour les détails sur la façon d'interpréter les combinaisons de segments d'image.

La mémoire de dictionnaire contient les informations obtenues par le décodage des segments de dictionnaire.

8.2 Composition d'une image de page

La matrice finale de chaque page est codée par 0, 1 ou plusieurs segments associés à cette page. Chaque segment de région décrit une partie du contenu d'une région rectangulaire de la page. Comme ces régions de page peuvent se chevaucher et comme certaines parties de la page peuvent être décrites comme des niveaux multiples de raffinement, il est important de définir quelles sont les règles de composition des segments de région. De même, comme un décodeur peut avoir à afficher des représentations intermédiaires d'une page sur la base d'informations partielles, il est utile de suggérer l'interprétation de pages partielles.

Comme décrit au 7.4.8, chaque page possède une valeur de pixel par défaut (**0** ou **1**) et un opérateur combinatoire sur quatre (OR, AND, XOR, XNOR), ces éléments étant spécifiés dans son segment d'informations de page. Chaque segment de région spécifie également un opérateur combinatoire qui lui est propre. Le bit-fanion "désactivation d'opérateur combinatoire de page", contenu dans le segment d'informations de page, spécifie si l'un des segments de région directe de la page a priorité sur l'opérateur combinatoire de page. Si ce bit est **0**, aucun segment de région directe, associé à cette page, n'a priorité sur l'opérateur combinatoire de page. Le décodeur peut utiliser cette information pour optimiser son décodage.

Le résultat du décodage d'un segment de région est une phototrame dont les dimensions et l'emplacement par rapport au tampon de page sont indiqués dans le champ d'information de ce segment de région.

Le contenu final du tampon de page que le décodeur doit produire comme résultat final du décodage d'une page est celui qui serait produit par les étapes suivantes:

1) Décoder le segment d'informations de page.

2) Créer le tampon de page, de la taille indiquée dans le segment d'informations de page.

   Si la hauteur de page est inconnue, cette création est impossible. Dans ce cas cependant, la page doit être découpée en bandes et la hauteur maximale de bande est spécifiée. Le tampon de page initial peut alors être créé, sa hauteur initiale étant égale à cette hauteur maximale de bande. A chaque apparition d'un segment de fin de bande, la hauteur du tampon de page peut être augmentée, de façon que la dernière rangée du nouveau tampon soit la hauteur de bande maximale plus la rangée finale de la bande précédente. Le segment de fin de page (ainsi que le dernier segment de fin de bande) permet la détermination de la hauteur réelle de la page.

   En variante, lorsque la hauteur de page est inconnue, le décodeur peut utiliser un tampon de taille fixe dont la hauteur est égale à la hauteur maximale de bande de la page. A chaque apparition d'un segment de fin de bande, le décodeur peut imprimer ou copier ailleurs toutes les rangées de ce tampon jusques et y compris la rangée finale de la bande. Il peut ensuite effacer ce tampon en prévision de la prochaine bande. Le décodeur peut suivre cette stratégie chaque fois que la page est découpée en bandes, même si la hauteur de page est connue d'avance.

   NOTE 1 – Les étapes ci-dessous peuvent être suivies quelle que soit la stratégie de découpage en bandes. Les restrictions imposées par le découpage en bandes garantissent que, lorsqu'un segment de fin de bande apparaît, aucune partie de la page située au-dessus ou au sommet de cette rangée finale de bande ne pourra être modifiée, de sorte que la présentation située au-dessous de cette rangée sera exprimée en termes de tampon de page (c'est-à-dire en dimensions complètes de la page), même si la hauteur de page n'est pas connue initialement.

3) Remplir le tampon de page avec la valeur par défaut des pixels de page.

4) Extraire le prochain segment de région associé à cette page.

5) Les cinq cas suivants peuvent se présenter:

   a) le segment de région est un segment de région directe immédiate. Dans ce cas, décoder ce segment de région, ce qui donne une phototrame. Combiner celle-ci avec le contenu actuel du tampon de page au moyen de l'opérateur combinatoire du segment de région;

   b) le segment de région est un segment de région directe intermédiaire. Dans ce cas, attribuer un nouveau tampon auxiliaire au moyen de la taille et de la localisation spécifiées dans le champ d'informations de ce segment de région. Ce tampon est initialement associé au segment de région. Décoder celui-ci et placer la phototrame qui en résulte dans le tampon auxiliaire;

   c) le segment de région est un segment de région immédiate de raffinement qui ne fait référence à aucun autre segment de région. Dans ce cas, le segment de région joue le rôle d'un raffinement d'une partie du tampon de page. Exécuter le raffinement indiqué par le segment de région sur la partie du tampon de page qui est spécifiée dans ce segment de région, en fonction des données contenues dans le segment de région de raffinement. Cette opération remplace une partie du tampon de page par une version raffinée;

   d) le segment de région est un segment de région immédiate de raffinement qui fait référence à un autre segment de région. Celui-ci doit être un segment de région intermédiaire déjà apparu qui n'a pas encore fait l'objet d'une référence par un segment de région de raffinement. Cet autre segment de région possède donc un tampon auxiliaire qui lui est associé. A noter que l'autre segment de région peut lui-même être un segment de région de raffinement intermédiaire. Exécuter l'opération de raffinement sur ce tampon auxiliaire, en fonction des données contenues dans le segment de région actuel puis combiner le tampon résultant avec le tampon de page au moyen de l'opérateur combinatoire du segment de région actuel, à l'emplacement associé au tampon auxiliaire, lequel est désenregistré;

   e) le segment de région est un segment de région intermédiaire de raffinement qui doit faire référence à un seul autre segment de région. Celui-ci doit être un segment de région intermédiaire déjà apparu qui n'a pas encore fait l'objet d'une référence par un segment de région de raffinement. Cet autre segment de région possède donc un tampon auxiliaire qui lui est associé. Exécuter l'opération de raffinement sur ce tampon auxiliaire, en fonction des données contenues dans le segment de région actuel. Remplacer le contenu précédent du tampon auxiliaire par la phototrame résultant du raffinement. Modifier l'association du tampon auxiliaire de façon qu'il soit désormais associé au segment de région actuel et qu'il ne soit plus associé à l'autre segment de région.

6) Répéter les étapes 4) et 5) jusqu'à ce qu'aucun segment de région ne soit plus associé à la page. A ce moment, tous les tampons auxiliaires qui ont été attribués doivent avoir été raffinés, insérés dans la page et désenregistrés, comme décrit à l'étape 5 d). Aucun tampon auxiliaire ne doit rester.

7) Le résultat de la décompression de cette page correspond au contenu final du tampon de page.

Les règles décrites à l'étape 5) sont tout à fait simples dans leur principe. Les segments de région immédiate doivent être insérés dans le tampon de page par insertion directe (segments directs, étape 5 a), par raffinement d'une partie du tampon de page (segments de raffinement ne faisant référence à aucun autre segment, étape 5 c), ou par raffinement puis insertion dans un tampon auxiliaire (segments de raffinement faisant référence à un autre segment, étape 5 d). Les segments de région intermédiaire impliquent la création d'un tampon auxiliaire contenant la matrice de région (segments directs, étape 5 b) ou le remplacement du contenu actuel d'un tampon auxiliaire (segments de raffinement, étape 5 e).

Quelques exemples d'application de ces règles sont donnés ci-dessous.

EXEMPLE 1 – Si la page ne contient pas de segments de région, le tampon de page est entièrement rempli avec la valeur par défaut des pixels de page.

EXEMPLE 2 – Le segment d'informations pour la page 1 spécifie que l'opérateur combinatoire par défaut de cette page est OR et que la valeur par défaut des pixels de page est **0**. Les segments de région associés à la page 1 sont, dans l'ordre, les suivants:

- le segment 3 qui est un segment de région alphanumérique immédiate sans pertes dont l'opérateur combinatoire externe est OR;
- le segment 4 qui est un segment de région générique immédiate sans pertes dont l'opérateur combinatoire externe est OR;
- le segment 6 qui est un segment de région de dégradé immédiate sans pertes dont l'opérateur combinatoire externe est OR.

La matrice de page correspondante peut être obtenue par décodage des segments 3, 4 et 6 et par insertion de chacun à son emplacement régional spécifié au moyen de l'opérateur OR, dans une phototrame contenant initialement **0** à chaque pixel. Noter que l'ordre de décodage puis d'insertion de ces trois segments n'a pas d'incidence sur la matrice de page résultante. De même, si le segment 3 possède un opérateur combinatoire interne OR et une valeur par défaut de pixel de **0**, il peut être représenté en insérant simplement les instances de symbole directement dans le tampon de page, sans qu'il soit nécessaire de le décoder en une matrice temporaire puis d'insérer cette matrice dans le tampon de page. Une observation similaire s'applique au segment 6.

EXEMPLE 3 – Le segment d'informations pour la page 2 spécifie que l'opérateur combinatoire par défaut de cette page est OR et que la valeur par défaut des pixels de page est **0**. Les segments de région associés à la page 2 sont, dans l'ordre, les suivants:

- le segment 7, qui est un segment de région alphanumérique intermédiaire;
- le segment 8, qui est un segment de région de matrice générique intermédiaire;
- le segment 13, qui est un segment de raffinement de région de matrice générique immédiate dont l'opérateur combinatoire externe est OR et qui fait référence au segment 8;
- le segment 14, qui est un segment de raffinement de région de matrice générique immédiate dont l'opérateur combinatoire externe est OR et qui fait référence au segment 7;
- le segment 19, qui est un segment de région alphanumérique immédiate dont l'opérateur combinatoire est OR;
- le segment 22, qui est un segment de région de matrice générique immédiate dont l'opérateur combinatoire est OR.

Le tampon de page correspondant est celui qui serait obtenu par les étapes suivantes:

1) remplir le tampon de page avec la valeur **0**;

2) décoder le segment 7 en un tampon auxiliaire;

3) décoder le segment 8 en un tampon auxiliaire;

4) raffiner le tampon auxiliaire du segment 8 en fonction des informations de raffinement contenues dans le segment 13, puis insérer le tampon raffiné dans le tampon de page au moyen de l'opérateur OR, le tampon auxiliaire étant désactivé immédiatement après;

5) raffiner le tampon auxiliaire du segment 7 en fonction des informations de raffinement contenues dans le segment 14 puis insérer le tampon raffiné dans le tampon de page au moyen de l'opérateur OR, le tampon auxiliaire étant désactivé immédiatement après;

6) décoder le segment 19 et insérer la matrice correspondante dans le tampon de page au moyen de l'opérateur OR;

7) décoder le segment 22 et insérer la matrice correspondante dans le tampon de page au moyen de l'opérateur OR.

Le résultat correct est également obtenu *quel que soit l'ordre* dans lequel les étapes 4) à 7) sont exécutées. Un décodeur conforme est donc libre de choisir un ordre quelconque pour décoder ces étapes. En fait, un ordre quelconque des étapes 2) à 7) produit le résultat correct, à condition que l'étape 2) soit effectuée avant l'étape 5) et l'étape 3) avant l'étape 4.

EXEMPLE 4 – Si une page contient plusieurs segments de région immédiate à codage direct qui n'ont pas priorité sur l'opérateur combinatoire de la page et si celle-ci contient un segment de région de raffinement immédiate qui ne fait pas référence à d'autres segments, le tampon de page résultant est celui qui serait obtenu par les opérations suivantes:

- remplissage du tampon de page avec la valeur par défaut des pixels de page;
- insertion de tous les segments de région à codage direct qui précèdent le segment de région de raffinement;
- raffinement de la partie de la région qui est couverte par le segment de région de raffinement;
- insertion de tous les segments de région à codage direct qui suivent le segment de région de raffinement.

Dans ce cas, l'ordre d'insertion est déterminant: tous les segments immédiats qui précèdent le segment de raffinement doivent être insérés avant celui-ci, lequel doit être inséré avant les segments immédiats qui le suivent.

NOTE 2 – Dans certains cas, le décodeur peut avoir à afficher une forme intermédiaire de la page. Par exemple, il peut avoir à présenter à l'utilisateur un affichage progressif du contenu de la page au fur et à mesure que les segments de page sont reçus par un moyen de transmission donné. Les éventuelles matrices de page intermédiaires qui seront ainsi affichées relèvent entièrement du décodeur et ne sont pas spécifiées par la présente Recommandation | Norme internationale.

Un décodeur peut utiliser une stratégie consistant à extraire le contenu actuel du tampon de page ainsi que de tous tampons auxiliaires actuellement actifs et à combiner tous ces tampons au moyen de l'opérateur combinatoire par défaut de la page, puis à afficher cet ensemble pour l'utilisateur. Si l'opérateur combinatoire de page est XOR ou XNOR, cette combinaison peut être effectuée de façon réversible. Elle peut alors être introduite dans le tampon de page réel puis être annulée après avoir été présentée à l'utilisateur. Si l'opérateur combinatoire de page est OR ou AND, cette combinaison n'est pas réversible et un tampon supplémentaire est requis pour mémoriser les résultats de cette combinaison.

La description par étapes qui précède ne vise à spécifier que les résultats de la décompression. Un décodeur conforme peut suivre les étapes à son gré, à condition que le tampon de page final soit identique à celui qui aurait été obtenu en suivant les étapes recommandées.

EXEMPLE 5 – Un décodeur peut remarquer qu'un segment de région intermédiaire fait référence à une région de la page qui n'est pas chevauchée par un autre segment de région, de sorte qu'il puisse pratiquement ne pas attribuer de tampon auxiliaire à ce segment de région mais qu'il puisse utiliser le tampon de page immédiatement. Il ne peut le faire que s'il est certain que cela ne modifiera pas les résultats définitifs du décodage des segments de région de la page.

Annexe A

Procédure de décodage arithmétique d'un entier

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

A.1 Description générale

La présente Recommandation | Norme internationale utilise un certain nombre de procédures de décodage arithmétique afin de décoder des valeurs d'entier. Il s'agit des suivantes:

IAAI Procédure de décodage arithmétique d'entier utilisée pour décoder le nombre d'instances de symbole contenues dans une agrégation

IADH Procédure de décodage arithmétique d'entier utilisée pour décoder la différence de hauteur entre deux classes de hauteur

IADS Procédure de décodage arithmétique d'entier utilisée pour décoder la coordonnée S de la deuxième instance de symbole d'une bande et les instances suivantes

IADT Procédure de décodage arithmétique d'entier utilisée pour décoder la coordonnée T de la deuxième instance de symbole d'une bande et les instances suivantes

IADW Procédure de décodage arithmétique d'entier utilisée pour décoder la différence de largeur entre deux symboles d'une classe de hauteur

IAEX Procédure de décodage arithmétique d'entier utilisée pour décoder des fanions d'exportation

IAFS Procédure de décodage arithmétique d'entier utilisée pour décoder la coordonnée S de la première instance de symbole dans une bande

IAID Procédure de décodage arithmétique d'entier utilisée pour décoder les identificateurs de symboles d'instances de symboles.

IAIT Procédure de décodage arithmétique d'entier utilisée pour décoder la coordonnée T des instances de symbole d'une bande

IARDH Procédure de décodage arithmétique d'entier utilisée pour décoder la différence de hauteur de raffinements d'instance de symbole

IARDW Procédure de décodage arithmétique d'entier utilisée pour décoder la différence de largeur de raffinements d'instance de symbole

IARDX Procédure de décodage arithmétique d'entier utilisée pour décoder les valeurs de décalage X de raffinements d'instance de symbole

IARDY Procédure de décodage arithmétique d'entier utilisée pour décoder les valeurs de décalage Y de raffinements d'instance de symbole

IARI Procédure de décodage arithmétique d'entier utilisée pour décoder le bit R_I d'instances de symbole

Chacune de ces procédures est utilisée pour décoder des valeurs d'entier (qui peuvent inclure la valeur hors bande (OOB). Le codage d'un entier est fondé sur un arbre décisionnel.

L'invocation d'une procédure de décodage arithmétique d'entier implique le décodage d'une séquence d'éléments binaires où chaque bit est décodé au moyen d'un contexte formé par les bits déjà décodés dans cette invocation. Chaque contexte de chaque procédure de décodage arithmétique d'entier possède son propre estimateur adaptatif de probabilité, qui est utilisé par le codeur arithmétique sous-jacent comme décrit dans l'Annexe E. La séquence de bits décodée est interprétée pour former une valeur.

Le Tableau A.1 est utilisé par toutes les procédures de décodage arithmétique d'entier, sauf la procédure IAID.

A.2 Procédure de décodage de valeurs (sauf procédure IAID)

L'organigramme de la Figure A.1 est utilisé dans le cadre de la procédure de décodage. Il produit deux valeurs: V et S. Le résultat de la procédure de décodage arithmétique d'entier est égal à:

- V si $S = \mathbf{0}$
- –V si $S = \mathbf{1}$ et $V > 0$
- OOB si $S = \mathbf{1}$ et $V = 0$

V représente donc la valeur absolue de l'entier décodé et S représente son signe. La valeur par ailleurs redondante –0 est interprétée comme étant une valeur hors bande (OOB).

Dans la Figure A.1, chaque bit est décodé dans un contexte formé à partir de la procédure particulière de décodage arithmétique d'entier qui est invoquée et à partir des bits déjà décodés lors de l'invocation de cette procédure de décodage. Ce contexte est formé comme suit:

1) Poser:

$$PREV = 1$$

2) Suivre l'organigramme de la Figure A.1. Décoder chaque bit avec CX = "IAx + PREV" où le terme "IAx" représente l'identificateur de la procédure actuelle de décodage arithmétique d'entier, où "+" représente une concaténation et où les 9 bits de droite de PREV sont utilisés.

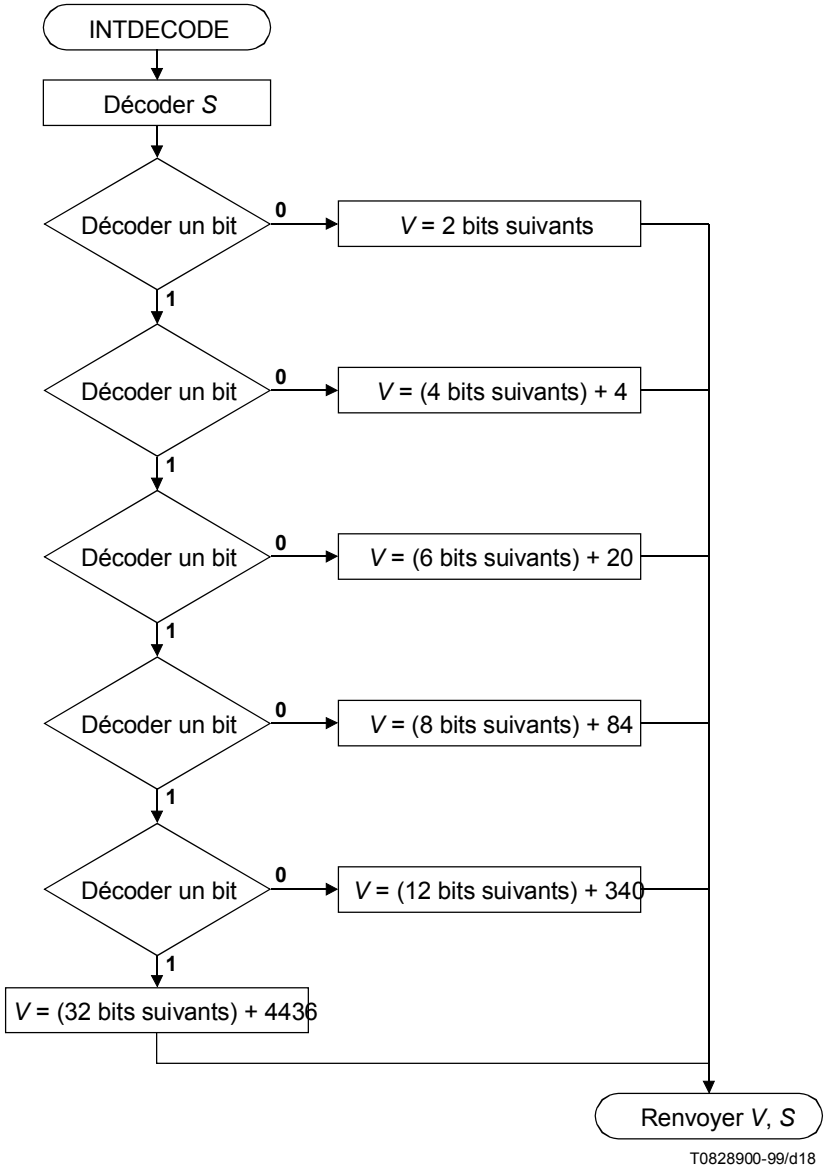


Figure A.1 – Organigramme des procédures de décodage arithmétique d'entiers (sauf IAID**)**

3) Une fois que chaque bit est décodé et si PREV < 256, poser:

$$PREV = (PREV << 1)\ OR\ D$$

Sinon, poser:

$$PREV = (((PREV << 1)\ OR\ D)\ AND\ 511)\ OR\ 256$$

où D représente la valeur du bit qui vient d'être décodé.

PREV contiendra donc toujours les valeurs des 8 bits décodés le plus récemment, plus **1** bit de départ qui sert à indiquer le nombre de bits déjà décodés.

4) La séquence des bits décodés, interprétée conformément au Tableau A.1, indique la valeur qui est le résultat de cette invocation de la procédure de décodage arithmétique d'entier.

Noter que chaque type de données et que chaque procédure de décodage arithmétique d'entier utilise un ensemble de contextes distinct: par exemple, les contextes utilisés pour la procédure IAFS sont distincts des contextes utilisés pour la procédure IADW.

Tableau A.1 – Table pour la procédure de décodage d'entiers arithmétiques

VAL	Encodage
0 . . . 3	**00** + VAL codage sur 2 bits
–1	**1001**
–3 . . . –2	**101** + (–VAL – 2) codage sur 1 bit
4 . . . 19	**010** + (VAL – 4) codage sur 4 bits
–19 . . . –4	**110** + (–VAL – 4) codage sur 4 bits
20 . . . 83	**0110** + (VAL – 20) codage sur 6 bits
–83 . . . –20	**1110** + (–VAL – 20) codage sur 6 bits
84 . . . 339	**01110** + (VAL – 84) codage sur 8 bits
–339 . . . –84	**11110** + (–VAL – 84) codage sur 8 bits
340 . . . 4435	**011110** + (VAL – 340) codage sur 12 bits
–4435 . . . –340	**111110** + (–VAL – 340) codage sur 12 bits
4436 . . . ∞	**011111** + (VAL – 4436) codage sur 32 bits
–∞ . . . –4436	**111111** + (–VAL – 4436) codage sur 32 bits
OOB	**1000**

EXEMPLE – Une invocation de la procédure IADW peut se dérouler comme suit:

- Mettre CX à "IADW**000000001**". Cette valeur désigne un estimateur adaptatif de probabilité particulièrement identifié. Décoder un bit et supposer que la valeur décodée (D) est **0**.
- Mettre CX à IADW**000000010**, décoder un bit et supposer que la valeur décodée est **1**.
- Mettre CX à IADW**000000101**, décoder un bit et supposer que la valeur décodée est **0**.
- Mettre CX à IADW**000001010**, décoder un bit et supposer que la valeur décodée est **1**.
- Mettre CX à IADW**000010101**, décoder un bit et supposer que la valeur décodée est **0**.
- Mettre CX à IADW**000101010**, décoder un bit et supposer que la valeur décodée est **0**.
- Mettre CX à IADW**001010100**, décoder un bit et supposer que la valeur décodée est **0**.
- La séquence des bits déjà décodés est **0101000**. Conformément au Tableau A.1 et à la Figure A.1, cela correspond à la valeur 12 ($S = 0$, $V=12$) qui est le résultat de cette invocation de la procédure IADW.

Un contexte est identifié par un nom de procédure de décodage arithmétique d'entier et par une séquence de 9 bits. Chaque procédure de décodage arithmétique d'entier nécessite donc 512 octets de capacité dans sa mémoire de contextes.

A.3 Procédure de décodage IAID

Cette procédure de décodage est différente de toutes les autres procédures de décodage arithmétique d'entier. Elle utilise des représentations de longueur fixe des valeurs en cours de décodage et ne limite pas le nombre de bits déjà décodés qui sont utilisés dans le cadre du contexte. La longueur est égale à **SBSYMCODELEN**. Cette procédure de décodage n'est invoquée qu'à partir de la procédure de décodage de région alphanumérique, de sorte qu'au moment de l'invocation, la longueur **SBSYMCODELEN** est connue.

La procédure IAID de décodage arithmétique d'entier est la suivante:

1) Poser:

$$PREV = 1$$

2) Décoder les bits de longueur **SBSYMCODELEN** comme suit:

   a) décoder un bit avec CX égal à "IAID + PREV" où "+" représente une concaténation et où l'on utilise les **SBSYMCODELEN** + 1 bit situés le plus à droite de PREV;

   b) une fois que chaque bit est décodé, poser:

$$PREV = (PREV << 1) \text{ OR } D$$

   où D représente la valeur du bit qui vient d'être décodé.

   Ainsi, PREV contient toujours les valeurs de tous les bits décodés jusque là, plus 1 bit de départ qui sert à indiquer le nombre de bits déjà décodés.

3) Une fois que les bits **SBSYMCODELEN** ont été décodés, poser:

$$PREV = PREV - 2^{\mathbf{SBSYMCODELEN}}$$

   Cette étape a pour effet d'effacer le bit supérieur (bit de départ **1**) de PREV avant de le retourner.

4) Le contenu de PREV est le résultat de cette invocation de la procédure de décodage IAID.

Le nombre de contextes requis est $2^{\mathbf{SBSYMCODELEN}}$. Ce nombre est inférieur à deux fois l'identificateur de symbole le plus grand. La capacité de mémoire requise pour les contextes peut donc être calculée à partir du nombre de symboles. Elle n'est normalement pas supérieure à 2 octets par symbole.

EXEMPLE – Supposons que **SBSYMCODELEN** = 3. Une invocation de la procédure IAID peut se dérouler comme suit:

- Utiliser l'estimateur adaptatif de probabilité identifié en posant CX = "IAID**0001**" afin de décoder un bit. Supposer que la valeur ainsi décodée soit **0**.
- Au moyen de CX = IAID**0010**, décoder un bit et supposer que la valeur décodée est **1**.
- Au moyen de CX = IAID**0101**, décoder un bit et supposer que la valeur décodée est **0**.
- A ce point, PREV = **1010**. Appliquer l'étape 3). PREV devient **010**. Le résultat de cette invocation de la procédure de décodage IAID est donc la valeur **010** ou (en notation décimale), le nombre 2.

L'identification du contexte utilisé dépend de la valeur de **SBSYMCODELEN**. Dans tous les cas, les contextes du codeur arithmétique seront réinitialisés entre les modifications du paramètre **SBSYMCODELEN**, qui ne change jamais pendant le décodage d'un segment isolé (mais qui peut changer entre deux segments).

Annexe B

Procédure de décodage par table de Huffman

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

B.1 Description générale

Des tables de séquences codées peuvent être utilisées pour coder n'importe quel type de données dans les codeurs à variante de Huffman. A de nombreux points d'utilisation d'une table, le codeur a la possibilité d'utiliser l'une des tables normalisées ou d'envoyer sa propre table. Un segment de table codée permet d'envoyer une telle table spéciale. La table codée est une liste de séquences linéaires codées dont chacune décrit comment coder une seule valeur ou une valeur extraite d'une étendue spécifiée. Une table peut, facultativement, être utilisée pour coder une valeur OOB qui représente un signal hors bande par rapport à la procédure de décodage qui utilise cette table.

B.2 Structure d'une table de codage

La Figure B.1 montre la structure interne d'une table de Huffman codée. Celle-ci se compose d'une série de lignes de table donc chacune décrit le codage d'une étendue de valeurs numériques. Deux lignes tabulaires additionnelles peuvent aussi exister afin de coder des étendues "extensibles". La plus petite valeur qui puisse être codée dans une table décrite conformément à la présente Recommandation | Norme internationale est -2147483648 (-2^{31}) et la plus grande valeur est 2147483647 ($2^{31} - 1$), de sorte que ces étendues ne sont pas vraiment extensibles. Il est également possible qu'une ligne additionnelle code une valeur hors bande (OOB).

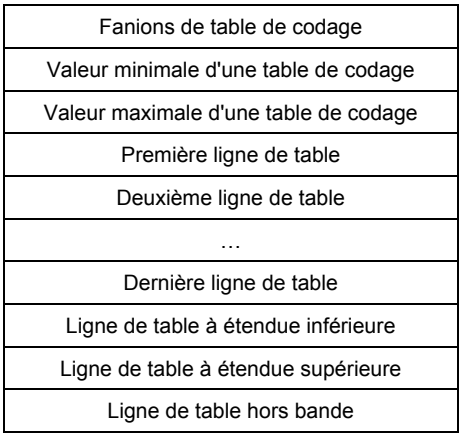


Figure B.1 – Structure codée d'une table de Huffman

Chaque ligne de la table spécifie la longueur du préfixe qui lui est associé ainsi que le nombre de bits qui suivent ce préfixe pour coder une valeur.

Lors du décodage d'une table de Huffman codée, un décodeur doit décoder la table qui est produite par les étapes suivantes:

1) Décoder le champ de fanions de table de codage comme décrit au B.2.1. Cela détermine les valeurs HTOOB, HTPS et HTRS.

2) Décoder le champ de fanions de table de codage comme décrit au B.2.2. Soit HTLOW la valeur ainsi décodée.

3) Décoder le champ de fanions de table de codage comme décrit au B.2.3. Soit HTHIGH la valeur ainsi décodée.

4) Poser:

$$\begin{aligned} \text{CURRANGELOW} &= \text{HTLOW} \\ \text{NTEMP} &= 0 \end{aligned}$$

5) Décoder chaque ligne de table comme suit:
   a) lire HTPS bits. Mettre PREFLEN[NTEMP] à la valeur ainsi décodée;
   b) lire HTRS bits. Soit RANGELEN[NTEMP] la valeur ainsi décodée;
   c) poser:

$$\begin{aligned} \text{RANGELOW[NTEMP]} &= \text{CURRANGELOW} \\ \text{CURRANGELOW} &= \text{CURRANGELOW} + 2^{\text{RANGELEN[NTEMP]}} \\ \text{NTEMP} &= \text{NTEMP} + 1 \end{aligned}$$

   d) si CURRANGELOW ≥ HTHIGH, passer à l'étape 6).
6) Lire HTPS bits. Soit LOWPREFLEN la valeur lue.
7) Poser:

$$\begin{aligned} \text{PREFLEN[NTEMP]} &= \text{LOWPREFLEN} \\ \text{RANGELEN[NTEMP]} &= 32 \\ \text{RANGELOW[NTEMP]} &= \text{HTLOW} - 1 \\ \text{NTEMP} &= \text{NTEMP} + 1 \end{aligned}$$

   Il s'agit de la ligne de table à étendue inférieure pour cette table.
8) Lire HTPS bits. Soit HIGHPREFLEN la valeur lue.
9) Poser:

$$\begin{aligned} \text{PREFLEN[NTEMP]} &= \text{HIGHPREFLEN} \\ \text{RANGELEN[NTEMP]} &= 32 \\ \text{RANGELOW[NTEMP]} &= \text{HTHIGH} \\ \text{NTEMP} &= \text{NTEMP} + 1 \end{aligned}$$

   Il s'agit de la ligne de table à étendue supérieure pour cette table.
10) Si HTOOB est **1**,
   a) lire HTPS bits. Soit OOBPREFLEN la valeur lue;
   b) poser:

$$\begin{aligned} \text{PREFLEN[NTEMP]} &= \text{OOBPREFLEN} \\ \text{NTEMP} &= \text{NTEMP} + 1 \end{aligned}$$

   Il s'agit de la ligne de table hors bande pour cette table. Noter qu'aucune étendue n'est associée à cette valeur.
11) Créer les codes de préfixe au moyen de l'algorithme décrit en B.3.

