UNIÓN INTERNACIONAL DE TELECOMUNICACIONES

UIT-T T.88

(02/2000)

SECTOR DE NORMALIZACIÓN
DE LAS TELECOMUNICACIONES
DE LA UIT

SERIE T: TERMINALES PARA SERVICIOS DE TELEMÁTICA

Tecnología de la información – Codificación con pérdida/sin pérdida de imágenes binivel

Recomendación UIT-T T.88

(Anteriormente Recomendación del CCITT)

NORMA INTERNACIONAL ISO/CEI 14492

RECOMENDACIÓN UIT-T T.88

TECNOLOGÍA DE LA INFORMACIÓN – CODIFICACIÓN CON PÉRDIDA/SIN PÉRDIDA DE IMÁGENES BINIVEL

Resumen

Esta Recomendación | Norma Internacional, llamada de manera oficiosa JBIG2, define un método de codificación de imágenes binivel (por ejemplo, material impreso en blanco y negro). Imágenes que constan de un solo plano de bits rectangular, en el que cada píxel puede tomar uno u otro de dos únicos colores posibles. Esta Recomendación | Norma Internacional se ha preparado de manera explícita para comprensión de imágenes, con pérdida, sin pérdida y de con pérdida a sin pérdida.

Orígenes

La Recomendación UIT-T T.88 se aprobó el 10 de febrero de 2000. Su texto se publica también, en forma idéntica, como Norma Internacional | ISO/CEI 14492.

PREFACIO

La UIT (Unión Internacional de Telecomunicaciones) es el organismo especializado de las Naciones Unidas en el campo de las telecomunicaciones. El UIT-T (Sector de Normalización de las Telecomunicaciones de la UIT) es un órgano permanente de la UIT. Este órgano estudia los aspectos técnicos, de explotación y tarifarios y publica Recomendaciones sobre los mismos, con miras a la normalización de las telecomunicaciones en el plano mundial.

La Conferencia Mundial de Normalización de las Telecomunicaciones (CMNT), que se celebra cada cuatro años, establece los temas que han de estudiar las Comisiones de Estudio del UIT-T, que a su vez producen Recomendaciones sobre dichos temas.

La aprobación de Recomendaciones por los Miembros del UIT-T es el objeto del procedimiento establecido en la Resolución 1 de la CMNT.

En ciertos sectores de la tecnología de la información que corresponden a la esfera de competencia del UIT-T, se preparan las normas necesarias en colaboración con la ISO y la CEI.

NOTA

En esta Recomendación, la expresión "Administración" se utiliza para designar, en forma abreviada, tanto una administración de telecomunicaciones como una empresa de explotación reconocida de telecomunicaciones.

PROPIEDAD INTELECTUAL

La UIT señala a la atención la posibilidad de que la utilización o aplicación de la presente Recomendación suponga el empleo de un derecho de propiedad intelectual reivindicado. La UIT no adopta ninguna posición en cuanto a la demostración, validez o aplicabilidad de los derechos de propiedad intelectual reivindicados, ya sea por los miembros de la UIT o por terceros ajenos al proceso de elaboración de Recomendaciones.

En la fecha de aprobación de la presente Recomendación, la UIT no ha recibido notificación de propiedad intelectual, protegida por patente, que puede ser necesaria para aplicar esta Recomendación. Sin embargo, debe señalarse a los usuarios que puede que esta información no se encuentre totalmente actualizada al respecto, por lo que se les insta encarecidamente a consultar la base de datos sobre patentes de la TSB.

ÍNDICE

Página

0 Introducción

La presente Recomendación | Norma Internacional, llamada de manera oficiosa JBIG2, define un método de codificación de imágenes binivel (por ejemplo, material impreso en blanco y negro). Imágenes que constan de un solo plano de bits rectangular, en el que cada píxel puede tomar uno u otro de dos únicos colores posibles. Los colores múltiples se han de tratar utilizando una norma de nivel superior, por ejemplo la Recomendación UIT-T T.44. Dicha norma está siendo redactada por el Grupo Mixto de Expertos en Imagen Binivel (JBIG), un "equipo de colaboración" establecido en 1988, que informa de sus resultados a ISO/IEC JTC1/SC29/WG1 y al UIT-T.

Las normas sobre facsímil existentes también se refieren a este tipo de compresión de imágenes, por ejemplo, al describir los algoritmos de compresión en las Recomendaciones UIT-T T.4 (MH, MR), T.6 (MMR), T.82 (JBIG1) y T.85 (Perfil de aplicación de la norma JBIG1 para facsímil). Además de la aplicación facsímil obvia, las especificaciones de la norma JBIG2 serán de utilidad para el almacenamiento y archivo de documentos, la codificación de imágenes en la WWW, la transmisión inalámbrica de datos, la gestión programada de impresiones e incluso las teleconferencias.

Como resultado del proceso que concluyó en 1993, el JBIG produjo una primera norma de codificación designada de manera oficial Recomendación UIT-T T.82 | Norma Internacional ISO/CEI 11544, a la que se conoce de manera oficiosa como JBIG o JBIG1. La JBIG1 tiene por objeto servir a modo de norma de codificación sin pérdida y progresiva (de con pérdida a sin pérdida). Aunque puede aplicarse a la codificación con pérdida, la calidad de las imágenes con pérdida, producidas de conformidad con la norma JBIG1 es notablemente inferior a la de las imágenes originales porque el número de píxels en la imagen con pérdida no puede superar a una cuarta parte de los de la imagen original.

Por el contrario, la norma JBIG2 se preparó de manera explícita para compresión de imágenes con pérdida, sin pérdida y de con pérdida a sin pérdida. El objetivo de diseño de la norma JBIG2 era hacer posible que el resultado de la compresión sin pérdida fuera mejor que lo especificado en las normas existentes y hacer posible también una compresión con pérdida con relaciones de compresión mucho más altas que las relaciones sin pérdida especificadas, asimismo, en las normas existentes, sin apenas degradación de calidad visible. Además, la norma JBIG2 permite la codificación de calidad progresiva, pasando de calidad inferior a superior (o sin pérdida), y codificación de contenido progresivo, añadiendo de manera sucesiva diferentes tipos de datos de imagen (por ejemplo, primero texto y a continuación semitonos). Un codificador JBIG2 típico descompone la imagen binivel de entrada en varias regiones y codifica cada una de ellas separadamente utilizando un método de codificación diferente. Esa descomposición en base al contenido es muy conveniente sobre todo en aplicaciones multimedios interactivas. Con la norma JBIG2 se puede tratar un conjunto de imágenes (documento de múltiples páginas) de manera explícita.

Como es habitual en las normas relativas a la compresión de imágenes, la JBIG2 define claramente los requisitos de un tren binario que se atenga a la misma, y define, por tanto, el comportamiento del decodificador. La norma JBIG2 no define explícitamente un codificador estándar, pero en cambio es lo bastante flexible como para permitir diseños de codificador complejos. De hecho, el diseño del codificador constituirá un factor diferenciador importante entre implementaciones de la norma JBIG2 competidoras.

Aunque esta Recomendación | Norma Internacional se ha redactado en términos de acciones a realizar por los decodificadores para interpretar un tren binario, se considera que cualquier decodificador que produzca el resultado correcto (definido por esas acciones) es conforme a la norma, con independencia de las acciones que finalmente lleve a cabo.

Los anexos A, B, C, D, E y F son normativos y, por tanto, son parte integrante de la presente Recomendación | Norma Internacional. Los anexos G y H son informativos y, por tanto, no son parte integrante de la presente.

0.1 Interpretación y utilización de los requisitos

Esta sección es informativa y tiene por objeto ayudar a interpretar los requisitos de la presente Recomendación | Norma Internacional. Los requisitos se han formulado de manera que sean lo más generales posible para que las implementaciones tengan un alto grado de flexibilidad. Por ello, el lenguaje con que se formulan los requisitos no es específico de ninguna aplicación o implementación. En lo que sigue se establece una correspondencia entre la redacción general de los requisitos y el uso pretendido de la presente Recomendación | Norma Internacional en aplicaciones típicas.

0.1.1 Aplicación de la codificación JBIG2

La norma JBIG2 se utiliza para codificar documentos binivel. Un documento binivel contiene una o más páginas. Una página típica contiene algunos datos de texto, es decir, algunos caracteres de pequeño tamaño dispuestos en filas horizontales o verticales. Los caracteres de la parte texto de una página se llaman *símbolos* en la JBIG2. Una página puede contener también "datos de semitonos", Es decir, imágenes de escala de grises o de color multinivel (fotografías) que han sido tremoladas para producir imágenes binivel. Las casillas de los mapas binarios periódicos de la parte semitonos de la página se llaman *patrones* en la JBIG2. Además, la página puede contener otros datos, por ejemplo, dibujo lineal y ruido. A esos datos que no son de texto ni de semitonos se les denomina datos *genéricos* en la JBIG2.

El modelo de imagen JBIG2 trata los datos de texto y los datos de semitonos como casos especiales. Lo previsto es que un codificador JBIG2 divida el contenido de una página en una región de texto que contenga texto digitalizado, una región de semitonos que contenga semitonos digitalizados y una región genérica que contenga los datos restantes de imágenes digitalizadas, por ejemplo, el dibujo lineal. En algunas circunstancias conviene considerar los datos de texto o semitonos como datos genéricos, ya sea por la calidad de la imagen o por el tamaño de los datos comprimidos y, a la inversa, en algunas circunstancias es mejor considerar los datos genéricos utilizando uno de los casos especiales.

Un codificador puede dividir una página en un cierto número de regiones, pero a menudo bastarán tres regiones, una para símbolos textuales, otra para patrones de semitonos y la tercera para los genéricos restantes. En algunos casos, es posible que no estén presentes todos los tipos de datos, y la página conste de menos de tres regiones.

Las diversas regiones se pueden superponer en la página física. La norma JBIG2 indica la manera de especificar cómo se recombinan las regiones que se superponen para formar la imagen de la página final.

Una región de texto consta de un cierto número de símbolos situados en ubicaciones especificadas en un plano de fondo. Los símbolos corresponden normalmente a caracteres de texto individuales. Gran parte de la efectividad de la JBIG2 se debe a la utilización de símbolos individuales más de una vez. Para reutilizar un símbolo, el codificador o el decodificador debe disponer de una manera sucinta de referirse a él. En la JBIG2, los símbolos se recogen en uno o más diccionarios de símbolos. Un diccionario de símbolos es un conjunto de mapas de bits de símbolos de texto, indizados de manera que se pueda hacer referencia al mapa de bits de un símbolo mediante un número de índice.

Una región de semitonos consta de un cierto número de patrones situados a lo largo de una cuadrícula regular. Los patrones corresponden normalmente a valores de la escala de grises. De hecho, el método de codificación de los índices de los patrones está concebido como si se tratara de codificar una escala de grises. La compresión se puede llevar a cabo representando los píxels binarios de una casilla de cuadrícula mediante un único entero, el índice del semitono (que normalmente es un valor de la escala de grises reproducido). Esta correspondencia entre muchos y uno (entre el patrón de una casilla y un valor de la escala de grises) puede tener como consecuencia el que la información de bordes presente en el mapa de bits original se pierda en la codificación de semitonos. Por este motivo, la codificación sin pérdida o casi sin pérdida de semitonos tendrá a menudo una calidad de imagen mejor (aunque de tamaño mayor) si el semitono se codifica con codificación genérica en vez de codificación de semitonos.

0.1.2 Relación entre segmentos y documentos

Un fichero JBIG2 contiene la información necesaria para decodificar un documento binivel y se compone de *segmentos*. Una página típica se codifica utilizando varios segmentos. En los casos sencillos, habrá un segmento de información de página, un segmento de diccionario de símbolos, un segmento de región de texto, un segmento de diccionario de patrones, un segmento de región de semitonos y un segmento fin de página. El segmento de información de página proporciona información general sobre la página, por ejemplo, su tamaño y su resolución. Los segmentos de diccionario contienen mapas de bits a los que se hace referencia en los segmentos de región. Los segmentos de región describen el aspecto de las regiones de texto y semitonos haciendo referencia a mapas de bits de un diccionario y especificando dónde deberán aparecer en la página. El segmento fin de página marca el final de la página.

0.1.3 Estructura y utilización de los segmentos

Cada segmento contiene un encabezamiento de segmento, un encabezamiento de datos, y datos. El encabezamiento de segmento se utiliza para llevar información de referencia del segmento y, en el caso de documentos multipáginas, información de asociación de páginas. Un encabezamiento de datos da información utilizada para decodificar los datos del segmento. Los datos describen una región de imagen o un diccionario, o proporcionan información de otro tipo.

Los segmentos se enumeran de manera secuencial. Un segmento puede hacer referencia a otro segmento con un número inferior, o *anterior*. Un segmento de región está asociado siempre con una página específica del documento. Un segmento de diccionario puede estar asociado con una página del documento, o puede estar asociado con el documento en su conjunto.

Un segmento de región puede hacer referencia a uno o más segmentos de diccionario anteriores. La finalidad de esa referencia es permitir al decodificador la identificación de símbolos de un segmento de diccionario que están presentes en la imagen.

Un segmento de región puede hacer referencia a un segmento de región anterior. La finalidad de esa referencia es combinar la imagen descrita por el segmento anterior con la representación actual de la página.

Un segmento de diccionario puede hacer referencia a segmentos de diccionario anteriores. Los símbolos añadidos a un segmento de diccionario pueden ser descritos directamente, o como refinamiento de los símbolos descritos anteriormente, ya sea en el mismo segmento de diccionario o en segmentos de diccionario anteriores.

Un fichero JBIG2 puede organizarse de dos maneras: secuencialmente o por acceso aleatorio. En la organización secuencial, cada encabezamiento de segmento de un segmento precede de manera inmediata al encabezamiento de datos y a los datos de ese segmento, todo en orden secuencial. En la organización de acceso aleatorio, todos los encabezamientos de segmento se reúnen al comienzo del fichero, y van seguidos por los datos (incluyendo los encabezamientos de datos) de todos los segmentos, en el mismo orden. Esta segunda organización permite que un decodificador determine todas las dependencias de segmentos sin leer el fichero en su totalidad.

Una tercera forma de encapsular datos codificados según la norma JBIG2 consiste en incorporarlos en un fichero no JBIG2; esto es lo que a veces se denomina *organización insertada*. En este caso, un formato de fichero diferente lleva los segmentos JBIG2. El encabezamiento de segmento, el encabezamiento de datos y los datos de cada segmento se almacenan juntos, pero el formato del fichero en que se insertan puede almacenar los segmentos en cualquier orden y en cualquier conjunto de ubicaciones dentro de su propia estructura.

0.1.4 Representaciones internas

Los datos decodificados deben ser almacenados antes de imprimirlos o visualizarlos. Aunque esta Recomendación | Norma Internacional no especifica cómo se almacenan, en su modelo de decodificación se suponen determinadas estructuras de datos, en concreto, memorias intermedias y diccionarios. La figura 1 ilustra los principales componentes de un decodificador y las memorias intermedias asociadas. En esa figura, los procedimientos de decodificación se señalan mediante líneas oscuras, y los componentes de memoria, mediante líneas claras. Además, flechas de línea oscura indican que un procedimiento de decodificación invoca otro procedimiento de codificación, por ejemplo, el procedimiento de decodificación de diccionario de símbolos invoca el procedimiento de decodificación de región genérica para decodificar los mapas de bits de los símbolos que define. Las flechas de línea clara indican el flujo de datos: el procedimiento de decodificación de la región de texto lee símbolos de la memoria de símbolos y los dibuja en la memoria intermedia de la página o en una memoria intermedia auxiliar. Aunque no se muestra en la figura 1, el tren de datos codificados fluye hacia los procedimientos de decodificación, y el bloque titulado "memorias intermedias de página y auxiliares" produce las imágenes de página decodificada finales.

Los recursos requeridos para decodificar cualquier tren binario JBIG2 dependen de la complejidad de ese tren binario. Se pueden utilizar algunas técnicas, por ejemplo, la de división en franjas, para reducir la necesidad de memoria del decodificador. Se estima que un decodificador con todas sus características funcionales puede necesitar dos memorias intermedias de página completa, más aproximadamente la misma cantidad de memoria de diccionario, más unos 100 kilobytes de memoria de contexto de codificación aritmética, para decodificar la mayoría de los trenes binarios.

Una memoria intermedia es la representación de un mapa de bits. Su finalidad es retener una gran cantidad de datos, normalmente los de una página completa. Una memoria intermedia puede contener la descripción de una región o de una página en su totalidad. Incluso si la memoria intermedia sólo describe una región, tiene información asociada a la misma que especifica su ubicación en la página. La decodificación de un segmento de región modifica el contenido de una memoria intermedia.

Sólo existe una memoria intermedia especial, la *memoria intermedia de página*. Se pretende que el decodificador acumule datos de región directamente en la memoria intermedia de página hasta que la página haya sido decodificada por completo; a continuación los datos se pueden enviar al dispositivo de salida o archivo. La decodificación de un segmento de región *intermedia* modifica el contenido de la memoria intermedia de página. La manera usual de preparar una página consiste en decodificar uno o más segmentos de región inmediata, cada uno de los cuales modifica la memoria intermedia de la página. El decodificador puede producir como salida una memoria intermedia de página incompleta, ya sea como parte de la transmisión progresiva o en respuesta a la introducción de datos por el usuario. Esa salida es facultativa, y su contenido no se especifica en la presente Recomendación | Norma Internacional.

Todas las demás memorias intermedias son auxiliares. Se pretende que el decodificador llene una memoria intermedia y más tarde la utilice para mejorar la memoria intermedia de página. A menudo, en una aplicación, no será necesario tener memorias intermedias auxiliares. La decodificación de un segmento de región *intermedia* modifica el contenido de una memoria intermedia auxiliar. El decodificador puede utilizar memorias intermedias auxiliares para producir como salida páginas distintas de las que figuran en una memoria intermedia de página completa, ya sea como parte de la transmisión progresiva o en respuesta a la introducción de datos por el usuario. Esa salida es facultativa, y su contenido no se especifica en la presente Recomendación | Norma Internacional.

Un diccionario de símbolos consta de un conjunto indizado de mapas de bits. Los mapas de bits de un diccionario son pequeños normalmente, de aproximadamente el tamaño de los caracteres del texto. A diferencia de una memoria intermedia, un mapa de bits de un diccionario no tiene información sobre ubicación de páginas asociada con él.

0.1.5 Resultados de la decodificación

La decodificación de un segmento conlleva la invocación de uno o más procedimientos de decodificación. Los procedimientos de decodificación que se han de invocar vienen determinados por el tipo de segmento.

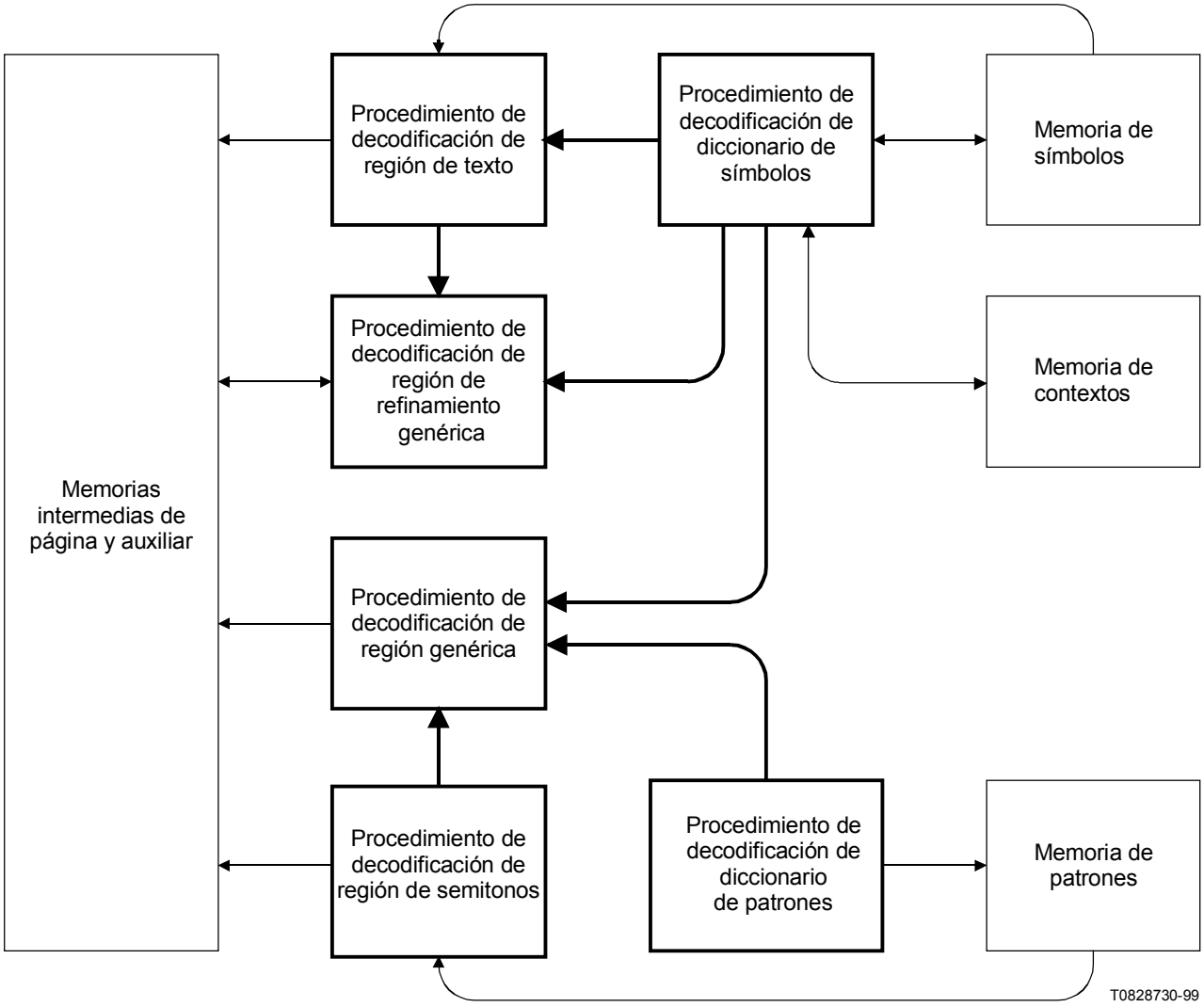


Figura 1 – Diagrama de bloques de los principales componentes de un decodificador

El resultado de la decodificación de un segmento de región es un mapa de bits almacenado en una memoria intermedia, posiblemente la memoria intermedia de la página. La decodificación de un segmento de región puede llenar una memoria intermedia nueva, o puede modificar una memoria intermedia existente. En aplicaciones típicas, la inserción de los datos en una memoria intermedia implica el cambio de píxels del color del plano de fondo al color del primer plano, pero esta Recomendación | Norma Internacional especifica otras posibles maneras de cambiar los píxels de una memoria intermedia.

Una página típica se describirá mediante uno o más segmentos de región inmediata, cada uno de los cuales da lugar a la modificación de la memoria intermedia de la página.

Del mismo modo que es posible especificar un símbolo nuevo en un diccionario refinando un símbolo especificado previamente, también lo es especificar una memoria intermedia nueva refinando una memoria intermedia existente. Sin embargo, una región sólo puede ser refinada mediante el procedimiento de decodificación de refinamiento genérica. Dicho refinamiento no utiliza la estructura interna de la región en la memoria intermedia que se refina. Una vez refinada una memoria intermedia, deja de estar disponible la memoria intermedia original.

El resultado de la decodificación de un segmento de diccionario es un diccionario nuevo. Los símbolos del diccionario pueden ser situados más tarde en una memoria intermedia por el procedimiento de decodificación de región de texto.

0.1.6 Procedimientos de decodificación

El *procedimiento de decodificación de región genérica* llena o modifica directamente una memoria intermedia, píxel por píxel si se utiliza codificación aritmética, o por pasadas de píxels de plano de fondo o de primer plano si se utilizan las codificaciones MMR y Huffman. En el caso de codificación aritmética, el contexto de predicción contiene sólo píxels determinados por datos ya decodificados dentro del segmento de que se trate.

El *procedimiento de decodificación de región de refinamiento genérico* modifica una memoria intermedia píxel por píxel utilizando la codificación aritmética. El contexto de predicción utiliza píxels determinados por datos ya decodificados dentro del segmento en curso así como píxels ya presentes en la memoria intermedia de página o en una memoria intermedia auxiliar.

El *procedimiento de decodificación de región de texto* toma símbolos de uno o más diccionarios de símbolos y los sitúa en una memoria intermedia. Este procedimiento es invocado durante la decodificación de un segmento de región de texto. El segmento de región de texto contiene la información de posición e índice de cada símbolo que se ha de situar en la memoria intermedia; los mapas de bits de los símbolos se toman de los diccionarios de símbolos.

El *procedimiento de decodificación de diccionario de símbolos* crea un diccionario de símbolos, es decir, un conjunto indizado de mapas de bits de símbolos. Un mapa de bits del diccionario puede ser codificado directamente; puede ser decodificado como un refinamiento de un símbolo ya existente en un diccionario; o puede ser decodificado como una agregación de dos o más símbolos existentes en diccionarios. Este procedimiento de decodificación es invocado durante la decodificación de un segmento de diccionario de símbolos.

El *procedimiento de decodificación de región de semitonos* toma patrones de un diccionario de patrones y los sitúa en una memoria intermedia. Este procedimiento es invocado durante la decodificación de un segmento de región de semitonos. El segmento de región de semitonos contiene la información de posición de todos los patrones que se han de situar en la memoria intermedia, así como información de índices de los propios patrones. Los patrones, los mapas de bits de tamaño fijo del semitono, se toman de los diccionarios de semitonos.

El *procedimiento de decodificación de diccionario de patrones* crea un diccionario, que es un conjunto indizado de mapas de bits de tamaño fijo (patrones). Los mapas de bits del diccionario se codifican directa y conjuntamente. Este procedimiento de decodificación es invocado durante la decodificación de un segmento de diccionario de patrones.

El *procedimiento de decodificación de control* decodifica encabezamientos de segmento, que incluyen información sobre el tipo de segmento. El tipo de segmento determina qué procedimiento de decodificación ha de ser invocado para decodificar el segmento. El tipo de segmento determina además dónde será situada la salida decodificada del segmento. La información de referencia del segmento, presente también en el encabezamiento del segmento y decodificada por el procedimiento de decodificación de control, determina qué otros segmentos se han de utilizar para decodificar el segmento en curso. El procedimiento de decodificación de control afecta a todo el proceso expuesto en la figura 1, por lo que no se muestra en ella como un bloque aparte.

El cuadro 1 presenta de forma resumida los tipos de datos que se decodifican, el procedimiento de decodificación seguido para decodificarlos y las representaciones finales de los datos decodificados.

Cuadro 1 – Entidades del proceso de decodificación

Concepto	Entidad de tren binario JBIG2	Entidad decodificadora JBIG2	Representación física
Documento	Fichero JBIG2	Decodificador JBIG2	Medio o dispositivo de salida
Página	Conjunto de segmentos	Implícita en el procedimiento de decodificación de control	Memoria intermedia de página
Región	Segmento de región	Procedimiento de decodificación de región	Memoria intermedia de página o memoria intermedia auxiliar
Diccionario	Segmento de diccionario	Procedimiento de decodificación de diccionario	Lista de símbolos
Carácter	Campo dentro de un segmento de diccionario de símbolos	Procedimiento de decodificación de diccionario de símbolos	Mapa de bits de símbolo
Valor de escala de grises	Campo dentro de un segmento de diccionario de semitonos	Procedimiento de decodificación de diccionario de patrones	Patrón

0.2 Codificación con pérdida

Esta Recomendación | Norma Internacional no define la manera de controlar la codificación con pérdida de imágenes binivel. Más bien define la manera de efectuar la reconstrucción perfecta de un mapa de bits elegido por el codificador para proceder a su codificación. Si el codificador elige un mapa de bits distinto del original, el proceso completo pasa a ser un proceso de codificación con pérdida. Los métodos de codificación diferentes permiten formas distintas de introducir la pérdida de manera provechosa.

0.2.1 Codificación de símbolos

La codificación de símbolos con pérdida es una manera natural de efectuar la codificación con pérdida de regiones de texto. Se trata de hacer posibles pequeñas diferencias entre el mapa de bits del símbolo original y el indizado en el diccionario de símbolos. La ganancia de compresión se produce al no tener que codificar un diccionario de gran tamaño y, después, porque se dispone de una codificación de índices de símbolos barata como consecuencia del menor tamaño del diccionario. El codificador ha de decidir cuándo dos mapas de bits son básicamente el mismo o básicamente diferentes. Esta técnica se describió por primera vez en [1].

En la codificación de símbolos con pérdida se corre el riesgo de incurrir en *errores de sustitución*, esto es, hacer que el codificador sustituya el mapa de bits correspondiente a un carácter por un mapa de bits que describe un carácter distinto, con lo que un lector humano leería erróneamente el carácter. El riesgo de los errores de sustitución se puede reducir utilizando medidas complejas de la diferencia entre mapas de bits y/o asegurando que los píxels críticos del mapa de bits indizado son correctos. Una manera de controlar esto, descrita en [5], consiste en indizar el símbolo posiblemente erróneo y aplicar a continuación la codificación de refinamiento al mapa de bits de ese símbolo. De lo que se trata es de codificar el perfil del carácter básico a un coste reducido y corregir a continuación los píxels que el codificador crea que alteran el significado del carácter.

El proceso consistente en introducir pérdida de manera provechosa en regiones del texto puede adoptar también formas más sencillas, por ejemplo, la eliminación de manchas de los documentos o la regularización de los bordes de las letras. Lo más probable es que esos cambios reduzcan la longitud de código de la región sin afectar al aspecto general de la misma, quizás incluso mejorándolo.

En [7] figuran varios ejemplos de cómo se realiza esta especie de codificación de símbolos con pérdida con la norma JBIG2.

NOTA – Aunque la expresión "región de texto" se utiliza para regiones de la página codificada utilizando la codificación de símbolos, otras posibles utilizaciones de la codificación de símbolos son la codificación del dibujo lineal y la de otros datos no textuales.

0.2.2 Codificación genérica

Para efectuar una codificación casi sin pérdida utilizando codificación genérica, el codificador aplica un preproceso a una imagen original y codifica la imagen cambiada sin pérdida. Las dificultades consisten en asegurar que los cambios dan lugar a una longitud de código menor y que la calidad de la imagen modificada no se deteriora debido a los cambios. En [11] se indican dos posibles preprocesos. Dichos preprocesos imprimen un giro de 90° a los píxels que, una vez girados, reducen notablemente la longitud de código total de la región, sin que ese giro degrade de manera importante la calidad visual. Los preprocesos permiten una codificación casi sin pérdida efectiva de semitonos periódicos y una ganancia moderada en la compresión de otros tipos de datos. Los preprocesos no sirven en el caso de imágenes de error difuso e imágenes tremoladas con ruido azul, ya que la compresión perceptivamente sin pérdida no puede conseguirse con una tasa de compresión bstante más baja que la tasa sin pérdida.

0.2.3 Codificación de semitonos

La codificación de semitonos es la manera natural de obtener una alta compresión para semitonos *periódicos* tales como las imágenes tremoladas ordenadas por conglomerados-puntos. Al contrario que en la codificación genérica con pérdida descrita más arriba, la codificación de semitonos no pretende preservar el mapa de bits original, aunque es posible en casos especiales. También se puede introducir pérdida para conseguir un mayor grado de compresión evitando poner todos los patrones de la imagen original en el diccionario, con lo que se reducen tanto el número de patrones de semitono como el número de bits requeridos para especificar qué patrón se utiliza en cada sitio.

Para la codificación con pérdida de imágenes de error difuso e imágenes tremoladas con ruido azul es aconsejable utilizar la codificación de semitonos con un tamaño de cuadrícula pequeño. La imagen reconstruida carecerá de nitidez y es posible que en ella se formen bloques pero será claramente reconocible. La formación de bloques se puede reducir en el lado codificador en un postproceso; por ejemplo, utilizando patrones de reconstrucción distintos de los que aparecen en el diccionario. Las imágenes de error difuso también pueden ser codificadas sin pérdida, o con pérdida controlada como se describe más arriba, utilizando la codificación genérica.

En [12] se dan más detalles sobre cómo se lleva a cabo esta codificación de semitonos.

0.2.4 Consecuencias de una segmentación inadecuada

Para obtener una codificación óptima, tanto en términos de calidad como de tamaño de fichero, deberá aplicarse la forma de codificación correcta a las regiones apropiadas de las páginas del documento. En esta subcláusula se describen brevemente las consecuencias de los errores cometidos en la segmentación.

Si se utiliza codificación de símbolos con pérdida para un documento que contiene datos de texto y de semitonos, se obtendrá un grado de compresión deficiente. Dependiendo del codificador, la calidad de los datos de semitonos puede ser buena o mala. Aplicando la forma de codificación de símbolos sin pérdida descrita en [5], la calidad visual probablemente no se deteriora.

La codificación genérica con pérdida (utilizando los preprocesos indicados en [11]), aplicada a un documento que contiene datos de símbolos y de semitonos, da de sí una buena calidad y compresión moderada.

El dibujo artístico y las regiones de texto manuscrito se pueden codificar eficazmente haciendo uso de la codificación genérica, pero, dependiendo del codificador, estos tipos de regiones también se pueden codificar de manera satisfactoria con codificación de símbolos.

NORMA INTERNACIONAL

RECOMENDACIÓN UIT-T

TECNOLOGÍA DE LA INFORMACIÓN – CODIFICACIÓN CON PÉRDIDA/SIN PÉRDIDA DE IMÁGENES BINIVEL

1 Alcance

Esta Recomendación | Norma Internacional define métodos de codificación de imágenes binivel y conjuntos de imágenes (documentos de múltiples páginas). Es particularmente adecuada para imágenes binivel que constan de texto y datos (semitonos) producidos por tremolación.

Los métodos definidos permiten la codificación sin pérdida (preservando los bits), la codificación con pérdida y la codificación progresiva. En la codificación progresiva, la primera imagen es con pérdida; las imágenes subsiguientes pueden ser con pérdida o sin pérdida.

La presente Recomendación | Norma Internacional define además los formatos de los ficheros en los que se introducen los datos de imágenes binivel codificados.

2 Referencias normativas

Las siguientes Recomendaciones | Normas Internacionales contienen disposiciones que, mediante referencia hecha en este texto, constituyen disposiciones de esta Recomendación | Norma Internacional. En el momento de la publicación, las ediciones indicadas eran válidas. Todas las Recomendaciones y Normas están sujetas a revisión por lo que se preconiza que los participantes en acuerdos basados en esta Recomendación | Norma Internacional investiguen la posibilidad de aplicar la edición más reciente de las Recomendaciones y Normas enumeradas a continuación. Los miembros de la CEI y de la ISO mantienen registros de las Normas Internacionales actualmente vigentes. El Sector de Normalización de las Telecomunicaciones (TSB) de la UIT mantiene una lista de las Recomendaciones actualmente vigentes.

– Recomendación CCITT T.6 (1988), *Esquemas de codificación facsímil y funciones de control de codificación para los aparatos facsímil del grupo 4.*

– ISO/CEI 8859-1:1988, *Information technology – 8 bit single byte coded graphic character sets – Part 1: Latin alphabet No. 1*.

– ISO/CEI 10646-11:2000, *Information technology – Universal multiple-octet coded character set (UCS) – Architecture and basic multilingual plane*.

3 Términos y definiciones

A los efectos de esta Recomendación | Norma Internacional, se aplican los siguientes términos y definiciones.

3.1 **píxel(s) de plantilla adaptativa**: Píxel(s) especial(es), en una plantilla, cuya ubicación no está fija.

3.2 **agregación**: Unión o fusión de varios símbolos individuales en un nuevo símbolo.

3.3 **imagen binivel**: Formación rectangular de bits.

3.4 **bit**: Dígito binario que representa los valores de **`0`** ó **`1`**.

3.5 **mapa de bits**: Imagen binivel.

3.6 **memoria intermedia, memoria tampón**: Zona de almacenamiento utilizada para retener un mapa de bits.

3.7 **byte**: Ocho bits de datos.

3.8 **operador de combinación**: Operador utilizado para combinar el contenido previo de un mapa de bits con valores nuevos dibujados en ese mapa de bits.

3.9 **sistema de coordenadas**: Sistema de numeración para ubicaciones bidimensionales en el que las ubicaciones se designan mediante dos números, el primero de los cuales aumenta de izquierda a derecha y el segundo, de arriba a abajo.

3.10 **S delta**: Diferencia entre las coordenadas S de dos ejemplares de símbolo sucesivas en una franja no vacía.

3.11 **T delta**: Diferencia entre las coordenadas T de dos franjas no vacías sucesivas.

3.12 **procedimiento de decodificación**: Componente de un decodificador que decodifica un determinado tipo de datos.

3.12.1 **procedimiento de decodificación de enteros**: Procedimiento de decodificación cuya salida es un valor único en cada invocación.

3.12.2 **procedimiento de decodificación aritmética de enteros**: Procedimiento de decodificación de enteros que utiliza la decodificación de entropía aritmética.

3.12.3 **procedimiento de decodificación de región**: Procedimiento de decodificación cuya salida es un mapa de bits.

3.12.4 **procedimiento de decodificación de región genérica**: Procedimiento de decodificación de una región que actúa decodificando píxels individualmente o por pasadas.

3.12.5 **procedimiento de decodificación de región de refinamiento genérica**: Procedimiento de decodificación de una región que actúa modificando un mapa de bits de referencia para producir un mapa de bits de salida.

3.12.6 **procedimiento de decodificación de escala de grises**: Procedimiento de decodificación cuya salida es una imagen en escala de grises.

3.12.7 **procedimiento de decodificación de diccionario de patrones**: Procedimiento de decodificación cuya salida es una lista de patrones.

3.12.8 **procedimiento de decodificación de región de semitonos**: Procedimiento de decodificación de una región que actúa dibujando un conjunto de patrones en un mapa de bits, colocando los patrones de acuerdo con una matriz de semitonos.

3.12.9 **procedimiento de decodificación de tabla Huffman**: Procedimiento de decodificación cuya salida es una tabla Huffman.

3.12.10 **procedimiento de decodificación de región de texto**: Procedimiento de decodificación de una región que actúa dibujando un conjunto de ejemplares de símbolo en un mapa de bits.

3.12.11 **procedimiento de decodificación de diccionario de símbolo**: Procedimiento de decodificación cuya salida es una lista de símbolos.

3.13 **decodificador**: Entidad que puede decodificar un tren binario de acuerdo con la presente Recomendación | Norma Internacional.

3.14 **diccionario**: Lista de mapas de bits.

3.14.1 **diccionario de patrones**: Lista de patrones.

3.14.2 **diccionario de símbolos**: Lista de símbolos.

3.15 **bandera de exportación**: Bit que indica que un símbolo está en la lista de exportación de un diccionario de símbolos.

3.16 **lista de exportación**: Lista de los símbolos de un diccionario de símbolos que se pueden utilizar haciendo referencia a ese diccionario de símbolos.

3.17 **imagen de escala de grises**: Formación rectangular de índices enteros no negativos.

3.18 **píxel de escala de grises**: Elemento de valor entero en una imagen de escala de grises.

3.19 **cuadrícula de semitonos**: Cuadrícula de ubicaciones que especifican dónde se han de dibujar los patrones.

3.20 **clase de altura**: Conjunto de símbolos de un diccionario de símbolos cuyas alturas son todas iguales.

3.21 **altura delta de dos clases de altura**: Diferencia entre la altura de dos clases de altura.

3.22 **anchura delta de dos clases de altura**: Diferencia entre la anchura de dos símbolos de una clase de altura.

3.23 **tabla Huffman**: Conjunto de líneas de tabla que especifican cómo se codifican los valores.

3.24 **codificación sin pérdida**: Método de codificación de datos de forma que los datos decodificados sean idénticos a los datos originales.

3.25 **codificación con pérdida**: Método de codificación de datos de forma que los datos codificados difieran, en teoría de manera insignificante, de los datos originales.

3.26 **ordinal**: Valor utilizado como contador.

3.27 **valor fuera de banda**: Valor no numérico que puede producirse en vez de un entero.

3.28 **patrón**: Mapa de bits producido por un procedimiento de decodificación de diccionario de patrones.

3.29 **píxel**: Elemento cuyo valor es **`0`** ó **`1`** en un mapa de bits.

3.30 **longitud de prefijo**: Longitud del prefijo de código Huffman en una línea de tabla.

3.31 **longitud de gama**: Número de bits de código adicionales en una línea de tabla.

3.32 **mapa de bits de referencia**: Mapa de bits utilizado como plan de referencia durante el procedimiento de decodificación de una región de refinamiento.

3.33 **segmento referenciado**: Otro segmento requerido para decodificar el segmento en curso.

3.34 **región**: Mapa de bits producido por un procedimiento de decodificación de región.

3.35 **segmento**: Encabezamiento de segmento y sus datos de segmento.

3.36 **franja**: Porción con la anchura total o la altura total del sistema de coordenadas de una región de texto.

3.36.1 **franja vacía**: Franja que no contiene la esquina de referencia de ningun ejemplar de símbolo.

3.36.2 **franja no vacía**: Franja que contiene la esquina de referencia de al menos un ejemplar de símbolo.

3.37 **tamaño de franja**: Número de píxels de la dimensión inferior a la máxima posible de una franja.

3.38 **símbolo**: Mapa de bits producido por un procedimiento de decodificación de diccionario de símbolos.

3.39 **ID de símbolo**: Entero utilizado para identificar un símbolo o como índice en una formación de símbolos para extraer el símbolo.

3.40 **ejemplar de símbolo**: Símbolo dibujado, posiblemente con refinamiento, en una determinada ubicación de una región de texto.

3.41 **altura delta de mejora de ejemplar de símbolo**: Diferencia en altura entre el mapa de bits de referencia de un ejemplar de símbolo y el mapa de bits producido por el procedimiento de decodificación de región de refinamiento genérica.

3.42 **anchura delta de mejora de ejemplar de símbolo**: Diferencia en anchura entre el mapa de bits de referencia de un ejemplar de símbolo y el mapa de bits producido por el procedimiento de decodificación de región de refinamiento genérica.

3.43 **X delta de mejora de ejemplar de símbolo**: Diferencia entre las coordenadas X de las esquinas superior izquierda de un mapa de bits de referencia de ejemplar de símbolo y el mapa de bits producido por el procedimiento de decodificación de región de refinamiento genérica.

3.44 **Y delta de mejora de ejemplar de símbolo**: Diferencia entre las coordenadas Y de las esquinas superior izquierda de un mapa de bits de referencia de ejemplar de símbolo y el mapa de bits producido por el procedimiento de decodificación de región de refinamiento genérica.

3.45 **línea de tabla**: Especificación de la codificación de un solo valor o una gama de valores como prefijo de código Huffman seguido por un número fijo de bits de código adicionales.

3.46 **predicción típica**: La predicción típica señala que toda una fila de una región genérica es idéntica a la fila precedente.

3.47 **valor**: Indicador entero o fuera de banda que se decodifica.

4 Símbolos y abreviaturas

NOTA – Debido a los requisitos de la ISO relativos a la nomenclatura, dentro del contexto de la cláusula 4, el término "símbolo" se utiliza para significar un nombre de variable.

4.1 Abreviaturas

A los efectos de esta Recomendación | Norma Internacional se aplican las siguientes siglas:

AT Plantilla adaptativa (*adaptative template*)

EOFB Fin de bloque facsímil (*end-of-facsimile block*)

ID Identificador (*identifier*)

LPS Símbolo menos probable, es decir, valor binario menos probable (*less probable symbol*)

LSB Bit menos significativo (*least significant bit*)

MMR READ modificado modificado (*modified modified READ*)

MPS Símbolo más probable, es decir, valor binario más probable (*more probable symbol*)

MSB Bit más significativo (*most significant bit*)

OOB Fuera de banda (*out-of-band*)

READ Designación de dirección de elemento relativo (*relative element address designate*)

TPGD Predicción típica para codificación de mapa de bits directa genérica (*typical prediction for generic direct bitmap coding*)

TPGR Predicción típica para codificación de mapa de bits de refinamiento genérica (*typical prediction for generic refinement bitmap coding*)

NOTA – El término "símbolo" de las abreviaturas LPS y MPS no se refiere a los símbolos (mapas de bits) de esta Recomendación | Norma Internacional. A pesar de ello, se utilizan las abreviaturas LPS y MPS porque corresponden a la terminología generalmente aceptada en la codificación aritmética.

4.2 Definiciones de símbolos

Los símbolos utilizados en esta Recomendación | Norma Internacional son los indicados a continuación. Se aplica el convenio de que los parámetros de cualquiera de los procedimientos de decodificación utilizados en esta Recomendación | Norma Internacional se indican en **negritas**.

A Intervalo de probabilidad

a Un número real

ARR Una formación

A_1, A_2, A_3, A_4 Píxels de plantilla adaptativa en el procedimiento de decodificación de región genérica

B Byte en curso de datos codificados aritméticamente

B1 Byte de datos codificados aritméticamente que sigue al byte en curso

B_{HC} Mapa de bits colectivo de clase de altura en un procedimiento de decodificación de diccionario de símbolos

B_{HDC} Mapa de bits colectivo de diccionario en un procedimiento de decodificación de diccionario de patrones

B_P Mapa de bits de patrón en un procedimiento de decodificación de diccionario de patrones

B_S Mapa de bits de símbolo en un procedimiento de decodificación de diccionario de símbolos

BM Mapa de bits

BP Puntero hacia el byte B

BPST Valor inicial de BP

C Valor de tren binario en un registro de códigos

Chigh 16 bits de orden alto de C

Clow 16 bits de orden bajo de C

CONTEXT Valores de los píxels de una plantilla utilizada en el procedimiento de decodificación genérica o de decodificación de refinamiento genérica

CT Contador de desplazamientos de renormalización

CURCODE Código Huffman de la línea de tabla en curso en una tabla Huffman

CUREXFLAG Bandera de exportación en curso

CURLEN	Longitud del prefijo de la línea de tabla en curso en una tabla Huffman
CURRANGELOW	Límite inferior de la gama de la línea de tabla en curso en una tabla Huffman
CURS	Coordenada S en curso en un procedimiento de decodificación de región de texto
CURT	Coordenada T del ejemplar de símbolo en curso en relación con la coordenada T de la franja en curso en un procedimiento de decodificación de región de texto
CX	Etiqueta identificadora de un contexto de codificación aritmética
D	Decisión de codificación aritmética
DFS	Diferencia entre las coordenadas S de los primeros ejemplares de dos franjas
DT	Número de franjas vacías entre dos franjas no vacías
DW	Diferencia en anchura entre dos mapas de bits de símbolo en un procedimiento de decodificación de diccionario de símbolos
EXFLAGS	Formación de banderas de exportación
EXINDEX	Índice para la formación EXFLAGS
EXRUNLENGTH	Longitud de un recorrido de valores de banderas de exportación idénticos
FIRSTS	Primera coordenada S de la franja en curso
FIRSTCODE	Primer código asignado a la longitud de un prefijo particular en una tabla Huffman
$GBATX_1$	Ubicación X del píxel de plantilla adaptativa 1 en un procedimiento de decodificación de región genérica
$GBATY_1$	Ubicación Y del píxel de plantilla adaptativa 1 en un procedimiento de decodificación de región genérica
$GBATX_2$	Ubicación X del píxel de plantilla adaptativa 2 en un procedimiento de decodificación de región genérica
$GBATY_2$	Ubicación Y del píxel de plantilla adaptativa 2 en un procedimiento de decodificación de región genérica
$GBATX_3$	Ubicación X del píxel de plantilla adaptativa 3 en un procedimiento de decodificación de región genérica
$GBATY_3$	Ubicación Y del píxel de plantilla adaptativa 3 en un procedimiento de decodificación de región genérica
$GBATX_4$	Ubicación X del píxel de plantilla adaptativa 4 en un procedimiento de decodificación de región genérica
$GBATY_4$	Ubicación Y del píxel de plantilla adaptativa 4 en un procedimiento de decodificación de región genérica
GB	Prefijo utilizado para muchas de las variables asociadas con un procedimiento de decodificación de región (mapa de bits) genérica
GBH	Altura de una región genérica
GBREG	Región producida por un procedimiento de decodificación de región genérica
GBTEMPLATE	Parámetro que indica el número y la disposición de los píxels en una plantilla utilizada en un procedimiento de decodificación de región genérica
GBW	Anchura de una región genérica
GI	Una formación de valores de escala de grises
GR	Prefijo utilizado para muchas de las variables asociadas con un procedimiento de decodificación de región de refinamiento genérica
$GRATX_1$	Ubicación X del píxel de plantilla adaptativa 1 en un procedimiento de decodificación de región de refinamiento genérica
$GRATY_1$	Ubicación Y del píxel de plantilla adaptativa 1 en un procedimiento de decodificación de región de refinamiento genérica
$GRATX_2$	Ubicación X del píxel de plantilla adaptativa 2 en un procedimiento de decodificación de región de refinamiento genérica
$GRATY_2$	Ubicación Y del píxel de plantilla adaptativa 2 en un procedimiento de decodificación de región de refinamiento genérica