B.2.1 Fanions d'une table de codage

Les bits de ce champ de 1 octet ont les définitions suivantes:

Bit 0 HTOOB. Si ce bit est **1**, la table peut coder pour une valeur hors bande.

Bits 1 et 3 Nombre de bits utilisés dans les champs de longueur de préfixe d'une ligne de table. La valeur de HTPS est celle de ce champ plus 1.

Bits 4 à 6 Nombre de bits utilisés dans les champs de longueur d'étendue de ligne de table codée. La valeur de HTRS est celle de ce champ plus 1.

Bit 7 Champ réservé qui doit être zéro.

B.2.2 Valeur minimale d'une table de codage

Ce champ signé de 4 octets est la limite inférieure de la première ligne de la table de codage.

B.2.3 Valeur maximale d'une table de codage

Ce champ signé de 4 octets est la limite supérieure de la dernière ligne normale de la table de codage.

B.3 Attribution des codes de préfixe

Sur la base de la table des longueurs de code de préfixe (PREFLEN) et du nombre de codes à attribuer (NTEMP), cet algorithme attribue un unique code de préfixe à chaque ligne de table, de la longueur indiquée par PREFLEN pour cette ligne de table.

Noter que la valeur 0 de PREFLEN indique que la ligne de table correspondante n'est jamais utilisée.

1) Construire un histogramme dans la série tabulaire LENCOUNT afin de compter le nombre d'apparitions de chaque valeur de longueur de préfixe dans la série tabulaire PREFLEN. Le paramètre LENCOUNT [*I*] indique le nombre d'apparitions de la valeur *I* dans la série tabulaire PREFLEN.
2) Soit LENMAX la plus grande valeur pour laquelle LENCOUNT[LENMAX] > 0. Poser:

$$\begin{aligned} \text{CURLEN} &= 1 \\ \text{FIRSTCODE}[0] &= 0 \\ \text{LENCOUNT}[0] &= 0 \end{aligned}$$

3) Tant que CURLEN ≤ LENMAX, effectuer les opérations suivantes:
   a) poser:

$$\begin{aligned} \text{FIRSTCODE}[\text{CURLEN}] &= (\text{FIRSTCODE}[\text{CURLEN} - 1] + \text{LENCOUNT}[\text{CURLEN} - 1]) \times 2 \\ \text{CURCODE} &= \text{FIRSTCODE}[\text{CURLEN}] \\ \text{CURTEMP} &= 0 \end{aligned}$$

   b) tant que CURTEMP < NTEMP, effectuer les opérations suivantes:
      i) si PREFLEN[CURTEMP] = CURLEN, poser:

$$\begin{aligned} \text{CODES}[\text{CURTEMP}] &= \text{CURCODE} \\ \text{CURCODE} &= \text{CURCODE} + 1 \end{aligned}$$

      ii) poser CURTEMP = CURTEMP + 1;
   c) poser:

$$\text{CURLEN} = \text{CURLEN} + 1$$

Une fois que cet algorithme a été exécuté, le numéro de ligne de table *I* a fait l'objet de l'attribution d'un code d'une longueur de PREFLEN[*I*] bits dont la valeur est mémorisée dans les PREFLEN[*I*] bits de poids faible de CODE[*I*], à moins que PREFLEN[*I*] ne soit égal à zéro, auquel cas cette ligne de table n'a fait l'objet de l'attribution d'aucun code.

B.4 Utilisation d'une table de Huffman

Pour décoder une valeur utilisant une table de Huffman, exécuter les étapes suivantes:

1) Lire un seul bit à la fois jusqu'à ce que la chaîne binaire lue corresponde au code attribué à une des lignes de la table, ce qui est possible étant donné qu'aucun code ne forme le préfixe d'un autre code. Soit *I* l'indice de la ligne de table dont le code a été ainsi décodé.

2) Lire RANGELEN[*I*] bits. Soit HTOFFSETT la valeur ainsi lue.

3) Si HTOOB = **1** pour cette table et si la ligne de table *I* est la ligne hors bande pour cette table, poser:

$$\text{HTVAL} = \text{OOB}$$

4) Sinon et si la ligne de table *I* est la ligne de table à étendue inférieure pour cette table, poser:

$$\text{HTVAL} = \text{RANGELOW}[I] - \text{HTOFFSET}$$

5) Sinon poser:

$$\text{HTVAL} = \text{RANGELOW}[I] + \text{HTOFFSET}$$

La valeur de HTVAL est celle qui est décodée au moyen de cette table. Noter qu'il peut s'agir d'une valeur numérique ou de la valeur spéciale OOB.

EXEMPLE – Le codage pour la table B.1 peut être la séquence d'octets suivante, exprimée en notation hexadécimale:

```
0x42  0x00  0x00  0x00  0x00  0x00  0x01
0x01  0x10  0x49  0x23  0x81  0x80
```

Le décodage de cette séquence au moyen de l'algorithme de B.2 se déroule comme suit:

- (0x42), le champ des fanions de table de codage, se subdivise en sous-champs exprimés en notation binaire par **0 100 001 0**, dont le décodage produit les attributions suivantes:

$$\begin{aligned}\text{HTOOB} &= \mathbf{0}\\ \text{HTPS} &= 2\\ \text{HTRS} &= 5\end{aligned}$$

- (`0x00000000`), le champ de la valeur minimale de la table de codage et la valeur de HTLOW.
- (`0X00010110`) ou 65808 en décimal), le champ de la valeur maximale de la table de codage et la valeur de HTHIGH.
- Trois lignes de table, la ligne de table à étendue inférieure et la ligne de table à étendue supérieure sont codées sous la forme de la séquence d'octets `0x49 0x23 0x81 0x80` ou, en notation binaire, par **01001001 00100011 10000001 10000000**. Cette chaîne binaire est subdivisée en lignes de table comme suit:

01 00100 Les deux premiers (HTPS) bits de cette ligne de table indiquent une longueur de préfixe de 1 et les cinq derniers (HTRS) bits de cette ligne de table indiquent une étendue de longueur 4.

10 01000 Cette ligne de table a une longueur de préfixe de 2 et une étendue de longueur 8.

11 10000 Cette ligne de table a une longueur de préfixe de 3 et une étendue de longueur 16.

00 Cette ligne de table a une longueur de préfixe de 0, indiquant que cette ligne n'est pas utilisée.

11 La ligne de table d'étendue supérieure a une longueur de préfixe de 3.

0000000 Sept bits de bourrage pour remplir le dernier octet.

Après le décodage de ces lignes de table, la valeur de NTEMP est 5. Les séries tabulaires PREFLEN, RANGELEN et RANGELOW sont les suivantes:

PREFLEN	1	2	3	0	3
RANGELEN	4	8	16	32	32
RANGELOW	0	16	272	–1	65808

L'application de l'algorithme B.3 à ces séries tabulaires produit la série tabulaire binaire de codes suivante:

CODES **0** **10** **110** X **111**

dans laquelle X indique que la ligne de table à étendue inférieure n'a pas fait l'objet de l'attribution d'un code. Le code de préfixe **0** précède donc un champ de 4 bits qui code une valeur comprise entre 0 et 15. Le code de préfixe **10** précède un champ de 8 bits qui code une valeur comprise entre 16 et 271, et ainsi de suite, comme indiqué dans le Tableau B.1.

B.5 Tables de Huffman normalisées

Ce paragraphe présente quelques tables de Huffman normalisées qui peuvent être utilisées dans les contextes appropriés sans avoir fait l'objet d'une transmission préalable.

Chaque table de Huffman est présentée sous une forme analogue à la transmission de table susmentionnée. Le paramètre de table HTOOB est indiqué (les valeurs HTPS, HTRS, HTLOW et HTHIGH peuvent être déduites des valeurs contenues dans la table), suivi d'une liste des lignes de table indiquant l'étendue à laquelle chaque ligne de table s'applique, la longueur de préfixe de ligne de table, la longueur d'étendue de ligne de table et le codage proprement dit (préfixe et valeur de base) pour chaque ligne de table. Ces lignes de table sont suivies d'une ligne de table à étendue inférieure et d'une ligne de table à étendue supérieure et, facultativement (selon la valeur de HTOOB) d'une ligne de table hors bande. Dans certains cas, les lignes de table à étendue inférieure ou supérieure sont omises des tables représentées afin d'indiquer que ces lignes ne sont pas utilisées dans la table (et feront l'objet de l'attribution d'une valeur zéro de PREFLEN).

Tableau B.1 – Table de Huffman normalisée A

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
0 . . . 15	1	4	**0** + VAL codage sur 4 bits
16 . . . 271	2	8	**10** + (VAL – 16) codage sur 8 bits
272 . . . 65807	3	16	**110** + (VAL – 272) codage sur 16 bits
65808 . . . ∞	3	32	**111** + (VAL – 65808) codage sur 32 bits

Tableau B.2 – Table de Huffman normalisée B

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codage
0	1	0	**0**
1	2	0	**10**
2	3	0	**110**
3 . . . 10	4	3	**1110** + (VAL – 3) codage sur 3 bits
11 . . . 74	5	6	**11110** + (VAL – 11) codage sur 6 bits
75 . . . ∞	6	32	**111110** + (VAL – 75) codage sur 32 bits
OOB	6		**111111**

Tableau B.3 – Table de Huffman normalisée C

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codage
–256 . . . –1	8	8	**11111110** + (VAL + 256) codage sur 8 bits
0	1	0	**0**
1	2	0	**10**
2	3	0	**110**
3 . . . 10	4	3	**1110** + (VAL – 3) codage sur 3 bits
11 . . . 74	5	6	**11110** + (VAL – 11) codage sur 6 bits
–∞ . . . –257	8	32	**11111111** + (–257 – VAL) codage sur 32 bits
75 . . . ∞	7	32	**1111110** + (VAL – 75) codage sur 32 bits
OOB	6		**111110**

Tableau B.4 – Table de Huffman normalisée D

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
1	1	0	**0**
2	2	0	**10**
3	3	0	**110**
4 . . . 11	4	3	**1110** + (VAL – 4) codage sur 3 bits
12 . . . 75	5	6	**11110** + (VAL – 12) codage sur 6 bits
76 . . . ∞	5	32	**11111** + (VAL – 76) codage sur 32 bits

Tableau B.5 – Table de Huffman normalisée E

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
–255 . . . 0	7	8	**1111110** + (VAL + 255) codage sur 8 bits
1	1	0	**0**
2	2	0	**10**
3	3	0	**110**
4 . . . 11	4	3	**1110** + (VAL – 4) codage sur 3 bits
12 . . . 75	5	6	**11110** + (VAL – 12) codage sur 6 bits
–∞ . . . –256	7	32	**1111111** + (–256 – VAL) codage sur 32 bits
76 . . . ∞	6	32	**111110** + (VAL – 76) codage sur 32 bits

Tableau B.6 – Table de Huffman normalisée F

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
–2048 . . . –1025	5	10	**11100** + (VAL + 2048) codage sur 10 bits
–1024 . . . –513	4	9	**1000** + (VAL + 1024) codage sur 9 bits
–512 . . . –257	4	8	**1001** + (VAL + 512) codage sur 8 bits
–256 . . . –129	4	7	**1010** + (VAL + 256) codage sur 7 bits
–128 . . . –65	5	6	**11101** + (VAL + 128) codage sur 6 bits
–64 . . . –33	5	5	**11110** + (VAL + 64) codage sur 5 bits
–32 . . . –1	4	5	**1011** + (VAL + 32) codage sur 5 bits
0 . . . 127	2	7	**00** + VAL codage sur 7 bits
128 . . . 255	3	7	**010** + (VAL – 128) codage sur 7 bits
256 . . . 511	3	8	**011** + (VAL – 256) codage sur 8 bits
512 . . . 1023	4	9	**1100** + (VAL – 512) codage sur 9 bits
1024 . . . 2047	4	10	**1101** + (VAL – 1024) codage sur 10 bits
–∞ . . . –2049	6	32	**111110** + (–2049 – VAL) codage sur 32 bits
2048 . . . ∞	6	32	**111111** + (VAL – 2048) codage sur 32 bits

Tableau B.7 – Table de Huffman normalisée G

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
–1024 . . . –513	4	9	**1000** + (VAL + 1024) codage sur 9 bits
–512 . . . –257	3	8	**000** + (VAL + 512) codage sur 8 bits
–256 . . . –129	4	7	**1001** + (VAL + 256) codage sur 7 bits
–128 . . . –65	5	6	**11010** + (VAL + 128) codage sur 6 bits
–64 . . . –32	5	5	**11011** + (VAL + 64) codage sur 5 bits
–32 . . . –1	4	5	**1010** + (VAL + 32) codage sur 5 bits
0 . . . 31	4	5	**1011** + VAL codage sur 5 bits
32 . . . 63	5	5	**11100** + (VAL – 32) codage sur 5 bits
64 . . . 127	5	6	**11101** + (VAL – 64) codage sur 6 bits
128 . . . 255	4	7	**1100** + (VAL – 128) codage sur 7 bits
256 . . . 511	3	8	**001** + (VAL – 256) codage sur 8 bits
512 . . . 1023	3	9	**010** + (VAL – 512) codage sur 9 bits
1024 . . . 2047	3	10	**011** + (VAL – 1024) codage sur 10 bits
–∞ . . . –1025	5	32	**11110** + (–1025 – VAL) codage sur 32 bits
2048 . . . ∞	5	32	**11111** + (VAL – 2048) codage sur 32 bits

Tableau B.8 – Table de Huffman normalisée H

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codage
–15 . . . –8	8	3	**11111100** + (VAL + 15) codage sur 3 bits
–7 . . . –6	9	1	**111111100** + (VAL + 7) codage sur 1 bit
–5 . . . –4	8	1	**11111101** + (VAL + 5) codage sur 1 bit
–3	9	0	**111111101**
–2	7	0	**1111100**
–1	4	0	**1010**
0 . . . 1	2	1	**00** + VAL codage sur 1 bit
2	5	0	**11010**
3	6	0	**111010**
4 . . . 19	3	4	**100** + (VAL – 4) codage sur 4 bits
20 . . . 21	6	1	**111011** + (VAL – 20) codage sur 1 bit
22 . . . 37	4	4	**1011** + (VAL – 22) codage sur 4 bits
38 . . . 69	4	5	**1100** + (VAL – 38) codage sur 5 bits
70 . . . 133	5	6	**11011** + (VAL – 70) codage sur 6 bits
134 . . . 261	5	7	**11100** + (VAL – 134) codage sur 7 bits
262 . . . 389	6	7	**111100** + (VAL – 262) codage sur 7 bits
390 . . . 645	7	8	**1111101** + (VAL – 390) codage sur 8 bits
646 . . . 1669	6	10	**111101** + (VAL – 646) codage sur 10 bits
–∞ . . . –16	9	32	**111111110** + (–16 – VAL) codage sur 32 bits
1670 . . . ∞	9	32	**111111111** + (VAL – 1670) codage sur 32 bits
OOB	2		**01**

Tableau B.9 – Table de Huffman normalisée I

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codage
–31 . . . –16	8	4	**11111100** + (VAL + 31) codage sur 4 bits
–15 . . . –12	9	2	**111111100** + (VAL + 15) codage sur 2 bits
–11 . . . –8	8	2	**11111101** + (VAL + 11) codage sur 2 bits
–7 . . . –6	9	1	**111111101** + (VAL + 7) codage sur 1 bit
–5 . . . –4	7	1	**1111100** + (VAL + 5) codage sur 1 bit
–3 . . . –2	4	1	**1010** + (VAL + 3) codage sur 1 bit
–1 . . . 0	3	1	**010** + (VAL + 1) codage sur 1 bit
1 . . . 2	3	1	**011** + (VAL – 1) codage sur 1 bit
3 . . . 4	5	1	**11010** + (VAL – 3) codage sur 1 bit
5 . . . 6	6	1	**111010** + (VAL – 5) codage sur 1 bit
7 . . . 38	3	5	**100** + (VAL – 7) codage sur 5 bits
39 . . . 42	6	2	**111011** + (VAL – 39) codage sur 2 bits
43 . . . 74	4	5	**1011** + (VAL – 43) codage sur 5 bits
75 . . . 138	4	6	**1100** + (VAL – 75) codage sur 6 bits
139 . . . 266	5	7	**11011** + (VAL – 139) codage sur 7 bits
267 . . . 522	5	8	**11100** + (VAL – 267) codage sur 8 bits
523 . . . 778	6	8	**111100** + (VAL – 523) codage sur 8 bits
779 . . . 1290	7	9	**1111101** + (VAL – 779) codage sur 9 bits
1291 . . . 3338	6	11	**111101** + (VAL – 1291) codage sur 11 bits
–∞ . . . –32	9	32	**111111110** + (–32 – VAL) codage sur 32 bits
3339 . . . ∞	9	32	**111111111** + (VAL – 3339) codage sur 32 bits
OOB	2		**00**

Tableau B.10 – Table de Huffman normalisée J

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codage
–21 . . . –6	7	4	**1111010** + (VAL + 21) codage sur 4 bits
–5	8	0	**11111100**
–4	7	0	**1111011**
–3	5	0	**11000**
–2 . . . 1	2	2	**00** + (VAL + 2) codage sur 2 bits
2	5	0	**11001**
3	6	0	**110110**
4	7	0	**1111100**
5	8	0	**11111101**
6 . . . 69	2	6	**01** + (VAL – 6) codage sur 6 bits
70 . . . 101	5	5	**11010** + (VAL – 70) codage sur 5 bits
102 . . . 133	6	5	**110111** + (VAL – 102) codage sur 5 bits
134 . . . 197	6	6	**111000** + (VAL – 134) codage sur 6 bits
198 . . . 325	6	7	**111001** + (VAL – 198) codage sur 7 bits
326 . . . 581	6	8	**111010** + (VAL – 326) codage sur 8 bits
582 . . . 1093	6	9	**111011** + (VAL – 582) codage sur 9 bits
1094 . . . 2117	6	10	**111100** + (VAL – 1094) codage sur 10 bits
2118 . . . 4165	7	11	**1111101** + (VAL – 2118) codage sur 11 bits
–∞ . . . –22	8	32	**11111110** + (–22 – VAL) codage sur 32 bits
4166 . . . ∞	8	32	**11111111** + (VAL – 4166) codage sur 32 bits
OOB	2		**10**

Tableau B.11 – Table de Huffman normalisée K

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
1	1	0	**0**
2 . . . 3	2	1	**10** + (VAL – 2) codage sur 1 bit
4	4	0	**1100**
5 . . . 6	4	1	**1101** + (VAL – 5) codage sur 1 bit
7 . . . 8	5	1	**11100** + (VAL – 7) codage sur 1 bit
9 . . . 12	5	2	**11101** + (VAL – 9) codage sur 2 bits
13 . . . 16	6	2	**111100** + (VAL – 13) codage sur 2 bits
17 . . . 20	7	2	**1111010** + (VAL – 17) codage sur 2 bits
21 . . . 28	7	3	**1111011** + (VAL – 21) codage sur 3 bits
29 . . . 44	7	4	**1111100** + (VAL – 29) codage sur 4 bits
45 . . . 76	7	5	**1111101** + (VAL – 45) codage sur 5 bits
77 . . . 140	7	6	**1111110** + (VAL – 77) codage sur 6 bits
141 . . . ∞	7	32	**1111111** + (VAL – 141) codage sur 32 bits

Tableau B.12 – Table de Huffman normalisée L

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
1	1	0	**0**
2	2	0	**10**
3 . . . 4	3	1	**110** + (VAL – 3) codage sur 1 bit
5	5	0	**11100**
6 . . . 7	5	1	**11101** + (VAL – 6) codage sur 1 bit
8 . . . 9	6	1	**111100** + (VAL – 8) codage sur 1 bit
10	7	0	**1111010**
11 . . . 12	7	1	**1111011** + (VAL – 11) codage sur 1 bit
13 . . . 16	7	2	**1111100** + (VAL – 13) codage sur 2 bit
17 . . . 24	7	3	**1111101** + (VAL – 17) codage sur 3 bit
25 . . . 40	7	4	**1111110** + (VAL – 25) codage sur 4 bit
41 . . . 72	8	5	**11111110** + (VAL – 41) codage sur 5 bit
73 . . . ∞	8	32	**11111111** + (VAL – 73) codage sur 32 bit

Tableau B.13 – Table de Huffman normalisée M

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
1	1	0	**0**
2	3	0	**100**
3	4	0	**1100**
4	5	0	**11100**
5 . . . 6	4	1	**1101** + (VAL – 5) codage sur 1 bit
7 . . . 14	3	3	**101** + (VAL – 7) codage sur 3 bits
15 . . . 16	6	1	**111010** + (VAL – 15) codage sur 1 bit
17 . . . 20	6	2	**111011** + (VAL – 17) codage sur 2 bits
21 . . . 28	6	3	**111100** + (VAL – 21) codage sur 3 bits
29 . . . 44	6	4	**111101** + (VAL – 29) codage sur 4 bits
45 . . . 76	6	5	**111110** + (VAL – 45) codage sur 5 bits
77 . . . 140	7	6	**1111110** + (VAL – 77) codage sur 6 bits
141 . . . ∞	7	32	**1111111** + (VAL – 141) codage sur 32 bits

Tableau B.14 – Table de Huffman normalisée N

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
–2	3	0	**100**
–1	3	0	**101**
0	1	0	**0**
1	3	0	**110**
2	3	0	**111**

Tableau B.15 – Table de Huffman normalisée O

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codage
–24 . . . –9	7	4	**1111100** + (VAL + 24) codage sur 4 bits
–8 . . . –5	6	2	**111100** + (VAL + 8) codage sur 2 bits
–4 . . . –3	5	1	**11100** + (VAL + 4) codage sur 1 bit
–2	4	0	**1100**
–1	3	0	**100**
0	1	0	**0**
1	3	0	**101**
2	4	0	**1101**
3 . . . 4	5	1	**11101** + (VAL – 3) codage sur 1 bit
5 . . . 8	6	2	**111101** + (VAL – 5) codage sur 2 bits
9 . . . 24	7	4	**1111101** + (VAL – 9) codage sur 4 bits
–∞ . . . –25	7	32	**1111110** + (–25 – VAL) codage sur 32 bits
25 . . . ∞	7	32	**1111111** + (VAL – 25) codage sur 32 bits

Annexe C

Procédure de décodage d'une image en échelle de gris

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

C.1 Description générale

Cette procédure de décodage est utilisée par la procédure de décodage de région de dégradé pour produire une série tabulaire de valeurs d'échelle de gris qui sont ensuite utilisées à l'intérieur d'un dictionnaire de structures.

C.2 Paramètres d'entrée

Les paramètres de cette procédure de décodage sont indiqués dans le Tableau C.1.

Tableau C.1 – Paramètres pour la procédure de décodage d'image en échelle de gris

Nom	Type	Longueur (en bits)	Signé?	Description et restrictions
GSMMR	Entier	1	N	Spécifie si le codage MMR est utilisé.
GSUSESKIP	Entier	1	N	Spécifie si l'omission de valeurs d'échelle de gris peut apparaître.
GSBPP	Entier	6	N	Nombre de bits par valeur d'échelle de gris.
GSW	Entier	32	N	Largeur de l'image en échelle de gris.
GSH	Entier	32	N	Hauteur de l'image en échelle de gris.
GSTEMPLATE	Entier	2	N	Gabarit utilisé pour coder les plans binaires d'échelle de gris[b)].
GSKIP	Matrice			Masque indiquant les valeurs à omettre. Largeur: **GSW** pixels, hauteur: **GSH** pixels[a)].

a) Paramètre inutilisé si **GSUSESKIP** = **0**.
b) Paramètre inutilisé si **GSMMR** = **1**.

C.3 Valeur de retour

La variable dont la valeur est le résultat de cette procédure de décodage est indiquée dans le Tableau C.2.

Tableau C.2 – Valeur de retour issue de la procédure de décodage d'image en échelle de gris

Nom	Type	Longueur (en bits)	Signé?	Description et restrictions
GSVALS	Série tabulaire			Image décodée en échelle de gris. La série tabulaire a pour largeur **GSW** et pour hauteur **GSH**.

C.4 Variables utilisées lors du décodage

Les variables utilisées par cette procédure de décodage sont indiquées dans le Tableau C.3.

Tableau C.3 – Variables utilisées dans la procédure de décodage d'image en échelle de gris

Nom	Type	Longueur (en bits)	Signé?	Description et restrictions
GSPLANES	Série tabulaire de matrices binaires			Plans binaires de l'image en échelle de gris. Il existe **GSBPP** plans binaires dans la variable GSPLANES. Chaque plan binaire a une largeur de **GSW** pixels et une hauteur de **GSH** pixels.
J	Entier	32	Y	Compteur de plans binaires.

C.5 Décodage de l'image en échelle de gris

L'image en échelle de gris s'obtient par décodage des **GSBPP** plans binaires. Ceux-ci sont désignés (dans le sens croissant des poids binaires) GSPLANES[0], GSPLANES[1]… GSPLANES[**GSBPP** – 1]. Les plans binaires sont codés en échelle de gris, de sorte que chaque valeur vraie d'un plan binaire est égale à sa valeur codée combinée par l'opérateur XOR avec le plan binaire de poids immédiatement supérieur.

L'image en échelle de gris s'obtient par la procédure suivante:

1) Décoder le plan GSPLANES[**GSBPP** – 1] au moyen de la procédure de décodage de région générique, dont les paramètres sont indiqués dans le Tableau C.4.

Tableau C.4 – Paramètres utilisés pour décoder un plan binaire de l'image en échelle de gris

Nombre	Valor
MMR	**GSMMR**
GBW	**GSW**
GBH	**GSH**
GBTEMPLATE	**GSTEMPLATE**
TPGDON	**0**
USESKIP	**GSUSESKIP**
SKIP	**GSKIP**
$\mathbf{GBATX}_1$	3 si **GSTEMPLATE** ≤ 1; 2 si **GSTEMPLATE** ≥ 2.
$\mathbf{GBATY}_1$	–1
$\mathbf{GBATX}_2$	–3
$\mathbf{GBATY}_2$	–1
$\mathbf{GBATX}_3$	2
$\mathbf{GBATY}_3$	–2
$\mathbf{GBATX}_4$	–2
$\mathbf{GBATY}_4$	–2

2) Poser J = **GSBPP** – 2.

3) Tant que $J \geq 0$, exécuter les étapes suivantes:

   a) décoder GSPLANES[J] au moyen de la procédure de décodage de région générique, dont les paramètres sont indiqués dans le Tableau C.4;

   b) pour chaque pixel de coordonnées (x, y) du plan GSPLANES[J], poser:

$$\text{GSPLANES}[J][x, y] = \text{GSPLANES}[J + 1][x, y] \text{ XOR } \text{GSPLANES}[J][x, y]$$

   c) poser $J = J - 1$.

4) Pour chaque pixel (x, y), poser:

$$\text{GSVALS}[x, y] = \sum_{J=0}^{\mathbf{GSBPP}-1} \text{GSPLANES}[J][x, y] \times 2^J$$

Annexe D

Formats de fichier

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

Deux modes autonomes d'organisation de fichier sont possibles pour un flux binaire JBIG2. Un troisième mode, non destiné à l'utilisation autonome, existe également afin de permettre l'imbrication de données à codage JBIG2 dans un autre format de fichier.

NOTE – Il est recommandé d'utiliser l'expression `".jbig2"` comme extension des fichiers JBIG2. Dans les environnements où trois caractères seulement sont autorisés, l'extension `".jb2"` est recommandée. Il est également recommandé que les décodeurs JBIG2 reconnaissent ces deux extensions.

D.1 Organisation séquentielle

Il s'agit d'un mode autonome d'organisation de fichier qui est destiné aux applications à flux continu dans lesquelles il est certain que le décodeur commencera au début du flux binaire et en décodera tous les éléments jusqu'à la fin de ce flux.

Dans cette organisation, la structure de fichier ressemble à celle de la Figure D.1. Un en-tête de fichier est suivi d'une séquence de segments. Les deux parties de chaque segment sont mémorisées ensemble: d'abord l'en-tête de segment, puis les données de segment.

Les segments doivent apparaître dans l'ordre croissant de leurs numéros: aucun segment ne peut précéder un segment de numéro inférieur.

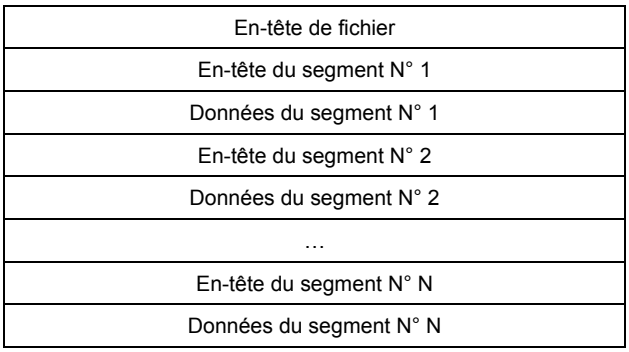


Figure D.1 – Organisation séquentielle

D.2 Organisation à accès aléatoire

Il s'agit d'une organisation de fichier autonome qui est destinée aux applications à accès aléatoire dans lesquelles le décodeur peut avoir à traiter des parties de fichier dans un ordre arbitraire, par exemple lors du décodage de toutes les pages de numéro impair avant les pages paires ou le décodage de pages individuelles en réponse à une certaine instruction de l'utilisateur. La capacité d'effectuer un accès aléatoire est donc importante.

Dans cette organisation, la structure de fichier ressemble à celle de la Figure D.2. Un en-tête de fichier est suivi d'une séquence d'en-têtes de segment dont le dernier est suivi des données pour le premier segment, puis des données pour le deuxième segment et ainsi de suite. Le dernier segment doit toujours être une fin de segment de fichier. Sinon, le décodeur n'a pas la possibilité de déterminer le moment où il a lu le dernier en-tête de segment.

Les segments doivent apparaître dans l'ordre croissant de leurs numéros de segment. Aucun segment ne peut précéder un segment ayant un numéro inférieur au sien.