GRAY	Valor de escala de grises vigente
GRAYMAX	El mayor valor de escala de grises para el que se da un patrón en un procedimiento de decodificación de diccionario de patrones
GRH	Altura de una región genérica que se codifica con codificación de refinamiento
GRREFERENCE	Mapa de bits de referencia en un procedimiento de decodificación de región de refinamiento genérica
GRREFERENCEDX	Desplazamiento X del mapa de bits de referencia con respecto al mapa de bits que se decodifica en un procedimiento de decodificación de región de refinamiento genérica
GRREFERENCEDY	Desplazamiento Y del mapa de bits de referencia con respecto al mapa de bits que se decodifica en un procedimiento de decodificación de región de refinamiento genérica
GRREG	Región producida por un procedimiento de decodificación de región de refinamiento genérica
GRTEMPLATE	Parámetro que indica el número y la disposición de los píxels en una plantilla utilizada en un procedimiento de decodificación de región genérica con codificación de refinamiento
GRW	Anchura de una región genérica que se codifica con codificación de refinamiento
GS	Prefijo utilizado para muchas de las variables asociadas con un procedimiento de decodificación de imagen de escala de grises
GSBPP	Número de bits por valor de la escala de grises en un procedimiento de decodificación de imagen de escala de grises
GSH	Altura de la imagen de escala de grises en un procedimiento de decodificación de imagen de escala de grises
GSKIP	Máscara indicadora de los valores de la escala de grises que se han de saltar
GSMMR	Indicación de si se utiliza MMR en un procedimiento de decodificación de imagen de escala de grises
GSTEMPLATE	Parámetro indicador del número y la disposición de los píxels en una plantilla utilizada en un procedimiento de decodificación de imagen de escala de grises
GSUSESKIP	Indicación de si deberán saltarse algunos valores de la escala de grises en un procedimiento de decodificación de imagen de escala de grises
GSVALS	Imagen de escala de grises decodificada
GSW	Anchura de la imagen de escala de grises en un procedimiento de decodificación de la imagen de escala de grises
HB	Prefijo utilizado para muchas de las variables asociadas con un procedimiento de decodificación de región (mapa de bits) de semitonos
HBH	Altura de una región de semitonos
HBPP	Número de bits por valor en una formación de valores de escala de grises
HBW	Anchura de una región de semitonos
HCHEIGHT	Altura de la clase de altura vigente en un procedimiento de decodificación de diccionario de símbolos
HCDH	Diferencia en altura entre dos clases de altura en un procedimiento de decodificación de diccionario de símbolos
HCFIRSTSYM	Índice del primer símbolo decodificado en una clase de altura
HCOMBOP	Operador de combinación utilizado en un procedimiento de decodificación de región de semitonos
HD	Prefijo utilizado para muchas de las variables asociadas con un procedimiento de decodificación de región de diccionario de patrones
HDEFPIXEL	Valor por defecto de píxels en una región de semitonos
HDMMR	Indicación de si se utiliza MMR en un procedimiento de decodificación de diccionario de patrones
HDPATS	Formación de patrones producida por un procedimiento de decodificación de diccionario de patrones
HDPH	Altura de los patrones en un diccionario de patrones
HDPW	Anchura de los patrones en un diccionario de patrones

HDTEMPLATE	Identificador de plantilla utilizado para decodificar patrones en un procedimiento de decodificación de diccionario de patrones
HENABLESKIP	Indicación de si se saltan los valores de la escala de grises no necesarios en un procedimiento de decodificación de región de semitonos
HGH	Altura de la imagen de escala de grises en un procedimiento de decodificación de región de semitonos
HGW	Anchura de la imagen de la escala de grises en un procedimiento de decodificación de región de semitonos
HGX	Desplazamiento horizontal de la cuadrícula en un procedimiento de decodificación de región de semitonos
HGY	Desplazamiento vertical de la cuadrícula en un procedimiento de decodificación de región de semitonos
H_I	Altura del mapa de bits de un ejemplar de símbolo
HIGHPREFLEN	Longitud del prefijo de la línea de tabla de gama superior en un tabla Huffman
HMMR	Indicación de si se utiliza codificación MMR en un procedimiento de decodificación de región de semitonos
HNUMPATS	Número de patrones que se pueden utilizar en un procedimiento de decodificación de región de semitonos
HO_I	Altura del mapa de bits original de un ejemplar de símbolo que contiene información de refinamiento
HPATS	Formación de patrones utilizados en una región de semitonos
HPH	Altura de cada patrón en una región de semitonos
HPW	Anchura de cada patrón en una región de semitonos
HRX	Coordinada horizontal de un vector de cuadrícula de semitonos
HRY	Coordenada vertical de un vector de cuadrícula de semitonos
HSKIP	Máscara indicadora de los valores de la escala de grises que se han de saltar
HT	Prefijo utilizado para muchas de las variables asociadas con un procedimiento de decodificación de tabla Huffman
HTEMPLATE	Parámetro indicador del número y la disposición de los píxels en una plantilla utilizada en un procedimiento de decodificación de región de semitonos
HTHIGH	Valor superior al mayor valor representado por cualquier línea de tabla normal en una tabla Huffman
HTLOW	Valor más bajo representado por cualquier línea de tabla normal en una tabla Huffman
HTOFFSET	Desplazamiento de gama de una línea de tabla cuando se decodifica utilizando una tabla Huffman
HTOOB	Indicación de si una tabla Huffman puede producir el valor fuera de banda (OOB)
HTPS	Longitud del campo de prefijo codificado en una línea de tabla de una tabla Huffman
HTREG	Región producida por un procedimiento de decodificación de región de semitonos
HTRS	Longitud del campo de gama codificada en una línea de tabla de una tabla Huffman
HTVAL	Valor decodificado utilizando una tabla Huffman
I	Formación, indizada por CX, de los índices de las estimaciones de probabilidad adaptativa
I	Índice de formación
IAAI	Procedimiento de decodificación aritmética de enteros utilizado para decodificar el número de ejemplares de símbolo en una agregación
IADH	Procedimiento de decodificación aritmética de enteros utilizado para decodificar la diferencia en altura entre dos clases de altura
IADS	Procedimiento de decodificación aritmética de enteros utilizado para decodificar la coordenada S del segundo ejemplar y ejemplares de símbolos subsiguientes en una franja
IADT	Procedimiento de decodificación aritmética de enteros utilizado para decodificar la coordenada T del segundo ejemplar y ejemplares de símbolo subsiguientes en una franja

IADW	Procedimiento de decodificación aritmética de enteros utilizado para decodificar la diferencia de anchura entre dos símbolos en una clase de altura
IAEX	Procedimiento de decodificación aritmética de enteros utilizado para decodificar banderas de exportación
IAFS	Procedimiento de decodificación aritmética de enteros utilizado para decodificar la coordenada S del primer ejemplar de símbolo en una franja
IAID	Procedimiento de decodificación aritmética de enteros utilizado para decodificar los ID símbolo de ejemplares de símbolo
IARDH	Procedimiento de decodificación aritmética de enteros utilizado para decodificar la altura delta (diferencia de) de refinamientos de ejemplares de símbolo
IARDW	Procedimiento de decodificación aritmética de enteros utilizado para decodificar la anchura delta (diferencia de) de refinamientos de ejemplares de símbolo
IARDX	Procedimiento de decodificación de entero aritmética utilizado para decodificar los valores de delta (diferencia de) X de refinamientos de ejemplares de símbolo
IARDY	Procedimiento de decodificación aritmética de enteros utilizado para decodificar los valores de delta (diferencia de) Y de refinamientos de ejemplares de símbolo
IARI	Procedimiento de decodificación aritmética de enteros utilizado para decodificar el bit R_I de ejemplares de símbolo
IAIT	Procedimiento de decodificación aritmética de enteros utilizado para decodificar la coordenada T de los ejemplares de símbolo en una franja
IB_I	Mapa de bits de un ejemplar de símbolo
IBO_I	Mapa de bits original de un ejemplar de símbolo que contiene información de refinamiento
ID_I	ID símbolo de un ejemplar de símbolo
IDS	Valor de delta S para un ejemplar de símbolo en un procedimiento de decodificación de región de texto
J	Índice de formación
K	Ordinal para un segmento referenciado
LENCOUNT	Histograma de longitudes de prefijo en una tabla Huffman
LENMAX	Longitud de prefijo máxima en una tabla Huffman
LOGSBSTRIPS	Logaritmo en base 2 del tamaño de franja utilizado para codificar una región de texto
LOWPREFLEN	Longitud del prefijo de la línea de tabla de gama inferior en una tabla Huffman
LTP	Indicación de si la línea en curso se codifica explícitamente en un procedimiento de decodificación de región genérica o en un procedimiento de decodificación de región de refinamiento genérica
m_g	Índice horizontal para el valor de escala de grises vigente
MMR	Indicación de si se utiliza codificación MMR en un procedimiento de decodificación de región genérica
MPS	Formación, indizada por CX, de los valores binarios vigentes más probables
NINSTANCES	Contador de ejemplares de símbolo
n_g	Índice vertical para el valor de escala de grises vigente
NLPS	Próximo índice para una renormalización de LPS
NMPS	Próximo índice para una renormalización de MPS
NSYMSDECODED	Número de símbolos decodificados hasta ahora en un procedimiento de decodificación de diccionario de símbolos
NTEMP	Número de líneas de tabla de una tabla Huffman
OOB	Un valor fuera de banda
P	Página con la que está asociado un segmento
PREFLEN	Formación de longitudes de prefijo que representan las líneas de tabla de una tabla Huffman
Qe	Estimación de la probabilidad de LPS

r	Bandera de retención de segmento
R	Número de segmentos a los que hace referencia algún segmento
RANGELEN	Formación de las longitudes de las gamas de las líneas de tabla de una tabla Huffman
RANGELOW	Formación de los límites inferiores de las gamas de las líneas de tabla de una tabla Huffman
RA_1, RA_2	Píxels de plantilla adaptativa en el procedimiento de decodificación de región de refinamiento genérica
RDH_I	Altura delta de un mapa de bits de refinamiento de ejemplar de símbolo
RDW_I	Anchura delta de un mapa de bits de refinamiento ejemplar de símbolo
RDX_I	Desplazamiento X de un refinamiento de ejemplar de símbolo
RDY_I	Desplazamiento Y de un refinamiento de ejemplar de símbolo
REFAGGNINST	Número de ejemplares de símbolo en una agregación
R_I	Bit indicador de si está presente información de refinamiento para un ejemplar de símbolo
REFCORNER	Esquina de un mapa de bits de ejemplar de símbolo que se ha de utilizar como referencia en un procedimiento de decodificación de región de texto
S	Una de las coordenadas del sistema de coordenadas utilizado en un procedimiento de decodificación de región de texto
S_I	Coordenada S de un ejemplar de símbolo
SB	Prefijo utilizado para muchas de las variables asociadas con un procedimiento de decodificación de región (mapa de bits) de símbolos
SBDSOFFSET	Desplazamiento para los valores de delta S codificados en una región de texto
SBCOMBOP	Operador de combinación utilizado en un procedimiento de decodificación de región de texto
SBDEFPIXEL	Valor por defecto de píxels en un región de texto
SBH	Altura de una región de texto
SBHUFF	Indicación de si se utiliza codificación Huffman en un procedimiento de decodificación de región de texto
SBHUFFDS	Tabla Huffman utilizada para decodificar la coordenada S de ejemplares de símbolo subsiguientes en una franja
SBHUFFDT	Tabla Huffman utilizada para decodificar la diferencia entre coordenadas T de franjas no vacías
SBHUFFFS	Tabla Huffman utilizada para decodificar la coordenada S del primer ejemplar de símbolo en una franja
SBHUFFRDH	Tabla Huffman utilizada para decodificar la diferencia entre la altura de un símbolo y la altura de un mapa de bits de ejemplar de símbolo codificado con refinamiento
SBHUFFRDW	Tabla Huffman utilizada para decodificar la diferencia entre la anchura de un símbolo y la anchura de un mapa de bits de ejemplar de símbolo codificado con refinamiento
SBHUFFRDX	Tabla Huffman utilizada para decodificar la diferencia entre la coordenada X de un ejemplar de símbolo y la coordenada X de un mapa de bits codificado con refinamiento
SBHUFFRDY	Tabla Huffman utilizada para decodificar la diferencia entre la coordenada Y de un ejemplar de símbolo y la coordenada Y de un mapa de bits de ejemplar de símbolo codificado con refinamiento
SBHUFFRSIZE	Tabla Huffman utilizada para decodificar el tamaño de los datos de un mapa de bits de refinamiento de ejemplar de símbolo
SBNUMINSTANCES	Número de ejemplares de símbolo en una región de texto
SBNUMSYMS	Número de símbolos que se pueden utilizar en una región de texto
$SBRATX_1$	Ubicación X del píxel de plantilla adaptativa RA_1 en un procedimiento de decodificación de región de texto
$SBRATY_1$	Ubicación Y del píxel de plantilla adaptativa RA_1 en un procedimiento de decodificación de región de texto

$SBRATX_2$	Ubicación X del píxel de plantilla adaptativa RA_2 en un procedimiento de decodificación de región de texto
$SBRATY_2$	Ubicación Y del píxel de plantilla adaptativa RA_2 en un procedimiento de decodificación de región de texto
SBREFINE	Indicación de si se utiliza codificación de mejora en un procedimiento de decodificación de región de texto
SBREG	Región producida por un procedimiento de decodificación de región de texto
SBRTEMPLATE	Identificador de plantilla para codificación de mejora de mapa de bits en un procedimiento de decodificación de región de texto
SBSTRIPS	Altura de las franjas de un ejemplar de símbolo
SBSYMCODELEN	Longitud de los códigos de símbolos usados en IAID
SBSYMCODES	Formación de códigos de longitud variable que identifican símbolos individuales
SBSYMS	Formación de símbolos utilizados en una región de texto
SBW	Anchura de una región de texto
SD	Prefijo utilizado para muchas de las variables asociadas con un procedimiento de decofidicación de región de diccionario de símbolos
$SDATX_1$	Ubicación X del píxel de plantilla adaptativa A_1 en un procedimiento de decodificación de diccionario de símbolos
$SDATY_1$	Ubicación Y del píxel de plantilla adaptativa A_1 en un procedimiento de decodificación de diccionario de símbolos
$SDATX_2$	Ubicación X del píxel de plantilla adaptativa A_2 en un procedimiento de decodificación de diccionario de símbolos
$SDATY_2$	Ubicación Y del píxel de plantilla adaptativa A_2 en un procedimiento de decodificación de diccionario de símbolos
$SDATX_3$	Ubicación X del píxel de plantilla adaptativa A_3 en un procedimiento de decodificación de diccionario de símbolos
$SDATY_3$	Ubicación Y del píxel de plantilla adaptativa A_3 en un procedimiento de decodificación de diccionario de símbolos
$SDATX_4$	Ubicación X del píxel de plantilla adaptativa A_4 en un procedimiento de decodificación de diccionario de símbolos
$SDATY_4$	Ubicación Y del píxel de plantilla adaptativa A_4 en un procedimiento de decodificación de diccionario de símbolos
SDEXSYMS	Símbolos exportados desde un diccionario de símbolos
SDHUFF	Indicación de si se utiliza codificación Huffman en un procedimiento de decodificación de diccionario de símbolos
SDHUFFAGGINST	Tabla Huffman utilizada para decodificar el número de ejemplares de símbolo de una agregación en un procedimiento de decodificación de diccionario de símbolos
SDHUFFDH	Tabla Huffman utilizada para decodificar la diferencia en altura entre dos clases de altura en un procedimiento de decodificación de diccionario de símbolos
SDHUFFDW	Tabla Huffman utilizada para decodificar la diferencia de anchura entre dos símbolos en un procedimiento de decodificación de diccionario de símbolos
SDHUFFBMSIZE	Tabla Huffman utilizada para decodificar el tamaño de un mapa de bits colectivo de clase altura en un procedimiento de decodificación de diccionario de símbolos
SDINSYMS	Formación de símbolos utilizada como parámetro de un procedimiento de decodificación de diccionario de símbolos
SDNEWSYMS	Símbolos decodificados en un diccionario de símbolos
SDNEWSYMWIDTHS	Anchuras de los símbolos decodificados en un diccionario de símbolos
SDNUMEXSYMS	Número de símbolos exportados desde un diccionario de símbolos
SDNUMINSYMS	Número de símbolos en la formación que se utiliza como parámetro en un procedimiento de decodificación de diccionario de símbolos

SDNUMNEWSYMS	Número de símbolos generados en un diccionario de símbolos
SDREFAGG	Indicación de si se utiliza codificación de refinamiento y agregación en un procedimiento de decodificación de diccionario de símbolos
$SDRATX_1$	Ubicación X del píxel de plantilla adaptativa RA_1 en un procedimiento de decodificación de diccionario de símbolos
$SDRATY_1$	Ubicación Y del píxel de plantilla adaptativa RA_1 en un procedimiento de decodificación de diccionario de símbolos
$SDRATX_2$	Ubicación X del píxel de plantilla adaptativa RA_2 en un procedimiento de decodificación de diccionario de símbolos
$SDRATY_2$	Ubicación Y del píxel de plantilla adaptativa RA_2 en un procedimiento de decodificación de diccionario de símbolos
SDRTEMPLATE	Identificador de plantilla para codificación de mejora de mapas de bits en un procedimiento de decodificación de diccionario de símbolos
SDTEMPLATE	Identificador de plantilla utilizado para decodificar mapas de bits de símbolo en un procedimiento de decodificación de diccionario de símbolos
SKIP	Máscara de los píxels que se han de saltar durante la decodificación de una región genérica
SLTP	Valor binario indicador de si la línea en curso fue predicha típicamente y la línea previa no lo fue, o viceversa
STRIPT	Coordenada T numéricamente más pequeña en la franja en curso
SWITCH	Indicación de si se han conmutado MPS y LPS en una renormalización de LPS
SYMWIDTH	Anchura del mapa de bits en curso en un procedimiento de decodificación de diccionario de símbolos
T	Una de las coordenadas del sistema de coordenadas utilizado en un procedimiento de decodificación de región de texto
TEMPC	Registro temporal en el codificador MQ
T_I	Coordenada T de un ejemplar de símbolo
TOTWIDTH	Anchura total de los mapas de bits en una clase de altura
TPGDON	Indicación de si se utiliza predicción típica en un procedimiento de decodificación de región genérica
TPGRON	Indicación de si se utiliza predicción típica en un procedimiento de decodificación de refinamiento de región genérica
TPGRPIX	Indicación de si el píxel en curso ha de ser decodificado implícitamente utilizando una predicción TPGR
TPGRVAL	Valor del píxel en curso predicho con TPGR
TRANSPOSED	Indicación de si las coordenadas del ejemplar de símbolo están transpuestos en un procedimiento de decodificación de región de texto
USESKIP	Indicación de si deben saltarse algunos píxels al decodificar una región genérica
V1	Un valor binario
V2	Un valor binario
W_I	Anchura de un mapa de bits de ejemplar de símbolos
WO_I	Anchura del mapa de bits original de un ejemplar de símbolos que contiene información de refinamiento
x	Coordenada horizontal de una ubicación en una cuadrícula de semitonos
X	Coordenada horizontal de un píxel en un mapa de bits
y	Coordenada vertical de una ubicación en una cuadrícula de semitonos
Y	Coordenada vertical de un píxel en un mapa de bits

4.3 Definiciones de operadores

Se definen los siguientes operadores:

OR — Si V1 y V2 son dos valores binarios, entonces V1 OR V2 es igual a **0** si tanto V1 como V2 son **0**. Es igual a **1** si bien V1 o bien V2 es **1**. Si V1 y V2 son dos valores enteros, entonces es el resultado de la aplicación lógica de OR.

AND — Si V1 y V2 son dos valores binarios, entonces V1 AND V2 es igual a **0** si bien V1 o bien V2 es **0**. Es igual a **1** si tanto V1 como V2 son **1**. Si V1 y V2 son valores enteros, entonces es el resultado de la aplicación lógica de AND.

XOR — Si V1 y V2 son dos valores enteros, entonces V1 XOR V2 es igual a **0** si V1 y V2 son iguales. Es igual a **1** si V1 y V2 difieren. Si V1 y V2 son dos valores enteros, entonces es el resultado de la aplicación lógica de XOR.

XNOR — Si V1 y V2 son dos valores binarios, entonces V1 XNOR V2 es igual a **0** si V1 y V2 difieren. Es igual a **1** si V1 y V2 son iguales.

REPLACE — Si V1 y V2 son dos valores binarios, entonces V1 REPLACE V2 es igual a V2.

NOT — Si V1 es un valor binario, entonces NOT V1 es **1** si V1 es **0**, y es **0** si V1 es **1**.

min — Si x e y son números, entonces min (x, y) es el menor de x e y.

max — Si x e y son números, entonces max (x, y) es el mayor de x e y.

$\lfloor\ \rfloor$ — Si a es un número, entonces $\lfloor a \rfloor$ es el mayor entero menor o igual a a.

$\lceil\ \rceil$ — Si a es un número, entonces $\lceil a \rceil$ es el menor entero mayor o igual que a.

<< — Si V1 y V2 son dos enteros, entonces V1 << V2 es el valor obtenido desplazando el valor de V1 hacia la izquierda en V2 bits, rellenando los bits V2 situados más a la derecha del valor nuevo con **0**.

>> — Si V1 y V2 son dos enteros, entonces V1 >> V2 es el valor obtenido desplazando el valor de V1 hacia la derecha en V2 bits, rellenando los bits V2 situados más a la izquierda del nuevo valor con **0**.

$>>_A$ — Si V1 y V2 son dos enteros, entonces V1 $>>_A$ V2 es el valor obtenido desplazando el valor de V1 hacia la derecha en V2 bits, rellenando los bits V2 situados más a la izquierda del valor nuevo con **0** si V1 es no negativo y **1** si V1 es negativo.

5 Convenios

5.1 Convenios tipográficos

Todos los nombres de parámetros se dan en **negritas**.

5.2 Notación binaria

Los dos valores binarios se indican como **0** y **1**.

5.3 Notación hexadecimal

El prefijo `0x` indica que el valor que sigue se ha de interpretar como un número hexadecimal (en base 16).

EJEMPLO – El valor `0x6A` es igual al valor decimal 106.

5.4 Sintaxis de valores enteros

5.4.1 Empaquetado de bits

Los bits se empaquetan en bytes empezando en el bit más significativo. Si un decodificador está leyendo una secuencia de bits de un tren binario, leerá primero el bit más significativo del primer byte, después el siguiente bit más significativo, y así sucesivamente, y a continuación procederá con el siguiente byte.

EJEMPLO – La secuencia de bytes `0x2F 0x05 0xC1`, si se interpreta como una secuencia de bits, es la secuencia

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 Valores multibytes

Los valores multibytes deberán interpretarse aplicando la regla de "primero el más significativo": el primer byte de cada valor es el más significativo, y el último byte es el menos significativo.

EJEMPLO – La secuencia de bytes `0x01 0x5C 0x99 0xFA`, si se interpreta como un valor de cuatro bytes, representa el valor `0x015C99FA`.

5.4.3 Numeración de los bits

El bit menos significativo de cualquier valor se numera como bit 0. Para un valor de un byte, el bit más significativo se numera como bit 7; para un valor de dos bytes, el bit más significativo se numera como bit 15; para un valor de cuatro bytes, el bit más significativo se numera como bit 31.

5.4.4 Signos

A menos que se especifique otra cosa, todos los valores multibits se tratarán como valores sin signo. Cuando un valor tenga que ser tratado como número con signo, se interpretará en la forma de complemento a dos.

5.5 Notación de formaciones y convenios

Las formaciones se numeran empezando a partir de cero.

EJEMPLO – Una formación ARR que contiene doce elementos es como sigue:

ARR[0], ARR[1], . . . , ARR[11]

5.6 Imagen y convenios de mapas de bits

Un mapa de bits es una formación rectangular. Cada elemento de esa formación tiene el valor **0** ó **1**. Un elemento de un mapa de bits se denomina píxel.

NOTA 1 – A lo largo de esta Recomendación | Norma Internacional, los píxels de los mapas de bits se tratan como si tuvieran los valores **0** ó **1**. En la mayoría de las aplicaciones de la presente Recomendación | Norma Internacional, la propia aplicación seleccionará alguna interpretación de esos dos valores. Una interpretación típica de estos píxels es que **0** representa blanco o plano de fondo, y **1** representa negro, o primer plano. Sin embargo, no es ése el caso en la presente Recomendación | Norma Internacional y las aplicaciones pueden hacer libremente otras interpretaciones de esos valores.

Los términos "izquierdo", "derecho", "superior", "inferior", "anchura" y "altura", se aplican a menudo a los mapas de bits. No se refieren a ningún aspecto físico del mapa de bits: si un mapa de bits se imprime en papel, se puede imprimir con su borde "izquierdo" a lo largo de cualquier borde del papel. En esta Recomendación | Norma Internacional se utilizan esos términos para referirse a los cuatro bordes del mapa de bits que se muestra en la figura 2.

A un píxel de un mapa de bits se hace referencia mediante un par de coordenadas X e Y, que a veces se escriben de la siguiente forma: (X, Y). La ubicación (0, 0) representa el píxel situado en la esquina superior izquierda. La coordenada X aumenta hacia la derecha y la coordenada Y aumenta en sentido descendente.

Si *BM* es un mapa de bits, el píxel cuyas coordenadas son X e Y se indica mediante *BM*[X, Y].

NOTA 2 – Estos convenios tienen por objeto facilitar la descripción de las operaciones en que intervienen mapas de bits, y no implican ninguna característica física de la imagen representada por el mismo.

6 Procedimientos de decodificación

6.1 Introducción a los procedimientos de decodificación

Esta Recomendación | Norma Internacional utiliza varios procedimientos de decodificación diferentes para diferentes tipos de datos. Cada uno de esos procedimientos produce como salida un determinado tipo de datos. El procedimiento de decodificación de región genérica, el procedimiento de decodificación de región de refinamiento genérica, el procedimiento de decodificación de región de semitonos y los procedimientos de decodificación de región de texto producen como salida regiones todos ellos. El procedimiento de decodificación de diccionario de símbolos produce como salida una formación de símbolos. El procedimiento de decodificación de diccionario de patrones genera como salida, de manera similar, una formación de mapas de bits de casillas de semitonos.

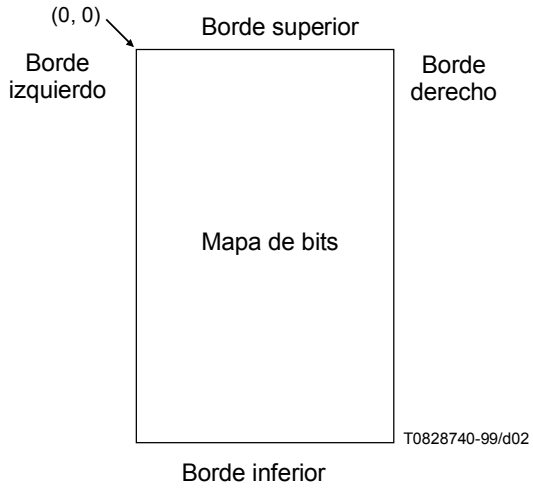


Figura 2 – Los cuatro bordes de un mapa de bits

Los diversos procedimientos de decodificación de región funcionan de maneras diferentes:

- El procedimiento de decodificación de región genérica decodifica un mapa de bits, tratándolo simplemente como una formación de píxels binarios.
- El procedimiento de decodificación de refinamiento de región genérica decodifica un mapa de bits tratándolo como una formación de píxels binarios, pero codifica cada píxel con respecto a algún mapa de bits de referencia.
- El procedimiento de decodificación de región de texto decodifica un mapa de bits dibujando en él un conjunto de símbolos, aplicando posiblemente el procedimiento de decodificación de región de refinamiento genérica a cada uno de ellos.
- El procedimiento de decodificación de región de semitonos decodifica un mapa de bits colocando en él un conjunto de patrones, en ubicaciones especificadas por una cuadrícula de semitonos.

Cada procedimiento de decodificación se especifica en términos de número de parámetros y una secuencia de operaciones, que dependen de los valores de los parámetros. Los parámetros los suministra el procedimiento de decodificación para cada invocación, y el mismo procedimiento de decodificación puede ser invocado múltiples veces en el transcurso de la decodificación de un tren binario, con parámetros diferentes cada vez.

Algunos de los parámetros del procedimiento de decodificación no se utilizan en determinadas circunstancias, dependiendo normalmente de los valores de otros parámetros. En tales casos, no es preciso especificar ningún valor para esos parámetros no utilizados.

En esta cláusula, en las cláusulas que siguen y en los anexos normativos, se imponen algunas restricciones al tren binario que se decodifica.

EJEMPLO 1 – En 7.3, algunos tipos de segmentos se describen como "Reservado; no se debe utilizar."

EJEMPLO 2 – En 7.4.2.1.1, si el campo **SDHUFF** es **0**, el campo **SDHUFFDH** debe contener el valor **0**.

Esas restricciones deberán interpretarse en el sentido de que el comportamiento de un decodificador que encuentra un tren binario que no cumple la restricción está indefinido, y queda fuera del ámbito de aplicación de la presente Recomendación | Norma Internacional.

NOTA – Esto significa que si un decodificador encuentra un tren binario que no satisface las restricciones, puede tomar cualquier medida: puede abandonar y abortar la decodificación; puede ignorar el error e intentar continuar; interpretar el error y cambiar su comportamiento (por ejemplo, utilizando el error para tratar de facilitar la recuperación tras futuros errores); y así sucesivamente.

6.2 Procedimiento de decodificación de región genérica

6.2.1 Descripción general

Este procedimiento de decodificación se utiliza para decodificar una formación rectangular de valores **0** ó **1**, que se codifican a razón de un píxel cada vez (es decir, se utiliza para decodificar un mapa de bits aplicando codificación genérica simple). El procedimiento de decodificación modifica también una formación de información de probabilidades que puede ser utilizada por otras invocaciones de este procedimiento de decodificación de región genérica.

El procedimiento de decodificación de región genérica se puede basar en la codificación secuencial de píxels de imagen utilizando la codificación aritmética especificada en el anexo E y una plantilla para determinar el estado de la codificación. Esta técnica fue aplicada en la Rec. UIT-T T.82 | ISO/CEI 11544 (JBIG). En 6.2.5 se describe este tipo de decodificación.

De manera alternativa, para mayor velocidad pero con menor compresión, el procedimiento de decodificación de región genérica se puede basar en la codificación Huffman de pasadas de píxels. Esta técnica se utilizó en el algoritmo MMR (READ modificado modificado) descrito en la Recomendación UIT-T T.6 (G4). En 6.2.6 se describe este tipo de decodificación.

6.2.2 Parámetros de entrada

En el cuadro 2 se muestran los parámetros de este procedimiento de decodificación.

Cuadro 2 – Parámetros del procedimiento de decodificación de región genérica

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
MMR	Entero	1	No	Indica si se utiliza codificación MMR
GBW	Entero	32	No	La anchura de la región
GBH	Entero	32	No	La altura de la región
GBTEMPLATE	Entero	2	No	El identificador de la plantilla[a)]
TPGDON	Entero	1	No	Indica si se utiliza predicción típica[a)]
USESKIP	Entero	1	No	Indica si deberán saltarse algunos píxels en la decodificación[a)]
SKIP	Mapa de bits			Un mapa de bits que indica qué píxels deberán saltarse. GBW píxels de ancho, GBH píxels de alto[c)]
$GBATX_1$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa A_1[a)]
$GBATY_1$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa A_1[a)]
$GBATX_2$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa A_2[b)]
$GBATY_2$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa A_2[b)]
$GBATX_3$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa A_3[b)]
$GBATY_3$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa A_3[b)]
$GBATX_4$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa A_4[b)]
$GBATY_4$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa A_4[b)]

a) No se utiliza si **MMR = 1**
b) No se utiliza si **MMR = 1** o **GBTEMPLATE** ≠ 0
c) No se utiliza si **USESKIP = 0** o **MMR = 1**

6.2.3 Valor de retorno

En el cuadro 3 se muestra la variable cuyo valor es el resultado de este procedimiento de decodificación.

Cuadro 3 – Valor de retorno del procedimiento de decodificación de región genérica

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
GBREG	Mapa de bits			El mapa de bits de la región decodificada

6.2.4 Variables utilizadas en la decodificación

En el cuadro 4 se muestran las variables utilizadas por este procedimiento de decodificación.

Cuadro 4 – Variables utilizadas en el procedimiento de decodificación de región genérica

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
LTP	Entero	1	No	Indica si la línea de imagen en curso se codifica explícitamente[a)]
SLTP	Entero	1	No	Indica si el valor LTP de la línea en curso difiere del valor LTP de la línea previa[a)]
CONTEXT	Entero	16	No	Los valores de los píxels de la plantilla[a)]
a) No se utiliza si **MMR** = **1.**				

6.2.5 Decodificación utilizando una plantilla y codificación aritmética

6.2.5.1 Descripción general

Si **MMR** es **0**, el procedimiento de decodificación de región genérica se basa en la codificación aritmética con una plantilla para determinar el estado de la codificación. En lo que queda de 6.2.5 se describe esta forma de decodificar, que sólo se aplica cuando **MMR** es **0**.

6.2.5.2 Orden de codificación y convenios de bordes

El algoritmo de codificación se aplica de forma iterativa a lo largo del mapa de bits en el orden de exploración raster, es decir, por filas de arriba a abajo y, dentro de cada columna, de izquierda a derecha. Durante el procesamiento del píxel objetivo en curso se hará referencia a algunos píxels en relación espacial fija con el píxel objetivo.

Cerca de los bordes del mapa de bits, esas referencias vecinas pueden no hallarse en el mapa de bits real. La regla para satisfacer las referencias fuera de límites es como sigue:

- Todos los píxels que se hallen fuera de los límites del mapa de bits real tendrán el valor **0**.

6.2.5.3 Plantillas fijas

Una plantilla define una vecindad en torno al píxel que se ha de codificar. Los valores de los píxels de esa vecindad definen un contexto. Cada contexto tiene su propia estimación de probabilidad adaptativa utilizada por el codificador aritmético (véase el anexo E). Aunque una plantilla es un patrón geométrico de píxels, se dice que los píxels de la plantilla toman valores cuando la plantilla está alineada con una determinada parte de la imagen.

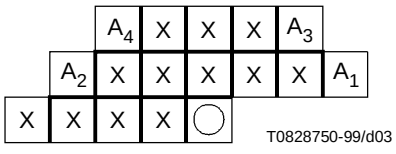


Figura 3 – Plantilla cuando GBTEMPLATE = 0, mostrando los píxels AT en sus ubicaciones nominales

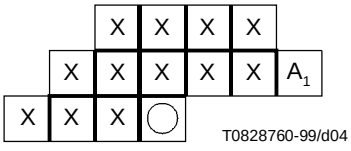


Figura 4 – Plantilla cuando GBTEMPLATE = 1, mostrando el píxel AT en su ubicación nominal

La figura 3 muestra la plantilla que será utilizada cuando **GBTEMPLATE** sea 0. La figura 4 muestra la plantilla que será utilizada cuando **GBTEMPLATE** sea 1. La figura 5 muestra la plantilla que será utilizada cuando **GBTEMPLATE** sea 2. La figura 6 muestra la plantilla que será utilizada cuando **GBTEMPLATE** sea 3. En cada una de estas figuras, el píxel señalado mediante un círculo corresponde al píxel que se ha de codificar y no forma parte de la plantilla. Los píxels marcados con una 'X' corresponden a píxels ordinarios de la plantilla. Los píxels indicados mediante A_1-A_4 son píxels especiales de la plantilla. Se les denomina "adaptativos" o píxels AT. Son píxels especiales en el sentido de que sus ubicaciones no son fijas y pueden ser situados en diferentes ubicaciones. Véase en 6.2.5.4 una descripción de los píxels AT. Las indicaciones A_1-A_4 se refieren a los píxels AT 1 a 4. Las ubicaciones reales de los píxels se especifican como parámetros de este procedimiento de codificación; las figuras 3 a 6 muestran las ubicaciones nominales de estos píxels AT para cada plantilla.

Los valores de los píxels de la plantilla serán combinados para formar un contexto. Cada píxel de la plantilla (incluidos los píxels adaptativos) deberá corresponder a un bit específico del contexto, aunque los píxels de la plantilla se pueden asignar a bits del contexto en cualquier orden. Puesto que la plantilla contiene hasta 16 píxels, los contextos pueden tomar hasta 65536 valores diferentes. El contexto se utilizará para identificar la estimación de probabilidad adaptativa que será utilizada por el codificador aritmético para codificar el píxel que ha de ser codificado (véase el anexo E).

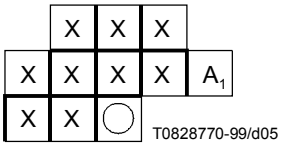


Figura 5 – Plantilla cuando GBTEMPLATE = 2, mostrando el píxel AT en su ubicación nominal

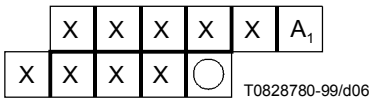


Figura 6 – Plantilla cuando GBTEMPLATE = 3, mostrando el píxel AT en su ubicación nominal

NOTA 1 – Una regla práctica consiste en utilizar plantillas grandes para mapas de bits grandes. Así pues, un semitono periódico de tamaño completo deberá codificarse con la plantilla de 16 píxels y los mapas de bits menores, tales como los mapas de bits de símbolo habituales, deberán codificarse con una de las plantillas de 10 píxels. En algunos casos conviene una plantilla intermedia, por requisitos de funcionamiento o de la memoria del decodificador; si tal cosa ocurre, deberá utilizarse la plantilla de 13 píxels. También es posible generar otras plantillas situando uno o más de los píxels AT por encima de un píxel de plantilla ordinario, con lo que se fija su valor.

NOTA 2 – Las plantillas de 10 píxels son las utilizadas en la Rec. UIT-T T.82 | ISO/CEI 11544 (JBIG). La velocidad de ejecución del programa informático es algo superior con la plantilla de dos líneas que con cualquiera de las plantillas de tres líneas. Para la mayoría de las imágenes, la plantilla de tres líneas y 10 píxels produce una compresión superior que la de la plantilla de dos líneas y 10 píxels.

6.2.5.4 Píxels de plantilla adaptativa

Al codificar la imagen, deberá permitirse a la plantilla cambiar según la manera restringida que se describe en esta subcláusula.

Los píxels a los que se permite cambiar se denominan píxels AT. Sus ubicaciones nominales vienen indicadas por 'A_1', 'A_2', 'A_3' y 'A_4' en las figuras 3, 4, 5 y 6. Se señala que algunas plantillas tienen menos de cuatro píxels AT. Por lo general, un píxel AT se puede situar en cualquier lugar del campo mostrado en la figura 7, sin incluir el píxel en curso. Por ello, es posible utilizar un tamaño de plantilla efectivo de 15, 14, 13, 12 ó 9 píxels haciendo que la ubicación desplazada del píxel AT se superponga a un píxel de plantilla ordinario. Las ubicaciones reales de los píxels AT para cualquier invocación de este procedimiento de decodificación se especifican como parámetros del procedimiento de decodificación. La ubicación del píxel A_1 viene dada por (**GBATX$_1$**, **GBATY$_1$**). Si **GBTEMPLATE** es 0:

- la ubicación del píxel A_2 viene dada por (**GBATX$_2$**, **GBATY$_2$**);
- la ubicación del píxel A_3 viene dada por (**GBATX$_3$**, **GBATY$_3$**);
- y la ubicación del píxel A_4 viene dada por (**GBATX$_4$**, **GBATY$_4$**).

NOTA 1 – Algunos perfiles pueden limitar las ubicaciones de los píxels AT a una gama más reducida que la que se muestra en la figura 7.

NOTA 2 – Los índices de los píxels AT de la figura 3 corresponden al buen funcionamiento esperado. Si sólo se desplaza un píxel AT de su ubicación nominal, conviene desplazar A_4. A continuación hay que desplazar el píxel A_3, y así sucesivamente.

NOTA 3 – En el cuadro 5 se muestran las ubicaciones nominales de los píxels AT. Deberán utilizarse esas ubicaciones a menos que otras distintas mejoren la calidad de la compresión. Algunos perfiles pueden limitar las ubicaciones de píxels AT a esas ubicaciones nominales solamente.

NOTA 4 – Si la ubicación de un píxel AT se superpone a la ubicación de cualquier píxel de plantilla ordinario, se puede ignorar el valor del píxel AT (ya que repite otro valor). Así se reducen las necesidades de memoria del decodificador, porque no se pueden producir todos los valores CX. Sin embargo, cuando la TPGD está habilitada (**TPGDON = 1**), se utiliza el contexto empleado para codificar el valor SLTP, con independencia de si los píxels AT se superponen a píxels de plantilla ordinarios. Esto significa que todavía pueden producirse contextos en los que el valor de un píxel AT difiera del valor de un píxel de plantilla ordinario, pero sólo para SLTP cuando la TPGD esté habilitada.

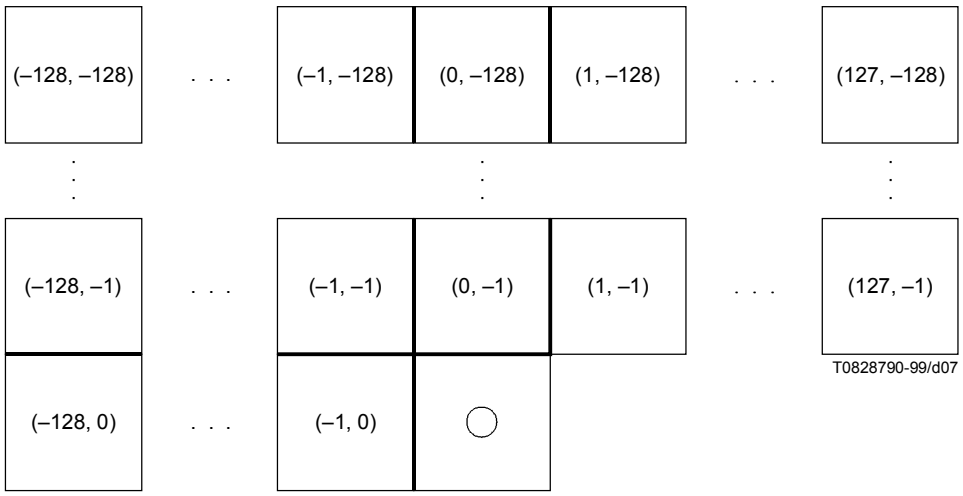


Figura 7 – Campo al que están limitadas las ubicaciones de los píxels AT

Cuadro 5 – Valores nominales de las ubicaciones de píxels AT

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
NOTA – NA significa que el parámetro no tiene un valor nominal.				

6.2.5.5 Predicción típica para codificación directa genérica (TPGD)

La predicción típica para codificación directa genérica puede ser habilitada o deshabilitada con el parámetro **TPGDON**. Si la predicción típica para codificación directa genérica está habilitada (**TPGDON** es **1**), antes de decodificar el primer píxel de cada fila, se decodificará un valor indicador de que esa fila es típica. Si la fila es típica, cada píxel de la misma es idéntico al píxel correspondiente de la fila situada inmediatamente por encima y, por consiguiente, no es necesario decodificar otros píxels de esta fila. Si la fila no es típica, es preciso decodificar cada uno de los píxels de la misma.

NOTA – El valor decodificado antes del primer píxel de cada fila no se utiliza en la plantilla de ningún píxel.

6.2.5.6 Píxels saltados

Si el parámetro **USESKIP** es **1**, el parámetro **SKIP** contiene un mapa de bits **GBW** por **GBH**. Cada píxel de **SKIP** corresponde a un píxel del mapa de bits que se decodifica; si el píxel de **SKIP** es **1**, el píxel correspondiente del mapa de bits que se codifica es **0**, y no se decodifica.

6.2.5.7 Decodificación del mapa de bits

La decodificación del mapa de bits procede como sigue:

1) Fijar:

$$LTP = \mathbf{0}$$

2) Crear un mapa de bits GBREG de **GBW** píxels de ancho y **GBH** píxels de alto.

3) Decodificar cada fila como sigue:

   a) Si todas las filas **GBH** han sido decodificadas, la decodificación se ha completado; avanzar al paso 4).

   b) Si **TPGDON** es **1**, decodificar un bit utilizando el codificador de entropía aritmético, dependiendo el contexto utilizado para decodificar este bit de la plantilla que se utiliza:

      - Si **GBTEMPLATE** es 0, se utiliza el contexto mostrado en la figura 8.
      - Si **GBTEMPLATE** es 1, se utiliza el contexto mostrado en la figura 9.
      - Si **GBTEMPLATE** es 2, se utiliza el contexto mostrado en la figura 10.
      - Si **GBTEMPLATE** es 3, se utiliza el contexto mostrado en la figura 11.

      Sea SLTP el valor de este bit. Fijar:

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

      NOTA – En las figuras 8 a 11, la plantilla se muestra con el píxel o los píxels AT en sus ubicaciones nominales. Deberán utilizarse los mismos valores de píxels (**0** ó **1**) para los píxels AT con independencia de cuáles son sus ubicaciones reales. Es decir, el desplazamiento de los píxels AT no afecta al contexto utilizado para decodificar SLTP.

   c) Si LTP = **1**, fijar cada uno de los píxels de la fila en curso de GBREG para que sea igual al píxel correspondiente de la fila situada inmediatamente por encima.

   d) Si LTP = **0**, decodificar, de izquierda a derecha, cada uno de los píxels de la fila en curso de GBREG. El procedimiento para cada píxel es como sigue:

      i) Si **USESKIP** es **1** y el píxel del mapa de bits **SKIP** en la ubicación correspondiente al píxel en curso es **1**, fijar el píxel en curso en **0**.

      ii) De no ser así:

         - Situar la plantilla dada por los parámetros **GBTEMPLATE**, $\mathbf{GBATX_1}$ a $\mathbf{GBATX_4}$ y $\mathbf{GBATY_1}$ a $\mathbf{GBATY_4}$ de manera que el píxel en curso esté alineado con la ubicación indicada con un círculo en la figura que describe el aspecto de la plantilla con el identificador **GBTEMPLATE**.
         - Formar un CONTEXT entero runiendo los valores de los píxeles de imagen cubiertos por la plantilla (incluidos los píxels AT) en su ubicación actual. El orden en que se efectúa esa recopilación no está normalizado, pero deberá ser coherente con la ubicación de los píxels AT e independiente de la misma.
         - Decodificar el píxel en curso invocando el procedimiento de decodificación de entropía aritmética, con CX fijada en el valor formado concatenando la etiqueta "GB" y los valores de los 10-16 píxeles reunidos en CONTEXT. El resultado de esta invocación es el valor del píxel en curso.

      EJEMPLO – Si **GBTEMPLATE** es 2, los píxels de imagen cubiertos por la plantilla son tal como se muestra en la figura 10 y los píxeles se han reunido en el orden de lectura (por filas de arriba a abajo y, dentro de cada fila, de izquierda a derecha), CX se fija en "GB**0011100101**".

4) Una vez que todas las filas han sido decodificadas, el contenido actual del mapa de bits GBREG es el resultado que obtendrá cada decodificador, tanto si ejecuta como si no esta secuencia exacta de pasos.

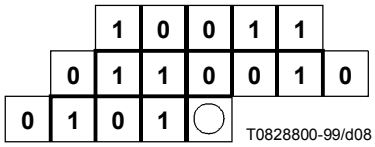


Figura 8 – Contexto reutilizado para la codificación del valor SLTP cuando GBTEMPLATE es 0

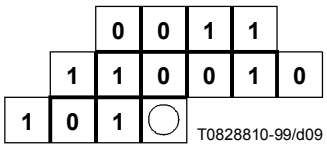


Figura 9 – Contexto reutilizado para la codificación del valor SLTP cuando GBTEMPLATE es 1

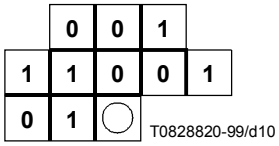


Figura 10 – Contexto reutilizado para la codificación del valor SLTP cuando GBTEMPLATE es 2

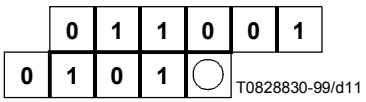


Figura 11 – Contexto reutilizado para la codificación del valor SLTP cuando GBTEMPLATE es 3

6.2.6 Decodificación utilizando la codificación MMR

Si **MMR** es **1**, el procedimiento de decodificación de región genérica es idéntico al del decodificador MMR (READ modificado modificado) descrito en la Recomendación UIT-T T.6, con las siguientes excepciones:

- Una invocación del procedimiento de decodificación de región genérica con **MMR** igual a **1** consumirá un número entero de bytes, empezando y terminando en una frontera de bytes. Con ello es posible que se salten algunos bits en el último byte leído.

- El decodificador que se especifica en la Recomendación UIT-T T.6 produce píxels cuyo valor puede ser "negro" o "blanco". A los efectos de la presente Recomendación | Norma Internacional, el resultado de utilizar el decodificador MMR se interpretará como sigue:

  – los píxels decodificados por el decodificador MMR que tengan el valor "negro" deberán tratarse como si tuvieran el valor **1**;

  – los píxels decodificados por el decodificador MMR que tengan el valor "blanco" deberán tratarse como si tuvieran el valor **0**.

- Si el número de bytes contenidos en el mapa de bits codificado se conoce por adelantado, puede permitirse que el tren de datos no contenga un EOFB (**000000000001000000000001**) al final de los datos codificados con MMR. Los casos en los que se conoce el número de bytes son aquellos en que este procedimiento de decodificación es invocado:

  – desde el procedimiento de decodificación del diccionario de patrones;

  – desde el procedimiento de decodificación del diccionario de símbolos; o

  – como parte de la decodificación de una región genérica cuya longitud de datos se conoce.

  El número de bytes no es conocido cuando este procedimiento de decodificación es invocado desde el procedimiento de decodificación de imagen de escala de grises o cuando es invocado como parte de la decodificación de una región genérica cuya longitud de datos no se conoce. En estos casos, EOFB debe estar presente.

NOTA 1 – Las fuentes de la cuenta de bytes, en los casos en que se conoce, son como sigue:

– En el procedimiento de decodificación de diccionario de patrones, se conoce la cuenta de bytes porque todos los datos del segmento – más allá del encabezamiento de datos de longitud fija – constituyen un único bloque de datos codificado con MMR, por lo que se puede calcular la longitud de los datos MMR a partir de la longitud de los datos del segmento.