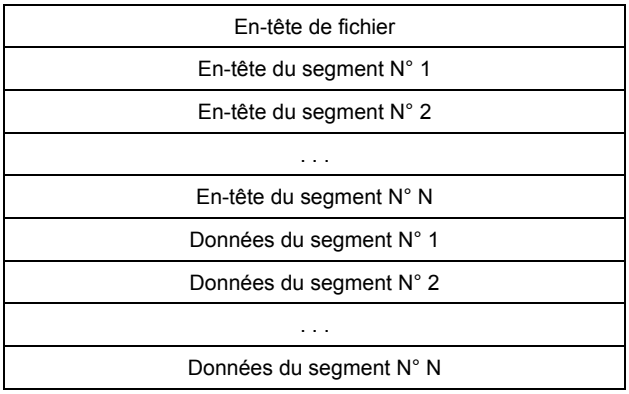


Figure D.2 – Organisation à accès aléatoire

D.3 Organisation imbriquée

Il ne s'agit pas d'une organisation de fichier autonome mais un mode qui se fonde sur un autre format de fichier pour transporter les segments JBIG2. Chaque segment est mémorisé par concaténation de son en-tête de segment et de ses parties de données de segment. Mais il n'y a pas d'ordre défini de mémorisation pour ces segments. Le format de fichier à imbrication permet de mémoriser ces segments dans un ordre quelconque et de les distinguer selon des données arbitraires.

Certaines applications peuvent avoir à précéder ou à suivre des données JBIG2 avec une combinaison de 2 octets unique (marqueur) de façon que ces données JBIG2 puissent être détectées à l'intérieur d'autres flux de données. Il est suggéré d'utiliser les octets `0xFF 0xAA` pour le marqueur de départ et les octets `0xFF 0xAB` pour le marqueur d'arrivée. Ces marqueurs ne sont pas considérés comme faisant partie des données JBIG2. Il y a lieu de noter que le premier octet d'un en-tête de segment n'est pas susceptible de prendre la valeur `0xFF` et que les séquences de deux octets `0xFF 0xAA` et `0xFF 0xAB` peuvent apparaître fortuitement dans des segments JBIG2.

NOTE – L'organisation imbriquée a pour objet de pouvoir faire bénéficier de nombreux systèmes actuels de l'incorporation d'une compression améliorée des images à deux niveaux. La meilleure façon d'y parvenir n'est cependant pas toujours d'insérer un flux JBIG2 complet en tant qu'entité monolithique car ce flux peut entrer en conflit avec d'autres contraintes. Par exemple, le système peut avoir ses propres vues quant à la façon de subdiviser les pages, qui peuvent ne pas s'accorder avec celles du format JBIG2. Le flux JBIG2 est donc flexible car il permet au système effectuant l'imbrication de mémoriser des données JBIG2 de toute façon paraissant la plus appropriée.

D.4 Syntaxe d'en-tête de fichier

Un en-tête de fichier contient les champs suivants, dans l'ordre suivant:

Chaîne d'identificateur – Voir D.4.1.

Fanions d'en-tête de fichier – Voir D.4.2.

Nombre de pages – Voir D.4.3.

D.4.1 Chaîne d'identificateur

Il s'agit de la séquence des 8 octets suivants: `0x97 0x4A 0x42 0x32 0x0D 0x0A 0x1A 0x0A`.

NOTE – Cette chaîne est similaire à la chaîne d'identification PNG. Le premier caractère n'est pas imprimable de façon que le fichier ne puisse pas être confondu avec un texte en ASCII. Le bit supérieur du premier caractère est réglé de façon à permettre à une personne examinant l'en-tête de déduire le type de fichier. Les octets suivants sont `CR LF CONTROL-Z LF`; toute corruption par conversion CR/LF et par troncature de fichier DOS peut être détectée immédiatement.

D.4.2 Fanions d'en-tête de fichier

Il s'agit d'un champ de 1 octet dont les bits sont définis comme suit:

Bit 0 Type d'organisation de fichier. Si ce bit est **0**, le fichier utilise l'organisation à accès aléatoire. S'il est **1**, le fichier utilise l'organisation séquentielle.

NOTE – Rien ne permet d'indiquer l'organisation imbriquée car cette organisation ne comporte pas d'en-tête de fichier JBIG2.

Bit 1 Nombre de pages inconnu. Si ce bit est **0**, le nombre de pages contenues dans le fichier est connu. S'il est **1**, le nombre de pages contenues dans le fichier n'était pas connu au moment où l'en-tête de fichier a été codé.

Bits 2 à 7 Champ réservé qui doit être **0**.

D.4.3 Nombre de pages

Il s'agit d'un champ de 4 octets qui n'est pas présent si le bit "nombre de pages inconnu" avait la valeur **1**. S'il est présent, il doit être égal au nombre de pages contenues dans le fichier.

Annexe E

Codage arithmétique

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

Un codeur arithmétique binaire adaptatif peut être utilisé comme codeur d'entropie si les modèles le permettent. Les modèles utilisés avec le codage arithmétique binaire adaptatif sont définis au 6.2, au 6.3 et dans l'Annexe A. La présente annexe définit les procédures de codage arithmétique de base.

Dans tous les paragraphes de cette annexe, les organigrammes et les tableaux ne sont normatifs que dans la mesure où ils visent à définir une sortie que des réalisations en variante doivent reproduire. Le paragraphe H.2 donne un exemple de test simple qui devrait être utile pour déterminer si une réalisation donnée est correcte.

E.1 Codage binaire

La Figure E.1 montre un simple schéma fonctionnel du codeur arithmétique binaire adaptatif. Les paires décision (D)-contexte (CX) sont traitées ensemble afin de produire en sortie un flux de données comprimées (CD). Aussi bien D que CX sont des données fournies par le modèle unitaire (non représenté) Le contexte CX choisit l'estimateur de probabilité à utiliser au cours du codage de la décision D. Dans la présente Recommandation | Norme internationale, CX est une étiquette de contexte formée par une certaine chaîne de caractères suivie d'une chaîne de bits.

EXEMPLE – Deux valeurs possibles de CX sont "IADW**001010100**" et "GB**1110110010000000**".

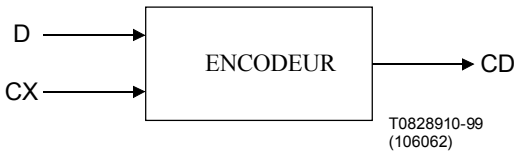


Figure E.1 – Entrées et sorties du codeur arithmétique

E.1.1 Subdivision récurrente des intervalles

La subdivision récurrente des intervalles de probabilité dans le codage Elias est la base du processus de codage arithmétique binaire. Avec chaque décision binaire, l'intervalle de probabilité actuel est subdivisé en deux sous-intervalles et la chaîne codée est modifiée (si nécessaire) de façon à pointer sur la base (ou borne inférieure) du sous-intervalle de probabilité assigné au symbole qui est apparu.

Lors de la subdivision de l'intervalle actuel en deux sous-intervalles, le sous-intervalle pour le symbole plus probable (MPS) est ordonné de façon à être au-dessus du sous-intervalle pour le symbole moins probable (LPS). Lorsque l'intervalle MPS est codé, le sous-intervalle LPS est donc ajouté à la chaîne codée. Cette convention de codage nécessite que les symboles soient reconnus comme étant soit de type MPS soit de type LPS plutôt que 0 ou 1. Par conséquent, la longueur de l'intervalle LPS et le sens de l'intervalle MPS doivent être connus pour chaque décision afin de coder celle-ci.

Comme la chaîne codée pointe toujours sur la base de l'intervalle actuel, le processus de décodage consiste à déterminer, pour chaque décision, le sous-intervalle sur lequel la chaîne codée pointe. Ce processus est également effectué de manière récurrente, au moyen du même processus de subdivision d'intervalle que dans le codeur. La chaîne codée est donc, dans le décodeur, un pointeur sur l'intervalle actuel par rapport à la base de celui-ci. Comme le processus de codage implique l'addition de fractions binaires plutôt que la concaténation de mots de code d'entiers, les décisions binaires les plus probables pourront souvent être codées avec consommation de beaucoup moins qu'un élément binaire par décision.

E.1.2 Conventions et approximations de codage

Les opérations de codage sont effectuées en arithmétique d'entiers à précision fixe, les entiers étant représentés par valeurs fractionnaires dans lesquelles `0x8000` est l'équivalent du nombre décimal 0,75. On maintient l'intervalle A dans l'étendue $0{,}75 \leq A < 1{,}5$ en le doublant chaque fois que la valeur d'entier tombe au-dessous de `0x8000`.

Le registre de code C est également doublé chaque fois que l'intervalle A est doublé. Périodiquement, pour éviter un débordement du registre C, un octet de données est retranché des bits d'ordre supérieur du registre C. Cet octet est placé dans un tampon externe contenant des données comprimées. Les reports dans ce tampon externe sont assurés par une procédure de bourrage binaire.

Le fait de maintenir l'intervalle A dans l'étendue $0{,}75 \leq A < 1{,}5$ permet d'utiliser une simple approximation arithmétique dans la subdivision de l'intervalle. Si celui-ci est A et que l'estimateur actuel de la probabilité LPS soit Qe, un calcul précis des sous-intervalles nécessitera:

$$A - (Qe \times A) = \text{sous-intervalle pour le symbole MPS}$$
$$Qe \times A = \text{sous-intervalle pour le symbole LPS}$$

Comme la valeur de A est de l'ordre de l'unité, ces sous-intervalles sont approchés comme suit:

$$A - Qe = \text{sous-intervalle pour le symbole MPS}$$
$$Qe = \text{sous-intervalle pour le symbole LPS}$$

Chaque fois que le symbole MPS est codé, la valeur de Qe est ajoutée au registre de code et l'intervalle est réduit à A – Qe. Chaque fois que le symbole LPS est codé, le registre de code est laissé inchangé et l'intervalle est réduit à Qe. L'étendue de précision pour A est alors rétablie, si nécessaire, par renormalisation de A comme de C.

Par le processus décrit ci-dessus, l'approximation dans le processus de subdivision d'intervalle peut parfois rendre le sous-intervalle LPS plus grand que le sous-intervalle MPS. Si par exemple la valeur de Qe est 0,5 et que A soit à la valeur minimale autorisée de 0,75, l'échelonnement approché donne 1/3 de l'intervalle au symbole MPS et 2/3 de l'intervalle au symbole LPS. Pour éviter cette inversion de grandeur, les intervalles MPS et LPS sont échangés chaque fois que l'intervalle LPS est plus grand que l'intervalle MPS. Cet échange conditionnel MPS/LPS ne peut se produire que lorsqu'une renormalisation est nécessaire.

Chaque fois qu'une renormalisation se produit, un processus d'estimation de probabilité est invoqué pour déterminer un nouvel estimateur de probabilité applicable au contexte qui est en cours de codage. Aucun comptage explicite de symboles n'est nécessaire pour l'estimation. Les probabilités relatives de renormalisation après codage d'un symbole LPS/MPS constituent un mécanisme de comptage approché des symboles qui est utilisé pour estimer directement les probabilités correspondantes.

E.2 Description du codeur arithmétique

Le CODEUR indiqué sur la Figure E.2 initialise le codeur par la procédure INITENC. Les paires CX et D sont lues et sont transmises à la procédure ENCODE jusqu'à ce que toutes les paires aient été lues. Les procédures d'estimation de probabilité qui fournissent des estimateurs adaptatifs de la probabilité de chaque contexte sont imbriquées dans la procédure ENCODE. Les octets de données comprimées sont extraits lorsqu'ils ne sont plus modifiables. Lorsque toutes les paires CX et D ont été lues (condition "Finished?"), la procédure FLUSH règle le contenu du registre C sur un aussi grand nombre que possible de bits 1 puis extrait les octets définitifs. FLUSH met également fin aux opérations de codage et produit le marqueur de terminaison requis.

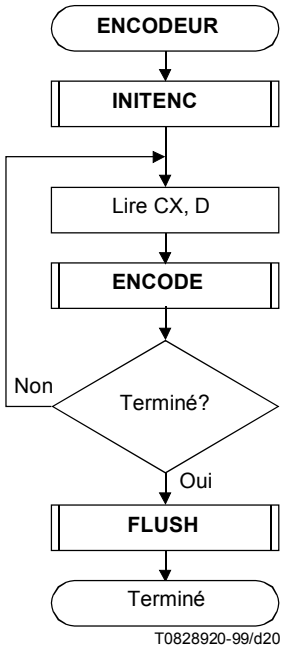


Figure E.2 – Codeur pour le codeur MQ

E.2.1 Conventions du registre de séquences du codeur

Les organigrammes figurant dans la présente annexe sont fondés sur les structures de registre suivantes pour le codeur:

	MSB			LSB
Registre C	0000cbbb	bbbbbsss	xxxxxxxx	xxxxxxxx
Registre A	00000000	00000000	aaaaaaaa	aaaaaaaa

Les bits "a" sont les bits fractionnaires contenus dans le registre A (qui indique la valeur actuelle de l'intervalle) et les bits "x" sont les bits fractionnaires contenus dans le registre de code. Les bits "s" sont des bits d'espacement qui fournissent des contraintes utiles pour le report et les bits "b" indiquent les positions binaires à partir desquelles les octets de données constitués sont extraits du registre C. Le bit "c" est un bit de report.

La description détaillée des processus de bourrage binaire et de traitement des reports sera insérée dans une partie ultérieure de la présente annexe.

E.2.2 Codage d'une décision (ENCODE)

La procédure ENCODE détermine si la décision D est un 0 ou non: selon le cas, une procédure CODE0 ou CODE1 est appelée. Les éléments imbriqués n'auront pas souvent de procédure ENCODE mais ils appelleront directement les procédures CODE0 ou CODE1 afin de coder une décision 0 ou une décision 1.

E.2.3 Codage d'un 1 ou d'un 0 (CODE1 et CODE0)

Lorsqu'une décision binaire donnée est codée, une des deux possibilités suivantes se produit: le symbole est soit le plus probable soit le moins probable. Les procédures CODE1 et CODE0 sont illustrées dans les Figures E.4 et E.5. Dans ces figures, CX est le contenu. Pour chaque contexte, l'on mémorise l'indice de l'estimateur de probabilité qui doit être utilisé dans les opérations de codage ainsi que la valeur MPS. Le terme MPS(CX) indique le sens (**0** ou **1**) du symbole MPS pour le contexte CX.

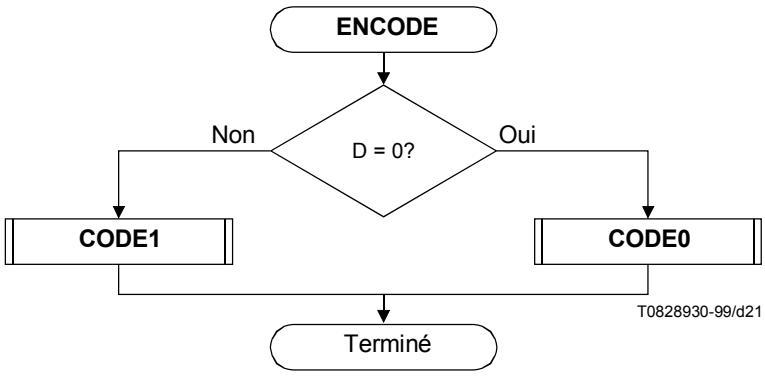


Figure E.3 – Procédure ENCODE

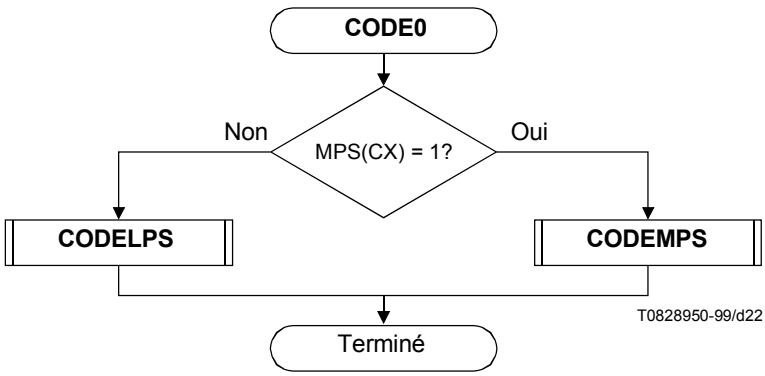


Figure E.4 – Procédure CODE1

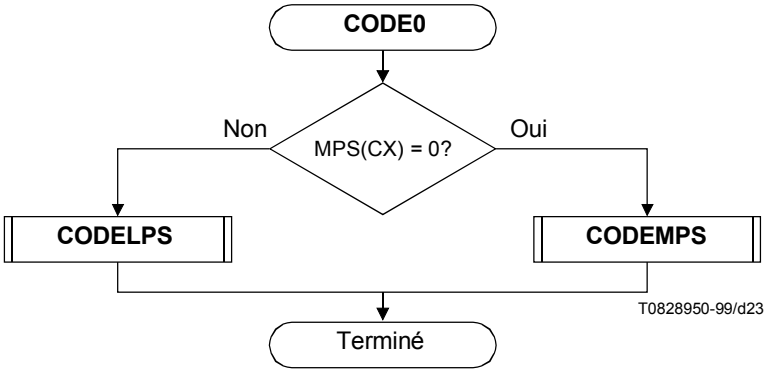


Figure E.5 – Procédure CODE0

E.2.4 Codage d'un symbole MPS ou LPS (CODEMPS et CODELPS)

La procédure CODELPS (Figure E.6) se compose habituellement d'un échelonnement de l'intervalle à Qe((I(CX)), qui est l'estimateur de probabilité du symbole LPS déterminé au moyen de l'indice I mémorisé pour le contexte CX. L'intervalle supérieur est d'abord calculé de façon qu'il puisse être comparé à l'intervalle inférieur pour confirmer que Qe possède la plus petite dimension. Cet intervalle est toujours suivi d'une renormalisation (RENORME). Si les dimensions des intervalles sont inversées, l'échange conditionnel MPS/LPS se produit cependant et l'intervalle supérieur est codé. Dans un cas comme dans l'autre, l'estimateur de probabilité est actualisé. Si le fanion SWITCH pour l'indice I(CX) est activé, le terme MPS(CX) est inversé. Un nouvel indice I est sauvegardé dans le contexte CX selon ce qui est déterminé au moyen de la colonne d'indice LPS suivant (NLPS) dans le Tableau E.1.

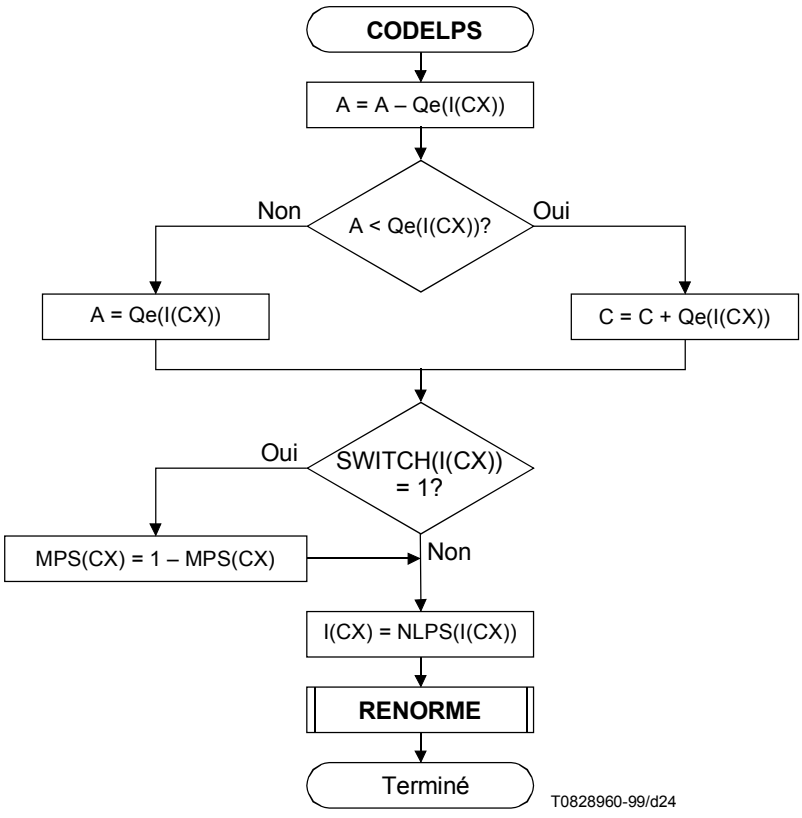


Figure E.6 – Procédure CODELPS avec échange conditionnel de symboles MPS/LPS

La procédure CODEMPS (Figure E.7) réduit habituellement la longueur de l'intervalle au sous-intervalle MPS puis règle le registre de code de façon qu'il pointe sur la base du sous-intervalle MPS. Si cependant les longueurs d'intervalle sont inversées, c'est le sous-intervalle LPS qui est codé au lieu du sous-intervalle MPS. Noter que l'inversion de longueur ne peut pas se produire si une renormalisation (RENORME) n'est pas requise après le codage du symbole. L'actualisation de l'estimateur de probabilité modifie l'indice I(CX) en fonction de la colonne de prochain indice MPS (NMPS) dans le Tableau E.1.

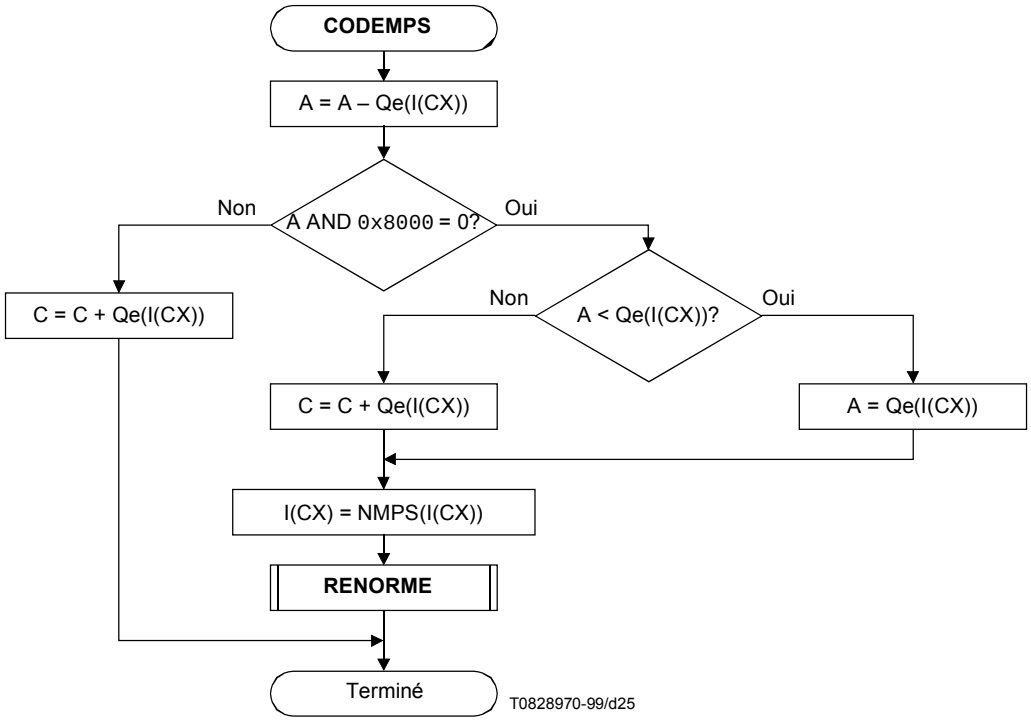


Figure E.7 – Procédure CODEMPS avec échange conditionnel de symboles MPS/LPS

E.2.5 Estimateur de probabilité

Le Tableau E.1 montre la valeur Qe associée à chaque indice Qe. Les valeurs Qe sont exprimées sous forme d'entiers hexadécimaux, d'entiers binaires et de fractions décimales. Pour convertir la valeur de Qe représentée par un entier de 15 bits afin d'obtenir la probabilité décimale, chaque valeur de Qe est divisée par (4/3) × (`0x8000`).

L'estimateur peut être défini comme une machine à états finis: une table d'indices Qe et des états suivants associés à chaque type de renormalisation (c'est-à-dire aux nouvelles positions dans la table), comme indiqué dans le Tableau E.1. Le changement d'état ne se produit que lorsque le registre d'intervalles du codeur arithmétique est renormalisé. Cela se produit toujours après le codage du symbole LPS et chaque fois que le registre d'intervalle est inférieur à `0x8000` (0,75 en notation décimale) après le codage du symbole MPS.

Après une renormalisation LPS, la colonne NLPS indique le nouvel indice pour l'estimateur de probabilité LPS. De même, si le fanion Switch = **1**, le sens du symbole MPS est inversé. Après une renormalisation MPS, la colonne NMPS indique le nouvel indice pour l'estimateur de probabilité MPS.

L'indice pointant sur l'estimateur actuel fait partie des informations mémorisées pour le contexte CX. Cet indice sert à pointer sur la table des valeurs de la colonne NMPS, qui indique l'indice suivant pour une renormalisation MPS. Cet indice est sauvegardé dans la mémoire de contextes de CX. Le terme MPS(CX) ne change pas.

La procédure d'estimation de la probabilité sur le trajet de renormalisation LPS est analogue à celle d'une renormalisation MPS sauf que lorsque le fanion Switch(I(CX)) = 1, le sens du terme MPS(CX) est inversé.

L'état d'indice final 46 peut être utilisé pour établir un estimateur de probabilité fixé à 0,5.

Tableau E.1 – Valeurs Qe et processus d'estimation de probabilité

Indice	Valeur Qe			NMPS	NLPS	SWITCH
	(hexadécimal)	(binaire)	(décimal)			
0	0x5601	**0101011000000001**	0,503937	1	1	1
1	0x3401	**0011010000000001**	0,304715	2	6	0
2	0x1801	**0001100000000001**	0,140650	3	9	0
3	0x0AC1	**0000101011000001**	0,063012	4	12	0
4	0x0521	**0000010100100001**	0,030053	5	29	0
5	0x0221	**0000001000100001**	0,012474	38	33	0
6	0x5601	**0101011000000001**	0,503937	7	6	1
7	0x5401	**0101010000000001**	0,492218	8	14	0
8	0x4801	**0100100000000001**	0,421904	9	14	0
9	0x3801	**0011100000000001**	0,328153	10	14	0
10	0x3001	**0011000000000001**	0,281277	11	17	0
11	0x2401	**0010010000000001**	0,210964	12	18	0
12	0x1C01	**0001110000000001**	0,164088	13	20	0
13	0x1601	**0001011000000001**	0,128931	29	21	0
14	0x5601	**0101011000000001**	0,503937	15	14	1
15	0x5401	**0101010000000001**	0,492218	16	14	0
16	0x5101	**0101000100000001**	0,474640	17	15	0
17	0x4801	**0100100000000001**	0,421904	18	16	0
18	0x3801	**0011100000000001**	0,328153	19	17	0
19	0x3401	**0011010000000001**	0,304715	20	18	0
20	0x3001	**0011000000000001**	0,281277	21	19	0
21	0x2801	**0010100000000001**	0,234401	22	19	0
22	0x2401	**0010010000000001**	0,210964	23	20	0
23	0x2201	**0010001000000001**	0,199245	24	21	0
24	0x1C01	**0001110000000001**	0,164088	25	22	0
25	0x1801	**0001100000000001**	0,140650	26	23	0
26	0x1601	**0001011000000001**	0,128931	27	24	0
27	0x1401	**0001010000000001**	0,117212	28	25	0
28	0x1201	**0001001000000001**	0,105493	29	26	0
29	0x1101	**0001000100000001**	0,099634	30	27	0
30	0x0AC1	**0000101011000001**	0,063012	31	28	0
31	0x09C1	**0000100111000001**	0,057153	32	29	0
32	0x08A1	**0000100010100001**	0,050561	33	30	0
33	0x0521	**0000010100100001**	0,030053	34	31	0
34	0x0441	**0000010001000001**	0,024926	35	32	0
35	0x02A1	**0000001010100001**	0,015404	36	33	0
36	0x0221	**0000001000100001**	0,012474	37	34	0
37	0x0141	**0000000101000001**	0,007347	38	35	0
38	0x0111	**0000000100010001**	0,006249	39	36	0
39	0x0085	**0000000010000101**	0,003044	40	37	0
40	0x0049	**0000000001001001**	0,001671	41	38	0
41	0x0025	**0000000000100101**	0,000847	42	39	0
42	0x0015	**0000000000010101**	0,000481	43	40	0
43	0x0009	**0000000000001001**	0,000206	44	41	0
44	0x0005	**0000000000000101**	0,000114	45	42	0
45	0x0001	**0000000000000001**	0,000023	45	43	0
46	0x5601	**0101011000000001**	0,503937	46	46	0

E.2.6 Renormalisation dans le codeur (RENORME)

La renormalisation est très semblable dans le codeur et dans le décodeur, la différence étant qu'elle produit des bits comprimés dans le codeur et qu'elle consomme des bits comprimés dans le décodeur.

La procédure RENORME de renormalisation dans le codeur est décrite dans la Figure E.8. Les deux registres A et C, d'intervalle et de code, sont décalés d'un bit à la fois. Le nombre de décalages est compté dans le compteur CT. Lorsque CT arrive à zéro, un octet de données comprimées est extrait du registre C par la procédure BYTEOUT. La renormalisation se poursuit jusqu'à ce que A ne soit plus inférieur à `0x8000`.

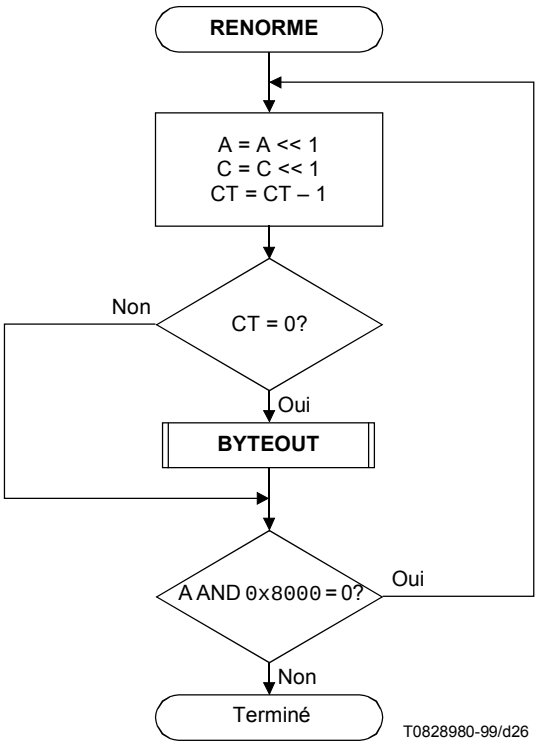


Figure E.8 – Procédure de renormalisation du codeur

E.2.7 Sortie de données comprimées (BYTEOUT)

La procédure BYTEOUT qui est appelée depuis RENORME est décrite dans la Figure E.9. Cette procédure contient les procédures de bourrage binaire qui sont nécessaires pour limiter la propagation des reports dans les octets constitués de données comprimées. Les conventions utilisées empêchent un report de se propager au-delà de l'octet écrit le plus récemment dans le tampon de données comprimées.

La procédure du bloc situé dans la partie droite inférieure effectue le bourrage de bits après un octet `0xFF`; la procédure analogue à gauche concerne le cas où le bourrage de bits n'est pas nécessaire.

B est l'octet visé par le pointeur de tampon (BP) de données comprimées. Si l'octet B n'est pas un octet `0xFF`, le bit de report est vérifié. Si le bit de report est activé, il est ajouté à B et l'octet B est de nouveau vérifié quant à la nécessité de bourrer un bit dans l'octet suivant. Cette nécessité ayant été déterminée, le trajet approprié est choisi. Le pointeur BP est incrémenté et la nouvelle valeur de B est extraite des bits "b" du registre de code.