– En el procedimiento de decodificación de diccionario de símbolos, se conoce la cuenta de bytes a partir de BMSIZE.

– En el procedimiento de decodificación de región genérica, se conoce de nuevo la cuenta de bytes a partir de la longitud de los datos del segmento.

  La razón por la cual se permite que EOFB sea opcional es que un EOFB tiene una longitud de tres bytes, mientras que la cuenta de bytes es a menudo conocida de antemano o se puede codificar en menos de tres bytes. Por lo tanto, omitir un EOFB reduce el tamaño de los datos codificados del mapa de bits; en los diccionarios de símbolos, en los cuales es frecuente que haya muchos pequeños mapas de bits codificados por separado, este ahorro puede resultar significativo.

NOTA 2 – Un decodificador puede efectuar el tratamiento del EOFB de diversas maneras en los casos en que es opcional. Esos distintos enfoques aprovechan la cuenta de bytes ya conocida y el hecho de que el EOFB – si está presente – ya está contabilizado en ese cómputo de bytes. Dos de los posibles enfoques son:

– Invocar el procedimiento de decodificación MMR y comprobar siempre si hay EOFB después de haber decodificado el mapa de bits. Sin embargo, no se debe permitir que el procedimiento de decodificación MMR examine más bytes que los que sabe que toman parte del bloque de datos comprimidos según MMR. Si el procedimiento de decodificación MMR se queda sin datos mientras está buscando el EOFB ello no se trata de un error sino de una condición normal que indica que no había EOFB presente.

– Invocar el procedimiento de decodificación MMR y no comprobar nunca si hay EOFB después de haber decodificado el mapa de bits, en los casos en que EOFB es opcional. Si el procedimiento de decodificación MMR consume menos bytes que los que se sabe que forma parte del bloque de datos comprimidos según MMR ello no se trata de un error sino de una condición normal que indica que había EOFB presente. Estos bytes no consumidos se han de saltear.

- Los códigos de ampliación de la Recomendación T.6, incluido el modo sin compresión, no deben estar presentes en los datos codificados con MMR.

NOTA 3 – El MMR permite un grado de compresión inferior al de la compresión de mapa de bits de imagen basada en la codificación aritmética. La decodificación de un mapa de bits de imagen utilizando MMR es más rápida que la decodificación de un mapa de bits de imagen basada en la codificación aritmética.

6.3 Procedimiento de decodificación de región de refinamiento genérico

6.3.1 Descripción general

Este procedimiento de decodificación se utiliza para decodificar una formación de valores **0** ó **1**, cuyos píxels se codifican uno por uno. Hay un mapa de bits de referencia conocido por el procedimiento de decodificación, que se utiliza como parte del proceso de decodificación. Se pretende que el mapa de bits de referencia se parezca al mapa de bits que se decodifica, y esta semejanza se utiliza para aumentar la compresión. Cada píxel se decodifica utilizando un contexto que comprende píxels extraídos del mapa de bits de referencia así como píxels decodificados previamente del mapa de bits que se decodifica.

6.3.2 Parámetros de entrada

En el cuadro 6 se muestran los parámetros de este procedimiento de decodificación.

Cuadro 6 – Parámetros del procedimiento de decodificación de región de refinamiento genérico

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
GRW	Entero	32	No	La anchura de la región
GRH	Entero	32	No	La altura de la región
GRTEMPLATE	Entero	1	No	El identificador de la plantilla
GRREFERENCE	Mapa de bits			El mapa de bits de referencia
GRREFERENCEDX	Entero	32	Sí	El desplazamiento X del mapa de bits de referencia con respecto al mapa de bits que se decodifica
GRREFERENCEDY	Entero	32	Sí	El desplazamiento Y del mapa de bits de referencia con respecto al mapa de bits que se decodifica
TPGRON	Entero	1	No	Indicación de si se utiliza predicción típica para refinamiento genérico
GRATX$_1$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa RA$_1$[a)]
GRATY$_1$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa RA$_1$[a)]
GRATX$_2$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa RA$_2$[a)].
GRATY$_2$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa RA$_2$[a)]
a) No se utiliza si **GRTEMPLATE ≠ 0**.				

6.3.3 Valor de retorno

En el cuadro 7 se muestra la variable cuyo valor es el resultado de este procedimiento de decodificación.

Cuadro 7 – Valor de retorno del procedimiento de decodificación de región de refinamiento genérico

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
GRREG	Mapa de bits			El mapa de bits de región decodificado

6.3.4 Variables utilizadas en la decodificación

En el cuadro 8 se muestran las variables utilizadas por este procedimiento de decodificación.

Cuadro 8 – Variables utilizadas en el procedimiento de decodificación de región de refinamiento genérico

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
CONTEXT	Entero	13	No	Los valores de los píxels de la plantillas
LTP	Entero	1	No	Indicación de si se decodifica explícitamente la línea de imagen en curso[a)]
SLTP	Entero	1	No	Indicación de si el valor LTP de la línea corriente difiere del valor LTP de la línea previa[a)]
TPGRPIX	Entero	1	No	Indicación de si se ha de codificar implícitamente el píxel corriente utilizando una predicción TPGR[a)]
TPGRVAL	Entero	1	No	Valor del píxel en curso predicho por TPGR[a)]
a) No se utiliza si **TPGRON = 0**.				

6.3.5 Decodificación utilizando una plantilla y codificación aritmética

6.3.5.1 Descripción general

El procedimiento de decodificación de región de refinamiento genérico se basa en la codificación aritmética con una plantilla para determinar el estado de la codificación. En lo que queda de 6.3.5 se describe esta forma de decodificación.

6.3.5.2 Orden de codificación y convenios de bordes

El algoritmo de codificación se aplica de forma iterativa a lo largo del mapa de bits de mejora que se decodifica, junto con un mapa de bits de referencia, en orden de exploración raster. Esto es, se aplica por filas de arriba a abajo y, dentro de cada fila, de izquierda a derecha. Durante el procesamiento del píxel objetivo en curso se hará referencia a algunos píxels en relación espacial fija con el píxel objetivo. Algunos de estos píxels se extraen de la versión de referencia del mapa de bits y algunos otros, de los píxels ya codificados del mapa de bits refinado.

Cerca de los bordes del mapa de bits, esas referencias vecinas pueden no hallarse en el mapa de bits real. La regla para satisfacer las referencias fuera de límites es como sigue:

- Todos los píxels que se hallen fuera de los límites del mapa de bits real o del mapa de bits de referencia tendrán el valor **0**.

6.3.5.3 Plantillas fijas y plantillas adaptativas

Una plantilla define una vecindad en torno al píxel que se ha de codificar. Los valores de los píxels de esa vecindad definen un contexto. Cada contexto tiene su propia estimación de probabilidad adaptativa utilizada por el codificador aritmético (véase el anexo E). Aunque una plantilla es un patrón geométrico de píxels, se dice que los píxels de la plantilla toman valores cuando la plantilla está alineada con una parte determinada de la imagen.

Figura 12 – Plantilla de refinamiento de 13 píxels mostrando los píxels AT en sus ubicaciones nominales

La figura 12 muestra la plantilla que será utilizada cuando **GRTEMPLATE** sea **0**. La figura 13 muestra la plantilla que será utilizada cuando **GRTEMPLATE** sea **1**. En cada una de estas figuras, el grupo del lado izquierdo indica los píxels de los píxels ya codificados del mapa de bits refinado que están en la plantilla, y el grupo del lado derecho indica los píxels de la versión de referencia de la plantilla que están en la plantilla. Cada grupo de la figura incluye un píxel señalado mediante un círculo; todos estos píxels corresponden al píxel que se ha de codificar. Los píxels marcados con una 'X' corresponden a píxels ordinarios de la plantilla. Los píxels indicados mediante RA_1-RA_2 son píxels especiales de la plantilla. Se les denomina "adaptativos" o píxels AT. Son píxels especiales en el sentido de que se permite que sus ubicaciones cambien durante el proceso de codificación de la imagen. Véase en 6.3.5.4 una descripción de los píxels AT. Las indicaciones RA_1-RA_2 señalan las ubicaciones nominales de los píxels AT 1 a 2.

El píxel AT RA_1 se puede situar en cualquier lugar del campo mostrado en la figura 7, sin incluir el píxel en curso. El píxel AT RA_2 se puede situar en cualquier lugar de la gama (–128, –128) a (127, 127) del mapa de bits de referencia.

Los píxels del grupo del lado izquierdo de cada plantilla se alinearán con los píxels ya decodificados del mapa de bits que se decodifica, estando el píxel señalado por un círculo en el píxel que se ha de decodificar. Sea (X, Y) la ubicación de este píxel. Los píxels del grupo del lado derecho de cada plantilla se alinearán con el mapa de bits de referencia **GRREFERENCE**, con el píxel señalado por un círculo situado en la ubicación (X – **GRREFERENCEDX,** Y - **GRREFERENCEDY**). Los valores de los píxels de la plantilla serán combinados para formar un contexto. Cada

píxel de la plantilla (incluidos los píxels adaptativos) deberá corresponder a un bit específico del contexto, aunque los píxels de la plantilla se pueden asignar a bits del contexto en cualquier orden. Puesto que la plantilla contiene hasta 13 píxels, los contextos pueden tomar hasta 8192 valores diferentes. El contexto se utilizará para identificar la estimación de probabilidad adaptativa que será utilizada por el codificador aritmético para codificar el píxel que ha de ser codificado (véase el anexo E).

Figura 13 – Plantilla de refinamiento de 10 píxels

6.3.5.4 Píxels de plantilla adaptativa

Al codificar la imagen, deberá permitirse a la plantilla cambiar según la manera restringida que se describe en esta subcláusula.

Los píxels a los que se permite cambiar se denominarán píxels AT. Sus ubicaciones nominales son indicadas por 'RA_1' y 'RA_2' en la figura 12. Se señala que sólo una plantilla tiene píxels AT.

6.3.5.5 Predicción típica para refinamiento genérico (TPGR)

La predicción típica para refinamiento genérico puede ser habilitada o deshabilitada con el parámetro **TPGRON**. Si la predicción típica para refinamiento genérico está habilitada (**TPGRON** es **1**), antes de decodificar el primer píxel de cada fila, se decodificará un valor indicador de si una fila es típica. Si la fila no es típica, cada píxel de la fila ha de ser decodificado de manera explícita. Si la fila es típica, todos los píxels predecibles típicamente pueden ser decodificados de manera implícita utilizando su valor predicho, siendo decodificados aun de manera explícita los píxels restantes. Para que un píxel sea predecible típicamente, debe cumplir los criterios definidos en 6.3.5.6, paso 3d).

NOTA – El valor decodificado antes del primer pixel de cada fila no se utiliza en la plantilla de ningún píxel.

6.3.5.6 Decodificación del mapa de bits de refinamiento

La decodificación del mapa de bits procede como sigue.

1) Fijar LTP = **0**.
2) Crear un mapa de bits GRREG de **GRW** píxels de ancho y **GRH** píxels de alto.
3) Decodificar cada fila como sigue:
   a) Si todas las filas **GRH** han sido decodificadas, la decodificación se ha completado; avanzar al paso 4).
   b) Si **TPGRON** es **1**, decodificar un bit utilizando el codificador de entropía aritmético, dependiendo del contexto utilizado para decodificar este bit de la plantilla que se utiliza:
      - si **GRTEMPLATE** es **0**, se utiliza el contexto mostrado en la figura 14;
      - si **GRTEMPLATE** es **1**, se utiliza el contexto mostrado en la figura 15.

      Sea SLTP el valor de este bit decodificado. Fijar:

      LTP = LTP XOR SLTP

   c) Si LTP = **0**, decodificar de manera explícita, de izquierda a derecha, cada uno de los píxels de la fila en curso de GRREG. El procedimiento para cada píxel es como sigue:
      i) Situar la plantilla dada por los parámetros **GRTEMPLATE** (y **$GRATX_1$**, **$GRATY_1$**, **$GRATX_2$** y **$GRATY_2$** si **GRTEMPLATE** es **0**) de manera que el píxel en curso esté alineado con la ubicación indicada con un círculo en la figura que describe el aspecto de la plantilla con el identificador **GRTEMPLATE**.
      ii) Formar un CONTEXT entero reuniendo los valores de los píxels de imagen cubiertos por la plantilla (incluidos los píxels AT) en su ubicación actual. El orden en que se efectúa esa recopilación no está normalizado, pero deberá ser coherente con la ubicación de los píxels AT e independiente de la misma.

iii) Decodificar el píxel en curso invocando el procedimiento de decodificación de entropía aritmética, con CX fijada en el valor formado concatenando la etiqueta "GR" y los valores de los 10-13 píxels reunidos en CONTEXT. El resultado de esta invocación es el valor del píxel en curso.

EJEMPLO – Si **GRTEMPLATE** es **1**, los píxels de imagen cubiertos por la plantilla son tal como se muestra en la figura 15 y los píxels se han reunido en el orden de la lectura (por filas de arriba a abajo y, dentro de cada fila, de izquierda a derecha, con los píxels de GRREG considerados antes que los píxels de GRREFERENCE), CX se fija en "GR**0000001000**".

d) Si LTP = **1**, decodificar de manera implícita, de izquierda a derecha, determinados píxels de la fila en curso de GRREG, y decodificar de manera explícita el resto. El procedimiento para cada píxel es como sigue:

i) Fijar TPGRPIX igual a **1** si:

- **TPGRON** es **1**; Y
- una formación de 3 × 3 píxels del mapa de bits de referencia (figura 16), centrada en la ubicación correspondiente al píxel en curso, contiene todos los píxels del mismo valor.

Cuando TPGRPIX se ha fijado a **1**, fijar TPGRVAL igual al valor de dicho píxel en curso, que es el valor común de los nueve píxels adyacentes en la formación de 3 × 3.

ii) Si TPGRPI es **1**, decodificar de manera implícita el píxel en curso fijándolo igual a su valor predicho (TPGRVAL).

iii) De otro modo, decodificar de manera explícita el píxel en curso utilizando la metodología de los pasos 3 c) i) a 3 c) iii) anteriores.

4) Una vez que todas las filas han sido decodificadas, el contenido actual del mapa de bits GRREG es el resultado que obtendrá cada decodificador, tanto si ejecuta como si no esta secuencia exacta de pasos.

Figura 14 – Contexto reutilizado para la codificación del valor SLTP cuando GRTEMPLATE es 0

Figura 15 –Contexto reutilizado para la codificación del valor SLTP cuando GRTEMPLATE es 1

Figura 16 –Plantilla TPGR

6.4 Procedimiento de decodificación de región de texto

6.4.1 Descripción general

Este procedimiento de decodificación se utiliza para decodificar un mapa de bits decodificando varios ejemplares de símbolo. Un ejemplar de símbolo contiene una ubicación y un ID símbolo y, posiblemente, un mapa de bits de refinamiento. Los ejemplares de símbolo se combinan para formar el mapa de bits decodificado.

NOTA – Este procedimiento de decodificación será utilizado normalmente para decodificar la parte texto de la página. Los símbolos son por lo general caracteres de texto únicos de algún conjunto de tipos de letra o alfabeto.

6.4.2 Parámetros de entrada

En el cuadro 9 se muestran los parámetros de este procedimiento de decodificación.

NOTA – Los valores de algunos de estos parámetros en una situación típica, en la que se decodifica un mapa de bits que contiene caracteres de texto en orden normal de lectura del inglés, y el valor del píxel del primer plano es **1**, son como sigue:

- **SBDEFPIXEL** es **0**
- **SBCOMBOP** es OR
- **TRANSPOSED** es **0**
- **REFCORNER** es BOTTOMLEFT

Cuadro 9 – Parámetros del procedimiento de decodificación de región de texto

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
SBHUFF	Entero	1	No	Indica si se utiliza codificación Huffman
SBREFINE	Entero	1	No	Indica si se utiliza codificación de refinamiento
SBW	Entero	32	No	La anchura de la región
SBH	Entero	32	No	La altura de la región
SBNUMINSTANCES	Entero	32	No	El número de ejemplares de símbolo en esta región
SBSTRIPS	Entero	4	No	El tamaño de las franjas de ejemplar de símbolo. Puede tomar los valores 1, 2, 4 u 8
SBNUMSYMS	Entero	32	No	El número de símbolos que pueden ser utilizados en esta región
SBSYMCODES	Formación de códigos Huffman			Una formación que contiene los códigos de los símbolos utilizados en esta región. Contiene códigos **SBNUMSYM**[a)]
SBSYMCODELEN	Entero	6	No	La longitud de los códigos de símbolo utilizados en IAID[d)]
SBSYMS	Formación de símbolos			Una formación que contiene los símbolos utilizados en esta región de texto. Contiene símbolos **SBNUMSYMS**
SBDEFPIXEL	Entero	1	No	El píxel por defecto de este mapa de bits
SBCOMBOP	Operador			El operador de combinación para esta región de texto. Puede tomar los valores OR, AND, XOR y XNOR
TRANSPOSED	Entero	1	No	Indica si las franjas están orientadas verticalmente
REFCORNER	Esquina			La esquina de referencia de cada mapa de bits de ejemplar de símbolo. Puede tomar los valores TOPLEFT, TOPRIGHT, BOTTMLEFT y BOTTOMRIGHT
SBDSOFFSET	Entero	5	Sí	Un desplazamiento para todos los valores de S delta
SBHUFFFS	Tabla Huffman			La tabla Huffman utilizada para decodificar la coordenada S del primer ejemplar de símbolo de cada franja[a)]

Cuadro 9 *(fin)*

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
SBHUFFFDS	Tabla Huffman			La tabla Huffman utilizada para decodificar la coordenada S de los ejemplares de símbolo subsiguientes de cada franja[a)]
SBHUFFDT	Tabla Huffman			La tabla Huffman utilizada para decodificar la primera diferencia entre coordenadas T de franjas no vacías[a)]
SBHUFFRDW	Tabla Huffman			La tabla Huffman utilizada para decodificar la diferencia entre la anchura de un símbolo y la anchura de un mapa de bits con codificación de mejora[b)]
SBHUFFRDH	Tabla Huffman			La tabla Huffman utilizada para decodificar la diferencia entre la altura de un símbolo y la altura de un mapa de bits con codificación de refinamiento[b)]
SBHUFFRDX	Tabla Huffman			La tabla Huffman utilizada para decodificar la diferencia entre la coordenada X de un ejemplar de símbolo y la coordenada X de un mapa de bits con codificación de refinamiento[b)]
SBHUFFRDY	Tabla Huffman			La tabla Huffman utilizada para decodificar la diferencia entre la coordenada Y de un ejemplar de símbolo y la coordenada Y de un mapa de bits con codificación de refinamiento[b)]
SBHUFFRSIZE	Tabla Huffman			La tabla Huffman utilizada para decodificar el tamaño de los datos del mapa de bits de refinamiento de un ejemplar de símbolo[b)]
SBRTEMPLATE	Entero	1	No	Identificador de plantilla para codifiación con refinamiento de mapas de bits de ejemplares de símbolo[c)]
$SBRATX_1$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa RA_1[c)]
$SBRATY_1$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa RA_1[c)]
$SBRATX_2$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa RA_2[c)]
$SBRATY_2$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa RA_2[c)]

a) No se utiliza si **SBHUFF** = **0**.
b) No se utiliza si **SBHUFF** = **0** o **SBREFINE** = **0**.
c) No se utiliza si **SBREFINE** = **0**.
d) No se utiliza si **SBHUFF** = **1**.

6.4.3 Valor de retorno

En el cuadro 10 se muestra la variable cuyo valor es el resultado de este procedimiento de decodificación.

Cuadro 10 – Valor de retorno del procedimiento de decodificación de región de texto

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
SBREG	Mapa de bits			El mapa de bits de región decodificado

6.4.4 Variables utilizadas en la decodificación

En el cuadro 11 se muestran las variables utilizadas por este procedimiento de decodificación.

Cuadro 11 – Variables utilizadas en el procedimiento de decodificación de región de texto

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
STRIPT	Entero	32	Sí	La coordenada T numéricamente más pequeña de la franja en curso
FIRSTS	Entero	32	Sí	La primera coordenada S de la franja en curso
NINSTANCES	Entero	32	No	Un contador de ejemplares de símbolo
DT	Entero	32	Sí	El número de franjas vacías entre dos franjas no vacías
DFS	Entero	32	Sí	La diferencia entre las coordenadas S de los primeros ejemplares de símbolo de dos franjas
CURS	Entero	32	Sí	La coordenada S actual
CURT	Entero	3	No	La coordenada T del ejemplar de símbolo actual con relación a la franja en curso
S_I	Entero	32	Sí	La coordenada S de un ejemplar de símbolo
T_I	Entero	32	Sí	La coordenada T de un ejemplar de símbolo
ID_I	Entero	32	No	ID símbolo de un ejemplar de símbolo
IB_I	Mapa de bits			Mapa de bits de símbolo de un ejemplar de símbolo
W_I	Entero	32	No	La anchura del mapa de bits de símbolo de un ejemplar de símbolo
H_I	Entero	32	No	La altura del mapa de bits de símbolo de un ejemplar de símbolo
IDS	Entero	32	Sí	La diferencia entre las coordenadas S de dos ejemplares de símbolo dentro de una franja
R_I	Entero	1	No	Indicación de si se codifica utilizando mejora el mapa de bits de símbolo de un ejemplar de símbolo
RDW_I	Entero	32	Sí	La anchura delta del mapa de bits de mejora de un ejemplar de símbolo[a)]
RDH_I	Entero	32	Sí	La altura delta del mapa de bits de mejora de un ejemplar de símbolo[a)]
RDX_I	Entero	32	Sí	La X delta del mapa de bits de refinamiento de un ejemplar de símbolo[a)]
RDY_I	Entero	32	Sí	La Y delta del mapa de bits de refinamiento de un ejemplar de símbolo[a)]
IBO_I	Mapa de bits			Mapa de bits de símbolo original de un ejemplar de símbolo[a)]
WO_I	Entero	32	No	La anchura de IBO_I[a)]
HO_I	Entero	32	No	La altura de IBO_I[a)]

a) No se utiliza si **SBREFINE** = **0**.

6.4.5 Decodificación de la región de texto

Un mapa de bits con codificación de símbolos se representa mediante un conjunto de ejemplares de símbolo. Cada ejemplar de símbolo codifica una ubicación, un ID símbolo y, posiblemente, información de refinamiento. La ubicación de cada ejemplar de símbolo consta de una coordenada S y una coordenada T. Si **TRANSPOSED** es **0**, el eje de coordenadas S corresponde al eje X del mapa de bits, y el eje de coordenadas T corresponde al eje Y del mapa de bits. Si **TRANSPOSED** es **1**, el eje de coordenadas S corresponde al eje Y del mapa de bits, y el eje de coordenadas T corresponde al eje X del mapa de bits.

NOTA 1 – La transposición de los ejes de coordenadas permite codificar de manera eficaz el texto orientado verticalmente. La esquina de referencia es variable porque la codificación más eficaz se obtiene normalmente cuando la esquina de referencia de cada ejemplar de símbolo se halla en una línea de base de texto, y las líneas de base de texto pueden orientarse en cualquier dirección.

Para mejorar la compresión, los ejemplares de símbolo se agrupan en franjas según sus valores T_I. Esto se hace de acuerdo con el valor de **SBSTRIPS**. Los ejemplares de símbolo que tienen valores T_I entre 0 y **SBSTRIPS** – 1 se agrupan en una franja, los ejemplares de símbolo que tienen valores T_I entre **SBSTRIPS** y 2 × **SBSTRIPS** – 1 en la siguiente, y así sucesivamente. Dentro de cada franja, los ejemplares de símbolo se codifican en el orden de la coordenada S creciente.

NOTA 2 – Normalmente, las franjas se presentan en el orden de las coordenadas T estrictamente crecientes, y los ejemplares de símbolo dentro de cada franja se presentan en el orden de las coordenadas S no decrecientes. Sin embargo, es posible que se produzcan valores negativos de S delta o T delta durante la decodificación, lo que significaría que las franjas y los ejemplares de símbolo podrían presentarse en cualquier orden.

En la figura 17 se muestra la estructura global de los datos que se han de decodificar para reconstruir la región de texto. En la figura 18 se muestra el formato de cada franja. Cuando **SBREFINE** es **0**, el formato de cada ejemplar de símbolo es tal como se muestra en la figura 19. Cuando **SBREFINE** es **1**, el formato de cada ejemplar de símbolo es como se muestra en la figura 20.

NOTA 3 – Puede haber algunos ejemplares de símbolo cuya esquina de referencia se encuentre fuera de la parte superior de la región. Si tienen que ser codificadas, ha de haber alguna manera de hacer que una franja se halle también por encima de la parte superior de la región. El valor inicial de STRIPT es la coordenada con respecto a la cual se sitúa la primera franja.