E.2.8 Initialisation du codeur (INITENC)

La procédure INITENC est utilisée pour initialiser le codeur arithmétique. Les étapes de base sont indiquées dans la Figure E.10.

Le registre d'intervalle et le registre de code sont réglés à leur valeur initiale et le compteur de bits est armé. Le réglage CT = 12 reflète le fait qu'il y a trois bits d'espacement dans le registre, qu'il y a lieu de bourrer avant que le champ dont les octets sont extraits soit atteint. Noter que BP pointe toujours sur l'octet qui précède la position de valeur initiale du pointeur (BPST), dans laquelle le premier octet est placé. Si l'octet précédent est de type `0xFF`, un bourrage de bit intempestif se produira donc, mais ce bourrage pourra être compensé par augmentation du compteur CT. Noter que l'initialisation par défaut des segments statistiques est MPS = 0 et I = 0 (c'est-à-dire Qe = 0x5601 ou la valeur décimale 0,503937).

E.2.9 Terminaison du codage (FLUSH)

La procédure FLUSH indiquée dans la Figure E.11 est utilisée pour mettre fin aux opérations de codage et pour produire le marqueur de terminaison approprié. Cette procédure garantit que le préfixe `0xFF` du code de marqueur se superpose aux bits finals des données comprimées, afin que tout code de marqueur situé à la fin des données comprimées soit reconnu et interprété avant que le décodage soit complet.

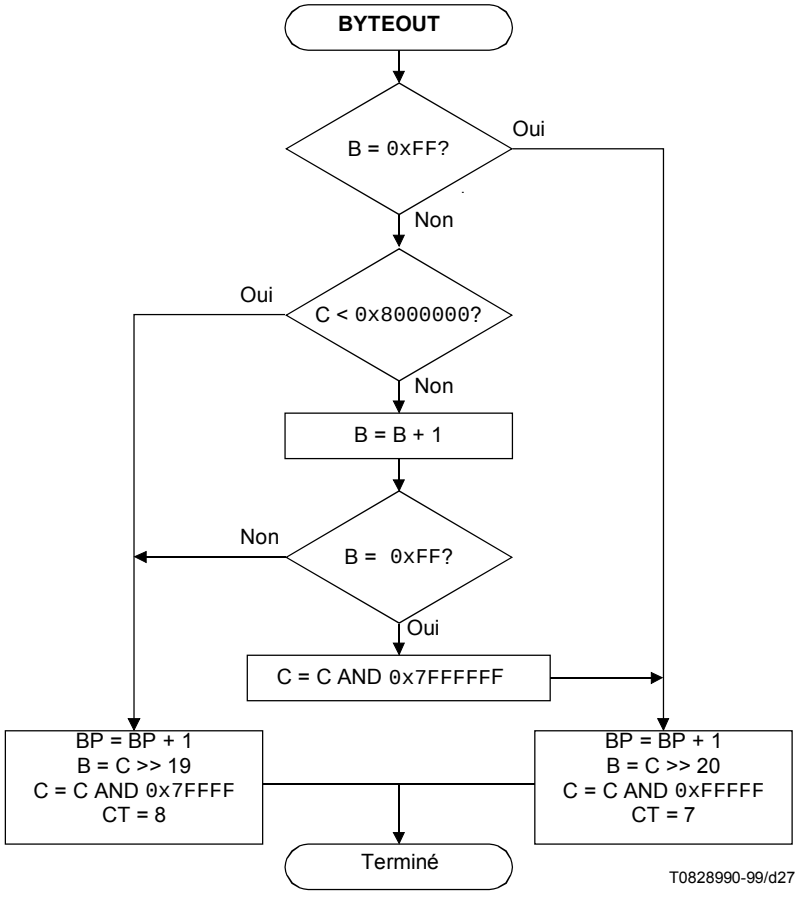


Figure E.9 – Procédure BYTEOUT pour le codeur

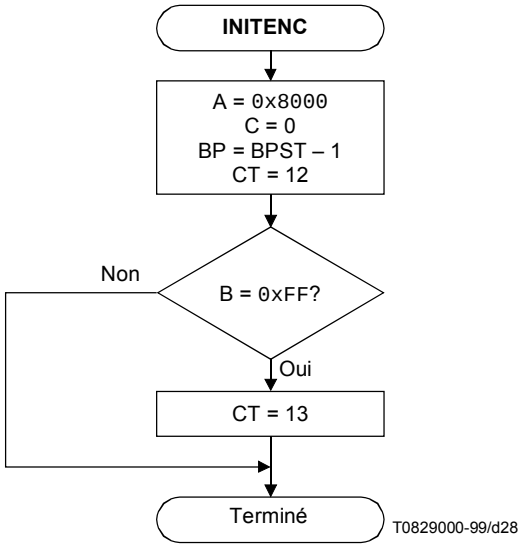


Figure E.10 – Initialisation du codeur

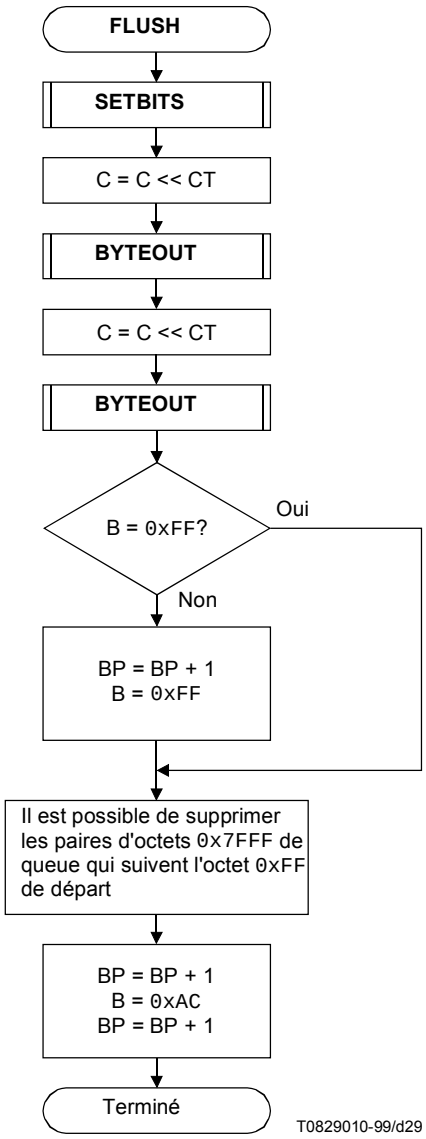


Figure E.11 – Procédure FLUSH

La première partie de la procédure FLUSH met à 1 autant de bits dans le registre C que possible, comme indiqué dans la Figure E.12. La limite supérieure exclusive pour le registre C est la somme de ce registre et du registre d'intervalle. Les 16 bits d'ordre inférieur de C sont forcés à 1 et le résultat est comparé à la limite supérieure. Si C est trop grand, le bit 1 de départ est supprimé, ce qui réduit C à une valeur située à l'intérieur de l'intervalle.

L'octet situé dans le registre C est alors constitué par décalage de C puis deux octets sont extraits. Si le deuxième octet n'est pas `0xFF`, un autre octet est ajouté aux données comprimées, qui est garanti être `0xFF`.

E.2.10 Minimisation des données comprimées

Au besoin, les données comprimées peuvent être tronquées une fois la procédure FLUSH terminée. Si une séquence de bits 1 est produite par le codeur arithmétique, le bourrage de bits produira des paires d'octets `0xFF`, `0x7F` qui pourront être extraites des données comprimées, à condition que le plus ancien octet `0xFF` de la séquence ne soit pas supprimé. Cet octet `0xFF` restant devient alors le préfixe du code de marqueur qui termine les données comprimées.

Le décodage n'est pas affecté par ce processus d'extraction car, dans le décodeur, on utilise la convention que, si un code marqueur est rencontré, 1 bit (sans bourrage binaire) sont fournis au décodeur jusqu'à ce que l'intervalle de codage soit complet.

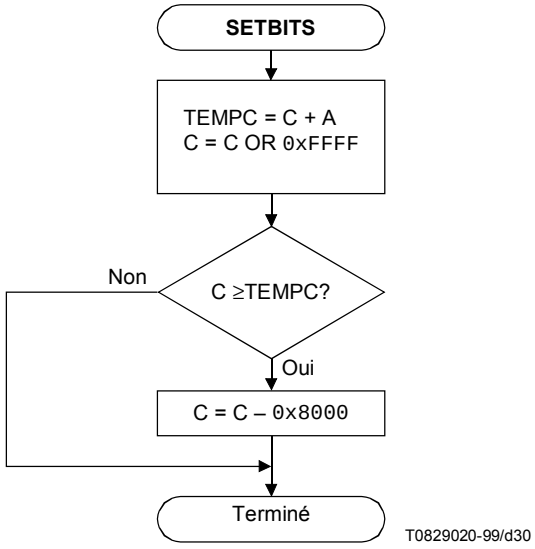


Figure E.12 – Réglage des bits définitifs dans le registre C

E.3 Procédure de décodage arithmétique

La Figure E.13 montre un simple schéma fonctionnel de décodeur arithmétique binaire adaptatif. Les données comprimées CD et un contexte CX issus du modèle de décodeur unitaire (non représenté) sont injectés dans le décodeur arithmétique. La sortie du décodeur est la décision D. Les modèles unitaires du codeur et du décodeur doivent fournir exactement le même contexte CX pour chaque décision donnée.

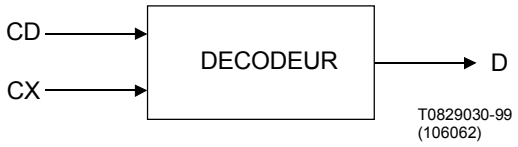


Figure E.13 – Entrées et sorties du décodeur arithmétique

Le DECODEUR de la Figure E.14 initialise le décodeur par la procédure INITDEC. Les contextes (CX) et les octets de données comprimées (le cas échéant) sont lus puis transmis à la routine DECODE jusqu'à ce que tous les contextes aient été lus. La routine DECODE décode la décision binaire D et renvoie une valeur de 0 ou 1. Les procédures d'estimation de probabilité qui fournissent des estimateurs adaptatifs de la probabilité pour chaque contexte sont imbriquées dans la routine DECODE. Lorsque tous les contextes ont été lus ("Finished?"), les données comprimées ont été décomprimées.

E.3.1 Conventions du registre de séquences du décodeur

Les organigrammes de la présente annexe sont fondés sur les structures de registre suivantes pour le décodeur:

	15	0
Registre Chigh	xxxxxxxx	xxxxxxxx
Registre Clow	bbbbbbbb	00000000
Registre A	aaaaaaaa	aaaaaaaa

Les registres Chigh et Clow peuvent être considérés comme un seul registre C de 32 bits en ce sens que la renormalisation de C décale un bit de nouvelle donnée du bit 15 de Clow au bit 0 de Chigh. Les comparaisons de décodage utilisent cependant le seul registre Chigh. Les nouvelles données sont insérées dans les "b" bits de Clow octet par octet.

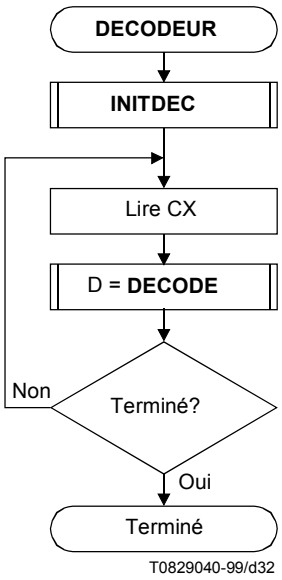


Figure E.14 – Décodeur pour le codeur MQ

La description détaillée du traitement de données avec des bits de bourrage sera présentée ultérieurement dans ce paragraphe.

Noter que les comparaisons indiquées dans les diverses procédures de ce paragraphe sont fondées sur des précisions supérieures à 16 bits. Des comparaisons logiques peuvent être utilisées avec une précision de 16 bits.

E.3.2 Décodage d'une décision (DECODE)

Le décodeur décode une seule décision binaire à la fois. Après le décodage d'une décision, le décodeur soustrait de la chaîne codée tout ce qui a été ajouté par le codeur. Les bits laissés dans la chaîne codée représentent le décalage à partir de la base de l'intervalle actuel jusqu'au sous-intervalle attribué à toutes les décisions binaires non encore décodées. Au cours du premier essai de la procédure de décodage (DECODE) décrite dans la Figure E.15, le registre Chigh est comparé à la longueur du sous-intervalle LPS. A moins qu'un échange conditionnel ne soit requis, cet essai détermine si un symbole MPS ou LPS est décodé. Si Chigh est logiquement supérieur ou égal à l'estimateur Qe de probabilité LPS pour l'indice actuel I mémorisé dans CX, le registre Chigh est décrémenté de cette longueur. Si le registre A n'est pas inférieur à `0x8000`, le sens du symbole MPS mémorisé dans CX est utilisé pour fixer la décision décodée D.

Lorsqu'une renormalisation est nécessaire, l'échange conditionnel MPS/LPS peut s'être produit. Pour le trajet MPS, la procédure d'échange conditionnel est décrite dans la Figure E.16. Du moment que la longueur A du sous-intervalle de MPS, calculée en première étape de la Figure E.16, n'est pas logiquement inférieure à l'estimateur de probabilité LPS Qe(I(CX)), un symbole MPS est en fait apparu et la décision peut être fixée à partir de MPS(CX). L'indice I(CX) est ensuite actualisé à partir de la colonne de l'indice MPS suivant (NMPS) dans le Tableau E.1. Si cependant le sous-intervalle LPS est plus grand, l'échange conditionnel s'est produit et un symbole LPS est apparu. L'actualisation de probabilité inverse le sens du symbole MPS si la colonne SWITCH contient un "1" et l'indice I(CX) est actualisé à partir de la colonne de l'indice LPS suivant (NLPS) du Tableau E.1. Noter que l'estimation de probabilité dans le décodeur doit être identique à l'estimation de probabilité dans le codeur.

Pour le trajet LPS du décodeur, la procédure d'échange conditionnel reçoit la procédure LPS_EXCHANGE indiquée dans la Figure E.17. La même comparaison logique entre le sous-intervalle A de symbole MPS et le sous-intervalle Qe(I(CX)) de symbole LPS détermine si un échange conditionnel a eu lieu. Sur les deux trajets, le nouveau sous-intervalle A est mis à Qe(I(CX)). Sur le trajet de gauche, l'échange conditionnel a eu lieu, de sorte que la décision et l'actualisation concernent le cas MPS. Sur le trajet de droite, la décision et l'actualisation concernent le cas LPS.

E.3.3 Renormalisation dans le décodeur (RENORMD)

La procédure RENORMD de renormalisation dans le décodeur est illustrée dans la Figure E.18. Un compteur garde trace du nombre de bits comprimés dans la section Clow du registre C. Lorsque CT est zéro, un nouvel octet est inséré dans Clow par la procédure BYTEIN.

Les deux registres A et C (d'intervalle et de code) sont décalés, bit par bit, jusqu'à ce que A ne soit plus inférieur à `0x8000.`

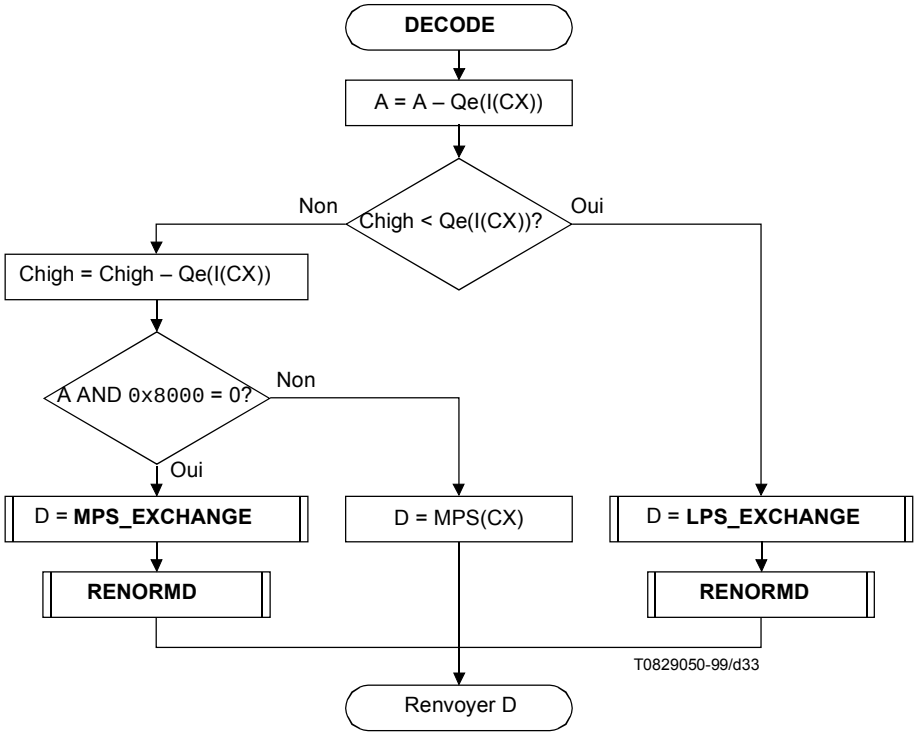


Figure E.15 – Décodage du symbole MPS ou LPS

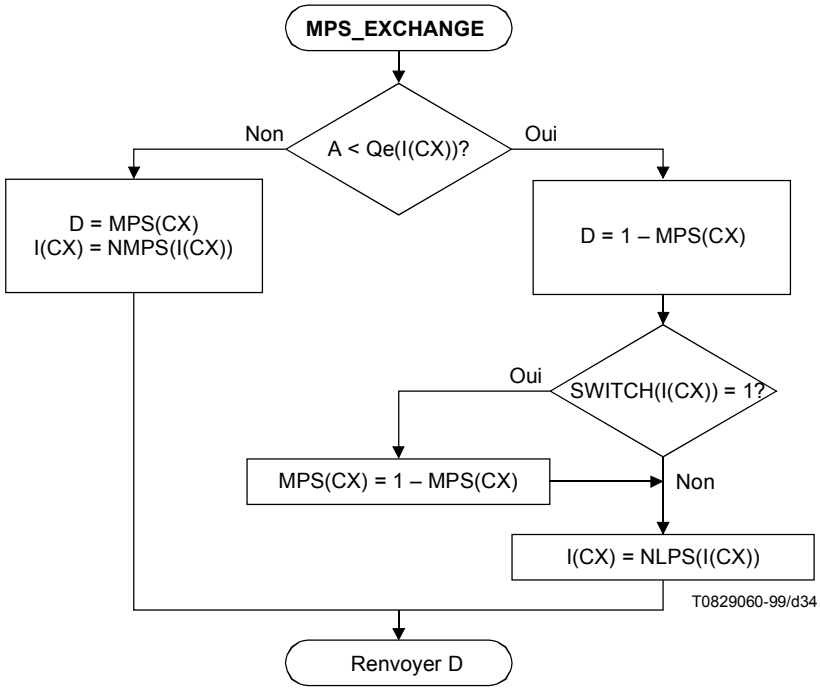


Figure E.16 – Procédure d'échange conditionnel dans le trajet MPS du décodeur

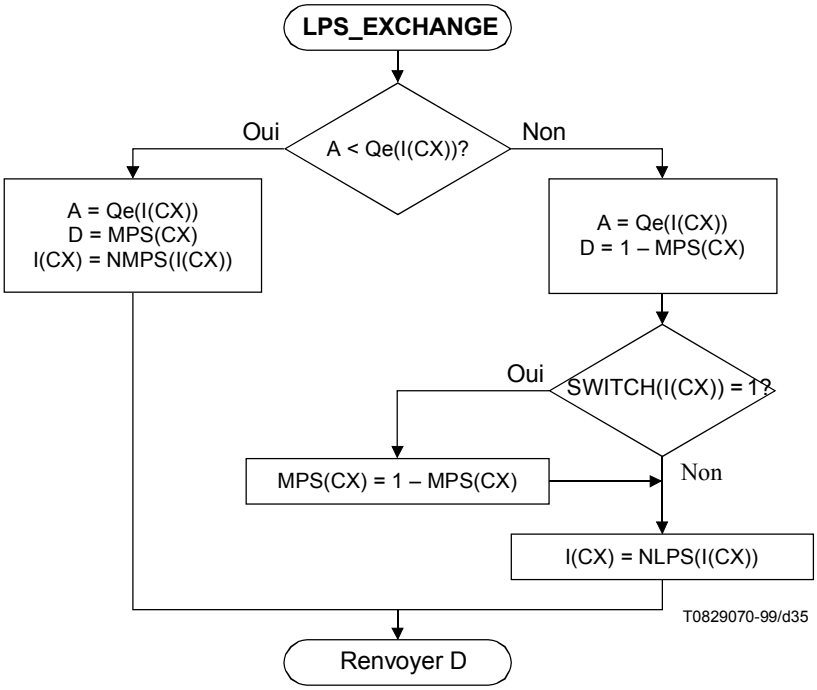


Figure E.17 – Procédure d'échange conditionnel dans le trajet LPS du décodeur

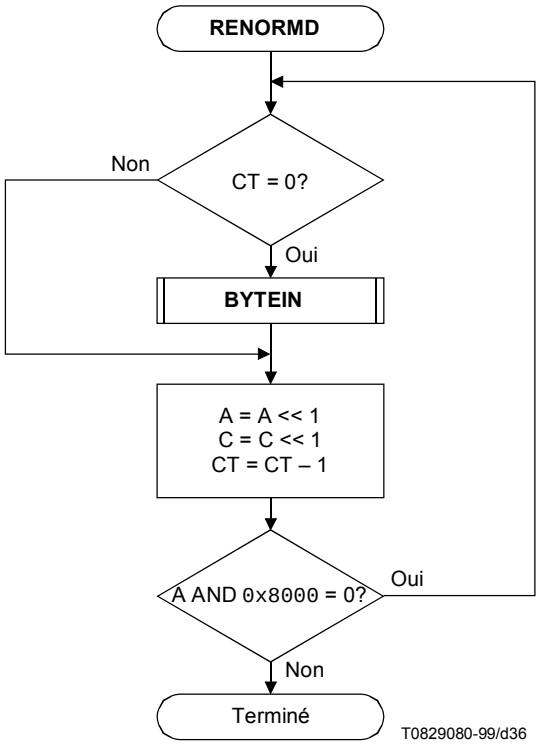


Figure E.18 – Procédure de renormalisation dans le décodeur

E.3.4 Entrée de données comprimées (BYTEIN)

La procédure BYTEIN, appelée par RENORMD, est illustrée dans la Figure E.19. Cette procédure lit en entrée un seul octet de données avec compensation des éventuels bits de bourrage suivant l'octet `0xFF` dans le processus. Elle détecte également les codes marqueurs qui doivent toujours apparaître à la fin d'un balayage ou d'un intervalle de synchronisation. Dans cette procédure, le registre C est la concaténation des registres Chigh et Clow.

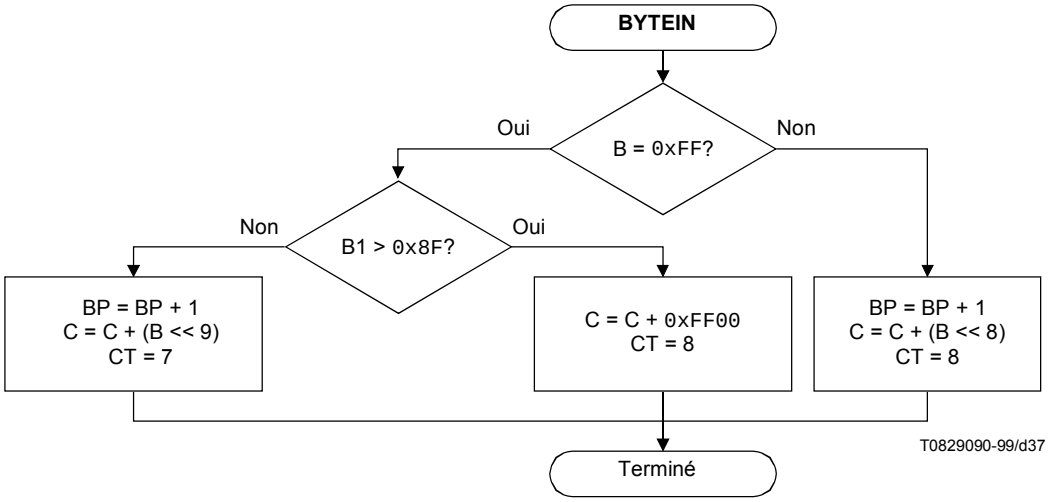


Figure E.19 – Procédure BYTEIN pour le décodeur

B est l'octet visé par le pointeur BP de tampon de données comprimées. Si B n'est pas un octet de type `0xFF`, le pointeur BP est incrémenté et la nouvelle valeur de B est insérée dans les 8 positions binaires supérieures de Clow.

Si B est un octet de type `0xFF`, l'on contrôle B1 (l'octet visé par BP+1). Si B1 est supérieur à `0x8F`, B1 doit être un des codes marqueurs. Le code marqueur est interprété comme requis et le pointeur de tampon reste pointé sur le préfixe `0xFF` du code marqueur terminant les données comprimées arithmétiquement. Les bits 1 sont ensuite injectés dans le décodeur jusqu'à ce que le décodage soit achevé, ce qui est indiqué par addition de la valeur `0xFF00` au registre C et réglage du compteur de bits CT sur 8.

Si B1 n'est pas un code marqueur, BP est incrémenté de façon à pointer sur l'octet suivant qui contient un bit de bourrage. B est ajouté au registre C avec un alignement tel que le bit de bourrage (qui contient un report quelconque) soit ajouté au bit de poids faible de Chigh.

E.3.5 Initialisation du décodeur (INITDEC)

La procédure INITDEC est utilisée pour initialiser le décodeur arithmétique. Les étapes de base sont indiquées dans la Figure E.20.

BP, pointeur sur les données comprimées, est initialisé à la valeur BPST pointant sur le premier octet comprimé. Celui-ci est décalé vers l'octet d'ordre inférieur de Chigh et un nouvel octet est lu en entrée. Le registre C est ensuite décalé de 7 bits puis CT est décrémenté de 7, ce qui met le registre C en alignement avec la valeur initiale de A. Le registre d'intervalle A est réglé de façon à correspondre à la valeur initiale dans le codeur.

E.3.6 Resynchronisation du décodeur

Habituellement, lorsque l'on atteint la fin des données arithmétiquement comprimées, le pointeur du tampon de données comprimées (BP) pointe sur l'octet `0xFF` du code marqueur final. Si, pour une raison ou une autre, le pointeur du tampon de données comprimées n'est pas à l'octet `0xFF` du marqueur, une procédure de synchronisation doit balayer les données comprimées jusqu'à ce qu'elle trouve le préfixe du code marqueur final. Si une recherche de ce type est requise, cela indique un état d'erreur. Cette procédure de reprise sur erreur n'est pas normalisée.

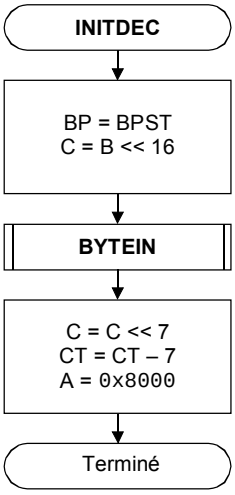


Figure E.20 – Initialisation du décodeur

E.3.7 Réinitialisation des statistiques de codage arithmétique

A certains moments au cours du décodage, tout ou partie des statistiques de codage arithmétique sont réinitialisées. Ce processus implique de rendre I(CX) et MPS(CX) égaux à zéro pour tout ou partie des valeurs de CX.

EXEMPLE – Au début du décodage d'un segment de région alphanumérique, toutes les statistiques de codage arithmétique sont réinitialisées.

E.3.8 Sauvegarde des statistiques de codage arithmétique

Dans certains cas, le décodeur a besoin de sauvegarder ou de restaurer certaines valeurs de I(CX) et de MPS(CX). Ces opérations sont effectuées dans le cadre du décodage d'un segment de dictionnaire de symboles. Dans ce cas, les valeurs qui sont sauvegardées ou restaurées sont toutes les valeurs indexées par les valeurs CX dont l'étiquette initiale est "GB" ou "GR" (c'est-à-dire toutes les valeurs CX qui sont utilisées par la procédure de décodage régionale générique ou par la procédure de décodage régionale générique par raffinement).

Annexe F

Profils

(Cette annexe fait partie intégrante de la présente Recommandation | Norme internationale)

Il est recommandé qu'un décodeur JBIG2 mette en œuvre un des profils décrits dans les Tableaux F.1 à F.7. Noter que le profil `0x00000001` (Tableau F.1) comporte toutes les capacités de la Recommandation | Norme internationale entière et que c'est le profil par défaut si aucun autre n'est spécifié.

Voir au 7.4.12 des informations sur la façon d'utiliser les numéros d'identification de profil.

Les numéros d'identification de profil allant de `0x00000000` à `0x00FFFFFF` sont réservés aux applications de l'ISO/CEI et de l'UIT. Dans cette fourchette, les numéros d'identification de profil `0x00000100` à `0x00000FFF` sont réservés aux applications de télécopieur UIT-T. Les entités autres qu'ISO/CEI et UIT souhaitant utiliser un numéro d'identification de profil non attribué doivent en choisir un dans l'étendue de `0x01000000` à `0xFFFFFFFF` à condition que ce numéro ne soit pas en conflit avec le choix d'une autre entité. Il est recommandé que les trois premiers octets du numéro d'identification de profil soient choisis de façon à correspondre aux trois premières lettres du nom de l'entité, ou qu'ils soient une abréviation appropriée de ce nom.

Tableau F.1 – Description de profil pour le profil `0x00000001`

Identification de profil	`0x00000001`
Exigences	Toutes les capacités JBIG2
Codage de région générique	Pas de restriction
Codage de région de raffinement	Pas de restriction
Codage de région de dégradé	Pas de restriction
Données numériques	Pas de restriction
Ressources requises	Processeur à grande vitesse
Exemples d'application	Impression à usage général; conversion de format

Tableau F.2 – Description de profil pour le profil `0x00000002`

Identification de profil	`0x00000002`
Exigences	Compression maximale
Codage de région générique	Codage arithmétique seulement; tout gabarit utilisé
Codage de région de raffinement	Pas de restriction
Codage de région de dégradé	Pas de restriction
Données numériques	Arithmétique seulement
Ressources requises	Processeur à grande vitesse
Exemples d'application	Archivage; WWW hertzien

Tableau F.3 – Description de profil pour le profil 0×00000003

NOTE 1 – Ce profil est un sous-ensemble du profil 0x00000002.

Identification de profil	0×00000003
Exigences	Complexité moyenne et compression moyenne
Codage de région générique	Arithmétique seulement; gabarits à 10 et à 13 pixels seulement
Codage de région de raffinement	Gabarit à 10 pixels seulement (arithmétique)
Codage de région de dégradé	Pas de masque d'omission
Données numériques	Arithmétique seulement
Ressources requises	Processeur à vitesse moyenne
Exemples d'application	WWW; télécopie évolué

Tableau F.4 – Description de profil pour le profil 0×00000004

Identification de profil	0x00000004
Exigences	Complexité basse avec capacité de progression sans pertes
Codage de région générique	MMR seulement
Codage de région de raffinement	Gabarit à 10 pixels seulement (arithmétique)
Codage de région de dégradé	Pas de masque d'omission
Données numériques	Huffman seulement
Ressources requises	Processeur à vitesse moyenne
Exemples d'application	WWW

Tableau F.5 – Description de profil pour le profil 0×00000005

NOTE 2 – Ce profil est un sous-ensemble du profil 0x00000004.

Identification de profil	0x00000005
Exigences	Complexité basse
Codage de région générique	MMR seulement
Codage de région de raffinement	Non disponible
Codage de région de dégradé	Pas de masque d'omission
Données numériques	Huffman seulement
Ressources requises	Processeur à faible vitesse
Exemples d'application	Télécopie classique; impression à grande vitesse; processeurs intégrés

Tableau F.6 – Description de profil pour le profil `0x00000006`

NOTE 3 – Ce profil est un sous-ensemble du profil `0x00000003`.

Identification de profil	`0x00000006`
Exigences	Peu de mémoire avec davantage de compression
Codage de région générique	Arithmétique uniquement; gabarits de 10 et de 13 pixels uniquement
Codage de région de raffinement	Gabarit de 10 pixels uniquement (arithmétique)
Codage de région de dégradé	Pas de masque d'omission
Données numériques	Arithmétique seulement
Ressources requises	Processeur à vitesse moyenne avec peu de mémoire
Exemples d'application	Dispositifs multifonction
Contraintes additionnelles	• Chaque page doit comporter au moins deux bandes. • Une bande qui contient un segment de région alphanumérique ne peut contenir aucun segment de région de dégradé ou de région générique. • Tous les segments de région doivent être des segments de région immédiate (pas de tampons auxiliaires). • Dans tout segment de dictionnaire de symboles où **SDREFAGG** est égal à **1**, REFAGGNINST doit être égal à 1 pour tous les symboles.

Tableau F.7 – Description de profil pour le profil `0x00000007`

NOTE 4 – Ce profil est un sous-ensemble du profil `0x00000005`.

Identification de profil	`0x00000007`
Exigences	Peu de mémoire et vitesse plus grande
Codage de région générique	MMR uniquement
Codage de région de raffinement	Non disponible
Codage de région de dégradé	Pas de masque d'omission
Données numériques	Huffman seulement
Ressources requises	Processeur à faible vitesse et peu de mémoire
Exemples d'application	Dispositifs multifonction
Contraintes additionnelles	• Chaque page doit comporter au moins deux bandes. • Une bande qui contient un segment de région alphanumérique ne peut contenir aucun segment de région de dégradé ou de région générique. • Tous les segments de région doivent être des segments de région immédiate (pas de tampons auxiliaires). • Dans tout segment de dictionnaire de symbole où **SDREFAGG** est égal à **1**, REFAGGNINST doit être égal à 1 pour tous les symboles.

Annexe G

Procédure de décodage arithmétique (conventions logicielles)

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

Cette annexe contient quelques variantes d'organigramme pour une version du décodeur entropique adaptatif. Cette version en variante peut être plus efficace lorsqu'elle est réalisée sous forme logicielle car elle comporte moins d'opérations dans le trajet rapide.

La version en variante est obtenue par exécution des substitutions suivantes:

- remplacer l'organigramme de la Figure E.20 par celui de la Figure G.1;
- remplacer l'organigramme de la Figure E.15 par celui de la Figure G.2;
- remplacer l'organigramme de la Figure E.19 par celui de la Figure G.3.

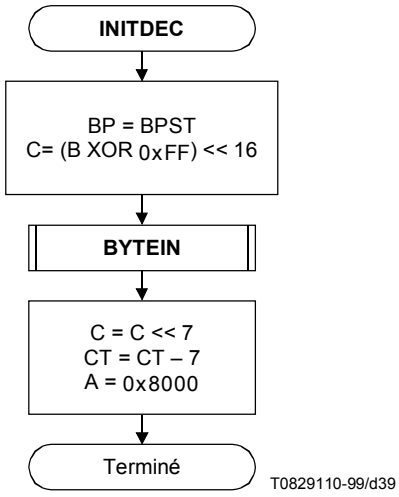


Figure G.1 – Initialisation du décodeur de conventions logicielles

118s4

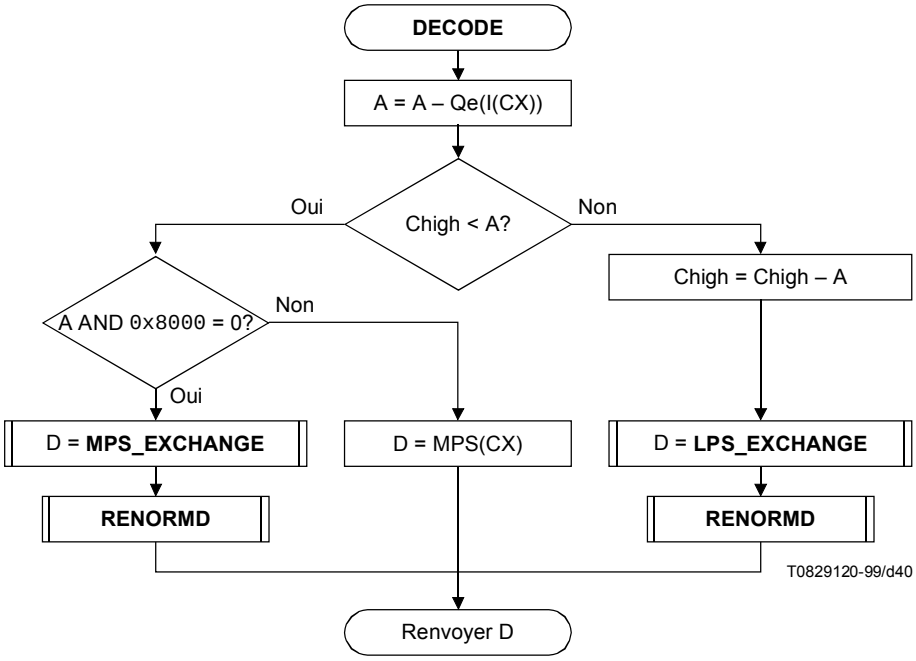


Figure G.2 – Décodage d'un symbole MPS ou LPS dans le décodeur de conventions logicielles

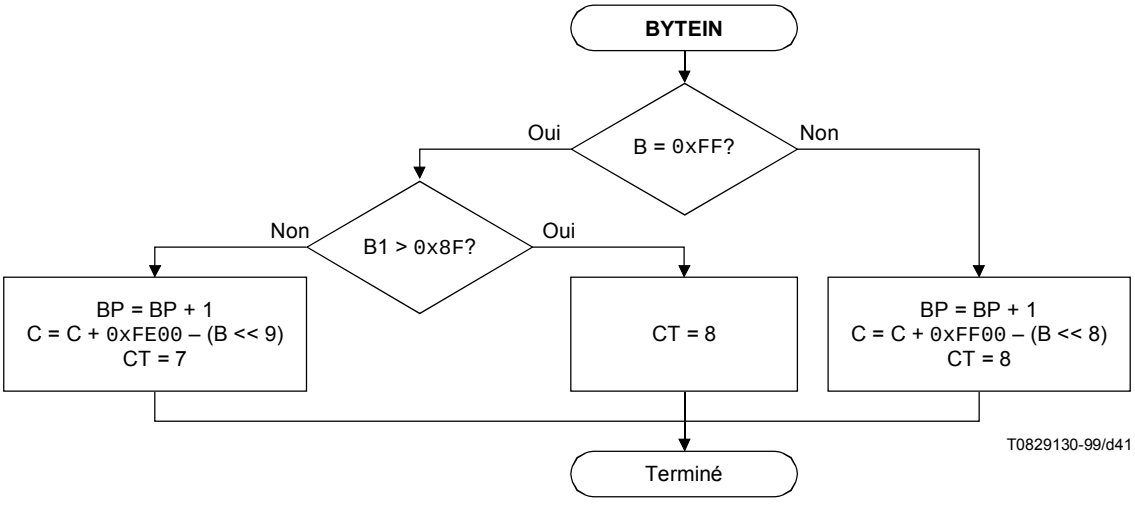


Figure G.3 – Insertion d'un nouvel octet dans le registre du décodeur de conventions logicielles

Annexe H

Exemple de flux de données et séquence d'essai

(Cette annexe ne fait pas partie intégrante de la présente Recommandation | Norme internationale)

H.1 Exemple de flux de données

Ce paragraphe présente un petit flux de données qui met en œuvre un grand nombre des caractéristiques du format JBIG2.

Les données brutes seront ici indiquées dans le format suivant:

```
0023: 01 23 45 67 89 AB CD EF
```

où le premier champ (avant les deux points) est le décalage d'octets effectué dans le flux des données en cours d'affichage et où le reste de la ligne est une séquence d'octets à partir de ce décalage. Toutes ces valeurs sont en notation hexadécimale.

En général, le décodage de la première manifestation d'un type de données quelconque est expliqué en détail. Les manifestations suivantes du même type de données ne sont pas expliquées avec autant de détails.

Ce flux de données code:

- un dictionnaire de symboles qui n'est associé à aucune page;
- et trois pages,

c'est-à-dire que les entités codées dans ce flux de données sont trois pages plus un seul objet qui n'est associé à aucune page.

Les deux premières pages contiennent chacune un segment d'informations de page, un segment de dictionnaire de symboles, un segment de région alphanumérique, un segment de région générique, un segment de dictionnaire de structures, un segment de région de dégradé et un segment de fin de page. Les phototrames codées par ces deux pages sont identiques et sont représentées sur la Figure H.1. Les données codées par les segments correspondants dans les deux pages sont également identiques (par exemple le segment de région alphanumérique pour la page 1 contient les mêmes données, dans le même ordre, que le segment de région alphanumérique pour la page 2). Cependant, les segments sont codés différemment dans les deux pages: dans la page 1, tous les segments utilisent une forme de codage de Huffman ou MMR; dans la page 2, tous les segments utilisent une forme de codage arithmétique. Les réalisateurs peuvent donc vérifier en fonction de leurs réalisations s'ils effectuent le décodage correctement.

La troisième page contient deux dictionnaires de symboles, dont l'un définit les symboles par raffinement et agrégation à partir de l'autre dictionnaire; elle contient également une région alphanumérique qui utilise les symboles extraits du dictionnaire, éventuellement après raffinement de l'un d'eux.

Dans tout ce paragraphe, les pixels ayant la valeur **1** sont indiqués sous forme de pixels noirs, alors que les pixels ayant la valeur **0** sont indiqués sous forme de pixels blancs. Il s'agit d'une interprétation typique des valeurs **0** et **1**, telle qu'une application utilisant la présente Recommandation | Norme internationale peut le faire.

Le flux de données est le suivant:

```
0000: 97 4A 42 32 0D 0A 1A 0A 01 00 00 00 03 00 00 00
0010: 00 00 01 00 00 00 00 18 00 01 00 00 00 01 00 00
0020: 00 01 E9 CB F4 00 26 AF 04 BF F0 78 2F E0 00 40
0030: 00 00 00 01 30 00 01 00 00 00 13 00 00 00 40 00
0040: 00 00 38 00 00 00 00 00 00 00 00 01 00 00 00 00
0050: 00 02 00 01 01 00 00 00 1C 00 01 00 00 00 02 00
0060: 00 00 02 E5 CD F8 00 79 E0 84 10 81 F0 82 10 86
0070: 10 79 F0 00 80 00 00 00 03 07 42 00 02 01 00 00
0080: 00 31 00 00 00 25 00 00 00 08 00 00 00 04 00 00
0090: 00 01 00 0C 09 00 10 00 00 00 05 01 10 00 00 00
00A0: 00 00 00 00 00 00 00 00 00 00 00 00 0C 40 07 08
00B0: 70 41 D0 00 00 00 04 27 00 01 00 00 00 2C 00 00
00C0: 00 36 00 00 00 2C 00 00 00 04 00 00 00 0B 00 01
00D0: 26 A0 71 CE A7 FF FF FF FF FF FF FF FF FF FF FF
00E0: FF FF FF FF FF FF FF FF F8 F0 00 00 00 05 10 01
00F0: 01 00 00 00 2D 01 04 04 00 00 00 0F 20 D1 84 61
0100: 18 45 F2 F9 7C 8F 11 C3 9E 45 F2 F9 7D 42 85 0A
0110: AA 84 62 2F EE EC 44 62 22 35 2A 0A 83 B9 DC EE
```

```
0120: 77 80 00 00 00 06 17 20 05 01 00 00 00 57 00 00
0130: 00 20 00 00 00 24 00 00 00 10 00 00 00 0F 00 01
0140: 00 00 00 08 00 00 00 09 00 00 00 00 00 00 00 00
0150: 04 00 00 00 AA AA AA AA 80 08 00 80 36 D5 55 6B
0160: 5A D4 00 40 04 2E E9 52 D2 D2 D2 8A A5 4A 00 20
0170: 02 23 E0 95 24 B4 92 8A 4A 92 54 92 D2 4A 29 2A
0180: 49 40 04 00 40 00 00 00 07 31 00 01 00 00 00 00
0190: 00 00 00 08 30 00 02 00 00 00 13 00 00 00 40 00
01A0: 00 00 38 00 00 00 00 00 00 00 00 01 00 00 00 00
01B0: 00 09 00 01 02 00 00 00 1B 08 00 02 FF 00 00 00
01C0: 02 00 00 00 02 4F E7 8C 20 0E 1D C7 CF 01 11 C4
01D0: B2 6F FF AC 00 00 00 0A 07 40 00 09 02 00 00 00
01E0: 1F 00 00 00 25 00 00 00 08 00 00 00 04 00 00 00
01F0: 01 00 0C 08 00 00 00 05 8D 6E 5A 12 40 85 FF AC
0200: 00 00 00 0B 27 00 02 00 00 00 23 00 00 00 36 00
0210: 00 00 2C 00 00 00 04 00 00 00 0B 00 08 03 FF FD
0220: FF 02 FE FE FE 04 EE ED 87 FB CB 2B FF AC 00 00
0230: 00 0C 10 01 02 00 00 00 1C 06 04 04 00 00 00 0F
0240: 90 71 6B 6D 99 A7 AA 49 7D F2 E5 48 1F DC 68 BC
0250: 6E 40 BB FF AC 00 00 00 0D 17 20 0C 02 00 00 00
0260: 3E 00 00 00 20 00 00 00 24 00 00 00 10 00 00 00
0270: 0F 00 02 00 00 00 08 00 00 00 09 00 00 00 00 00
0280: 00 00 00 04 00 00 00 87 CB 82 1E 66 A4 14 EB 3C
0290: 4A 15 FA CC D6 F3 B1 6F 4C ED BF A7 BF FF AC 00
02A0: 00 00 0E 31 00 02 00 00 00 00 00 00 00 0F 30 00
02B0: 03 00 00 00 13 00 00 00 25 00 00 00 08 00 00 00
02C0: 00 00 00 00 00 01 00 00 00 00 00 10 00 01 00 00
02D0: 00 00 16 08 00 02 FF 00 00 00 01 00 00 00 01 4F
02E0: E7 8D 68 1B 14 2F 3F FF AC 00 00 00 11 00 21 10
02F0: 03 00 00 00 20 08 02 02 FF FF FF FF FF 00 00 00
0300: 03 00 00 00 02 4F E9 D7 D5 90 C3 B5 26 A7 FB 6D
0310: 14 98 3F FF AC 00 00 00 12 07 20 11 03 00 00 00
0320: 25 00 00 00 25 00 00 00 08 00 00 00 00 00 00 00
0330: 00 00 8C 12 00 00 00 04 A9 5C 8B F4 C3 7D 96 6A
0340: 28 E5 76 8F FF AC 00 00 00 13 31 00 03 00 00 00
0350: 00 00 00 00 14 33 00 00 00 00 00 00
```

Ce flux de données est décodé comme suit:

1) L'en-tête de fichier:

   `0000: 97 4A 42 32 0D 0A 1A 0A 01 00 00 00 02`

   a) La chaîne d'identification de 8 octets:

      `0000: 97 4A 42 32 0D 0A 1A 0A`

   b) Le champ des fanions d'en-tête de fichier, de 1 octet:

      `0008: 01`

      Cette valeur indique que le fichier utilise l'organisation séquentielle et que le nombre de pages est connu.

   c) Le champ du nombre de pages, de 4 octets:

      `0009: 00 00 00 03`

      Cette valeur indique que le fichier a trois pages.

2) L'en-tête du premier segment:

   `000D: 00 00 00 00 00 01 00 00 00 00 18`

   a) Le champ du numéro de segment, de 4 octets:

      `000D: 00 00 00 00`

      Cette valeur indique que le segment porte le numéro 0.

   b) Le champ des fanions d'en-tête de segment, de 1 octet:

      `0011: 00`

      Cette valeur indique que le segment est du type "dictionnaire de symboles" (type 0), qu'il possède un champ court d'association de page et que son bit de non-rétention différée n'est pas activé.

   c) Le champ de comptage de segments référencés et de fanions de rétention, de 1 octet:

      `0012: 01`

      Cette valeur indique que le segment ne fait référence à aucun autre segment et qu'il doit être retenu.

d) Le champ d'association de page du segment, de 1 octet (forme courte):

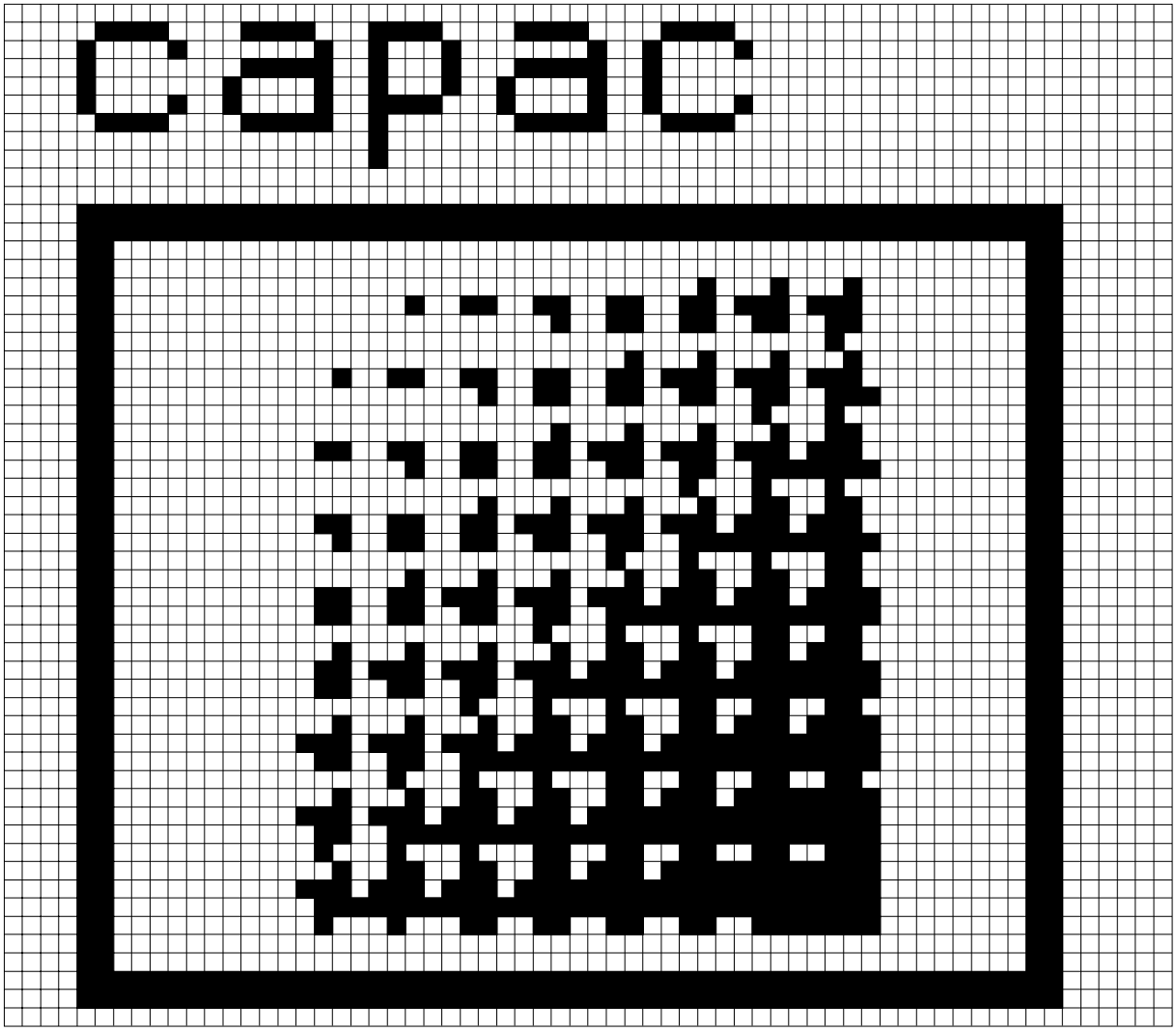
```
0013: 00
```

Cette valeur indique que ce segment n'est associé à aucune page.

e) Le champ de longueur de données du segment, de 4 octets:

```
0014: 00 00 00 18
```

Cette valeur indique que la partie données du segment a une longueur de 24 octets.



T0829140-99/d42

Figure H.1 – Phototrame

3) La partie donnée du premier segment:

```
0018: 00 01 00 00 00 01 00 00 00 01 E9 CB F4 00 26 AF
0028: 04 BF F0 78 2F E0 00 40
```

a) Le champ des fanions de dictionnaire de symbole, de 2 octets:

```
0018: 00 01
```

Cette valeur indique que le segment est codé au moyen de la variante de codage de Huffman, qu'il ne fait pas appel au codage de raffinement/d'agrégation, qu'il utilise le Tableau B.4 pour **SDHUFFDH** et le Tableau B.2 pour **SDHUFFDW**.

b) Le champ **SDNUMEXSYMS**, de 4 octets:

```
001A: 00 00 00 01
```

Cette valeur indique que **SDNUMEXSYMS** = 1, c'est-à-dire que 1 symbole est exporté par ce dictionnaire de symboles.

c) Le champ **SDNUMNEWSYMS** de 4 octets:

```
001E: 00 00 00 01
```

Cette valeur indique que **SDNUMNEWSYMS** = 1, c'est-à-dire que 1 symbole est défini par ce dictionnaire de symboles.

d) Les données codées du dictionnaire de symboles:

```
0022: E9 CB F4 00 26 AF 04 BF F0 78 2F E0 00 40
```

Ces données sont décodées comme suit:

i) au moyen du Tableau B.4, décoder une valeur de hauteur différentielle de classe de hauteurs. Cette valeur consomme les bits **1110 100**, indiquant une hauteur différentielle de classe de hauteurs égale à 8. La première classe de hauteurs a donc une hauteur de 8 pixels;

ii) au moyen du Tableau B.2, décoder une valeur de largeur différentielle. Cette valeur consomme les bits **1110 010**, indiquant une largeur différentielle égale à 5. Le premier symbole a donc une largeur de 5 pixels;

iii) au moyen du Tableau B.2, décoder une valeur de largeur différentielle. Cette valeur consomme les bits **111111**, indiquant une largeur différentielle de OOB. Cela termine la classe de hauteurs, qui contient un seul symbole dont la largeur est 5 pixels et la hauteur 8 pixels;

iv) au moyen du Tableau B.1, décoder la longueur en octets de la matrice collective des classes de hauteur. Cette valeur consomme les bits **01000**, indiquant une longueur de 8 octets;

v) omettre les bits restant dans le dernier octet lu. Cela consomme les bits **0000000**;

vi) décoder les 8 octets suivants:

```
0026: 26 AF 04 BF F0 78 2F E0
```

au moyen de l'algorithme MMR, ce qui produit la matrice collective des classes de hauteur qui est également la phototrame pour le symbole isolé dans la classe de hauteurs, représenté sur la Figure H.2;

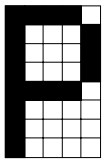


Figure H.2 – Premier symbole du premier dictionnaire de symboles

vii) comme **SDNUMNEWSYMS** = 1, le dernier symbole a donc été décodé;

viii) au moyen du Tableau B.1, décoder une longueur de séquence d'exportation. Cela consomme les bits **00000**, indiquant que les premiers symboles 0 ne sont pas exportés;

ix) au moyen du Tableau B.1, décoder une longueur de séquence d'exportation. Cela consomme les bits **00001**, indiquant que les prochains symboles 1 seront exportés. Ce dictionnaire de symboles définit donc un seul symbole, qui est exporté;

x) omettre les bits restant dans le dernier octet, ce qui consomme les bits **000000**.

4) Le deuxième en-tête de segment:

```
0030: 00 00 00 01 30 00 01 00 00 00 13
```

Ce segment possède le numéro 1, le type "informations de page" (type 48) et un champ court d'association de page. Son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et n'est pas retenu. Il est associé à la page 1 et a une longueur de données de 19 octets.

5) La partie donnée du deuxième segment:

```
003B: 00 00 00 40 00 00 00 38 00 00 00 00 00 00 00 00
004B: 01 00 00
```

a) Le champ de largeur de matrice de page:

```
003B: 00 00 00 40
```

Cette valeur indique que la page a une largeur de 64 pixels.

b) Le champ de hauteur de matrice de page:

```
003F: 00 00 00 38
```

Cette valeur indique que la page a une hauteur de 56 pixels.

c) Le champ de résolution horizontale:

```
0043: 00 00 00 00
```

Cette valeur indique que la résolution horizontale de la page est inconnue.

d) Le champ de résolution verticale:

```
0047: 00 00 00 00
```

Cette valeur indique que la résolution verticale de la page est inconnue.

e) Le champ de fanions de segment de page:

```
004B: 01
```

Cette valeur indique que la page sera finalement sans pertes, qu'elle ne contient pas de raffinements, que sa valeur de pixel par défaut est **0**, que son opérateur combinatoire par défaut est OR, qu'elle n'exige pas de tampons auxiliaires et que son opérateur combinatoire par défaut sera utilisé par chacun de ses segments de région.

f) Le champ d'informations de découpage en bande de la page:

```
004C: 00 00
```

Cette valeur indique que la page n'est pas découpée en bandes.

6) L'en-tête du troisième segment:

```
004E: 00 00 00 02 00 01 01 00 00 00 1C
```

Ce segment porte le numéro 2, est du type "dictionnaire de symboles" (type 0), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et est retenu. Il est associé à la page 1 et ses données ont une longueur de 28 octets.

7) La partie données du troisième segment::

```
0059: 00 01 00 00 00 02 00 00 00 02 E5 CD F8 00 79 E0
0069: 84 10 81 F0 82 10 86 10 79 F0 00 80
```

a) Le champ de fanions du dictionnaire de symboles, de 2 octets:

```
0059: 00 01
```

Cette valeur indique que le segment est codé au moyen de la variante de codage de Huffman, qu'il n'utilise pas le codage de raffinement/d'agrégation, qu'il utilise le Tableau B.4 pour **SDHUFFDH** et le Tableau B.2 pour **SDHUFFDW**.

b) Le champ **SDNUMEXSYMS**, de 4 octets:

```
005B: 00 00 00 02
```

Cette valeur indique que **SDNUMEXSYMS** = 2, c'est-à-dire que 2 symboles sont exportés par ce dictionnaire de symboles.

c) Le champ **SDNUMNEWSYMS**, de 4 octets:

```
005F: 00 00 00 02
```

Cette valeur indique que **SDNUMNEWSYMS** = 2, c'est-à-dire que 2 symboles sont définis par ce dictionnaire de symboles.

d) Les données codées du dictionnaire de symboles:

```
0063: E5 CD F8 00 79 E0 84 10 81 F0 82 10 86 10 79 F0
0073: 00 80
```

Ces données sont décodées comme suit:

i) au moyen du Tableau B.4, décoder une valeur de hauteur différentielle de classe de hauteurs. Cette valeur consomme les bits **1110 010**, indiquant une hauteur différentielle de classe de hauteurs égale à 6. La première classe de hauteurs a donc une hauteur de 6 pixels;

ii) au moyen du Tableau B.2, décoder une valeur de largeur différentielle. Cette valeur consomme les bits **1110 011**, indiquant une largeur différentielle égale à 6. Le premier symbole a donc une largeur de 6 pixels;

iii) au moyen du Tableau B.2, décoder une valeur de largeur différentielle. Cette valeur consomme les bits **0**, indiquant une largeur différentielle de 0. Le deuxième symbole a donc une largeur de 6 pixels;

iv) au moyen du Tableau B.2, décoder une valeur de largeur différentielle. Cette valeur consomme les bits **111111**, indiquant une largeur différentielle de OOB. Cela termine la classe de hauteurs, qui contient deux symboles ayant tous les deux une largeur de 6 pixels et une hauteur de 6 pixels;

iv) au moyen du Tableau B.1, décoder la longueur en octets de la matrice collective des classes de hauteur. Cette valeur consomme les bits **00000**, indiquant une longueur de 0 octet et le fait que la matrice collective des classes de hauteur est mémorisée sans compression. Comme la largeur totale des symboles dans la classe de hauteurs est égale à 12, chaque rangée de la classe de hauteurs est bourrée de façon à avoir une largeur de 16 bits (2 octets);

vi) omettre les bits restant dans le dernier octet lu. Cela consomme les bits **000000;**

vii) lire les 12 octets suivants (6 rangées de 2 octets chacune) et utiliser les 12 bits situés le plus à gauche dans chaque rangée comme matrice collective de classes de hauteur. Ces 12 octets sont les suivants:

```
0067: 79 E0 84 10 81 F0 82 10 86 10 79 F0
```

soit, en notation binaire:

01111001	**11100000**
10000100	**00010000**
10000001	**11110000**
10000010	**00010000**
10000110	**00010000**
01111001	**11110000**

La matrice collective de classes de hauteur est donc conforme à la Figure H.3 et les deux symboles sont conformes à la Figure H.4 a) et b).

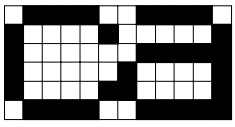


Figure H.3 – Matrice collective de classes de hauteur dans le second dictionnaire de symboles

viii) comme **SDNUMNEWSYMS** = 2, le dernier symbole a maintenant été décodé;

ix) au moyen du Tableau B.1, décoder une longueur de séquence d'exportation. Cela consomme les bits **00000**, indiquant que les premiers symboles 0 ne sont pas exportés;

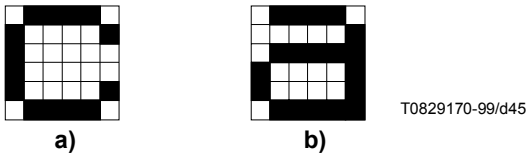


Figure H.4 – Symboles dans le second dictionnaire de symboles

x) au moyen du Tableau B.1, décoder une longueur de séquence d'exportation. Cela consomme les bits **00010**, indiquant que les 2 symboles suivants sont exportés. Ce dictionnaire de symboles définit donc 2 symboles, qui sont tous les deux exportés;

xi) omettre les bits restant dans le dernier octet. Cela consomme les bits **000000**.