Valor inicial de STRIPT
Primera franja
Segunda franja
. . .
Última franja

Figura 17 – Estructura codificada de una región de texto

T Delta
Primer ejemplar de símbolo
Segundo ejemplar de símbolo
. . .
Último ejemplar de símbolo
OOB

Figura 18 – Estructura de una franja

Coordenada S de ejemplar de símbolo
Coordenada T de ejemplar de símbolo
ID símbolo de ejemplar de símbolo

Figura 19 – Estructura de un ejemplar de símbolo cuando SBREFINE es 0

Coordenada S de ejemplar de símbolo
Coordenada T de ejemplar de símbolo
ID símbolo de ejemplar de símbolo
Información de refinamiento de ejemplar de símbolo

Figura 20 – Estructura de un ejemplar de símbolo cuando SBREFINE es 1

El resultado de la decodificación de una región de texto será el mapa de bits producido cuando se siguen los pasos que se indican a continuación.

1) Llenar un mapa de bits SBREG, del tamaño dado por **SBW** y **SBH**, con el valor **SBDEFPIXEL**.
2) Decodificar el valor de STRIPT inicial como se describe en 6.4.6. Anular el valor decodificado y asignar este valor anulado a la variable STRIPT. Asignar el valor 0 a FIRSTS. Asignar el valor 0 a NINSTANCES.
3) Decodificar cada franja como sigue:
   a) Si INSTANCES es igual a **SBNUMINSTANCES**, no hay más franjas para decodificar y el proceso de decodificación de la región de texto se ha completado; avanzar el paso 4).
   b) Decodificar el valor de T delta de la franja como se describe en 6.4.6. Sea DT el valor decodificado. Fijar:

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

   c) Decodificar cada ejemplar de símbolo de la franja como sigue:
      i) Si el ejemplar de símbolo en curso es el primer ejemplar de símbolo de la franja, decodificar la coordenada S del primer ejemplar de símbolo como se describe en 6.4.7. Sea DFS el valor decodificado. Fijar:

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

      ii) De otro modo, si el ejemplar de símbolo en curso no es el primer ejmplar de símbolo de la franja, decodificar la coordenada S del ejemplar de símbolo como se describe en 6.4.8. Si el resultado de esta decodificación es OOB quiere decir que se ha decodificado el último ejemplar de símbolo de la franja; avanzar el paso 3 d). De otro modo, sea IDS el valor decodificado. Fijar:

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

NOTA 4 – El uso pretendido de **SBDSOFFSET** es hacer que el valor más frecuentemente decodificado en 6.4.8 sea cero. El código más corto de todas las tablas por defecto utilizadas en 6.4.8 es para el valor cero.

      iii) Decodificar la coordenada T del ejemplar de símbolo como se describe en 6.4.9. Sea CURT el valor decodificado. Fijar:

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

      iv) Decodificar el ID símbolo del ejemplar de símbolo como se describe en 6.4.10. Sea ID_I el valor decodificado.
      v) Determinar el mapa de bits del ejemplar de símbolo IB_I como se describe en 6.4.11. La anchura y la altura de este mapa de bits se indicarán mediante W_I y H_I respectivamente.
      vi) Actualizar CURS como sigue:
         - Si **TRANSPOSED** es **0**, y **REFCORNER** es TOPRIGHT o BOTTOMRIGHT, fijar:

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

         - Si **TRANSPOSED** es **1**, y **REFCORNER** es BOTTOMLEFT o BOTTOMRIGHT, fijar:

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

         - De otro modo, no se cambia CURS en este paso.
      vii) Fijar:

$$S_I = \text{CURS}$$

      viii) Determinar la ubicación del mapa de bits del ejemplar de símbolo con respecto a SBREG como sigue:
         - Si **TRANSPOSED** es **0**:
            – Si **REFCORNER** es TOPLEFT, el píxel superior izquierdo del mapa de bits de ejemplar de símbolo IB_I deberá situarse en SBREG[S_I, T_I].

– Si **REFCORNER** es TOPRIGHT, el píxel superior derecho del mapa de bits de ejemplar de símbolo IB_I deberá situarse en SBREG[S_I, T_I].

– Si **REFCORNER** es BOTTOMLEFT, el píxel inferior izquierdo del mapa de bits de ejemplar de símbolo IB_I deberá situarse en SBREG[S_I, T_I].

– Si **REFCORNER** es BOTTOMRIGHT, el píxel inferior derecho del mapa de bits de ejemplar de símbolo IB_I deberá situarse en SBREG[S_I, T_I].

- Si **TRANSPOSED** es **1**:

  – Si **REFCORNER** es TOPLEFT, el píxel superior izquierdo del mapa de bits de ejemplar de símbolo IB_I deberá situarse en SBREG[T_I, S_I].

  – Si **REFCORNER** es TOPRIGHT, el píxel superior derecho del mapa de bits de ejemplar de símbolo IB_I deberá situarse en SBREG[T_I, S_I].

  – Si **REFCORNER** es BOTTOMLEFT, el píxel inferior izquierdo del mapa de bits de ejemplar de símbolo IB_I deberá situarse en SBREG[T_I, S_I].

  – Si **REFCORNER** es BOTTOMRIGHT, el píxel inferior derecho del mapa de bits de ejemplar de símbolo IB_I deberá situarse en SBREG[T_I, S_I].

  Si cualquier parte de IB_I, cuando está situada en esta ubicación, queda fuera de los límites de SBREG, se ignora esta parte de IB_I en el paso 3 c) ix).

ix) Dibujar IB_I en SBREG. Combinar cada píxel IB_I con el valor actual del píxel correspondiente en SBREG, utilizando el operador de combinación especificado por **SBCOMBOP**. Escribir los resultados de cada combinación en ese píxel en SBREG.

x) Actualizar CURS como sigue:

- Si **TRANSPOSED** es **0**, y **REFCORNER** es TOPLEFT o BOTTOMLEFT, fijar:

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

- Si **TRANSPOSED** es **1**, y **REFCORNER** es TOPLEFT o TOPRIGHT, fijar

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

- De otro modo, no se cambia CURS en este paso.

NOTA 5 – Las reglas de actualización de CURS están concebidas para hacer posible la reparación entre ejemplares de símbolos adyacentes que se han de codificar, en vez del ejemplar entre sus esquinas de referencia; para ello se requiere un origen de la variación (la anchura o la altura del mapa de bits en el ejemplar de símbolos) y permite una mejor compresión.

xi) Fijar:

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

d) Cuando la franja haya sido decodificada por completo, decodificar la franja siguiente.

4) Una vez que todas las franjas han sido decodificadas, el contenido actual del mapa de los bits SBREG es el resultado que obtendrá cada decodificador, tanto si ejecuta como si no, esta secuencia exacta de pasos.

6.4.6 T delta de franja

Si **SBHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SBHUFFDT** y multiplicar el valor resultante por **SBSTRIPS**.

Si **SBHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IADT (véase el anexo A) y multiplicar el valor resultante por **SBSTRIPS**.

6.4.7 Coordenada S de primer ejemplar de símbolo

NOTA – El valor de la coordenada S de ejemplar de símbolo del ejemplar instancia de símbolo de cada franja se codifica de manera diferente a partir de los ejemplares de símbolo subsiguientes en cada franja. De esta manera se aprovechan los principios de línea que se alinean.

Si **SBHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SBHUFFS**.

Si **SBHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IAFS (véase el anexo A).

6.4.8 Coordenada S de ejemplar de símbolo subsiguiente

Si **SBHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SBHUFFDS.**

Si **SBHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IADS (véase el anexo A).

En cualquiera de ambos casos es posible que el resultado de esta decodificación sea el valor fuera de banda (OOB).

6.4.9 Coordenada T de ejemplar de símbolo

Si **SBTRIPS** = 1, el valor decodificado es siempre cero. De no ser así:

- Si **SBHUFF** es **1**, decodificar un valor leyendo $\lceil \log_2 \textbf{SBSTRIP} \rceil$ bits directamente desde el tren binario.
- Si **SBHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IAIT (véase el anexo A)

  NOTA – Si **SBSTRIPS** = 1, no se consumen bits y no se invoca nunca el procedimiento de decodificación aritmética de enteros IAIT.

6.4.10 ID símbolo de ejemplar de símbolo

Si **SBHUFF** es **1**, decodificar un valor leyendo un bit cada vez hasta que la cadena de bits resultante sea igual a una de las entradas en **SBSYMCODES**. El valor resultante, que es ID_I, es el índice de la entrada en **SBSYMCODES** que se lee.

Si **SBHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IAID (véase el anexo A). Fijar ID_I en el valor resultante.

6.4.11 Mapa de bits de ejemplar de símbolo

En algunos casos, el mapa de bits de ejemplar de símbolo IB_I es simplemente el mapa de bits del símbolo identificado por ID_I. En otros casos, sin embargo, el mapa de bits de ejemplar de símbolo es ese mapa de bits modificado por información de refinamiento adicional. El bit que indica cuál de las operaciones es verdadera para un ejemplar de símbolo se llama R_I.

Si **SBREFINE** es **0**, fijar R_I a **0**.

Si **SBREFINE** es **1**, decodificar R_I como sigue:

- Si **SBHUFF** es **1**, leer un bit y fijar R_I en el valor de ese bit.
- Si **SBHUFF** es **0**, decodificar un bit utilizando el procedimiento de decodificación aritmética de enteros IARI y fijar R_I en el valor de ese bit.

Si R_I es **0**, fijar el mapa de bits de instancia de símbolo IB_I a **SBSYMS**[ID_I].

Si R_I es **1**, determinar el mapa de bits de instancia de símbolo como sigue:

1) Decodificar la anchura delta de refinamiento de ejemplar de símbolo como se describe en 6.4.11.1. Sea RDW_I el valor decodificado.
2) Decodificar la altura delta de refinamiento de ejemplar de símbolo como se describe en 6.4.11.2. Sea RDH_I el valor decodificado.
3) Decodificar el desplazamiento X de refinamiento de ejemplar de símbolo como se describe en 6.4.11.3. Sea RDX_I el valor decodificado.
4) Decodificar el desplazamiento Y de refinamiento de ejemplar de símbolo como se describe en 6.4.11.4. Sea RDY_I el valor decodificado.
5) Si **SBHUFF** es **1**:
   a) Decodificar el tamaño de datos del mapa de bits de refinamiento de ejemplar de símbolo como se describe en 6.4.11.5.
   b) Saltar los bits restantes en el último byte leído.
6) Sea IBO_I **SBSYMS**[ID_I]. Sea WO_I la anchura de IBO_I y HO_I la altura de IBO_I. El mapa de bits de ejemplar de símbolo IB_I es el resultado de aplicar el procedimiento de decodificación de región de refinamiento genérica descrito en 6.3. Fijar los parámetros de este procedimiento de decodificación como se muestra en el cuadro 12.
7) Si **SBHUFF** es **1**, saltar los bits restantes en el último byte leído. El número total de bytes procesados por el procedimiento de decodificación de mapa de bits de refinamiento genérica debe ser igual al valor leído en el paso 5 a).

6.4.11.1 Anchura delta (diferencia de) anchura de refinamiento de ejemplar de símbolo

Este campo, y los campos siguientes, indican el tamaño, la ubicación y el contenido del mapa de bits de símbolo mejorado, ya que el tamaño podría no ser el mismo que el tamaño del mapa de bits del símbolo cuyo ID se da en este ejemplar de símbolo; además, el cambio del tamaño del mapa de bits se puede extender a la izquierda y por arriba, no simplemente a la derecha y por debajo, por lo que es necesario suministrar tanto un desplazamiento como un tamaño. Se señala que los desplazamientos se dan en términos de X e Y, no de S y T.

Si **SBHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SBHUFFRDW**.

Si **SBHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IARDW (véase el anexo A).

Cuadro 12 – Parámetros utilizados para decodificar el mapa de bits de un ejemplar de símbolo utilizando refinamiento

Nombre	Valor
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 Altura delta (diferencia de) de refinamiento de ejemplar de símbolo

Si **SBHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SBHUFFRDH**.

Si **SBHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IARDH (véase el anexo A).

6.4.11.3 Desplazamiento X de refinamiento de ejemplar de símbolo

Si **SBHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SBHUFFRDX**.

Si **SBHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IARDX (véase el anexo A).

6.4.11.4 Desplazamiento Y de refinamiento de ejemplar de símbolo

Si **SBHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SBHUFFRDY**.

Si **SBHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IARDY (véase el anexo A).

6.4.11.5 Tamaño de datos de mapa de bits de refinamiento de ejemplar de símbolo

Decodificar un valor utilizando la tabla Huffman especificada por **SBHUFFRSIZE**.

6.5 Procedimiento de decodificación de diccionario de símbolos

6.5.1 Descripción general

Este procedimiento de decodificación se utiliza para decodificar un conjunto de símbolos; estos símbolos pueden ser utilizados a continuación por procedimientos de decodificación de región de texto, o en algunos casos por otros procedimientos de decodificación de diccionario de símbolos.

6.5.2 Parámetros de entrada

En el cuadro 13 se muestran los parámetros de este procedimiento de decodificación.

Cuadro 13 – Parámetros del procedimiento de decodificación de diccionario de símbolos

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
SDHUFF	Entero	1	No	Indicación de si se utiliza codificación Huffman
SDREFAGG	Entero	1	No	Indicación de si se utiliza codificación de refinamiento y agregación
SDNUMINSYMS	Entero	32	No	El número de símbolos que se utilizan como entrada a este procedimiento de decodificación de diccionario de símbolos
SDINSYMS	Formación de símbolos			Una formación que contiene los símbolos que se utilizan como entrada a este procedimiento de decodificación de diccionario de símbolos. Contiene símbolos **SDNUMINSYMS**.
SDNUMNEWSYMS	Entero	32	No	El número de símbolos que se han de definir en este diccionario de símbolos
SDNUMEXSYMS	Entero	32	No	El número de símbolos que se han de exportar desde este diccionario de símbolos
SDHUFFDH	Tabla Huffman			La tabla Huffman utilizada para decodificar la diferencia en altura entre dos clases de altura[a)]
SDHUFFDW	Tabla Huffman			La tabla Huffman utilizada para decodificar la diferencia de anchura entre dos símbolos[a)]
SDHUFFBMSIZE	Tabla Huffman			La tabla Huffman utilizada para decodificar el tamaño de un mapa de bits colectivo de clase de altura[a)]
SDHUFFAGGINST	Tabla Huffman			La tabla Huffman utilizada para decodificar el número de ejemplares de símbolo en una agregación[b)]
SDTEMPLATE	Entero	2	No	El identificador de plantilla utilizado para decodificar mapas de bits de símbolo[c)]
$SDATX_1$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa A_1[c)]
$SDATY_1$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa A_1[c)]
$SDATX_2$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa A_2[c)]
$SDATY_2$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa A_2[c)]
$SDATX_3$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa A_3[c)]
$SDATY_3$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa A_3[c)]
$SDATX_4$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa A_4[c)]
$SDATY_4$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa A_4[c)]
SDRTEMPLATE	Entero	1	No	Identificador de plantilla para la codificación con refinamiento de mapas de bits[d)]
$SDRATX_1$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa RA_1[d)]
$SDRATY_1$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa RA_1[d)]
$SDRATX_2$	Entero	8	Sí	La ubicación X del píxel de plantilla adaptativa RA_2[d)]
$SDRATY_2$	Entero	8	Sí	La ubicación Y del píxel de plantilla adaptativa RA_2[d)]

a) No se utiliza si **SDHUFF = 0**.
b) No se utiliza si **SDHUFF** = **0** o **SDREFAGG = 0**.
c) No se utiliza si **SDHUFF = 1**.
d) No se utiliza si **SDREFAGG = 0**.

El parámetro **SDREFAGG** determina cómo se codifican los símbolos de este diccionario de símbolos. Si **SDREFAGG** es **0**, cada mapa de bits de símbolo se codifica vía codificación directa de mapa de bits. Si **SDREFAGG** es **1**, cada mapa de bits de símbolo se codifica refinando o agregando mapas de bits de símbolo definidos previamente. Estos mapas de bits de símbolo definidos previamente pueden ser dibujados a partir de otros diccionarios y proporcionados como entrada a este procedimiento de decodificación en **SDINSYMS**, o pueden ser definidos en el diccionario vigente.

6.5.3 Valor de retorno

En el cuadro 14 se muestra la variable cuyo valor es el resultado de este procedimiento de decodificación.

Cuadro 14 – Valor de retorno del procedimiento de decodificación de diccionario de símbolos

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
SDEXSYMS	Formación de símbolos			Los símbolos exportados por este diccionario de símbolos. Contiene símbolos **SDNUMEXSYMS**

6.5.4 Variables utilizadas en la decodificación

En el cuadro 15 se muestran las variables utilizadas por este procedimiento de decodificación.

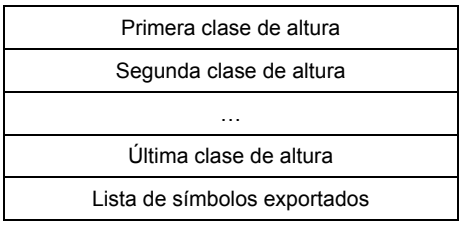
Cuadro 15 – Variables utilizadas en el procedimiento de decodificación de diccionario de símbolos

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
SDNEWSYMS	Formación de símbolos			Los símbolos definidos en este diccionario de símbolos. Contiene símbolos **SDNUMNEWSYMS**
SDNEWSYMWIDTHS	Formación de enteros			Las anchuras de los símbolos en SDNEWSYMS. Contiene enteros **SDNUMNEWSYMS**. Cada entero es un valor sin signo de 32 bits
HCHEIGHT	Entero	32	No	La altura de la clase de altura vigente
NSYMSDECODED	Entero	32	No	Cuántos símbolos han sido decodificados hasta ahora
HCDH	Entero	32	Sí	La diferencia en altura entre dos clases de altura
SYMWIDTH	Entero	32	No	La anchura del símbolo en curso
TOTWIDTH	Entero	32	No	La anchura de la clase de altura vigente
HCFIRSTSYM	Entero	32	No	El índice del primer símbolo en la clase de altura vigente
DW	Entero	32	Sí	La diferencia de anchura entre dos símbolos
B_S	Mapa de bits			El mapa de bits del símbolo en curso
B_{HC}	Mapa de bits			El mapa de bits colectivo de clase de altura vigente
I	Entero	32	No	Un índice de formación
J	Entero	32	No	Un índice de formación
REFAGGNINST	Entero	32	No	El número de ejemplares de símbolos en una agregación
EXFLAGS	Formación de enteros			Las banderas de exportación para este diccionario. Contiene valores **SDNUMINSYMS** + **SDNUMNEWSYMS**. Cada valor es un bit
EXINDEX	Entero	32	No	Un índice de formación
CUREXFLAG	Entero	1	No	La bandera de exportación actual
EXRUNLENGTH	Entero	32	No	La longitud de una serie de valores de bandera de exportación idénticos

6.5.5 Decodificación de diccionario de símbolos

En la figura 21 se muestra la estructura interna de un diccionario de símbolos. Los símbolos definidos en el diccionario se ordenan en clase de altura: una clase de altura contiene un cierto número de símbolos cuyos mapas de bits tienen la misma altura.

NOTA 1 – En la mayoría de los casos, las clases de altura se presentan en el orden estricto de altura creciente, de menor a mayor altura. Sin embargo, si **SDREFAGG** es **1**, un símbolo puede ser codificado como un refinamiento de un símbolo mayor definido en el mismo diccionario. En este caso, la clase de altura de ese símbolo base debe ser decodificada (y puede presentarse por tanto) antes que la clase de altura menor del símbolo que se codifica refinándolo. Por este motivo, las alturas delta (diferencias de) de clase de altura (y anchuras delta (diferencias de) de símbolos) pueden ser cero o negativas, tanto como positivas.



Fgura 21 – Estructura de un diccionario de símbolos

Si **SDHUFF** es **1** y **SDREFAGG** es **0**, el formato de una clase de altura es como se muestra en la figura 22. De otro modo, el formato de una clase de altura es como se muestra en la figura 23. Los campos mencionados en estas figuras se describen por completo en lo que sigue.

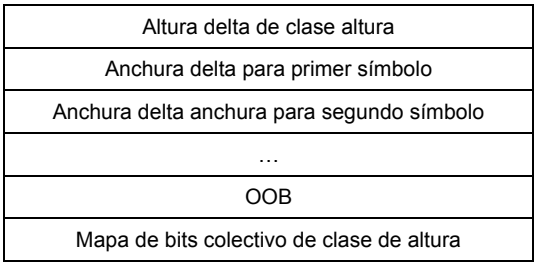


Figura 22 – Codificación de clases de altura cuando SDHUFF es 1 y SDREFAGG es 0

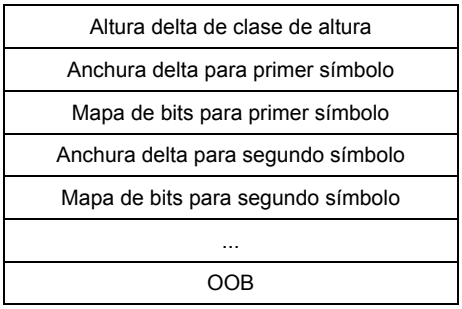


Figura 23 – Codificación de clase de altura cuando SDHUFF es 0 o SDREFAGG es 1

El resultado de la decodificación de un diccionario de símbolos es una formación SDEXSYMS que contiene mapas de bits **SDNUMEXSYMS**. Esta formación será la formación producida por los pasos siguientes:

1) Crear una formación SDNEWSYMS de mapas de bits, con **SDNUMNEWSYMS** entradas.

2) Si **SDHUFF** es **1** y **SDREFAGG** es **0**, crear una formación SDNEWSYMWIDTHS de enteros, con **SDNUMNEWSYMS** entradas.

3) Fijar:

$$HCHEIGHT = 0$$
$$NSYMSDECODED = 0$$

4) Decodificar cada clase de altura como sigue:

a) Si NSYMSDECODED = **SDNUMNEWSYMS**, todos los símbolos del diccionario han sido decodificados; avanzar al paso 5).

b) Decodificar la delta altura de clase de altura como se describe en 6.5.6. Sea HCDH el valor decodificado. Fijar:

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

c) Decodificar cada símbolo de la clase de altura como sigue:

i) Decodificar la anchura delta para el símbolo como se describe en 6.5.7. Si el resultado de esta decodificación es OOB, todos los símbolos de esta clase de altura han sido decodificados; avanzar al paso 4 d). De otro modo, sea DW el valor decodificado y fijar:

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

ii) Si **SDHUFF** es **0** o **SDREFAGG** es **1**, decodificar el mapa de bits de símbolo como se describe en 6.5.8. Sea B_S el mapa de bits decodificado (este mapa de bits tiene una anchura de SYMWIDTH y una altura de HCHEIGHT). Fijar:

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

iii) Si **SDHUFF** es **1** y **SDREFAGG** es **0**, fijar:

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

iv) Fijar:

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

d) Si **SDHUFF** es **1** y **SDREFAGG** es **0**, decodificar el mapa de bits colectivo de clase de altura como se describe en 6.5.9. Sea B_{HC} el mapa de bits decodificado. Este mapa de bits tiene una anchura de TOTWIDTH y una altura de HCHEIGHT. Descomponer el mapa de bits B_{HC} para obtener los símbolos SDNEWSYMS[HCFIRSTSYM] a SDNEWSYMS[NSYMSDECODED – 1].

B_{HC} contiene los símbolos NSYMSDECODE – HCFIRSTSYM concatenados de izquierda a derecha, sin separaciones intermedias. Para cada I entre HCFIRSTSYM y NSYMSDECODED – 1:

- la anchura de SDNEWSYMS[I] es el valor de SDNEWSYMWIDTHS[I],
- la altura de SDNEWSYMS[I] es HCHEIGHT, y
- el mapa de bits SDNEWSYMS[I] se puede obtener extrayendo las columnas de B_{HC} de:

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

a

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

EJEMPLO – Las columnas 0 a SDNEWSYMWIDTHS[HCFIRSTSYM] – 1 de B_{HC} contienen el mapa de bits de SDNEWSYMS[HCFIRSTSYM], el primer símbolo de la clase de altura.

5) Determinar qué mapas de bits de símbolo se exportan desde este diccionario de símbolos, como se describe en 6.5.10. Estos mapas de bits se pueden dibujar a partir de los símbolos utilizados como entrada al procedimiento de decodificación de diccionario de símbolos así como los nuevos símbolos producidos por el procedimiento de decodificación.

NOTA 2 – No es preciso que todos los nuevos símbolos sean exportados; esto permite al diccionario definir algún símbolo, utilizarlo vía codificación de refinamiento/agregación para construir otros símbolos, y no tener que exportar realmente el símbolo original. Además, puesto que los símbolos de entrada pueden ser exportados, este diccionario puede, de hecho, copiar símbolos de otros diccionarios.

6.5.6 Altura delta (diferencia de) de clase de altura

Si **SDHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SDHUFFDH**.

Si **SDHUFF** es **0**, decodificar un valor utilizando el procedimiento decodificación aritmética de enteros IADH (véase el anexo A).

6.5.7 Anchura delta (diferencia de) anchura

Si **SDHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SDHUFFDW**.

Si **SDHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IADW (véase el anexo A).