8) L'en-tête du 4e segment:

```
0075: 00 00 00 03 07 42 00 02 01 00 00 00 31
```

Ce segment a le numéro de segment 3, est du type "région alphanumérique immédiate sans pertes" (type 7), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il fait référence à deux autres segments, portant les numéros 0 et 2. Le segment 0 doit être retenu mais le segment 2 et le segment actuel ne doivent pas être retenus. Ce segment est associé à la page 1 et ses données ont une longueur de 49 octets.

9) La partie donnée du 4e segment:

```
0082: 00 00 00 25 00 00 00 08 00 00 00 04 00 00 00 01
0092: 00 0C 09 00 10 00 00 00 05 01 10 00 00 00 00 00
00A2: 00 00 00 00 00 00 00 00 00 00 0C 40 07 08 70 41
00B2: D0
```

a) Le champ de largeur de matrice de segment de région:

```
0082: 00 00 00 25
```

Cette valeur indique que la matrice de région a une largeur de 37 pixels.

b) Le champ de hauteur de matrice de segment de région:

```
0086: 00 00 00 08
```

Cette valeur indique que la matrice de segment de région a une hauteur de 8 pixels.

c) Le champ de décalage horizontal de la matrice de segment de région:

```
008A: 00 00 00 04
```

Cette valeur indique que le bord gauche de la matrice de région est décalé de 4 pixels à droite du bord gauche de la page.

d) Le champ de décalage vertical de la matrice de segment de région:

```
008E: 00 00 00 01
```

Cette valeur indique que le bord supérieur de la matrice de région est décalé de 1 pixel au-dessous du bord supérieur de la page.

e) Le champ de fanions de segment de région:

```
0092: 00
```

Cette valeur indique que la région doit être insérée dans la page au moyen de l'opérateur combinatoire OR.

f) Le champ de fanions de segment de région alphanumérique, de 2 octets:

```
0093: 0C 09
```

Cette valeur indique que le segment est codé au moyen de la variante de Huffman, qu'il ne contient pas de raffinements, qu'il possède une valeur **SBSTRIPS** = 4, qu'il a un coin de référence égal à BOTTOMLEFT, qu'il n'est pas transposé, qu'il combine ses symboles au moyen de l'opérateur OR, qu'il a une valeur de pixel par défaut de **0** et qu'il a une valeur **SBDSOFFSET** = 3.

g) Le champ de fanions de codage de Huffman de segment de région alphanumérique, de 2 octets:

```
0095: 00 10
```

Cette valeur indique que le segment utilise le Tableau B.6 pour **SBHUFFFS**, le Tableau B.8 pour **SBHUFFDS** et le Tableau B.12 pour **SBHUFFDT**.

h) Le champ **SBNUMINSTANCES**, de 4 octets:

```
0097: 00 00 00 05
```

Ce champ indique que **SBNUMINSTANCES** = 5, c'est-à-dire que 5 instances de symbole sont codées dans cette région alphanumérique.

i) La table de décodage de Huffman des identificateurs de symbole de segment de région alphanumérique:

```
009B: 01 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00AB: 00 0C
```

Les deux dictionnaires de symboles auxquels ce segment fait référence contiennent un total de 3 symboles. Le décodage de la table de Huffman RUNCODE consomme toutes les données sauf les 4 derniers bits et donne l'attribution suivante des longueurs de code aux séquences RUNCODE:

RUNCODE1	1	RUNCODE2	1

et donne donc la table de Huffman suivante pour les séquences RUNCODE:

RUNCODE1	**0**	RUNCODE2	**1**

Le décodage au moyen de cette table produit la séquence RUNCODE2, RUNCODE2, RUNCODE1 (consommant les bits **110**). Le premier symbole (le "p" extrait du premier dictionnaire de symboles) a donc une longueur codée de 2; le deuxième symbole (le "c" extrait du deuxième dictionnaire de symboles) a une longueur de code de Huffman égale à 2 et le troisième symbole (le "a" extrait du deuxième dictionnaire de symboles) a une longueur de code égale à 1. L'application de l'algorithme d'attribution des codes de Huffman produit la table:

p	**10**
c	**11**
a	**0**

A ce point, il reste 1 bit (**0**) dans le dernier octet de données. Ce bit est donc omis.

j) Les données de région alphanumérique codées:

```
00AD: 40 07 08 70 41 D0
```

Ces données sont décodées comme suit:

i) au moyen du Tableau B.12, décoder une valeur delta T. Cela consomme les bits **0**, indiquant une valeur delta T de 4 (soit la valeur de table décodée à 1, multipliée par **SBSTRIPS**). La valeur initiale de STRIPT est –4;

ii) au moyen du Tableau B.12, décoder une valeur delta T. Cela consomme les bits **10**, indiquant une valeur delta T de 8 (2 fois **SBSTRIPS**). STRIPT devient donc 4;

iii) au moyen du Tableau B.6, décoder une première valeur S. Cela consomme les bits **00 00000000**, indiquant une première valeur S = 0;

iv) la lecture de 2 bits (depuis **SBSTRIPS**) consomme les bits **01**. La coordonnée T de la première instance de symbole est donc 5 (STRIPT plus la valeur décodée de 1);

v) au moyen de la table de Huffman des identificateurs de symbole, décoder une valeur d'identificateur de symbole. Cela consomme les bits **11**, indiquant le symbole "c". Celui-ci sera donc inséré avec son coin gauche inférieur à (0, 5);

vi) a ce point, CURS = 8 (0 plus **SBDSOFFSET**) plus la largeur du symbole précédent (6-1);

vii) au moyen du Tableau B.8, décoder une valeur delta S. Cela consomme les bits **00 0**, indiquant une valeur delta S = 0;

viii) la lecture de 2 bits consomme les bits **01**. La coordonnée T de la deuxième instance de symbole est donc 5;

ix) au moyen de la table de Huffman des identificateurs de symbole, décoder une valeur d'identificateur de symbole. Cela consomme les bits **0**, indiquant le symbole "a". Celui-ci sera donc inséré avec son coin gauche inférieur à (8, 5);

x) a ce point, CURS = 16 (8 plus **SBDSOFFSET** plus la largeur du symbole précédent de 6–1);

xi) au moyen du Tableau B.8, décoder une valeur delta S. Cela consomme les bits **00 0**, indiquant une valeur delta S = 0;

xii) la lecture de 2 bits consomme les bits **11**. La coordonnée T de la troisième instance de symbole est donc 7;

xiii) au moyen de la table de Huffman des identificateurs de symbole, décoder une valeur d'identificateur de symbole. Cela consomme les bits **10**, indiquant le symbole "p". Celui-ci sera donc inséré avec son coin gauche inférieur à (16, 7);

xiv) a ce point, CURS = 23 (16 + **SBDSOFFSET** + la largeur 5 du symbole précédent 1);

xv) au moyen du Tableau B.8, décoder une valeur delta S. Cela consomme les bits **00 0**, indiquant une valeur delta S de 0;

xvi) la lecture de 2 bits consomme les bits **01**. La coordonnée T de la quatrième instance de symbole est donc 5;

xvii) au moyen de la table de Huffman des identificateurs de symbole, décoder une valeur d'identificateur de symbole. Cela consomme les bits **0**, ce qui indique le symbole "a". Celui-ci doit donc être inséré avec son coin inférieur gauche à (8, 5);

xviii) a ce point, CURS = 31 (23 + **SBDSOFFSET** + la largeur 6 du symbole précédent – 1);

xix) au moyen du Tableau B.8, décoder une valeur delta S. Cela consomme les bits **00 0**, indiquant une valeur delta S = 0;

xx) la lecture de 2 bits consomme les bits **01**. La coordonnée T de la cinquième instance de symbole est donc 5;

xxi) au moyen de la table de Huffman des identificateurs de symbole, décoder une valeur d'identificateur de symbole. Cela consomme les bits **11**, indiquant le symbole "c". Celui-ci sera donc inséré avec son coin inférieur gauche à (31, 5);

xxii) au moyen du Tableau B.8, décoder une valeur delta S. Cela consomme les bits **01**, indiquant une valeur delta S = OOB, c'est-à-dire la fin de cette bande. Comme **SBNUMINSTANCES** = 5 et comme 5 instances de symbole ont été décodées, il n'y a plus de bandes;

xxiii) omettre les bits restant dans le dernier octet lu. Cela consomme les bits **0000**;

k) Le décodage des données produit la liste suivante d'instances de symbole et de coordonnées (indiquant où doit être placé le coin inférieur gauche du symbole):

Symbole	Coordonnées
c	(0, 5)
a	(8, 5)
p	(16, 7)
a	(23, 5)
c	(31, 5)

L'insertion de ces instances de symbole produit la matrice de région de 37 × 8 pixels représentée sur la Figure H.5.

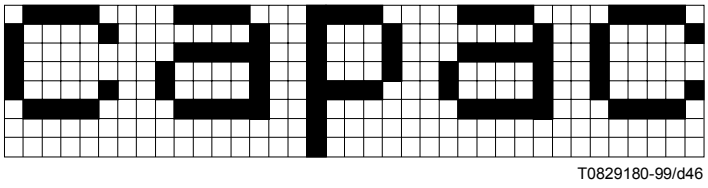


Figure H.5 – Phototrame de région alphanumérique

10) L'en-tête du 5e segment:

```
00B3: 00 00 00 04 27 00 01 00 00 00 2C
```

Ce segment porte le numéro 4, est de type "région générique immédiate sans pertes" (type 39), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et n'est pas retenu. Il est associé à la page 1 et ses données ont une longueur de 44 octets.

11) La partie donnée du 5e segment:

```
00BE: 00 00 00 36 00 00 00 2C 00 00 00 04 00 00 00 0B
00CE: 00 01 26 A0 71 CE A7 FF FF FF FF FF FF FF FF FF
00DE: FF FF FF FF FF FF FF FF FF FF F8 F0
```

a) Le champ d'informations de segment de région:

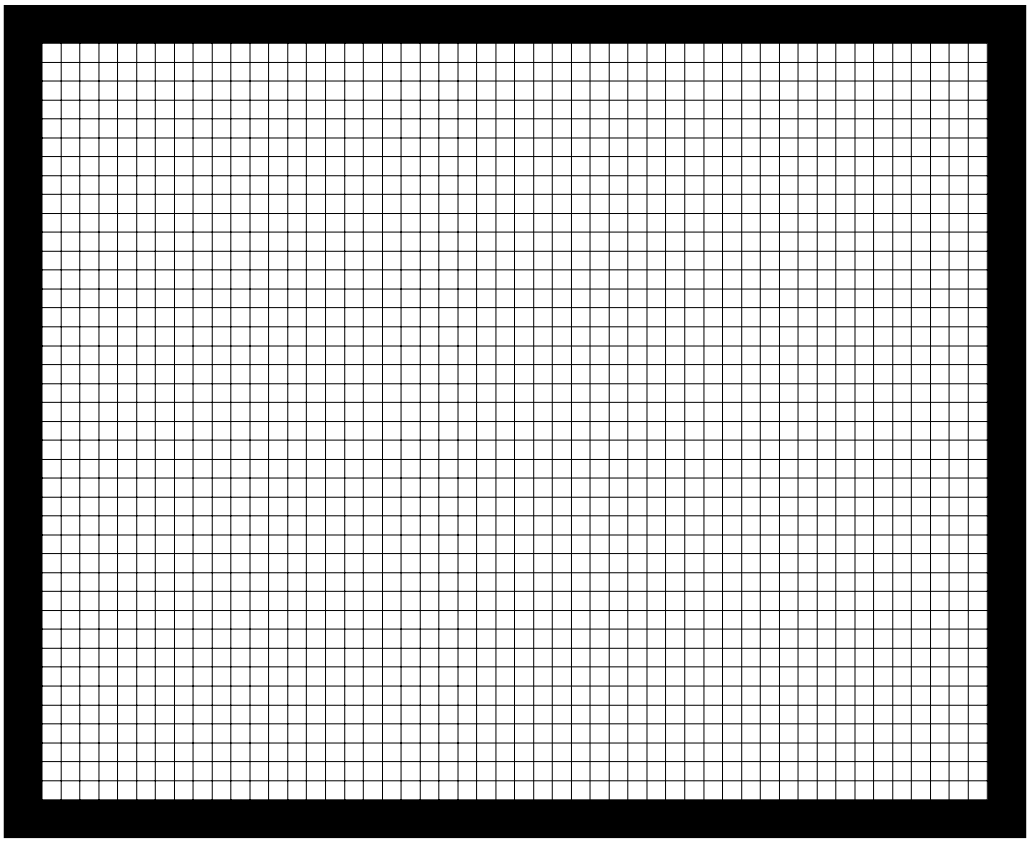
```
00BE: 00 00 00 36 00 00 00 2C 00 00 00 04 00 00 00 0B
00CE: 00
```

Cette valeur indique que la matrice de région codée par ce segment a une largeur de 54 pixels, une hauteur de 4 pixels et que son coin supérieur gauche est décalé de 4 pixels à droite du bord gauche de la page et de 11 pixels au-dessous du bord supérieur de la page. Il doit être inséré dans la page au moyen de l'opérateur OR.

b) Les données de région:

```
00CF: 01 26 A0 71 CE A7 FF FF FF FF FF FF FF FF FF FF
00DF: FF FF FF FF FF FF FF FF FF F8 F0
```

Le premier octet (01) est celui des fanions de segment de région générique. Il indique que la région est codée en MMR. Les octets suivants sont les données à codage MMR pour la matrice de région. Ces octets MMR se décompriment pour former la matrice de région de 54×44 pixels représentée sur la Figure H.6.



T0829190-99/47

Figure H.6 – Phototrame de région générique

12) L'en-tête du 6e segment:

```
00EA: 00 00 00 05 10 01 01 00 00 00 2D
```

Ce segment porte le numéro 5, est du type "dictionnaire de structures" (type 16), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait pas référence à d'autres segments et est retenu. Il est associé à la page 1 et ses données ont une longueur de 45 octets.

13) La partie donnée du 6e segment:

```
00F5: 01 04 04 00 00 00 0F 20 D1 84 61 18 45 F2 F9 7C
0105: 8F 11 C3 9E 45 F2 F9 7D 42 85 0A AA 84 62 2F EE
0115: EC 44 62 22 35 2A 0A 83 B9 DC EE 77 80
```

a) Le champ de fanions de dictionnaire de structures, de 1 octet:

```
00F5: 01
```

Cette valeur indique que le segment est codé au moyen de la variante de codage MMR.

b) Le champ **HDPW**, de 1 octet:

```
00F6: 04
```

Cette valeur indique que **HDPW**, largeur des structures définies dans ce dictionnaire, est 4.

c) Le champ **HDPH**, de 1 octet:

```
00F7: 04
```

Cette valeur indique que **HDPH**, hauteur des structures définies dans ce dictionnaire, est 4.

d) Le champ **GRAYMAX**, de 4 octets:

```
00F8: 00 00 00 0F
```

Cette valeur indique que **GRAYMAX** = 15 et qu'il y a donc 16 structures dans ce dictionnaire de structures (numérotées de 0 à 15).

e) Les 38 octets restant dans le segment:

```
00FC: 20 D1 84 61 18 45 F2 F9 7C 8F 11 C3 9E 45 F2 F9
010C: 7D 42 85 0A AA 84 62 2F EE EC 44 62 22 35 2A 0A
011C: 83 B9 DC EE 77 80
```

Ces octets sont décomprimés en MMR pour former la matrice collective du dictionnaire de structures. La largeur de cette matrice est (**GRAYMAX** + 1) × **HDPW** et sa hauteur est de **HDPH** pixels. La matrice de 64 × 4 pixels qui résulte de cette décompression MMR est représentée sur la Figure H.7.

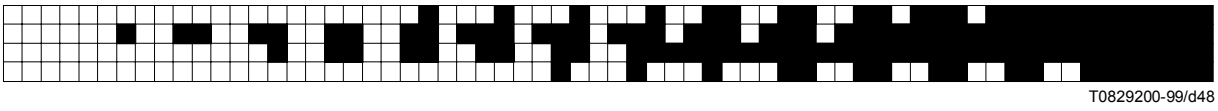


Figure H.7 – Matrice collective du dictionnaire de structures

Les 16 structures individuelles sont obtenues à partir de la matrice collective par subdivision de celle-ci en éléments de 4 pixels de largeur. Ces éléments sont représentés sur la Figure H.8, dans laquelle la structure numéro 0 est celle du coin supérieur gauche, la structure numéro 1 celle qui est à droite de la précédente, et ainsi de suite.

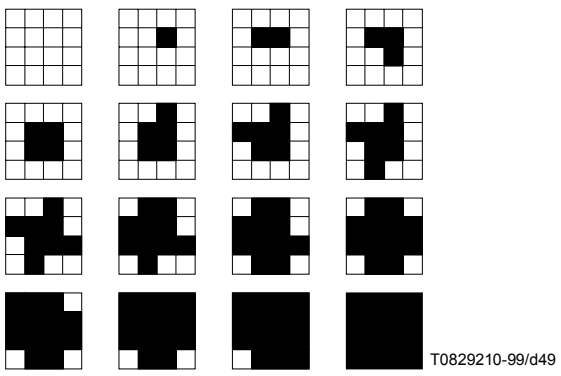


Figure H.8 – Structures définies dans le dictionnaire de structures

14) L'en-tête du 7e segment:

```
0122: 00 00 00 06 17 20 05 01 00 00 00 57
```

Ce segment porte le numéro 6, est du type "région de dégradé immédiate sans pertes" (type 23), possède un champ court d'association de page et son bit de non-rétention différée n'est pas activé. Il fait référence à un seul autre segment: le segment numéro 5. Ni le segment numéro 5 ni ce segment ne doivent être retenus. Il est associé à la page 1 et ses données ont une longueur de 87 octets.

15) La partie donnée du 7e segment:

```
012E: 00 00 00 20 00 00 00 24 00 00 00 10 00 00 00 0F
013E: 00 01 00 00 00 08 00 00 00 09 00 00 00 00 00 00
014E: 00 00 04 00 00 00 AA AA AA AA 80 08 00 80 36 D5
015E: 55 6B 5A D4 00 40 04 2E E9 52 D2 D2 D2 8A A5 4A
016E: 00 20 02 23 E0 95 24 B4 92 8A 4A 92 54 92 D2 4A
017E: 29 2A 49 40 04 00 40
```

a) Le champ d'informations de segment de région:

```
012E: 00 00 00 20 00 00 00 24 00 00 00 10 00 00 00 0F
013E: 00
```

Cette valeur indique que la matrice de région codée par ce segment a une largeur de 32 pixels et une hauteur de 36 pixels, que son coin supérieur gauche est décalé de 16 pixels à droite du bord gauche de la page et de 15 pixels au-dessous du bord supérieur de la page. Il doit être inséré dans la page au moyen de l'opérateur OR.

b) Le champ de fanions de segment de région de dégradé:

```
013F: 01
```

Cette valeur indique que la région de dégradé est codée par la variante de codage MMR. Les structures doivent être combinées au moyen de l'opérateur OR. La valeur de pixel par défaut est **0**.

c) Le champ **HGW**:

```
0140: 00 00 00 08
```

Cette valeur indique que la séquence tabulaire de valeurs d'échelle de gris a une largeur de 8.

d) Le champ **HGH**:

```
0144: 00 00 00 09
```

Cette valeur indique que la séquence tabulaire de valeurs d'échelle de gris a une hauteur de 9.

e) Le champ **HGX**:

```
0148: 00 00 00 00
```

Cette valeur indique que le décalage horizontal de la grille de dégradé est de 0 pixel.

f) Le champ **HGY**:

```
014C: 00 00 00 00
```

Cette valeur indique que le décalage vertical de la grille de dégradé est de 0 pixel.

g) Le champ **HRX**:

```
0150: 04 00
```

Cette valeur indique que **HRX** = 1024 et que la coordonnée horizontale du vecteur de grille est donc de 1024/256 = 4 pixels.

h) Le champ **HRY**:

```
0152: 00 00
```

Cette valeur indique que la coordonnée verticale du vecteur de grille est de 0 pixel.

i) Le premier plan binaire:

```
0154: AA AA AA AA 80 08 00 80
```

La décompression de cette valeur par l'algorithme MMR donne la matrice représentée sur la Figure H.9 a). Noter que les 7 derniers bits du dernier octet sont sautés une fois que toutes les données à codage MMR ont été décodées (c'est-à-dire que le décodage du plan binaire est forcé de consommer un nombre entier d'octets).

j) Le deuxième plan binaire:

```
015C: 36 D5 55 6B 5A D4 00 40 04
```

La décompression de cette valeur par l'algorithme MMR donne la matrice représentée sur la Figure H.9 b).

k) Le troisième plan binaire:

```
0165: 2E E9 52 D2 D2 D2 8A A5 4A 00 20 02
```

La décompression de cette valeur par l'algorithme MMR donne la matrice représentée sur la Figure H.9 c).

l) Le quatrième plan binaire:

```
0171: 23 E0 95 24 B4 92 8A 4A 92 54 92 D2 4A 29 2A 49
0181: 40 04 00 40
```

La décompression de cette valeur par l'algorithme MMR donne la matrice représentée sur la Figure H.9 d).

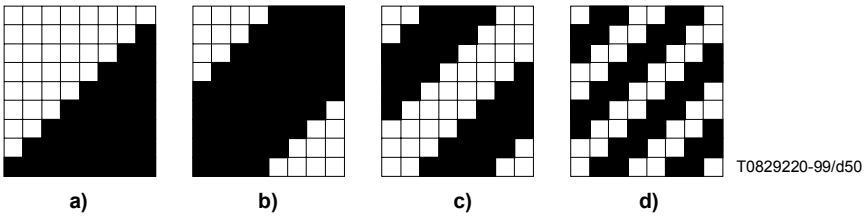


Figure H.9 – Plans binaires bruts de la région de dégradé

m) Ces plans binaires sont ensuite décodés par l'algorithme de Gray comme décrit en C.5 après combinaison par l'opérateur XOR du premier vers le second puis du plan binaire résultant vers le troisième et ainsi de suite. Les plans binaires résultats sont représentés par la Figure H.10.

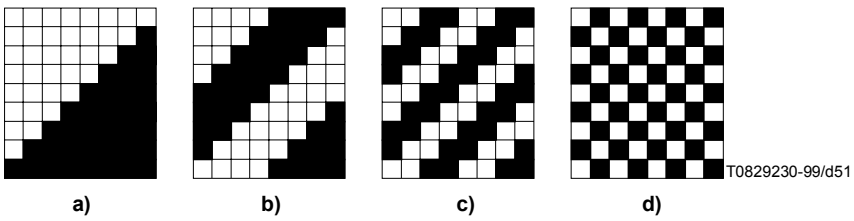


Figure H.10 – Plans binaires de la région de dégradé

n) L'empilement de ces plans binaires, le premier étant le plus significatif, produit la séquence tabulaire des valeurs suivantes:

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

o) Le vecteur de grille de dégradé et le décalage produisent la séquence tabulaire suivante de coordonnées. La combinaison de cette séquence avec celle des valeurs produit une liste d'opérations de dessin qui indique que la structure dont l'indice contenu dans le dictionnaire de structures du segment numéro 5 est la valeur dont le pixel supérieur gauche doit être placé à l'emplacement indiqué.

(0, 0)	(4, 0)	(8, 0)	(12, 0)	(16, 0)	(20, 0)	(24, 0)	(28, 0)
(0, 4)	(4, 4)	(8, 4)	(12, 4)	(16, 4)	(20, 4)	(24, 4)	(28, 4)
(0, 8)	(4, 8)	(8, 8)	(12, 8)	(16, 8)	(20, 8)	(24, 8)	(28, 8)
(0, 12)	(4, 12)	(8, 12)	(12, 12)	(16, 12)	(20, 12)	(24, 12)	(28, 12)
(0, 16)	(4, 16)	(8, 16)	(12, 16)	(16, 16)	(20, 16)	(24, 16)	(28, 16)
(0, 20)	(4, 20)	(8, 20)	(12, 20)	(16, 20)	(20, 20)	(24, 20)	(28, 20)
(0, 24)	(4, 24)	(8, 24)	(12, 24)	(16, 24)	(20, 24)	(24, 24)	(28, 24)
(0, 28)	(4, 28)	(8, 28)	(12, 28)	(16, 28)	(20, 28)	(24, 28)	(28, 28)
(0, 32)	(4, 32)	(8, 32)	(12, 32)	(16, 32)	(20, 32)	(24, 32)	(28, 32)

p) L'exécution de ces opérations de dessin par insertion produit la matrice de région de 32 × 36 pixels qui est représentée sur la Figure H.11.

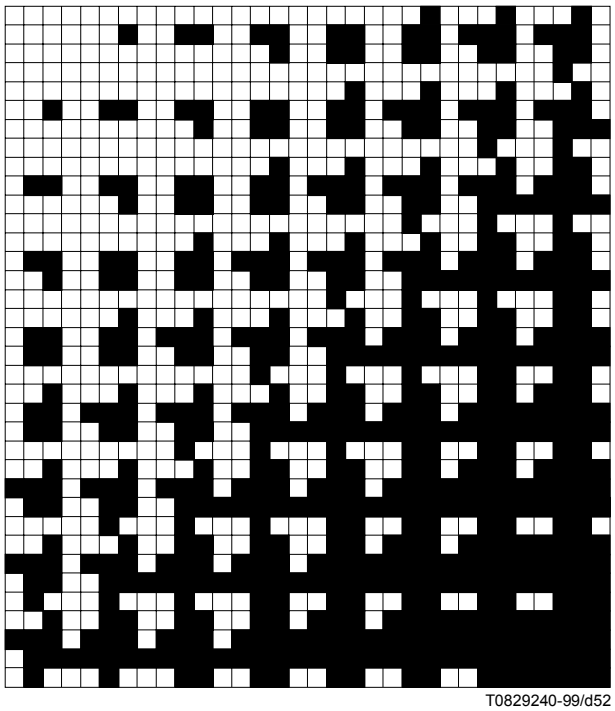


Figure H.11 – Phototrame de la région de dégradé

16) L'en-tête du 8e segment:

```
0185: 00 00 00 07 31 00 01 00 00 00 00
```

Ce segment porte le numéro 7, est du type "fin de page" (type 49), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et n'est pas retenu. Il est associé à la page 1 et ses données ont une longueur de 0 octet.

17) Les matrices de segment de région sont combinées comme suit, compte tenu de la valeur par défaut des pixels de page et de l'opérateur combinatoire du segment de chaque région. Tout d'abord, la matrice de page (large de 64 pixels et haute de 56 pixels) est remplie de **0** (valeur de pixel de page par défaut). Ensuite, la phototrame représentée sur la Figure H.5 est insérée dans la matrice de page au moyen de l'opérateur OR, avec son pixel supérieur gauche aux coordonnées (4, 1). Ensuite, la phototrame representée sur la Figure H.6 est insérée dans la matrice de page au moyen de l'opérateur OR, avec son pixel supérieur gauche aux coordonnées (4, 11). Finalement, au moyen de l'opérateur OR, la phototrame représentée sur la Figure H.11 est insérée dans la matrice de page avec son pixel supérieur gauche aux coordonnées (16, 15). Cela fait, la matrice résultante est celle qui est représentée dans la Figure H.1.

18) L'en-tête du 9e segment:

```
0190: 00 00 00 08 30 00 02 00 00 00 13
```

Ce segment porte le numéro 8, est du type "informations de page" (type 48), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait pas référence à d'autres segments et n'est pas retenu. Il est associé à la page 2 et ses données ont une longueur de 19 octets.

19) La partie donnée du 9e segment

```
019B: 00 00 00 40 00 00 00 38 00 00 00 00 00 00 00 00
01AB: 01 00 00
```

Cette valeur contient les mêmes informations que le segment numéro 1.

20) L'en-tête du 10e segment:

```
01AE: 00 00 00 09 00 01 02 00 00 00 1B
```

Ce segment porte le numéro 9, est du type "dictionnaire de symboles" (type 0), possède un champ court d'association de page et son fanion de non rétention différée n'est pas activé. Il ne fait pas référence à d'autres segments et il est retenu. Il est associé à la page 2 et ses données ont une longueur de 27 octets.

21) La partie donnée du 10e segment:

```
01B9: 08 00 02 FF 00 00 00 02 00 00 00 02 4F E7 8C 20
01C9: 0E 1D C7 CF 01 11 C4 B2 6F FF AC
```

a) Le champ de fanions du dictionnaire de symboles, de 2 octets:

```
01B9: 08 00
```

Cette valeur indique que le segment est codé au moyen de la variante de codage arithmétique et ne fait pas appel au codage de raffinement/d'agrégation. **SDTEMPLATE** a la valeur 2. Les fanions "contexte de codage de matrice utilisé" et "contexte de codage de matrice retenu" ont tous les deux la valeur **0**.

b) Le champ de fanions de gabarit adaptatif de dictionnaire de symboles:

```
01BB: 02 FF
```

Cette valeur indique que $\mathbf{SDATX_1}$ = 2 et $\mathbf{SDATY_1}$ = –1. Donc, le pixel de gabarit adaptatif A_1 est aux coordonnées (2, –1) ce qui est la valeur nominale pour le gabarit 2.

c) Le champ **SDNUMEXSYMS**, de 4 octets:

```
01BD: 00 00 00 02
```

Cette valeur indique que **SDNUMEXSYMS** = 2, c'est-à-dire que 2 symboles sont exportés par ce dictionnaire de symboles.

d) Le champ **SDNUMNEWSYMS**, de 4 octets:

```
01C1: 00 00 00 02
```

Cette valeur indique que **SDNUMNEWSYMS** = 2, c'est-à-dire que 2 symboles sont définis par ce dictionnaire de symboles.

e) Les données codées du dictionnaire de symboles:

```
01C5: 4F E7 8C 20 0E 1D C7 CF 01 11 C4 B2 6F FF AC
```

Le décodage de cette valeur donne les deux symboles qui ont été représentés sur les Figures H.4 a) et b).

22) L'en-tête du 11e segment:

```
01D4: 00 00 00 0A 07 40 00 09 02 00 00 00 1F
```

Ce segment porte le numéro 10, est du type "région alphanumérique immédiate sans pertes" (type 7), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il fait référence à deux autres segments (portant les numéros 0 et 9; segment 0 et segment 9) et ce segment ne doit pas être retenu. Il est associé à la page 2 et ses données ont une longueur de 31 octets.

23) La partie donnée du 11e segment:

```
01E1: 00 00 00 25 00 00 00 08 00 00 00 04 00 00 00 01
01F1: 00 0C 08 00 00 00 05 8D 6E 5A 12 40 85 FF AC
```

a) Le champ d'informations de segment de région:

```
01E1: 00 00 00 25 00 00 00 08 00 00 00 04 00 00 00 01
01F1: 00
```

Cette valeur indique que la matrice de région codée par ce segment a une largeur de 37 pixels, une hauteur de 8 pixels et que son coin supérieur gauche est à 4 pixels à droite du bord gauche de la page et à 1 pixel au-dessous du bord supérieur de la page. Il doit être inséré dans la page par l'opérateur OR.

b) Le champ de fanions de segment de région alphanumérique, de 2 octets:

```
01F4: 00 00 00 05
```

Cette valeur indique que le segment est codé au moyen de la variante de codage arithmétique, qu'il ne contient aucun raffinement, qu'il a une valeur **SBSTRIPS** = 4, un coin de référence BOTTOMLEFT, qu'il n'est pas transposé, qu'il combine ses symboles par OR, qu'il a une valeur de pixel par défaut de **0** et une valeur **SBDSOFFSET** = 3.

c) Le champ **SBNUMINSTANCES**, de 4 octets:

```
01F4: 00 00 00 05
```

Cette valeur indique que **SBNUMINSTANCES** = 5, c'est-à-dire que 5 instances de symbole sont codées dans cette région alphanumérique.

d) Les données codées de région alphanumérique:

```
01F8: 8D 6E 5A 12 40 85 FF AC
```

Le décodage de cette valeur suit exactement la séquence qui a été présentée lors du décodage du segment numéro 3 (segment de région alphanumérique de la page 1). Il produit la matrice de région représentée sur la Figure H.5.

24) L'en-tête du 12e segment

```
0200: 00 00 00 0B 27 00 02 00 00 00 23
```

Ce segment porte le numéro 11, est du type "région générique immédiate sans pertes" (type 39), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait pas référence à d'autres segments et n'est pas retenu. Il est associé à la page 2 et ses données ont une longueur de 35 octets.

25) La partie donnée du 12e segment:

```
020B: 00 00 00 36 00 00 00 2C 00 00 00 04 00 00 00 0B
021B: 00 08 03 FF FD FF 02 FE FE FE 04 EE ED 87 FB CB
022B: 2B FF AC
```

a) Le champ d'information de segment de région:

```
020B: 00 00 00 36 00 00 00 2C 00 00 00 04 00 00 00 0B
021B: 00
```

Cette valeur indique que la matrice de région codée par ce segment a une largeur de 54 pixels, une hauteur de 44 pixels et que son coin supérieur gauche est à 4 pixels à droite du bord gauche de la page et à 11 pixels au-dessous du bord supérieur de la page. Il doit être inséré dans la page par l'opérateur OR.

b) Le champ de fanions de région générique, de 1 octet:

```
021C: 08
```

Cette valeur indique que la région est codée au moyen du codage arithmétique, que **GBTEMPLATE** = 0 et que **TPGDON** = **1**.

c) Le champ de fanions AT de segment de région générique:

```
021D: 03 FF FD FF 02 FE FE FE
```

Ce champ a une longueur de 8 octets parce que **GBTEMPLATE** = 0 et qu'il y a donc 4 pixels AT dont les positions doivent être déterminées. Les pixels AT sont situés avec A_1 aux coordonnées (3, –1); avec A_2 à (–3, –1); avec A_3 à (2, –2); et avec A_4 à (–2, –2). Il s'agit des positions nominales de ces pixels dans ce gabarit.

d) Les données de région:

```
0225: 04 EE ED 87 FB CB 2B FF AC
```

Le décodage de ces données au moyen des valeurs décodées de **GBTEMPLATE**, de **TPGDON** et des coordonnées de pixel AT produit la matrice de région 54 × 44 représentée sur la Figure H.6.

26) L'en-tête du 13e segment:

```
022E: 00 00 00 0C 10 01 02 00 00 00 1C
```

Ce segment porte le numéro 12, est du type "dictionnaire de structures" (type 16), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et est retenu. Il est associé à la page 2 et ses données ont une longueur de 28 octets.

27) La partie donnée du 13e segment:

```
0239: 06 04 04 00 00 00 0F 90 71 6B 6D 99 A7 AA 49 7D
0249: F2 E5 48 1F DC 68 BC 6E 40 BB FF AC
```

a) Le champ de fanions de dictionnaire de symboles, de 1 octet:

```
0239: 06
```

Cette valeur indique que le segment est codé en variante arithmétique et que **HDTEMPLATE** = 3.

b) Le champ **HDPW**, de 1 octet:

```
023A: 04
```

Cette valeur indique que **HDPW** = 4.

c) Le champ **HDPH**, de 1 octet:

```
023B: 04
```

Cette valeur indique que **HDPH** = 4.

d) Le champ **GRAYMAX**, de 4 octets:

```
023C: 00 00 00 0F
```

Cette valeur indique que **GRAYMAX** = 15 et qu'il y a donc 16 structures dans ce dictionnaire de structures.

e) Les 21 octets restant dans ce segment:

```
0240: 90 71 6B 6D 99 A7 AA 49 7D F2 E5 48 1F DC 68 BC
0250: 6E 40 BB FF AC
```

Ces octets se décompriment, par la procédure de décodage à dictionnaire de structures, en la matrice collective représentée sur la Figure H.7. Les 16 structures définies par ce dictionnaire de structures sont donc comme indiqué dans la Figure H.8.

28) L'en-tête du 14e segment:

```
0255: 00 00 00 0D 17 20 0C 02 00 00 00 3E
```

Ce segment porte le numéro 13, est du type "région de dégradé immédiate sans pertes" (type 23), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à un seul autre segment: celui qui porte le numéro 12. Ni le segment 12 ni ce segment ne doivent être retenus. Il est associé à la page 2 et ses données ont une longueur de 62 octets.

29) La partie donnée du 14e segment:

```
0261: 00 00 00 20 00 00 00 24 00 00 00 10 00 00 00 0F
0271: 00 02 00 00 00 08 00 00 00 09 00 00 00 00 00 00
0281: 00 00 04 00 00 00 87 CB 82 1E 66 A4 14 EB 3C 4A
0291: 15 FA CC D6 F3 B1 6F 4C ED BF A7 BF FF AC
```

a) Le champ d'informations de segment de région:

```
0261: 00 00 00 20 00 00 00 24 00 00 00 10 00 00 00 0F
0271: 00
```

Cette valeur indique que la matrice de région codée par ce segment a une largeur de 32 pixels, une hauteur de 36 pixels et que son coin supérieur gauche est à 16 pixels à droite du bord gauche de la page et à 15 pixels au-dessous du bord supérieur de la page. Il doit être inséré dans la page par l'opérateur OR.

b) Le champ de fanions de segment de région de dégradé:

```
0272: 02
```

Cette valeur indique que la région de dégradé est codée en variante arithmétique et que **HTEMPLATE** = 1. Les structures doivent être combinées par l'opérateur OR. La valeur par défaut du pixel est **0**.

c) Les autres paramètres:

```
0273: 00 00 00 08 00 00 00 09 00 00 00 00 00 00 00 00
0283: 04 00 00 00
```

Les champs suivants indiquent que **HGW** = 8, **HGH** = 9, **HGX** = 0, **HGY** = 0, **HRX** = 1024 et **HRY** = 0.

d) Les quatre plans binaires:

```
0287: 87 CB 82 1E 66 A4 14 EB 3C 4A 15 FA CC D6 F3 B1
0297: 6F 4C ED BF A7 BF FF AC
```

Le décodage des quatre plans binaires de dimensions 8×9 à partir de ces données produit les quatre plans binaires représentés sur la Figure H.9. Comme dans le segment numéro 6, le décodage par l'algorithme de Gray de ces plans binaires, leur combinaison dans une séquence tabulaire de valeurs et l'insertion des structures issues du dictionnaire de structures conformément à cette séquence tabulaire et aux paramètres de grille de dégradé produisent la matrice de région représentée dans la Figure H.11.

30) L'en-tête du 15e segment:

```
029F: 00 00 00 0E 31 00 02 00 00 00 00
```

Ce segment porte le numéro 14, est du type "Fin de page" (type 49), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et n'est pas retenu. Il est associé à la page 2 et ses données ont une longueur de 0 octet.

31) La matrice de page est composée par combinaison des trois matrices de région selon une méthode identique à leur combinaison lors de la formation de la page 1, ce qui donne la même matrice de page.

32) L'en-tête du 16e segment:

```
02AA: 00 00 00 0F 30 00 03 00 00 00 13
```

Ce segment porte le numéro 15, est du type "Informations de page" (type 48), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et n'est pas retenu. Il est associé à la page 3 et ses données ont une longueur de 19 octets.

33) La partie donnée du 16e segment:

```
02B5: 00 00 00 25 00 00 00 08 00 00 00 00 00 00 00 00
02C5: 01 00 00
```

Cette valeur indique que la page a une largeur de 37 pixels et une hauteur de 8 pixels, que sa résolution horizontale et verticale est inconnue, qu'elle est finalement sans pertes, qu'elle ne contient pas de raffinements, que sa valeur de pixel par défaut est **0**, que son opérateur combinatoire par défaut est OR, qu'elle n'exige pas de tampons auxiliaires et qu'elle utilise l'opérateur combinatoire par défaut de la page dans toutes les régions de celle-ci.

34) L'en-tête du 17e segment:

```
02C8: 00 00 00 10 00 01 00 00 00 00 16
```

Ce segment porte le numéro 16, est du type "dictionnaire de symboles" (type 0), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et est retenu. Il n'est associé à aucune page et ses données ont une longueur de 22 octets.

35) La partie donnée du 17e segment:

```
02D3: 08 00 02 FF 00 00 00 01 00 00 00 01 4F E7 8D 68
02E3: 1B 14 2F 3F FF AC
```

a) Le champ de fanions de dictionnaire de symboles, de 2 octets:

```
02D3: 08 00
```

Cette valeur indique que le segment est codé en variante arithmétique et ne fait pas appel au codage de raffinement/d'agrégation. **SDTEMPLATE** = 2. Les fanions "utilisation du contexte de codage de matrice" et "rétention du contexte de codage de matrice" ont tous les deux la valeur **0**.

b) Le champ de fanions AT de dictionnaire de symboles:

```
02D5: 02 FF
```

Cette valeur indique que $\mathbf{SDATX_1} = 2$ et $\mathbf{SDATY_1} = -1$. Dont le pixel AT A_1 est à l'emplacement (2, –1), qui est la valeur nominale pour le gabarit 2.

c) Le champ **SDNUMEXSYMS**, de 4 octets:

```
02D7: 00 00 00 01
```

Cette valeur indique que **SDNUMEXSYMS** = 1, c'est-à-dire qu'un seul symbole est exporté par ce dictionnaire de symboles.

d) Le champ **SDNUMNEWSYMS**, de 4 octets:

```
02DB: 00 00 00 01
```

Cette valeur indique que **SDNUMNEWSYMS** = 1, c'est-à-dire qu'un seul symbole est défini par ce dictionnaire de symboles.

e) Les données codées du dictionnaire de symboles:

```
02DF: 4F E7 8D 68 1B 14 2F 3F FF AC
```

Cette séquence est décodée comme suit:

i) remettre à zéro toutes les statistiques de codage arithmétique;

ii) au moyen de la procédure de décodage arithmétique d'entier IADH, décoder une valeur de hauteur différentielle de classe de hauteurs. La valeur décodée est 6, indiquant que la première classe de hauteurs a une hauteur de 6 pixels;

iii) au moyen de la procédure de décodage arithmétique d'entier IADW, décoder une valeur de largeur différentielle. La valeur décodée est 6. Le premier symbole a donc une largeur de 6 pixels;

iv) au moyen de la procédure de décodage de région générique, avec **GBTEMPLATE** et le pixel AT A_1 réglés comme décrit dans l'en-tête de données du dictionnaire de symboles, décoder une matrice 6×6. Cela produit la matrice représentée dans la Figure H.12 a);

v) au moyen de la procédure de décodage arithmétique d'entier IADW, décoder une valeur de largeur différentielle. La valeur décodée est OOB, indiquant la fin de la classe de hauteurs;

vi) comme **SDNUMNEWSYMS** = 1, le dernier symbole a donc été décodé;

vii) au moyen de la procédure de décodage arithmétique d'entier IAEX, décoder une longueur de séquence d'exportation. La valeur décodée est 0, indiquant que les premiers symboles 0 ne sont pas exportés;

viii) au moyen de la procédure de décodage arithmétique d'entier IAEX, décoder une longueur de séquence d'exportation. La valeur décodée est 1, indiquant que les premiers symboles 1 sont exportés. Donc ce dictionnaire de symboles définit un seul symbole, qui est exporté.

Figure H.12 – Symboles contenus dans les dictionnaires de symboles de la troisième page

36) L'en-tête du 18e segment:

```
02E9: 00 00 00 11 00 21 10 03 00 00 00 20
```

Ce segment porte le numéro 17, est du type "dictionnaire de symboles" (type 0), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il fait référence à un seul autre segment, portant le numéro 16. Le segment 16 n'est pas retenu mais ce segment est retenu. Il est associé à la page 3 et ses données ont une longueur de 32 octets.

37) La partie donnée du 18e segment:

```
02F5: 08 02 02 FF FF FF FF FF 00 00 00 03 00 00 00 02
0305: 4F E9 D7 D5 90 C3 B5 26 A7 FB 6D 14 98 3F FF AC
```

a) Le champ de fanions du dictionnaire de symboles, de 2 octets:

```
02F5: 08 02
```

Cette valeur indique que le segment est codé par la variante arithmétique et qu'il utilise le codage de raffinement/d'agrégation. **SDTEMPLATE** = 2. **SDRTEMPLATE** = 0. Les fanions "utilisation de contexte de codage de matrice" et "rétention de contexte de codage de matrice" sont tous les deux égaux à **0**.

b) Le champ de fanions AT du dictionnaire de symboles:

```
02F7: 02 FF
```

Cette valeur indique que $\mathbf{SDATX_1} = 2$ et $\mathbf{SDATY_1} = -1$. Donc le pixel AT A_1 est à l'emplacement (2, –1), ce qui est la valeur nominale pour le gabarit 2.

c) Les fanions AT de raffinement de dictionnaire de symboles:

```
02F9: FF FF FF FF
```

Cette valeur indique que $\mathbf{SDRATX_1} = -1$, $\mathbf{SDRATY_1} = -1$, $\mathbf{SDRATX_2} = -1$ et $\mathbf{SDRATY_2} = -1$. Le pixel AT RA_1 est donc à l'emplacement (–1, –1) et le pixel AT RA_2 est à l'emplacement (–1, –1), qui sont les emplacements nominaux pour le gabarit de raffinement 0.

d) Le champ **SDNUMEXSYMS**, de 4 octets:

```
02FD: 00 00 00 03
```

Cette valeur indique que **SDNUMEXSYMS** = 3, c'est-à-dire que 3 symboles sont exportés par ce dictionnaire de symboles.

e) Le champ **SDNUMNEWSYMS**, de 4 octets:

```
0301: 00 00 00 02
```

Ce champ indique que **SDNUMNEWSYMS** = 2, c'est-à-dire que 2 symboles sont définis par ce dictionnaire de symboles.

f) Les données codées du dictionnaire de symboles:

```
0305: 4F E9 D7 D5 90 C3 B5 26 A7 FB 6D 14 98 3F FF AC
```

Cette séquence est décodée comme suit:

i) remettre les statistiques de codage arithmétique à zéro;

ii) au moyen de la procédure de décodage arithmétique d'entier IADH, décoder une valeur de hauteur différentielle de classe de hauteurs. La valeur décodée est 6, indiquant que la première classe de hauteurs a une hauteur de 6 pixels;

iii) au moyen de la procédure de décodage arithmétique d'entier IADW, décoder une valeur de largeur différentielle. La valeur décodée est 6, indiquant que le premier symbole a une largeur de 6 pixels;

iv) au moyen de la procédure de décodage arithmétique d'entier IAAI, décoder un certain nombre d'instances de symbole dans l'agrégation qui forme le premier symbole. La valeur décodée est 1;

v) au moyen de la procédure de décodage arithmétique d'entier IAID, décoder un identificateur de symbole. La valeur décodée est 0. Le premier symbole est donc celui qui est identifié par l'identificateur de symbole 0, qui est le premier (et le seul) symbole dans le dictionnaire (segment 16) référencé par ce segment;

vi) au moyen de la procédure de décodage arithmétique d'entier IARDX, décoder une valeur de décalage horizontal de raffinement d'instance de symbole. La valeur décodée est 0;

vii) au moyen de la procédure de décodage arithmétique d'entier IARDY, décoder une valeur de décalage vertical de raffinement d'instance de symbole. La valeur décodée est 0. Le raffinement est donc effectué avec alignement du symbole raffiné sur le symbole de référence (c'est-à-dire que **GRREFERENCEDX** et **GRREFERENCEDY** ont chacun la valeur 0);

viii) au moyen de la procédure de décodage de région générique par raffinement, décoder la matrice raffinée d'instance de symbole. La matrice de référence est celle qui est représentée sur la Figure H.12 a) et la matrice raffinée, qui est celle du premier symbole défini dans ce dictionnaire de symboles, est représentée sur la Figure H.12 b).

ix) au moyen de la procédure de décodage arithmétique d'entier IADW, décoder une valeur de largeur différentielle. La valeur décodée est 8. Le deuxième symbole a donc une largeur de 14 pixels;

x) au moyen de la procédure de décodage arithmétique d'entier IAAI, décoder un certain nombre d'instances de symbole dans l'agrégation qui forme le deuxième symbole. La valeur décodée est 2;

xi) au moyen de la procédure de décodage de région alphanumérique, avec les paramètres réglés comme décrit au 6.5.8.2, décoder une région alphanumérique de 14 × 6 pixels:

- au moyen de la procédure de décodage arithmétique d'entier IADT, décoder la valeur initiale STRIPT. La valeur décodée est 0;
- au moyen de la procédure de décodage arithmétique d'entier IADT, décoder une valeur delta T. La valeur décodée est 0;
- au moyen de la procédure de décodage arithmétique d'entier IAFS, décoder une première valeur S. La valeur décodée est 0. Le coin de référence (supérieur gauche dans ce cas) de la première instance de symbole dans l'agrégation est donc au point (0,0);
- au moyen de la procédure de décodage arithmétique d'entier IAID, décoder un identificateur de symbole. La valeur décodée est 0. Le premier symbole est donc celui qui est identifié par l'identificateur de symbole 0, qui désigne le premier (et seul) symbole dans le dictionnaire (segment 16) référencé par ce segment;
- au moyen de la procédure de décodage arithmétique d'entier IARI, décoder un fanion de raffinement. La valeur décodée est **0**, indiquant que la première instance de symbole n'est pas raffinée;
- a ce point, CURS = 5. Au moyen de la procédure de décodage arithmétique d'entier IADS, décoder une valeur delta S. La valeur décodée est 3. Le coin de référence de la deuxième instance de symbole dans l'agrégation est donc à (8, 0);
- au moyen de la procédure de décodage arithmétique d'entier IAID, décoder un identificateur de symbole. La valeur décodée est 1. Le deuxième symbole est donc celui qui est identifié par l'identificateur de symbole 1, qui désigne le premier (et jusqu'ici le seul) symbole défini dans ce dictionnaire de symboles;
- au moyen de la procédure de décodage arithmétique d'entier IARI, décoder un fanion de raffinement. La valeur décodée est **0**, indiquant que la première instance de symbole n'est pas raffinée;
- au moyen de la procédure de décodage arithmétique d'entier IADS, décoder une valeur delta S. La valeur décodée est OOB, indiquant la fin de la bande;
- le décodage de la région alphanumérique est donc complet. La matrice de région alphanumérique, qui est celle du deuxième symbole défini dans le dictionnaire de symboles, est représentée dans la Figure H.12 c). Elle est obtenue par insertion de la Figure H.12 a) au point (0, 0) et de la Figure H.12 b) au point (8, 0);

xii) au moyen de la procédure de décodage arithmétique d'entier IADW, décoder une valeur de largeur différentielle. La valeur décodée est OOB, indiquant la fin de la classe de hauteurs;

xiii) Comme **SDNUMNEWSYMS** = 2, le dernier symbole a maintenant été décodé;

xiv) au moyen de la procédure de décodage arithmétique d'entier IAEX, décoder une longueur de séquence d'exportation. La valeur décodée est 0, indiquant que les premiers symboles 0 ne sont pas exportés;

xv) au moyen de la procédure de décodage arithmétique d'entier IAEX, décoder une longueur de séquence d'exportation. La valeur décodée est 3, indiquant que les 3 prochains symboles sont exportés. Ce dictionnaire de symboles importe donc un seul symbole et définit deux symboles, les trois étant exportés;

Noter que le symbole issu du segment 16 est réexporté par ce dictionnaire. Les trois symboles exportés par ce dictionnaire sont donc les trois symboles représentés dans la Figure H.12. Une région alphanumérique peut donc utiliser le symbole représenté sur la Figure H.12 a) (initialement défini dans le segment 16) en faisant référence au segment 17, bien que le segment 16 ne soit plus retenu une fois passé la fin du segment 17.

38) L'en-tête du 19e segment:

```
0315: 00 00 00 12 07 20 11 03 00 00 00 25
```

Ce segment porte le numéro 18, est du type "région alphanumérique immédiate sans pertes" (type 7), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il fait référence à un seul autre segment, le segment 17. Celui-ci et ce segment ne doivent pas être retenus. Il est associé à la page 3 et ses données ont une longueur de 37 octets.

39) La partie donnée du 19e segment:

```
0321: 00 00 00 25 00 00 00 08 00 00 00 00 00 00 00 00
0331: 00 8C 12 00 00 00 04 A9 5C 8B F4 C3 7D 96 6A 28
0341: E5 76 8F FF AC
```

a) Le champ d'informations de segment de région:

```
0321: 00 00 00 25 00 00 00 08 00 00 00 00 00 00 00 00
0331: 00
```

Cette valeur indique que la matrice de région codée par ce segment a une largeur de 37 pixels, une hauteur de 8 pixels et que son coin supérieur gauche est à 0 pixel à droite du bord gauche de la page et à 0 pixel au-dessous du bord supérieur de la page. Il doit être inséré dans la page par l'opérateur OR.

b) Le champ de fanions de segment de région alphanumérique, de 2 octets:

```
0332: 8C 12
```

Cette valeur indique que le segment est codé en variante de codage arithmétique, qu'il contient des raffinements, qu'il a une valeur **SBSTRIPS** = 1, qu'il a un coin de référence de type TOPLEFT, qu'il n'est pas transposé, qu'il combine ses symboles au moyen de l'opérateur OR, qu'il a une valeur par défaut de pixel de **0**, qu'il a une valeur **SBDOFFSET** = 3 et une valeur **SBRTEMPLATE** = 1.

c) Le champ **SBNUMINSTANCES**; de 4 octets:

```
0334: 00 00 00 04
```

Cette valeur indique que **SBNUMINSTANCES** = 4, c'est-à-dire que quatre instances de symbole sont codées dans cette région alphanumérique.

d) Les données codées de région alphanumérique:

```
0338: A9 5C 8B F4 C3 7D 96 6A 28 E5 76 8F FF AC
```

Ces données sont décodées comme suit:

i) remettre à zéro toutes les statistiques de codage arithmétique;

ii) au moyen de la procédure de décodage arithmétique d'entier IADT, décoder la valeur STRIPT initiale. La valeur décodée est 0;

iii) au moyen de la procédure de décodage arithmétique d'entier IADT, décoder une valeur delta T. La valeur décodée est 0;

iv) au moyen de la procédure de décodage arithmétique d'entier IAFS, décoder une première valeur S. La valeur décodée est 0. Le coin de référence (supérieur gauche dans ce cas) de la première instance de symbole dans la région alphanumérique est donc au point (0, 0);

v) au moyen de la procédure de décodage arithmétique d'entier IAID, décoder un identificateur de symbole. La valeur décodée est 1. La première instance de symbole utilise donc le symbole indiqué dans la Figure H.12 b);

vi) au moyen de la procédure de décodage arithmétique d'entier IARI, décoder un fanion de raffinement. La valeur décodée est **0**, indiquant que la première instance de symbole n'est pas raffinée;

vii) a ce point CURS = 5. Au moyen de la procédure de décodage arithmétique d'entier IADS, décoder une valeur delta S. La valeur décodée est 0. Après addition dans **SBDSOFFSET**, le coin de référence de la deuxième instance de symbole dans l'agrégation est donc à (8, 0);

viii) au moyen de la procédure de décodage arithmétique d'entier IAID, décoder un identificateur de symbole. La valeur décodée est 0. La deuxième instance de symbole utilise donc le symbole indiqué dans la Figure H.12 a);

ix) au moyen de la procédure de décodage arithmétique d'entier IARI, décoder un fanion de raffinement. La valeur décodée est **0**, indiquant que la première instance de symbole n'est pas raffinée;

x) a ce point, CURS = 13. Au moyen de la procédure de décodage arithmétique d'entier IADS, décoder une valeur delta S. La valeur décodée est 0. Après addition dans **SBDSOFFSET**, le coin de référence de la troisième instance de symbole dans l'agrégation est donc au point (16, 0);

xi) au moyen de la procédure de décodage arithmétique d'entier IAID, décoder un identificateur de symbole. La valeur décodée est 1. La troisième instance de symbole utilise donc le symbole représenté dans la Figure H.12 b);

xii) au moyen de la procédure de décodage arithmétique d'entier IARI, décoder un fanion de raffinement. La valeur décodée est **1**, indiquant que la troisième instance de symbole est raffinée;

xiii) au moyen de la procédure de décodage arithmétique d'entier IARDW, décoder une valeur de largeur différentielle pour le raffinement d'instance de symbole. La valeur décodée est –1;

xiv) au moyen de la procédure de décodage arithmétique d'entier IARDH, décoder une valeur de hauteur différentielle pour le raffinement d'instance de symbole. La valeur décodée est 2. Comme le symbole de référence mesure 6×6 pixels, l'instance de symbole raffinée mesure donc 5×8 pixels;

xv) au moyen de la procédure de décodage arithmétique d'entier IARDX, décoder une valeur delta X pour le raffinement d'instance de symbole. La valeur décodée est 1;

xvi) au moyen de la procédure de décodage arithmétique d'entier IARDY, décoder une valeur delta Y pour le raffinement d'instance de symbole. La valeur décodée est –2. **GRREFERENCEDX** et **GRREFERENCEDY** sont donc réglés comme suit:

$$\mathbf{GRREFERENCEDX} = \lfloor -1/2 \rfloor + 1 = -1 + 1 = 0$$

$$\mathbf{GRREFERENCEDY} = \lfloor 2/2 \rfloor - 2 = 1 - 2 = -1$$

de sorte que le raffinement soit effectué avec le symbole raffiné à un emplacement tel que le pixel supérieur gauche du symbole raffiné soit aligné sur le pixel (0, 1) du symbole de référence;

xvii) au moyen de la procédure de décodage de région générique par raffinement, décoder la matrice raffinée d'instance de symbole. La matrice de référence est celle qui est indiquée dans la Figure H.12 b) et la matrice raffinée, qui est celle de la troisième instance de symbole, est représentée sur la Figure H.2;

xviii) a ce point, CURS = 18. Au moyen de la procédure de décodage arithmétique d'entier IADS, décoder une valeur delta S. La valeur décodée est 0. Après addition dans **SBDSOFFSET**, le coin de référence de la troisième instance de symbole dans l'agrégation est donc au point (23, 0).

xix) au moyen de la procédure de décodage arithmétique d'entier IAID, décoder un identificateur de symbole. La valeur décodée est 2. La quatrième instance de symbole utilise donc le symbole représenté dans la Figure H.12 c).

xx) au moyen de la procédure de décodage arithmétique d'entier IADS, décoder une valeur delta S. La valeur décodée est OOB, indiquant la fin de la bande;

xxi) le décodage de la région alphanumérique est maintenant complet. La matrice de région alphanumérique est celle qui est représentée sur la Figure H.5. Elle est obtenue par insertion de la Figure H.12 b) avec son coin supérieur gauche à (0,0), de la Figure H.12 a) avec son coin supérieur gauche à (8, 0), de la Figure H.2 avec son coin supérieur gauche à (16, 0), et de la Figure H.12 c) avec son coin supérieur gauche à (23, 0).

40) L'en-tête du 20e segment:

```
0346: 00 00 00 13 31 00 03 00 00 00 00
```

Ce segment porte le numéro 19, est du type "fin de page" (type 49), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et n'est pas retenu. Il est associé à la page 3 et ses données ont une longueur de 0 octet.

41) La matrice de page est formée à partir du seul segment de région, le segment numéro 18. Elle est égale à la matrice de ce segment de région (Figure H.5).

42) L'en-tête du 21e segment:

```
0351: 00 00 00 14 33 00 00 00 00 00 00
```

Ce segment porte le numéro 20, est du type "fin de fichier" (type 51), possède un champ court d'association de page et son fanion de non-rétention différée n'est pas activé. Il ne fait référence à aucun autre segment et n'est associé à aucune page. Ses données ont une longueur de 0 octet.

H.2 Séquence d'essai pour codeur arithmétique

Dans ce paragraphe, un petit ensemble de données est fourni afin de contrôler le codeur et le décodeur arithmétiques. L'essai est structuré de façon à contrôler un grand nombre des trajets du codeur et du décodeur, mais il est impossible de les vérifier tous au cours d'une brève séquence d'essai, de sorte que la concordance avec les résultats de cet essai ne garantit malheureusement pas une réalisation absolument correcte.

Les décisions à coder, insérées dans 32 octets et représentées en notation hexadécimale, sont les suivantes:

```
00 02 00 51 00 00 00 C0 03 52 87 2A AA AA AA AA
82 C0 20 00 FC D7 9E F6 BF 7F ED 90 4F 46 A3 BF
```

Les données codées qui doivent être obtenues par codage de cette séquence, représentées sous la forme de 30 octets hexadécimaux, sont les suivantes:

```
84 C7 3B FC E1 A1 43 04 02 20 00 00 41 0D BB 86
F4 31 7F FF 88 FF 37 47 1A DB 6A DF FF AC
```

Le décodage de ces 30 octets doit produire les 32 octets initiaux.

Le Tableau H.1 présente une liste bit par bit du fonctionnement du codeur et du décodeur arithmétiques. La première ligne de ce tableau correspond aux opérations INITENC et INITDEC. La valeur de l'octet qui précède le premier octet dans le tampon de sortie est censée être 0x00, produisant la valeur initiale de l'octet B 0x00. La dernière ligne du tableau correspond à l'opération FLUSH.

Pour l'ensemble de cet essai, une seule valeur de CX est utilisée. Le terme I(CX) a la valeur initiale 0 et le terme MPS(CX) a la valeur initiale **0**.

La première colonne est le compteur d'événements EC. La deuxième colonne est la décision D à coder. Les troisième et quatrième colonnes indiquent les valeurs de I(CX) et de MPS(CX). La cinquième colonne contenant un "X" indique que l'échange conditionnel se produira lors du codage (ou du décodage) de la décision actuelle. La sixième colonne montre la valeur Qe actuelle correspondant à I(CX) (voir Tableau E.1). La septième colonne montre la valeur du registre A avant le codage (décodage) de la décision. Noter que le registre A est toujours supérieur ou égal à `0x8000`.

Les variables jusqu'à ce point étaient communes au codeur et au décodeur. Les quatre colonnes suivantes (C, CT, B, OUT) ne concernent que le codeur. Les quatre colonnes suivantes (C, CT, B, IN) ne concernent que le décodeur. La colonne finale (C) montre le registre C pour le décodeur de conventions logicielles car c'est la seule valeur qui diffère pour le décodeur de conventions logicielles. Toutes les valeurs indiquées pour les registres C sont données avant le codage (décodage) de la décision actuelle.