En cualquiera de ambos casos es posible que el resultado de esta decodificación sea un valor fuera de banda (OOB).

6.5.8 Mapa de bits de símbolo

Este campo sólo está presente si **SDHUFF** = **0** o **SDREFAGG** = **1**. Este campo adopta una de dos formas posibles; **SDREFAGG** determina qué forma se utiliza.

6.5.8.1 Mapa de bits de símbolo codificado directamente

Si **SDREFAGG** es **0**, decodificar el mapa de bits del símbolo utilizando un procedimiento de decodificación de región genérica como se describe en 6.2. Fijar los parámetros de este procedimiento de decodificación como se muestra en el cuadro 16.

Cuadro 16 – Parámetros utilizados para decodificar el mapa de bits de un símbolo utilizando codificación de mapa de bits genérica

Nombre	Valor
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 Mapa de bits de símbolo codificado con refinamiento/agregación

Si **SDREFAGG** es **1**, el mapa de bits del símbolo es codificado por refinamiento y agregación de otros símbolos definidos previamente. Decodificar el mapa de bits como sigue:

1) Decodificar el número de ejemplares de símbolo contenido en la agregación, como se especifica en 6.5.8.2.1. Sea REFAGGNINST el valor decodificado.

2) Si REFAGGNINST es mayor que uno, decodificar el propio mapa de bits utilizando un procedimiento de decodificación de región de texto como se describe en 6.4. Fijar los parámetros de este procedimiento de decodificación como se muestra en el cuadro 17.

3) Si REFAGGNINST es igual a uno, decodificar el mapa de bits como se describe en 6.5.8.2.2.

Cuadro 17 – Parámetros utilizados para decodificar el mapa de bits de un símbolo utilizando decodificación de refinamiento/agregación

Nombre	Valor
SBHUFF	**SDHUFF**
SBREFINE	**1**
SBW	SYMWIDTH
SBH	HCHEIGHT
SBNUMINSTANCES	REFAGGNINST
SBSTRIPS	1
SBNUMSYMS	**SDNUMINSYMS** + NSYMSDECODED
SBSYMCODES	Véase 6.5.8.2.3.[a)]
SBSYMCODELEN	Véase 6.5.8.2.3.[b)]
SBSYMS	Véase 6.5.8.2.4.
SBDEFPIXEL	**0**
SBCOMBOP	OR
TRANSPOSED	**0**
REFCORNER	TOPLEFT
SBDSOFFSET	0
SBHUFFFS	Cuadro B.6[a)]
SBHUFFDS	Cuadro B.8[a)]
SBHUFFDT	Cuadro B.11[a)]
SBHUFFRDW	Cuadro B.15[a)]
SBHUFFRDH	Cuadro B.15[a)]
SBHUFFRDX	Cuadro B.15[a)]
SBHUFFRDY	Cuadro B.15[a)]
SBHUFFRSIZE	Cuadro 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**, este parámetro no tiene valor.
b) Si **SDHUFF** = **1**, este parámetro no tiene valor.

6.5.8.2.1 Número de ejemplares de símbolo en una agregación

Si **SDHUFF** es **1**, decodificar un valor utilizando la tabla Huffman especificada por **SDHUFFAGGINST**.

Si **SDHUFF** es **0**, decodificar un valor utilizando el procedimiento de decodificación aritmética de enteros IAAI (véase el anexo A).

6.5.8.2.2 Decodificación de un mapa de bits cuando REFAGGNINST = 1

Si un mapa de bits de símbolo se codifica mediante codificación de refinamiento/agregación, y sólo hay un símbolo en la agregación, el mapa de bits se decodifica como sigue. Se trata básicamente del método seguido por el procedimiento de decodificación de región de símbolo, con la salvedad de que cuando un valor es conocido, no se decodifica.

1) Fijar **SBHUFF = SDHUFF**.

2) Decodificar un ID símbolo como se describe en 6.4.10, utilizando los valores de **SBSYMCODES** y **SBSYMCODELEN** descritos en 6.5.8.2.3. Sea ID_I el valor decodificado.

3) Decodificar el desplazamiento X de refinamiento de ejemplar como se describe en 6.4.11.3. Si **SDHUFF** es **1**, utilizar el cuadro B.15 para **SBHUFFRDX**. Sea RDX_I el valor decodificado.

4) Decodificar el desplazamiento Y de refinamiento de ejemplar como se describe en 6.4.11.4. Si **SDHUFF** es **1**, utilizar el cuadro B.15 para **SBHUFFRDX**. Sea RDY_I el valor decodificado.

5) Si **SDHUFF** es **1**:

   a) decodificar el tamaño de datos del mapa de bits de refinamiento de ejemplar de símbolo como se describe en 6.4.11.5, utilizando el cuadro B.1 para **SBHUFFRSIZE**;

   b) saltar los bits restantes en el último byte leído.

6) Sea IBO_I **SBSYMS**[ID_I], donde **SBSYMS** es como se muestra en 6.5.8.2.4. El mapa de bits del símbolo es el resultado de aplicar el procedimiento de decodificación de región de refinamiento genérica descrito en 6.3. Fijar los parámetros de este procedimiento de decodificación como se muestra en el cuadro 18.

7) Si **SBHUFF** es **1**, saltar los bits restantes en el último byte leído. El número total de bytes procesados por el procedimiento de decodificación de región de refinamiento genérica debe ser igual al valor leído en el paso 5 a).

Cuadro 18 –Parámetros utilizados para decodificar el mapa de bits de un símbolo cuando REFAGGNINST = 1

Nombre	Valor
GRW	SYMWIDTH
GRH	HCHEIGHT
GRTEMPLATE	**SDRTEMPLATE**
GRREFERENCE	IBO_I
GRREFERENCEDX	RDX_I
GRREFERENCEDY	RDY_I
TPGRON	**0**
$\mathbf{GRATX_1}$	$\mathbf{SDRATX_1}$
$\mathbf{GRATY_1}$	$\mathbf{SDRATY_1}$
$\mathbf{GRATX_2}$	$\mathbf{SDRATX_2}$
$\mathbf{GRATY_2}$	$\mathbf{SDRATY_2}$

6.5.8.2.3 Fijación de SBSYMCODES y SBSYMCODELEN

Cuando **SDHUFF = 1**, fijar **SBSYMCODES** en una formación de códigos **SBNUMSYS**, en la que la longitud de cada código es:

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

y el código **SBSYMCODES** [*I*] es *I* (para *I* entre 0 y **SBNUMSYMS** – 1).

NOTA – De esta manera se fijan los códigos como códigos de igual longitud, asignados desde cero. Las longitudes de los códigos se calculan a partir del número máximo de símbolos disponibles en este diccionario de símbolos: todos los símbolos importados y todos los símbolos aquí definidos. Se produce un cierto grado de pérdida al elegir esta longitud de código y asignar estos códigos. Sin embargo, actuar de esta manera significa que ni las longitudes de códigos ni los códigos finalmente asignados a cada símbolo cambian durante el proceso de decodificación de este diccionario de símbolos.

De manera similar, cuando **SDHUFF** es **0**, **SBSYMCODELEN** deberá fijarse en:

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

con lo que la longitud de las cadenas de bits decodificadas utilizando IAID no cambiarán durante la decodificación de este diccionario de símbolos.

6.5.8.2.4 Fijación de SBSYMS

Fijar **SBSYMS** en una formación de símbolos **SDNUMINSYMS** + NSYMSDECODED, formada por la concatenación de la formación **SDINSYMS** y las primeras entradas NSYMSDECODED de la formación SDNEWSYMS.

6.5.9 Mapa de bits colectivo de clase de altura

Este campo sólo está presente si **SDHUFF = 1** y **SDREFAGG = 0**.

Este campo contiene los mapas de bits de todos los símbolos de la clase de altura, concatenados de izquierda a derecha, y codificados con MMR. Va precedido por la cuenta de su tamaño en bytes.

Este campo se decodifica como sigue:

1) Leer el tamaño de bytes utilizando la tabla Huffman **SDHUFFBMSIZE**. Sea BMSIZE el valor decodificado.

2) Saltar los bits restantes en el último byte leído.

3) Si BMSIZE es cero, el mapa de bits se almacena sin compresión, y el tamaño real en bytes es:

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

   Decodificar el mapa de bits leyendo todos estos bytes y tratarlo como HCHEIGHT filas de TOTWIDTH píxels, rellenándose cada fila hasta el límite de un byte con 0-7 bits **0**.

4) De otro modo, decodificar el mapa de bits utilizando un procedimiento de decodificación de mapa de bits genérica como se describe en 6.2. Fijar los parámetros de este procedimiento de decodificación como se muestra en el cuadro 19.

Cuadro 19 – Parámetros utilizados para decodificar el mapa de bits colectivo de una clase de altura

Nombre	Valor
MMR	**1**
GBW	TOTWIDTH
GBH	HCHEIGHT

5) Saltar los bits restantes en el último byte leído.

NOTA – BMSIZE se utiliza para determinar el número de bytes de datos codificados con MMR, y permitir así al codificador omitir el EOFB (véase 6.2.6); esto por lo general da lugar a una reducción en el tamaño de los datos codificados. También permite al codificador transmitir el mapa de bits sin comprimir en los casos en los cuales la codificación MMR daría lugar a una expansión.

6.5.10 Símbolos exportados

Los símbolos que pueden ser exportados desde un diccionario determinado incluyen cualquiera de los símbolos que son entradas al diccionario, más cualquiera de los símbolos definidos en el diccionario.

La formación de símbolos exportados desde el diccionario se produce decodificando un bit para cada uno de esos símbolos. Los bits constituyen una formación EXFLAGS de valores binarios **SDNUMINSYMS** + **SDNUMNEWSYMS**, cada uno de los cuales corresponde a un símbolo de entrada o un símbolo recién definido. Un bit **1** para un símbolo indica que el símbolo es exportado. Desde el diccionario se deben exportar exactamente **SDNUMEXSYMS** símbolos. El orden de los símbolos exportados es el orden producido concatenando la formación **SDINSYMS** y la formación SDNEWSYMS.

El procedimiento siguiente produce esta formación de símbolos exportados.

1) Fijar:

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

2) Decodificar un valor utilizando el cuadro B.1 si **SDHUFF** es **1**, o el procedimiento de decodificación aritmética de enteros IAEX si **SDHUFF** es **0**. Sea EXRUNLENGTH el valor que ha de ser decodificado.

3) Fijar de EXFLAGS[EXINDEX] a EXFLAGS[EXINDEX + EXRUNLENGTH – 1] en CUREXFLAG. Si EXRUNLENGTH = 0, este paso no cambia ningún valor.

4) Fijar:

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

5) Repetir los pasos 2) a 4) hasta que EXINDEX = **SDNUMINSYMS** + **SDNUMNEWSYMS**.

6) La configuración EXFLAGS contiene ahora **1** para cada símbolo que es exportado desde el diccionario, y **0** para cada símbolo que no es exportado.

7) Fijar:

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

8) Para cada valor *I* de 0 a **SDNUMINSYMS** + **SDNUMNEWSYMS** – 1, si EXFLAGS[*I*] = **1**, ejecutar los pasos siguientes:

   a) Si *I* < **SDNUMINSYMS**, fijar:

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

   b) Si *I* ≥ **SDNUMINSYMS** fijar

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

NOTA – La mayoría de los diccionarios exportarán exactamente los símbolos nuevos que definen; no exportarán ninguno de los símbolos de **SDINSYMS**. En este caso, los primeros valores **SDNUMINSYMS** en EXFLAGS son **0**, y los restantes valores **SDNUMNEWSYMS** son **1**.

6.6 Procedimiento de decodificación de región de semitonos

6.6.1 Descripción general

Este procedimiento de decodificación se utiliza para decodificar un mapa de bits decodificando una formación de valores, que se utilizan para dibujar patrones en una cuadrícula de semitonos. Estos patrones se combinan para formar el mapa de bits decodificado.

NOTA – Esta manera de codificar sirve para transmitir eficazmente un mapa de bits que contenga datos de imagen de semitonos periódicos, tales como los datos tremolados ordenados por conglomerados-puntos. Otras formas de datos de imagen de semitonos, tales como los datos de error difuso, se pueden convertir en esta forma mediante un decribado, o se pueden codificar en una forma que se parezca mucho al original utilizando codificación de mapa de bits genérica.

6.6.2 Parámetros de entrada

En el cuadro 20 se muestran los parámetros de este procedimiento de decodificación.

6.6.3 Valor de retorno

En el cuadro 21 se muestra la variable cuyo valor es el resultado de este procedimiento de decodificación.

6.6.4 Variables utilizadas en la decodificación

En el cuadro 22 se muestran las variables utilizadas por este procedimiento de decodificación.

Cuadro 20 – Parámetros del procedimiento de decodificación de región de semitonos

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
HBW	Entero	32	No	La anchura de la región
HBH	Entero	32	No	La altura de la región
HMMR	Entero	1	No	Indicación de si se utiliza codificación MMR
HTEMPLATE	Entero	2	No	El identificador de la plantilla[a)]
HNUMPATS	Entero	32	No	El número de patrones que pueden ser utilizados en esta región
HPATS	Formación de patrones			Una formación que contiene los patrones utilizados en esta región. Contiene patrones **HNUMPATS**.
HDEFPIXEL	Entero	1	No	El píxel por defecto de este mapa de bits
HCOMBOP	Operador			El operador de combinación utilizado en esta región de semitono. Puede tomar los valores OR, AND, XOR, XNOR y REPLACE.
HENABLESKIP	Entero	1	No	Indicación de si se saltan los valores de escala grises no necesarios[a)]
HGW	Entero	32	No	La anchura de la imagen de escala de grises
HGH	Entero	32	No	La altura de la imagen de escala grises
HGX	Entero	32	Sí	256 veces el desplazamiento horizontal del origen de la cuadrícula
HGY	Entero	32	Sí	256 veces el desplazamiento vertical del origen de la cuadrícula
HRX	Entero	16	No	256 veces la coordenada horizontal del vector de la cuadrícula
HRY	Entero	16	No	256 veces la coordenada vertical del vector de la cuadrícula
HPW	Entero	8	No	La anchura de cada patrón
HPH	Entero	8	No	La altura de cada patrón
a) No se utiliza si **HMMR = 1**.				

Cuadro 21 – Valor de retorno del procedimiento de decodificación de región de semitonos

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
HTREG	Mapa de bits			El mapa de bits de región decodificado

Cuadro 22 – Variables utilizadas en el procedimiento de decodificación de región de semitonos

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
n_g	Entero	32	No	Índice horizontal para el valor de escala de grises vigente
m_g	Entero	32	No	Índice vertical para el valor de escala de grises vigente
x	Entero	32	Sí	La coordenada horizontal para el patrón correspondiente al valor de escala de grises vigente
y	Entero	32	Sí	La coordenada vertical para el patrón correspondiente al valor de escala de grises vigente
HSKIP	Mapa de bits			Soltar máscara. HSKIP es **HGW** por **HGH** píxels[a)]
HBPP	Entero	32	No	El número de bits por valor en la formación de valores de escala de grises
GI	De formación			Formación de valores de escala de grises. GI es una formación de **HGW** por **HGH**, cada entrada en la cual es un entero sin signo de HBPP bits
a) No se utiliza si **HENABLESKIP = 0**.				

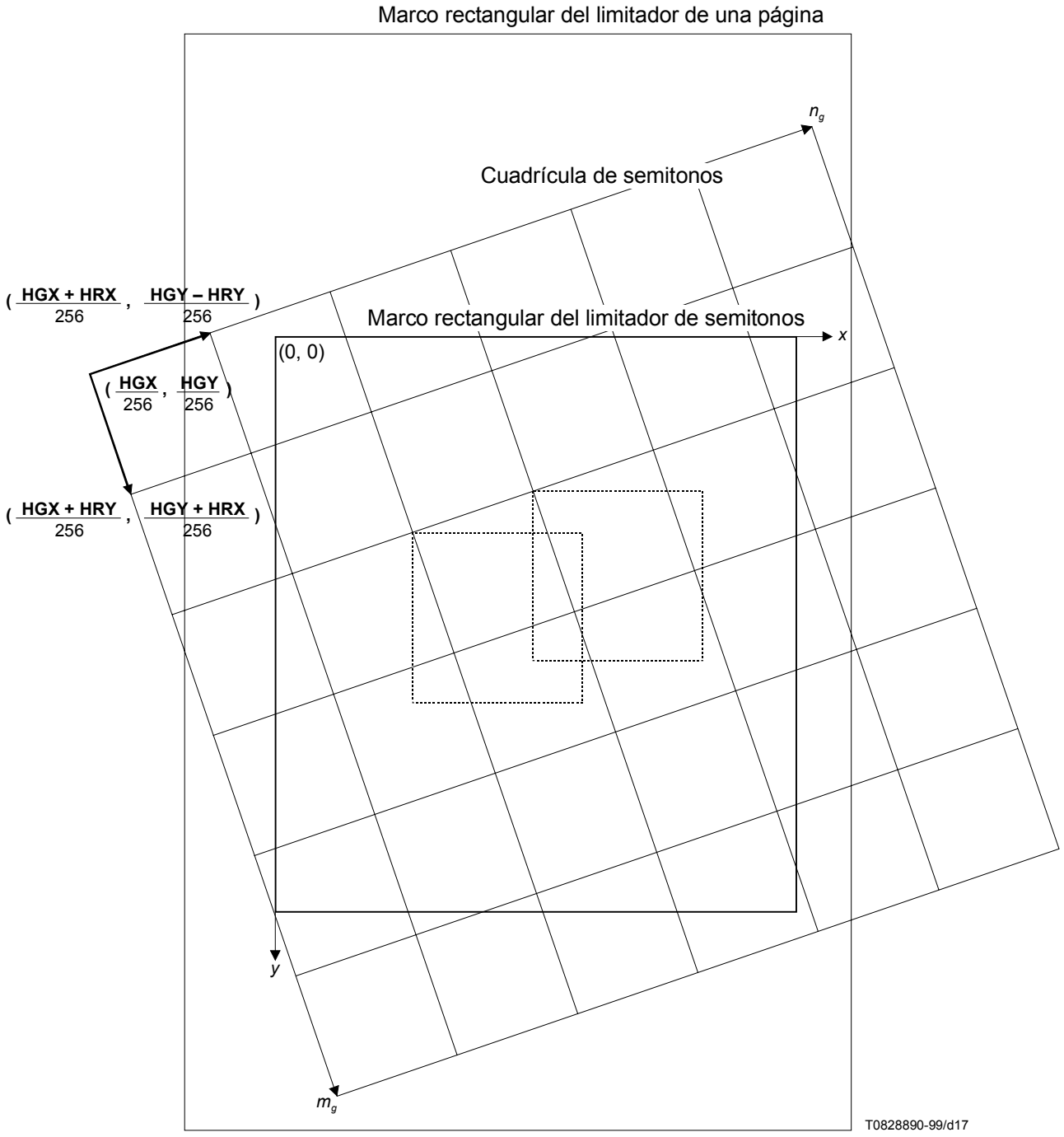


Figura 24 - Especificación de sistemas de coordenadas y parámetros de cuadrícula

6.6.5 Decodificación de la región de semitonos

Un mapa de bits de semitonos codificados se representa mediante un conjunto de ejemplares de patrón. Cada ejemplar codifica un patrón. La ubicación de cada patrón no se codifica explícitamente sino que viene dada por una cuadrícula que abarca la totalidad del mapa de bits de semitonos. El origen de la cuadrícula de semitonos es especificado por los parámetros **HGX** y **HGY**. El periodo de la cuadrícula es especificado por los parámetros **HRX** y **HRY** (véase la figura 24). A **HGX**, **HGY**, **HRX** y **HRY** se les aplica un factor de escala de 256, lo que significa que el origen de la cuadrícula y el periodo de la cuadrícula tienen una parte fraccionaria de 8 bits.

NOTA 1 – Se señala que **HRX** y **HRY** son valores sin signo; es decir, son valores siempre superiores o iguales a cero. Esto significa que el vector de cuadrícula sólo puede hallarse en un único cuadrante. A pesar de esta restricción, cualquier cuadrícula de semitonos puede codificarse mediante un ajuste adecuado de **HGX** y **HGY**: **HGX** y **HGY** se deben fijar de manera que el origen de la cuadrícula esté en la esquina situada más a la izquierda. Ésta es la esquina superior izquierda en el caso en que la cuadrícula esté alineada con los ejes, o bien se trata de una cuadrícula alineada con los ejes con una ligera rotación en el sentido contrario a las agujas del reloj (como se muestra en la figura 24), y es la esquina inferior izquierda si se trata de una cuadrícula alineada con los ejes con una ligera rotación en el sentido de las agujas del reloj.

Los patrones posibles se dan en un diccionario. La identidad de un patrón se especifica mediante un índice que normalmente representará el valor de escala de grises del patrón.

NOTA 2 – Se utiliza la expresión valor de escala de grises del índice para ilustrar la idea de compresión. En la presente Recomendación | Norma Internacional no se exige que el índice corresponda de hecho al valor de escala de grises.

El resultado de la decodificación de un mapa de bits de semitonos es un mapa de bits que se produce ejecutando los pasos siguientes:

1) Llenar un mapa de bits HTREG, o el tamaño dado por **HBW** y **HBH**, con el valor **HDEFPIXEL**.
2) Si **HENABLESKIP** es **1**, calcular un mapa de bits HSKIP como se muestra en 6.6.5.1.
3) Fijar HFPP en $\lceil \log_2 (\mathbf{HNUMPATS}) \rceil$.
4) Decodificar una GI de imagen de tamaño **HGW** por **HGH** con HBPP bits por píxel utilizando el procedimiento de decodificación de imagen de escala de grises que se describe en el anexo C. Fijar los parámetros de este procedimiento de decodificación como se muestra en el cuadro 23.

Cuadro 23 – Parámetros utilizados para decodificar la formación de valores de escala de grises de una región de semitonos

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

a) Si **HENABLESKIP** = **0**, este parámetro no tiene valor.
b) Si **HMMR** = **1**, este parámetro no tiene valor.

Sea GI el resultado de invocar este procedimiento de decodificación

5) Situar secuencialmente los patrones correspondientes a los valores de GI en HTREG por el procedimiento descrito en 6.6.5.2. En la figura 24 se ilustra el procedimiento de reproducción. El esquema de dos patrones se indica mediante cuadrados de líneas de puntos.
6) Una vez que todos los patrones han sido situados en el mapa de bits, el contenido actual del mapa de bits de semitonos codificados es el resultado que obtendrá cada decodificador, tanto si ejecuta como si no, esta secuencia exacta de pasos.

NOTA 3 – Si **HGX** es 0, **HGY** es 0, **HRX** es igual a **HPW** × 256 y **HRY** es 0, la cuadrícula es simple: está alienada con los ejes, la dirección principal es la horizontal y el paso de cuadrícula es igual al tamaño de los patrones. En este caso, es posible optimizar el proceso de dibujo, ya que los patrones no pueden superponerse.

6.6.5.1 Cálculo de HSKIP

El mapa de bits HSKIP contiene **1** en un píxel si el dibujo de un patrón en la ubicación correspondiente en la cuadrícula de semitonos no afecta a ninguno de los píxels de HTREG. Se calcula como sigue:

1) Para cada valor de m_g entre 0 y **HGH** – 1, empezando en 0, ejecutar los pasos siguientes:
   a) Para cada valor de n_g entre 0 y **HGW** – 1, empezando en 0, ejecutar los pasos siguientes:
      i) Fijar:

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

      ii) Si ((x + **HPW** ≤ 0) OR (x ≥ **HBW**) OR (y + **HPH** ≤ 0) OR (y ≥ **HBH**)), fijar:

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

      de no ser así, fijar:

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

6.6.5.2 Reproducción de los patrones

Dibujar los patrones en HTREG utilizando el procedimiento siguiente:

1) Para cada valor de n_g entre 0 y **HGH** – 1, empezando en 0, ejecutar los pasos siguientes:

   a) Para cada valor de m_g entre 0 y **HGW** – 1, empezando en 0, ejecutar los pasos siguientes:

      i) Fijar:

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

      ii) Dibujar el patrón **HPATS**[GI[n_g, m_g]] en HTREG de tal manera que su píxel superior izquierdo quede en la ubicación (x, y) en HTREG.

      En HTREG se dibuja un patrón como sigue. Cada píxel del patrón se combinará con el valor vigente del píxel correspondiente en el mapa de bits de semitonos codificado, utilizando el operador de combinación especificado por **HCOMBOP**. Los resultados de cada combinación se escribirán en ese píxel en el mapa de bits de semitonos codificado.

      Si alguna parte de un patrón decodificado, cuando esté situado en la ubicación (x, y), queda fuera del mapa de bits de semitonos codificados real, dicha parte del patrón será ignorada en el proceso de combinación del patrón con el mapa de bits.

NOTA – La imagen de escala de grises puede ser utilizada por el decodificador para obtener una buena reproducción del semitono en un dispositivo de salida multinivel de resolución espacial limitada, por ejemplo, la pantalla de un ordenador. La utilización de la imagen de escala de grises a tales fines queda fuera del ámbito de aplicación de la presente Recomendación | Norma Internacional.