Pour le codeur, CT est un compteur indiquant le moment où un octet est prêt à être extrait d'un registre C. La colonne sous B montre l'octet de la variable B qui est en attente d'extraction. Cet octet peut parfois changer en raison d'un report. Finalement, les octets comprimés du codeur sont énumérés dans la colonne OUT. Un octet est considéré comme "extrait" lorsque le pointeur de données comprimées BP est arrivé à pointer au-delà de cet octet.

Le compteur CT du décodeur indique le moment où l'octet suivant des données comprimées est prêt à être introduit. La colonne sous B montre la valeur du registre B, qui est utilisée pour déterminer le moment où un bourrage de bits s'est produit. La colonne sous IN montre les octets qui sont consommés. Un octet est considéré comme "consommé" lorsque le pointeur de données comprimées BP est arrivé à pointer sur cet octet. Noter que l'octet `0xAC` final n'est jamais consommé, conformément à cette définition, bien qu'il soit lu dans le cadre de la procédure BYTEIN.

Tableau H.1 – Données de trace de codeur et de décodeur

Variables communes							Codeur				Décodeur				Décodeur de convention logicielle
EC	D	I	MPS	CE	Qe hex	A hex	C hex	CT	B hex	OUT hex	C hex	CT	B hex	IN hex	C hex
0									00		00000000			84 C7	00000000
1	0	0	0	X	5601	8000	00000000	12	00		42638000	1	C7		3D9C0000
2	0	1	0		3401	AC02	00000000	11	00		84C70000	0	C7	3B	273A0000
3	0	2	0		1801	F002	00006802	10	00		A18C7600	7	3B		4E758800
4	0	2	0		1801	D801	00008003	10	00		898B7600	7	3B		4E758800
5	0	2	0		1801	C000	00009804	10	00		718A7600	7	3B		4E758800
6	0	2	0		1801	A7FF	0000B005	10	00		59897600	7	3B		4E758800
7	0	2	0		1801	8FFE	0000C806	10	00		41887600	7	3B		4E758800
8	0	3	0		0AC1	EFFA	0001C00E	9	00		530EEC00	6	3B		9CEB1000
9	0	3	0		0AC1	E539	0001CACF	9	00		484DEC00	6	3B		9CEB1000
10	0	3	0		0AC1	DA78	0001D590	9	00		3D8CEC00	6	3B		9CEB1000
11	0	3	0		0AC1	CFB7	0001E051	9	00		32CBEC00	6	3B		9CEB1000
12	0	3	0		0AC1	C4F6	0001EB12	9	00		280AEC00	6	3B		9CEB1000
13	0	3	0		0AC1	BA35	0001F5D3	9	00		1D49EC00	6	3B		9CEB1000
14	0	3	0		0AC1	AF74	00020094	9	00		1288EC00	6	3B		9CEB1000
15	1	3	0		0AC1	A4B3	00020B55	9	00		07C7EC00	6	3B		9CEB1000
16	0	12	0		1C01	AC10	0020B550	5	00		7C7EC000	2	3B		2F910000
17	0	12	0		1C01	900F	0020D151	5	00		607DC000	2	3B		2F910000
18	0	13	0		1601	E81C	0041DAA4	4	00		88F98000	1	3B		5F220000
19	0	13	0		1601	D21B	0041F0A5	4	00		72F88000	1	3B		5F220000
20	0	13	0		1601	BC1A	004206A6	4	00		5CF78000	1	3B		5F220000
21	0	13	0		1601	A619	00421CA7	4	00		46F68000	1	3B		5F220000
22	0	13	0		1601	9018	004232A8	4	00		30F58000	1	3B		5F220000
23	0	29	0		1101	F42E	00849152	3	00		35E90000	0	3B		BE440000
24	0	29	0		1101	E32D	0084A253	3	00		24E80000	0	3B		BE440000
25	0	29	0		1101	D22C	0084B354	3	00		13E70000	0	3B		BE440000
26	1	29	0		1101	C12B	0084C455	3	84		02E60000	0	3B	FC	BE440000
27	0	27	0		1401	8808	000622A8	8	84		1737E000	5	FC		70D01800
28	1	28	0		1201	E80E	000C6D52	7	84		066DC000	4	FC		E1A03000
29	0	26	0		1601	9008	00636A90	4	84		336E0000	1	FC		5C998000
30	0	27	0		1401	F40E	00C70122	3	84		3ADA0000	0	FC		B9330000
31	0	27	0		1401	E00D	00C71523	3	84		26D90000	0	FC		B9330000
32	1	27	0		1401	CC0C	00C72924	3	C7	84	12D80000	0	FC	E1	B9330000
33	0	25	0		1801	A008	00014920	8	C7		96C70800	5	E1		0940F000
34	0	25	0		1801	8807	00016121	8	C7		7EC60800	5	E1		0940F000
35	0	26	0		1601	E00C	0002F244	7	C7		CD8A1000	4	E1		1281E000
36	0	26	0		1601	CA0B	00030845	7	C7		B7891000	4	E1		1281E000
37	0	26	0		1601	B40A	00031E46	7	C7		A1881000	4	E1		1281E000
38	0	26	0		1601	9E09	00033447	7	C7		8B871000	4	E1		1281E000
39	0	26	0		1601	8808	00034A48	7	C7		75861000	4	E1		1281E000
40	0	27	0		1401	E40E	0006C092	6	C7		BF0A2000	3	E1		2503C000
41	0	27	0		1401	D00D	0006D493	6	C7		AB092000	3	E1		2503C000
42	0	27	0		1401	BC0C	0006E894	6	C7		97082000	3	E1		2503C000
43	0	27	0		1401	A80B	0006FC95	6	C7		83072000	3	E1		2503C000
44	0	27	0		1401	940A	00071096	6	C7		6F062000	3	E1		2503C000
45	0	27	0		1401	8009	00072497	6	C7		5B052000	3	E1		2503C000
46	0	28	0		1201	D810	000E7130	5	C7		8E084000	2	E1		4A078000
47	0	28	0		1201	C60F	000E8331	5	C7		7C074000	2	E1		4A078000
48	0	28	0		1201	B40E	000E9532	5	C7		6A064000	2	E1		4A078000
49	0	28	0		1201	A20D	000EA733	5	C7		58054000	2	E1		4A078000
50	0	28	0		1201	900C	000EB934	5	C7		46044000	2	E1		4A078000
51	0	29	0		1101	FC16	001D966A	4	C7		68068000	1	E1		940F0000
52	0	29	0		1101	EB15	001DA76B	4	C7		57058000	1	E1		940F0000
53	0	29	0		1101	DA14	001DB86C	4	C7		46048000	1	E1		940F0000
54	0	29	0		1101	C913	001DC96D	4	C7		35038000	1	E1		940F0000
55	0	29	0		1101	B812	001DDA6E	4	C7		24028000	1	E1		940F0000
56	0	29	0		1101	A711	001DEB6F	4	C7		13018000	1	E1		940F0000
57	1	29	0		1101	9610	001DFC70	4	C7		02008000	1	E1	A1	940F0000
58	1	27	0		1401	8808	00EFE380	1	3B	C7	10068400	6	A1		78017800
59	0	25	0		1801	A008	001F1C00	6	3B		80342000	3	A1		1FD3C000
60	0	25	0		1801	8807	001F3401	6	3B		68332000	3	A1		1FD3C000
61	0	26	0		1601	E00C	003E9804	5	3B		A0644000	2	A1		3FA78000
62	0	26	0		1601	CA0B	003EAE05	5	3B		8A634000	2	A1		3FA78000
63	0	26	0		1601	B40A	003EC406	5	3B		74624000	2	A1		3FA78000
64	0	26	0		1601	9E09	003EDA07	5	3B		5E614000	2	A1		3FA78000

Tableau H.1 – Données de trace de codeur et de décodeur *(suite)*

Variables communes							Codeur				Décodeur				Décodeur de convention logicielle
EC	D	I	MPS	CE	Qe hex	A hex	C hex	CT	B hex	OUT hex	C hex	CT	B hex	IN hex	C hex
65	0	26	0		1601	8808	003EF008	5	3B		48604000	2	A1		3FA78000
66	0	27	0		1401	E40E	007E0C12	4	3B		64BE8000	1	A1		7F4F0000
67	0	27	0		1401	D00D	007E2013	4	3B		50BD8000	1	A1		7F4F0000
68	0	27	0		1401	BC0C	007E3414	4	3B		3CBC8000	1	A1		7F4F0000
69	0	27	0		1401	A80B	007E4815	4	3B		28BB8000	1	A1		7F4F0000
70	0	27	0		1401	940A	007E5C16	4	3B		14BA8000	1	A1		7F4F0000
71	1	27	0		1401	8009	007E7017	4	3B		00B98000	1	A1	43	7F4F0000
72	1	25	0		1801	A008	03F380B8	1	FC	3B	05CD0C00	6	43		9A3AF000
73	0	23	0		2201	C008	001C05C0	6	FC		2E686000	3	43		919F8000
74	1	23	0		2201	9E07	001C27C1	6	FC		0C676000	3	43		919F8000
75	0	21	0		2801	8804	00709F04	4	FC		319D8000	1	43		56660000
76	1	22	0		2401	C006	00E18E0A	3	FC		13390000	0	43	04	ACCC0000
77	0	20	0		3001	9004	03863828	1	E1	FC	4CE41000	6	04		431FEC00
78	0	21	0		2801	C006	0004D052	8	E1		39C62000	5	04		863FD800
79	1	21	0		2801	9805	0004F853	8	E1		11C52000	5	04		863FD800
80	0	19	0		3401	A004	0013E14C	6	E1		47148000	3	04		58EF6000
81	1	20	0		3001	D806	00282A9A	5	E1		26270000	2	04		B1DEC000
82	0	19	0		3401	C004	00A0AA68	3	E1		989C0000	0	04		27670000
83	0	19	0		3401	8C03	00A0DE69	3	E1		649B0000	0	04	02	27670000
84	0	20	0		3001	B004	014224D4	2	E1		61340400	7	02		4ECFFA00
85	0	20	0		3001	8003	014254D5	2	E1		31330400	7	02		4ECFFA00
86	1	21	0		2801	A004	028509AC	1	A1	E1	02640800	6	02		9D9FF400
87	1	19	0		3401	A004	000426B0	7	A1		09902000	4	02		9673D000
88	1	18	0		3801	D004	00109AC0	5	A1		26408000	2	02		A9C34000
89	0	17	0		4801	E004	00426B00	3	A1		99020000	0	02		47010000
90	0	17	0		4801	9803	0042B301	3	A1		51010000	0	02	20	47010000
91	1	18	0		3801	A004	0085F604	2	42	A1	12004000	7	20		8E03BE00
92	0	17	0		4801	E004	0007D810	8	42		48010000	5	20		9802F800
93	1	17	0		4801	9803	00082011	8	42		00000000	5	20		9802F800
94	0	16	0	X	5101	9002	00104022	7	42		00000000	4	20		9001F000
95	1	17	0		4801	A202	00208044	6	42		00000000	3	20		A201E000
96	0	16	0	X	5101	9002	00410088	5	42		00000000	2	20		9001C000
97	1	17	0		4801	A202	00820110	4	42		00000000	1	20		A2018000
98	0	16	0	X	5101	9002	01040220	3	42		00000000	0	20	00	90010000
99	1	17	0		4801	A202	02080440	2	42		00000000	7	00		A201FE00
100	0	16	0	X	5101	9002	04100880	1	04	43	00000000	6	00		9001FC00
101	1	17	0		4801	A202	00001100	8	04		00000000	5	00		A201F800
102	0	16	0	X	5101	9002	00002200	7	04		00000000	4	00		9001F000
103	1	17	0		4801	A202	00004400	6	04		00000000	3	00		A201E000
104	0	16	0	X	5101	9002	00008800	5	04		00000000	2	00		9001C000
105	1	17	0		4801	A202	00011000	4	04		00000000	1	00		A2018000
106	0	16	0	X	5101	9002	00022000	3	04		00000000	0	00		90010000
107	1	17	0		4801	A202	00044000	2	04		00000000	7	00		A201FE00
108	0	16	0	X	5101	9002	00088000	1	02	04	00000000	6	00		9001FC00
109	1	17	0		4801	A202	00010000	8	02		00000000	5	00		A201F800
110	0	16	0	X	5101	9002	00020000	7	02		00000000	4	00		9001F000
111	1	17	0		4801	A202	00040000	6	02		00000000	3	00		A201E000
112	0	16	0	X	5101	9002	00080000	5	02		00000000	2	00		9001C000
113	1	17	0		4801	A202	00100000	4	02		00000000	1	00		A2018000
114	0	16	0	X	5101	9002	00200000	3	02		00000000	0	00	41	90010000
115	1	17	0		4801	A202	00400000	2	02		00008200	7	41		A2017C00
116	0	16	0	X	5101	9002	00800000	1	20	02	00010400	6	41		9000F800
117	1	17	0		4801	A202	00000000	8	20		00020800	5	41		A1FFF000
118	0	16	0	X	5101	9002	00000000	7	20		00041000	4	41		8FFDE000
119	1	17	0		4801	A202	00000000	6	20		00082000	3	41		A1F9C000
120	0	16	0	X	5101	9002	00000000	5	20		00104000	2	41		8FF18000
121	1	17	0		4801	A202	00000000	4	20		00208000	1	41		A1E10000
122	0	16	0	X	5101	9002	00000000	3	20		00410000	0	41	0D	8FC00000
123	1	17	0		4801	A202	00000000	2	20		00821A00	7	0D		A17FE400
124	0	16	0	X	5101	9002	00000000	1	00	20	01043400	6	0D		8EFDC800
125	1	17	0		4801	A202	00000000	8	00		02086800	5	0D		9FF99000
126	0	16	0	X	5101	9002	00000000	7	00		0410D000	4	0D		8BF12000
127	1	17	0		4801	A202	00000000	6	00		0821A000	3	0D		99E04000
128	0	16	0	X	5101	9002	00000000	5	00		10434000	2	0D		7FBE8000

Tableau H.1 – Données de trace de codeur et de décodeur *(suite)*

Variables communes							Codeur				Décodeur				Décodeur de convention logicielle
EC	D	I	MPS	CE	Qe hex	A hex	C hex	CT	B hex	OUT hex	C hex	CT	B hex	IN hex	C hex
129	1	17	0		4801	A202	00000000	4	00		20868000	1	0D		817B0000
130	0	16	0	X	5101	9002	00000000	3	00		410D0000	0	0D	BB	4EF40000
131	0	17	0		4801	A202	00000000	2	00		821B7600	7	BB		1FE68800
132	0	18	0		3801	B402	00009002	1	00	00	7434EC00	6	BB		3FCD1000
133	0	19	0		3401	F802	00019006	8	00		7867D800	5	BB		7F9A2000
134	0	19	0		3401	C401	0001C407	8	00		4466D800	5	BB		7F9A2000
135	1	19	0		3401	9000	0001F808	8	00		1065D800	5	BB		7F9A2000
136	0	18	0		3801	D004	0007E020	6	00		41976000	3	BB		8E6C8000
137	1	18	0		3801	9803	00081821	6	00		09966000	3	BB		8E6C8000
138	1	17	0		4801	E004	00206084	4	00		26598000	1	BB		B9AA0000
139	0	16	0	X	5101	9002	0040C108	3	00		4CB30000	0	BB	86	434E0000
140	0	17	0		4801	A202	00818210	2	00		99670C00	7	86		089AF200
141	0	18	0		3801	B402	01039422	1	40	00	A2CC1800	6	86		1135E400
142	0	19	0		3401	F802	00079846	8	40		D5963000	5	86		226BC800
143	0	19	0		3401	C401	0007CC47	8	40		A1953000	5	86		226BC800
144	0	19	0		3401	9000	00080048	8	40		6D943000	5	86		226BC800
145	0	20	0		3001	B7FE	00106892	7	40		73266000	4	86		44D79000
146	0	20	0		3001	87FD	00109893	7	40		43256000	4	86		44D79000
147	1	21	0		2801	AFF8	00219128	6	40		2648C000	3	86		89AF2000
148	0	19	0		3401	A004	008644A0	4	40		99230000	1	86		06E08000
149	0	20	0		3001	D806	010CF142	3	40		CA440000	0	86		0DC10000
150	0	20	0		3001	A805	010D2143	3	40		9A430000	0	86	F4	0DC10000
151	0	21	0		2801	F008	021AA288	2	40		D485E800	7	F4		1B821600
152	0	21	0		2801	C807	021ACA89	2	40		AC84E800	7	F4		1B821600
153	0	21	0		2801	A006	021AF28A	2	40		8483E800	7	F4		1B821600
154	0	22	0		2401	F00A	04363516	1	40		B905D000	6	F4		37042C00
155	0	22	0		2401	CC09	04365917	1	40		9504D000	6	F4		37042C00
156	0	22	0		2401	A808	04367D18	1	40		7103D000	6	F4		37042C00
157	0	22	0		2401	8407	0436A119	1	0D	41	4D02D000	6	F4		37042C00
158	0	23	0		2201	C00C	00058A34	8	0D		5203A000	5	F4		6E085800
159	0	23	0		2201	9E0B	0005AC35	8	0D		3002A000	5	F4		6E085800
160	0	24	0		1C01	F814	000B9C6C	7	0D		1C034000	4	F4		DC10B000
161	1	24	0		1C01	DC13	000BB86D	7	0D		00024000	4	F4		DC10B000
162	1	22	0		2401	E008	005DC368	4	0D		00120000	1	F4	31	DFF58000
163	1	20	0		3001	9004	01770DA0	2	BB	0D	00486200	7	31		8FBB9C00
164	1	19	0		3401	C004	00043680	8	BB		01218800	5	31		BEE27000
165	1	18	0		3801	D004	0010DA00	6	BB		04862000	3	31		CB7DC000
166	1	17	0		4801	E004	00436800	4	BB		12188000	1	31		CDEB0000
167	0	16	0	X	5101	9002	0086D000	3	BB		24310000	0	31	7F	6BD00000
168	0	17	0		4801	A202	010DA000	2	BB		4862FE00	7	7F		599F0000
169	1	18	0		3801	B402	021BD002	1	86	BB	00C3FC00	6	7F		B33E0000
170	1	17	0		4801	E004	000F4008	7	86		030FF000	4	7F		DCF40000
171	0	16	0	X	5101	9002	001E8010	6	86		061FE000	3	7F		89E20000
172	1	17	0		4801	A202	003D0020	5	86		0C3FC000	2	7F		95C20000
173	0	16	0	X	5101	9002	007A0040	4	86		187F8000	1	7F		77820000
174	1	17	0		4801	A202	00F40080	3	86		30FF0000	0	7F	FF	71020000
175	1	16	0	X	5101	9002	01E80100	2	F4	86	61FFFE00	7	FF		2E020000
176	1	15	0		5401	FC04	00014804	8	F4		43FBF800	5	FF		B8080000
177	1	14	0	X	5601	A802	00029008	7	F4		87F7F000	4	FF		200A0000
178	0	14	1	X	5601	A402	0005CC12	6	F4		63EDE000	3	FF		40140000
179	0	14	0	X	5601	9C02	000C4426	5	F4		1BD9C000	2	FF		80280000
180	1	15	0		5401	AC02	0018884C	4	F4		37B38000	1	FF		744E0000
181	1	14	0	X	5601	A802	00311098	3	F4		6F670000	0	FF	88	389A0000
182	1	14	1	X	5601	A402	0062CD32	2	F4		32CE2000	6	88		7133DC00
183	1	15	1		5401	AC02	00C59A64	1	31	F4	659C4000	5	88		4665B800
184	0	16	1		5101	B002	0003DCCA	8	31		23368000	4	88		8CCB7000
185	1	15	1	X	5401	A202	0007B994	7	31		466D0000	3	88		5B94E000
186	1	16	1		5101	A802	000F7328	6	31		8CDA0000	2	88		1B27C000
187	1	17	1		4801	AE02	001F8852	5	31		77B20000	1	88		364F8000
188	1	18	1		3801	CC02	003FA0A6	4	31		5F620000	0	88		6C9F0000
189	0	18	1		3801	9401	003FD8A7	4	31		27610000	0	88	FF	6C9F0000
190	1	17	1		4801	E004	00FF629C	2	31		9D87FC00	6	FF		427C0000
191	1	17	1		4801	9803	00FFAA9D	2	31		5586FC00	6	FF		427C0000
192	0	18	1		3801	A004	01FFE53C	1	7F	31	1B0BF800	5	FF		84F80000

Tableau H.1 – Données de trace de codeur et de décodeur *(fin)*

Variables communes							Codeur				Décodeur				Décodeur de convention logicielle
EC	D	I	MPS	CE	Qe hex	A hex	C hex	CT	B hex	OUT hex	C hex	CT	B hex	IN hex	C hex
193	1	17	1		4801	E004	000F94F0	7	7F		6C2FE000	3	FF		73D40000
194	0	17	1		4801	9803	000FDCF1	7	7F		242EE000	3	FF		73D40000
195	1	16	1	X	5101	9002	001FB9E2	6	7F		485DC000	2	FF		47A40000
196	1	17	1		4801	A202	003F73C4	5	7F		90BB8000	1	FF		11460000
197	1	18	1		3801	B402	007F778A	4	7F		91750000	0	FF	37	228C0000
198	1	19	1		3401	F802	00FF5F16	3	7F		B2E8DC00	6	37		45192000
199	1	19	1		3401	C401	00FF9317	3	7F		7EE7DC00	6	37		45192000
200	1	19	1		3401	9000	00FFC718	3	7F		4AE6DC00	6	37		45192000
201	0	20	1		3001	B7FE	01FFF632	2	FF	7F	2DCBB800	5	37		8A324000
202	1	19	1		3401	C004	0007D8C8	8	FF		B72EE000	3	37		08D50000
203	1	19	1		3401	8C03	00080CC9	8	FF		832DE000	3	37		08D50000
204	1	20	1		3001	B004	00108194	7	FF		9E59C000	2	37		11AA0000
205	1	20	1		3001	8003	0010B195	7	FF		6E58C000	2	37		11AA0000
206	1	21	1		2801	A004	0021C32C	6	FF		7CAF8000	1	37		23540000
207	1	22	1		2401	F006	0043D65A	5	FF		A95D0000	0	37		46A80000
208	1	22	1		2401	CC05	0043FA5B	5	FF		855C0000	0	37		46A80000
209	1	22	1		2401	A804	00441E5C	5	FF		615B0000	0	37		46A80000
210	1	22	1		2401	8403	0044425D	5	FF		3D5A0000	0	37	47	46A80000
211	1	23	1		2201	C004	0088CCBC	4	FF		32B28E00	7	47		8D517000
212	0	23	1		2201	9E03	0088EEBD	4	FF		10B18E00	7	47		8D517000
213	1	21	1		2801	8804	0223BAF4	2	FF		42C63800	5	47		453DC000
214	1	22	1		2401	C006	0447C5EA	1	FF		358A7000	4	47		8A7B8000
215	0	22	1		2401	9C05	0447E9EB	1	88	FF	11897000	4	47		8A7B8000
216	1	20	1		3001	9004	001FA7AC	6	88		4625C000	2	47		49DE0000
217	1	21	1		2801	C006	003FAF5A	5	88		2C498000	1	47		93BC0000
218	0	21	1		2801	9805	003FD75B	5	88		04488000	1	47	1A	93BC0000
219	0	19	1		3401	A004	00FF5D6C	3	88		11223400	7	1A		8EE1CA00
220	1	18	1		3801	D004	03FD75B0	1	88		4488D000	5	1A		8B7B2800
221	0	18	1		3801	9803	03FDADB1	1	FF	88	0C87D000	5	1A		8B7B2800
222	0	17	1		4801	E004	0006B6C4	7	FF		321F4000	3	1A		ADE4A000
223	0	16	1	X	5101	9002	000D6D88	6	FF		643E8000	2	1A		2BC34000
224	0	15	1		5401	FC04	0036FA24	4	FF		4CF60000	0	1A	DB	AF0D0000
225	0	14	1	X	5601	A802	006DF448	3	FF		99EDB600	7	DB		0E144800
226	1	14	0	X	5601	A402	00DC9492	2	FF		87D96C00	6	DB		1C289000
227	0	14	1	X	5601	9C02	01B9D526	1	37	FF	63B0D800	5	DB		38512000
228	0	14	0	X	5601	8C02	0004564E	7	37		1B5FB000	4	DB		70A24000
229	1	15	0		5401	AC02	0008AC9C	6	37		36BF6000	3	DB		75428000
230	1	14	0	X	5601	A802	00115938	5	37		6D7EC000	2	DB		3A830000
231	1	14	1	X	5601	A402	00235E72	4	37		2EFB8000	1	DB		75060000
232	1	15	1		5401	AC02	0046BCE4	3	37		5DF70000	0	DB	6A	4E0A0000
233	0	16	1		5101	B002	008E21CA	2	37		13ECD400	7	6A		9C152A00
234	1	15	1	X	5401	A202	011C4394	1	47	37	27D9A800	6	6A		7A285400
235	0	16	1		5101	A802	00008728	8	47		4FB35000	5	6A		584EA800
236	0	15	1	X	5401	A202	00010E50	7	47		9F66A000	4	6A		029B5000
237	0	14	1	X	5601	9C02	0002C4A2	6	47		96CB4000	3	6A		0536A000
238	1	14	0	X	5601	8C02	00063546	5	47		81948000	2	6A		0A6D4000
239	1	14	1		5601	D804	001A2D1C	3	47		AE4E0000	0	6A		29B50000
240	0	14	1	X	5601	8203	001A831D	3	47		584D0000	0	6A	DF	29B50000
241	1	14	0		5601	B008	006B6478	1	1A	47	09337C00	6	DF		A6D48000
242	0	14	1		5601	AC02	0006C8F0	8	1A		1266F800	5	DF		999B0000
243	1	14	0		5601	AC02	000D91E0	7	1A		24CDF000	4	DF		87340000
244	0	14	1		5601	AC02	001B23C0	6	1A		499BE000	3	DF		62660000
245	0	14	0		5601	AC02	00364780	5	1A		9337C000	2	DF		18CA0000
246	0	15	0		5401	AC02	006D3B02	4	1A		7A6D8000	1	DF		31940000
247	1	16	0		5101	B002	00DB1E06	3	1A		4CD90000	0	DF	FF	63280000
248	1	15	0	X	5401	A202	01B63C0C	2	1A		99B3FE00	7	FF		084E0000
249	1	14	0	X	5601	9C02	036D201A	1	DB	1A	8B65FC00	6	FF		109C0000
250	0	14	1	X	5601	8C02	0002EC36	8	DB		6AC9F800	5	FF		21380000
251	1	14	0		5601	D804	000D08DC	6	DB		5323E000	3	FF		84E00000
252	1	14	1		5601	AC02	001A11B8	5	DB		A647C000	2	FF		05BA0000
253	1	15	1		5401	AC02	0034CF72	4	DB		A08D8000	1	FF		0B740000
254	1	16	1		5101	B002	006A46E6	3	DB		99190000	0	FF		16E80000
255	1	17	1		4801	BE02	00D52FCE	2	DB		9031FE00	7	FF		2DD00000
256	1	18	1		3801	EC02	01AAEF9E	1	DB		9061FC00	6	FF		5BA00000
257										DB 6A DF FF AC					

Bibliographie

[1] ASCHER (R. N.), NAGY (G.): A means for achieving a high degree of compaction on scan-digitized printed text, *IEEE Transactions on Computers*, C-23:1174-1179, novembre 1974.

[2] BILGIN (A.), SEMENTILLI (P.), MARCELLIN (M.): Progressive image coding using trellis coded quantization, *IEEE Transactions on Image Processing*, novembre 1997. (submitted).

[3] CONSTANTINESCU (C.), ARPS (R.): Fast residue coding for lossless textual image compression, in *Proc. of Data Compression Conference*, pages 397-406, Snowbird, Utah, USA, 1997.

[4] FORCHHAMMER (S.), JENSEN (K. S.): Data compression of scanned halftone images, *IEEE Trans. Commun.*, 42:1881–1893, février-avril 1994.

[5] HOWARD (P. G.): Lossless and lossy compression of text images by soft pattern matching, in *Proc. of Data Compression Conference*, pages 210-219, 1996.

[6] HOWARD (P. G.): Text image compression using soft pattern matching, *Computer Journal*, 40:2-3, 1997.

[7] HOWARD (P. G.), KOSSENTINI (F.), MARTINS (B.), FORCHHAMMER (S.), RUCKLIDGE (W. J.): The emerging JBIG2 standard, *IEEE Transactions on Circuits and Systems for Video Technology*, 8(7):838-848, novembre 1998.

[8] HUNTER (R.), ROBINSON (A. H.): International digital facsimile coding standards, in *Proceedings of IEEE*, Volume 68, pages 854-867, juillet 1980.

[9] INGLIS (S.), WITTEN (I. H.): Compression-based template matching, in J. Storer and M. Cohn, editors, *Proc. IEEE Data Compression Conference*, 1994.

[10] KOSSENTINI (F.), WARD (R.): An analysis-compression technique for black and white documents, in *IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 141-144, San Antonio, TX, USA, avril 1996.

[11] MARTINS (B.), FORCHHAMMER (S.): Lossless/lossy compression of bi-level images, in *Proc. of IS&T/SPIE Symp. on Electr. Im.: Science and Technology 1997*, Volume 3018, pages 38-49, 1997.

[12] MARTINS (B.), FORCHHAMMER (S.): Halftone Coding with JBIG2, submitted to Journal of Electronic Imaging, septembre 1998.

[13] MARTINS (B.), FORCHHAMMER (S.): Tree Coding of Bi-level Images, *IEEE Trans. Image Proc.*, 7(4):517-528, 1998.

[14] MARTINS (B.), FORCHHAMMER (S.): Lossless, near-lossless, and refinement coding of bi-level images, *IEEE Trans. Image Processing*, 8(5):601-613, mai 1999.

[15] MOHIUDDIN (K.), RISSANEN (J. J.), ARPS (R.): Lossless binary image compression based on pattern matching, in *Proceedings of International Conference on Computers, Systems, and Signal Processing*, páginas 447-451, Bangalore, India, 1984.

[16] PRATT (W. K.), CAPITANT (P. J.), CHEN (W. H.), HAMILTON (E. R.), WALLS (R. H.): Combined symbol matching facsimile data compression system, in *Proceedings of the IEEE*, Volume 68, pages 786-796, juillet 1980.

[17] SAID (A.), PEARLMAN (W. A): A new fast and efficient image codec based on set partitioning in hierarchical trees, *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243-250, juin 1996.

[18] SAYOOD (K.): *Introduction to Data Compression*, Morgan Kaufmann Publishers, San Francisco, CA, 1996.

[19] TOU (J. T.), GONZALEZ (R. C.): *Pattern Recognition Principles*, Addison-Wesley Publishing Company, Reading, MA, 1974.

[20] WITTEN (I. H.), MOFFAT (A.), BELL (T. C.): *Managing Gigabytes*, Van Nostrand Reinhold, New York, 1994.

SÉRIES DES RECOMMANDATIONS UIT-T

Série A	Organisation du travail de l'UIT-T
Série B	Moyens d'expression: définitions, symboles, classification
Série C	Statistiques générales des télécommunications
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 et protocole Internet
Série Z	Langages et aspects généraux logiciels des systèmes de télécommunication

Genève, 2002