La imagen de escala de grises se codifica aplicando la codificación de plano de bits de forma que el decodificador reciba la imagen de escala de grises de manera progresiva. El decodificador puede, por consiguiente, reproducir una imagen en semitonos utilizando los valores de la escala de grises cuantificados como índices. Esas imágenes en semitonos intermedias no deberán influir en el mapa de bits de semitonos codificado final.

6.7 Procedimiento de decodificación de diccionario de patrones

6.7.1 Descripción general

Este procedimiento de decodificación se utiliza para decodificar un conjunto de patrones de tamaño fijo; los patrones pueden ser utilizados a continuación por los procedimientos de decodificación de región de semitonos.

6.7.2 Parámetros de entrada

En el cuadro 24 se muestran los parámetros de este procedimiento de decodificación.

Cuadro 24 – Parámetros del procedimiento de decodificación de diccionario de patrones

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
HDMMR	Entero	1	No	Indica si se utiliza MMR
HDPW	Entero	32	No	La anchura de cada patrón
HDPH	Entero	32	No	La altura de cada patrón
GRAYMAX	Entero	32	No	El mayor valor de la escala de grises para el que se da un patrón
HDTEMPLATE	Entero	2	No	La plantilla utilizada para codificar los patrones[a)]
[a)] No se utiliza si **HDMMR** = **1**.				

6.7.3 Valor de retorno

En el cuadro 25 se muestra la variable cuyo valor es el resultado de este procedimiento de decodificación.

Cuadro 25 –Valor de retorno del procedimiento de decodificación de diccionario de patrones

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
HDPATS	Formación de patrones			Los patrones exportados por este diccionario de patrones. Contiene **GRAYMAX** + 1 patrones

6.7.4 Variables utilizadas en la decodificación

En el cuadro 26 se muestran las variables utilizadas por este procedimiento de decodificación.

Cuadro 26 – Variables utilizadas en el procedimiento de decodificación del diccionario de patrones

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
GRAY	Entero	32	No	Índice de escala de grises
B_{HDC}	Mapa de bits			El mapa de bits colectivo de diccionario
B_P	Mapa de bits			Un mapa de bits de tamaño **HDPW** por **HDPH**

6.7.5 Decodificación del diccionario de patrones

El resultado de la decodificación de un diccionario de patrones es un conjunto de patrones: **HDPATS[0]** … **HDPATS[GRAYMAX]**. Estos patrones serán los producidos por los pasos siguientes:

1) Crear un mapa de bits B_{HDC}. La altura de este mapa de bits es **HDPH**. La anchura del mapa de bits es (**GRAYMAX** + 1) × **HDPW**. Este mapa de bits contiene todos los patrones concatenados de izquierda a derecha.

2) Decodificar el mapa de bits colectivo utilizando un procedimiento de decodificación de región genérica como se describe en 6.2. Fijar los parámetros de este procedimiento de decodificación como se muestra en el cuadro 27.

Cuadro 27 – Parámetros utilizados para decodificar el mapa de bits colectivo de un diccionario de patrones

Nombre	Valor
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) Si **HDMMR** = **1**, este parámetro no tiene valor.
b) Si **HDMMR** = **1** o **DHTEMPLATE** ≠ **0**, este parámetro no tiene valor.

3) Fijar:

$$\text{GRAY} = 0$$

4) Mientras GRAY ≤ **GRAYMAX**:

   a) Sea B_P la subimagen de B_{HDC} que consta de **HPH** filas y **HDPW** × GRAY a **HDPW** × (GRAY + 1) – 1 columnas. Fijar:

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

   b) Fijar:

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

7 Procedimiento de decodificación de control

7.1 Descripción general

Este procedimiento de decodificación controla la invocación de todos los demás procedimientos de decodificación. El tren binario codificado consta de un conjunto de segmentos, cada uno de los cuales contiene una parte de los datos necesarios para la decodificación. Hay varios tipos de segmentos diferentes.

Un segmento tiene dos partes: una parte encabezamiento de segmento y una parte datos de segmento. Todos los tipos de segmento utilizan un formato común para el encabezamiento de segmento, pero formatos diferentes para los datos de segmento.

Algunos segmentos dan información sobre la estructura del documento: comienzo de página, fin de página, y así sucesivamente. Otros segmentos codifican regiones, utilizadas a su vez para producir la imagen decodificada de una determinada página. Otros, en fin, ("segmentos de diccionario") no hacen ni una cosa ni otra, pero definen en cambio recursos que pueden ser utilizados por segmentos que codifican regiones.

Un segmento puede ser asociado con alguna página, o no ser asociado con ninguna página. Un segmento puede hacer referencia a otros segmentos precedentes. Un segmento puede incluir también bits de retención para el segmento al que hace referencia, y para él mismo; indican cuándo puede descartar el decodificador los datos creados al decodificar un segmento.

EJEMPLO – Un segmento de región de texto puede hacer uso de los símbolos definidos en segmentos de diccionario de símbolos precedentes. Esto se indica en el encabezamiento de segmento de la región de texto haciendo referencia a esos segmentos de diccionario de símbolos.

En 7.2 se describe el formato de los encabezamientos de segmento. En 7.3 se definen los tipos de segmentos. En 7.4 se define la sintaxis de cada tipo de segmento.

En lo que sigue, se hacen algunas referencias a segmentos "precedentes" y "siguientes" (y otras indicaciones que implican un orden de los segmentos). Estos términos se definen con referencia al orden impuesto a los segmentos por sus números de segmento: un segmento precede a todos los segmentos cuyos números de segmento son mayores que su número de segmento. En las organizaciones secuenciales y de acceso aleatorio (véanse D.1 y D.2) los segmentos deben aparecer en el archivo en orden creciente de número de segmento. Sin embargo, en la organización insertada (véase D.3) no es así porque los segmentos JBIG2 están encapsulados en otro formato de archivo.

NOTA – Es posible que haya discontinuidad en la numeración de los segmentos. Un fichero JBIG2 puede contener segmentos con los números 2, 3, 4, 8 y 10, por ejemplo, debido a la edición. Los números de segmento podrían haber sido contiguos al principio pero, en algún momento de la vida del fichero, se suprimieron algunas páginas y los segmentos restantes no se renumeraron.

La parte encabezamiento de un segmento empieza y termina siempre en una frontera de bytes.

La parte datos de un segmento empieza y termina siempre en una frontera de bytes. Cualesquiera bits no utilizados del byte final de un segmento deben contener **0**, y no serán examinados por el decodificador.

La parte encabezamiento de segmento y la parte datos de segmento de un segmento no tienen por qué presentarse de forma contigua en el tren binario que se decodifica. Véase en el anexo D una organización en la que la parte encabezamiento de segmento de un segmento se puede almacenar a cierta distancia de la parte datos de segmento de ese segmento.

Esta cláusula contiene figuras que describen diversas partes de los datos codificados, por ejemplo, las figuras 25 y 31. En dichas figuras se utilizan los convenios siguientes:

- El primer byte encontrado en el tren binario es el situado en el extremo izquierdo.
- Los campos cuyos tamaños son fijos, y que están siempre presentes, se destacan con líneas finas.

- Los campos cuyos tamaños no son fijos, o que no están presentes en todos los casos, o cuyas estructuras se describen por completo en otro lugar, se señalan con líneas gruesas.
- Algunas figuras (por ejemplo, la 25) están divididas en campos, cada uno de los cuales tiene una longitud que es un número entero de bytes. En estas figuras, marcas divisoras que descienden desde la parte superior de la figura indican fronteras de bytes, y los campos están separados por líneas cuyo trazado abarca la altura total de la figura.
- Las demás figuras se dividen en campos, cado uno de los cuales tiene una longitud que es un número entero de bits, que corresponde a un número entero de bytes. En estas figuras, cortas marcas divisoras que ascienden desde la parte inferior de la figura muestran las fronteras de bits. Los campos están separados por marcas divisoras más largas que ascienden desde la parte inferior de la figura. El número de cada bit se muestra debajo de la figura.

7.2 Sintaxis de encabezamiento de segmento

7.2.1 Campos de encabezamiento de segmento

Un encabezamiento de segmento contiene los campos que se muestran en la figura 25 y se describen a continuación.

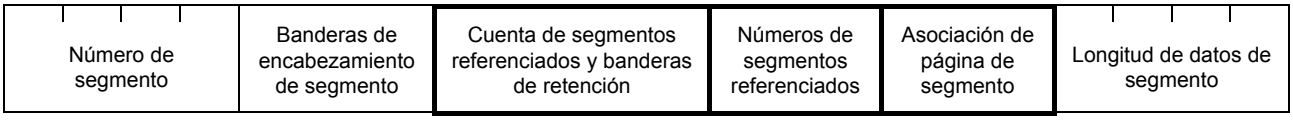


Figura 25 – Estructura de encabezamiento de segmento

Número de segmento – Véase 7.2.2.

Bandera de encabezamiento de segmento – Véase 7.2.3.

Cuenta de segmentos referenciados y banderas de retención – Véase 7.2.4.

Campos de números de segmentos referenciados – Véase 7.2.5.

Asociación de página de segmento – Véase 7.2.6.

Longitud de datos de segmento – Véase 7.2.7.

7.2.2 Número de segmento

Este campo de cuatro bytes contiene el número de segmento del segmento. La gama válida de números de segmentos es de 0 a 4294967295 (`0xFFFFFFFF`) inclusive. Como se mencionó previamente, es posible que existan separaciones intermedias en la numeración de los segmentos.

7.2.3 Banderas de encabezamiento de segmento

Este campo es de un byte. Los bits definidos se muestran en la figura 26 y se describen a continuación.

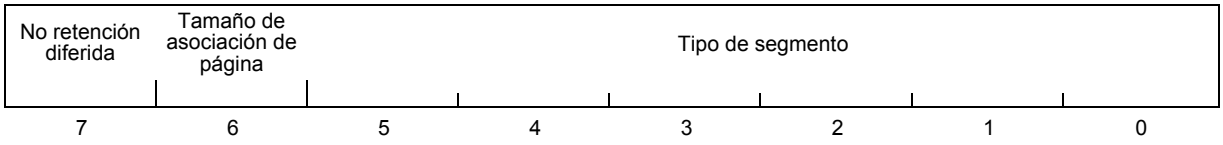


Figura 26 – Banderas de encabezamiento de segmento

Bits 0-5 Tipo de segmento. Véase 7.3.

Bit 6 Tamaño de campo de asociación de página. Véase 7.2.6.

Bit 7 No retención diferida. Si este bit es **1**, este segmento sólo es señalado como retenido por sí mismo y por sus segmentos de extensión anejos, y es señalado como no retenido por los últimos segmentos de extensión anejos. Un segmento de extensión es un segmento de extensión anejo cuando sólo hace referencia a un segmento, y los únicos segmentos (si hay alguno) entre él y ese segmento referenciado son otros segmentos de extensión que también se refieren únicamente a ese segmento referenciado.

NOTA – La finalidad de este bit es indicar al decodificador que al segmento sólo hacen referencia un número pequeño de segmentos de extensión. El decodificador puede efectuar algunas acciones costosas cuando se señalan segmentos como retenidos, pero si la retención sólo beneficia a los segmentos de extensión anejos al segmento, es posible que esas acciones puede que no sean necesarias. Esto es algo que conviene saber por adelantado.

7.2.4 Cuenta de segmentos referenciados y banderas de retención

Este campo contiene uno o más bytes que indican a cuántos otros segmentos hace referencia este segmento, y cuáles contienen datos necesarios según este segmento.

NOTA – Las necesidades de memoria del decodificador se pueden reducir haciéndole saber cuándo puede olvidar los datos representados por algún segmento previo.

El número de bytes de este campo depende del número de segmentos a los que hace referencia este segmento. Si este segmento se refiere a cuatro o menos segmentos, este campo tiene una longitud de un byte. Si se refiere a más de cuatro segmentos, este campo tiene una longitud de $4 + \lceil (R + 1)/8 \rceil$ bytes, donde R es el número de segmentos a los que hace referencia este segmento.

EJEMPLO – Si este segmento hace referencia a otros cinco a siete segmentos, el campo tiene una longitud de cinco bytes; si hace referencia a otros ocho a quince segmentos, el campo tiene una longitud de seis bytes.

Los tres bits más significativos del primer byte de este campo determinan la longitud del campo. Si el valor de este subcampo de tres bits está comprendido entre 0 y 4, el campo tiene una longitud de un byte. Si el valor de este subcampo de tres bits es 7, el campo tiene una longitud de por lo menos cinco bytes. Este subcampo de tres bits no debe contener valores de 5 y 6.

Si el campo tiene una longitud de un byte, ese byte se formatea como se muestra en la figura 27 y se describe a continuación.

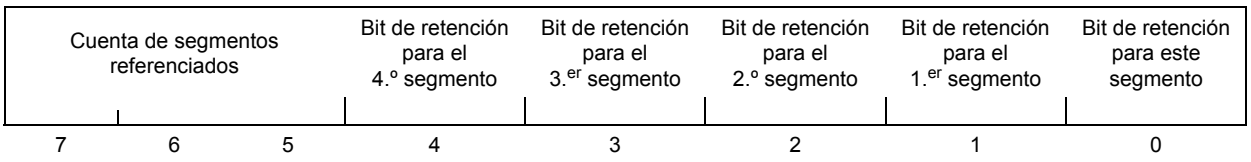


Figura 27 – Cuenta de segmentos referenciados y banderas de retención – Forma corta

Bit 0 Bit de retención para este segmento.

Bit 1 Bit de retención para el primer segmento referenciado. Si este segmento no hace referencia a ningún otro segmento, este campo debe contener **0**.

Bit 2 Bit de retención para el segundo segmento referenciado. Si este segmento hace referencia a menos de otros dos segmentos, este campo debe contener **0**.

Bit 3 Bit de retención para el tercer segmento referenciado. Si este segmento hace referencia a menos de otros tres segmentos, este campo debe contener **0**.

Bit 4 Bit de retención para el cuarto segmento referenciado. Si este segmento hace referencia a menos de otros cuatro segmentos, este campo debe contener **0**.

Bits 5-7 Cuenta de segmentos referenciados. Este campo puede tomar valores comprendidos entre cero y cuatro. Especifica el número de segmentos a los que hace referencia este segmento.

Si el campo es de formato largo (su longitud es de al menos cinco bytes), consta de un campo inicial de cuatro bytes, seguido por una sucesión de campos de un byte. El campo inicial de cuatro bytes se formatea como sigue.

Bits 0-28 Cuenta de segmentos referenciados. Este campo especifica el número de segmentos a los que hace referencia este segmento.

Bits 29-31 Indicación de formato largo. Este campo debe contener el valor 7.

El primer campo de un byte que sigue al campo inicial de cuatro bytes se formatea como sigue.

Bit 0 Bit de retención para este segmento.

Bit 1 Bit de retención para el primer segmento referenciado.

Bit 2 Bit de retención para el segundo segmento referenciado.

Bit 3 Bit de retención para el tercer segmento referenciado.

Bit 4 Bit de retención para el cuarto segmento referenciado.

Bit 5 Bit de retención para el quinto segmento referenciado. Si este segmento hace referencia a menos de otros cinco segmentos, este campo debe contener **0**.

Bit 6 Bit de retención para el sexto segmento referenciado. Si este segmento hace referencia a menos de otros seis segmentos, este campo debe contener **0**.

Bit 7 Bit de retención para el séptimo segmento referenciado. Si este segmento hace referencia a menos de otros siete segmentos, este campo debe contener **0**.

El segundo campo de un byte, si está presente, contiene bits de retención para los segmentos referenciados octavo a decimoquinto; los bits correspondientes a cualesquiera segmentos más allá de la cuenta de segmentos efectivamente referenciados deben ser **0**. Los campos de un byte subsiguientes se formatean de manera similar.

Si el valor del bit de retención para este segmento es **0**, ningún segmento puede hacer referencia a este segmento.

Si el valor del bit de retención para el primer segmento referenciado es **0**, ningún segmento después de éste puede hacer referencia al primer segmento al que hace referencia este segmento (es decir, este segmento es el último que hace referencia a ese otro segmento). Los valores de bits de retención posteriores tendrán significados similares: si el valor del bit de retención para el K-ésimo segmento referenciado es **0**, ningún segmento posterior a éste puede hacer referencia al K-ésimo segmento al que este segmento hace referencia.

7.2.5 Números de segmentos referenciados

Este campo contiene los números de segmento de los segmentos a los que este segmento hace referencia, si es que hay alguno. El número de valores en este campo viene determinado por la cuenta de segmentos referenciados y el campo de banderas de retención. Cada valor es el número de segmento de un segmento al que hace referencia este segmento. Si un segmento se refiere a otros segmentos sólo debe referirse a segmentos con número de segmento más bajo. Cuando el número del segmento actual es 256 o menos, cada número de segmento referenciado tiene una longitud de un byte. De otro modo, cuando el número del segmento actual es 65536 o menos, cada número de segmento referenciado tiene una longitud de dos bytes. De no ser así, cada número de segmento referenciado tiene una longitud de cuatro bytes.

7.2.6 Asociación de página de segmento

Este campo codifica el número de la página a la que pertenece el segmento. La primera página debe ser numerada "1". Este campo puede contener un valor de cero; este valor indica que este segmento no está asociado con ninguna página.

Un segmento que tiene una asociación de página de segmento distinta de cero, sólo puede ser referenciado por segmentos que tengan el mismo valor de asociación de página de segmento que él.

Este campo tiene una longitud de un byte si el bit bandera de tamaño de campo de asociación de página de este segmento es **0**, y tiene una longitud de cuatro bytes si el bit bandera de tamaño de campo de asociación de página de este segmento es **1**.

NOTA – La mayoría de los documentos tienen menos de 256 páginas, por lo que este campo tiene un formato corto que puede retener valores de 0 a 255 en un solo byte. El campo de asociación de página para segmentos no asociados puede tener también una longitud de un solo byte.

7.2.7 Longitud de datos de segmento

Este campo de cuatro bytes contiene la longitud de la parte datos de segmento del segmento, en bytes.

Si el tipo de segmento es "Región genérica inmediata", el campo de longitud puede contener el valor `0xFFFFFFFF`. Este valor significa que la longitud de la parte datos del segmento es desconocida en el momento en que se escribe el encabezamiento del segmento (por ejemplo, en una aplicación de reproducción inmediata, tal como el facsímil). En este caso, la longitud verdadera de la parte datos del segmento se determinará examinando los datos: si el segmento utiliza codificación aritmética basada en la plantilla, la parte datos del segmento termina con la secuencia de dos bytes `0xFF 0xAC` seguida por una cuenta de filas de cuatro bytes. Si el segmento utiliza codificación MMR, la parte datos del segmento termina con una secuencia de dos bytes `0×00 0×00` seguida por una cuenta de filas de cuatro bytes. La forma de codificación utilizada por el segmento se puede determinar examinando el decimoctavo byte de su parte datos del segmento, y las secuencias finales pueden producirse en cualquier lugar después de ese decimoctavo byte.

NOTA – Dada una lista de encabezamientos de segmento en la organización de acceso aleatorio (véase la figura D.2), un decodificador puede elaborar un mapa del resto del fichero conociendo la longitud de los datos asociados con cada segmento. Esto le permite acceder de manera aleatoria.

7.2.8 Ejemplo de encabezamiento de segmento

EJEMPLO 1 – Un encabezamiento de segmento formado por la secuencia de bytes

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

se descompone como sigue:

0x00 0x00 0x00 0x20 Este número de segmento es 0x00000020, o 32 en base decimal.

0x86 El tipo de este segmento es 6. Su campo de asociación de página tiene una longitud de un byte. Sólo es retenido por sus segmentos de extensión anejos.

0x6B Este segmento hace referencia a otros tres segmentos. A él hacen referencia algunos otros segmentos. Ésta es la última referencia al segundo de los tres segmentos a los que él hace referencia.

0x02 0x1E 0x05 Los tres segmentos a los que él hace referencia son los de número 2, 30 y 5.

0x04 Este segmento está asociado con la página número 4.

EJEMPLO 2 – Un encabezamiento de segmento formado por la secuencia de bytes, en hexadecimal:

```
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
```

se descompone como sigue:

00 00 02 34 El número de este segmento es 0x00000234, o 564 en base decimal.

40 El tipo de este segmento es 0. Su campo de asociación de página tiene una longitud de cuatro bytes.

E0 00 00 09 El campo de cuenta de segmentos referenciados de este segmento tiene un formato largo. Este segmento hace referencia a otros nueve segmentos.

02 FD A este segmento hacen referencia algunos otros segmentos. Ésta es la última referencia a los segmentos primero y octavo de los nueve segmentos a los que él hace referencia.

01 00 ... 02 04 Los nueve segmentos a los que él hace referencia se identifican, cada uno, mediante dos bytes, ya que este número de segmento se halla entre 256 y 65535. Los segmentos a los que él hace referencia son, en base decimal, los de número 256, 2, 30, 5, 512, 513, 514, 515 y 516.

00 00 04 01 Este segmento está asociado con la página número 1025.

7.3 Tipos de segmento

Cada segmento tiene un tipo determinado. Este tipo especifica el tipo de los datos asociados con el segmento. El tipo restringe los otros segmentos a los que él puede hacer referencia, y los otros segmentos que pueden hacer referencia a él. Estas restricciones se detallan en 7.3.1.

El tipo de segmento es un número comprendido entre 0 y 63, ambos inclusive. No todos los valores están permitidos. La lista de tipos de segmento permitidos, sus nombres completos, y la subcláusula en que se define sus formato son:

0 Diccionario de símbolos – Véase 7.4.2.

4 Región de texto intermedia – Véase 7.4.3.

6 Región de texto inmediata – Véase 7.4.3.

7 Región de texto sin pérdida inmediata – Véase 7.4.3.

16 Diccionario de patrones – Véase 7.4.4.

20 Región de semitonos intermedia – Véase 7.4.5.

22 Región de semitonos inmediata – Véase 7.4.5.

23 Región de semitonos sin pérdida inmediata – Véase 7.4.5.

36 Región genérica intermedia – Véase 7.4.6.

38 Región genérica inmediata – Véase 7.4.6.

39 Región genérica sin pérdida inmediata – Véase 7.4.6.

40 Región de refinamiento genérica intermedia – Véase 7.4.7.

42 Región de refinamiento genérica inmediata – Véase 7.4.7.

43 Región de refinamiento genérica sin pérdida inmediata – Véase 7.4.7.

48 Información de página – Véase 7.4.8.

49 Fin de página – Véase 7.4.9.

50 Fin de franja – Véase 7.4.10.

51 Fin de fichero – Véase 7.4.11.

52 Perfiles – Véase 7.4.12.

53 Tablas – Véase 7.4.13.

62 Extensión – Véase 7.4.14.

Todos los demás tipos de segmento están reservados y no deben ser utilizados.

NOTA – Estos números de tipo de segmento se atribuyen de acuerdo con las reglas siguientes. Los dos bits de orden alto (bits 4-5) de este número especifican el tipo primario del segmento, y los cuatro bits de orden bajo (bits 0-3) especifican el tipo secundario del segmento.

Los tipos primarios son:

0 Datos de mapa de bits de símbolo.

1 Datos de mapa de bits de semitono.

2 Datos de mapa de bits genérico.

3 Metadatos.

A los tipos primarios 0-2 se hace referencia colectivamente como tipos de región.

Para los tipos de región la interpretación de los cuatro bits de orden bajo es:

Bit 0 Si este bit es **1**, indica que el segmento forma parte de alguna región de la página sin pérdida.

Bit 1 Si este bit es **1**, indica que el segmento se puede dibujar inmediatamente en el mapa de bits de la página. Si este bit es **0**, indica que el segmento es un segmento intermedio. Véase 8.2.

Bits 2-3 Estos dos bits definen un subtipo del tipo primario:

0 Diccionario
1 Región directa
2 Región de refinamiento

Para los metadatos, las interpretaciones de los cuatro bits de orden bajo son:

0 Información de página.

1 Fin de página.

2 Fin de franja.

3 Fin de fichero.

4 Perfiles.

5 Tablas.

6-13 Reservado.

14 Extensión.

15 Reservado.

A los segmentos de tipos "región de texto intermedia", "región de texto inmediata", "región de texto sin pérdida inmediata", "región de semitonos intermedia", "región de semitonos inmediata", "región de semitonos sin pérdida inmediata", "región genérica intermedia", "región genérica inmediata", "región genérica sin pérdida inmediata", "región de refinamiento genérica intermedia", "región de refinamiento genérica inmediata" y "región de refinamiento genérica sin pérdida inmediata" se hace referencia colectivamente como "segmentos de región".

A los segmentos de tipos "región de texto intermedia", "región de texto inmediata", "región de texto sin pérdida inmediata", "región de semitonos intermedia", "región de semitonos inmediata", "región de semitonos sin pérdida inmediata", "región genérica intermedia", "región genérica inmediata", "región genérica sin pérdida inmediata" se hace referencia colectivamente como "segmentos de región genérica".

A los segmentos de tipos "región de texto intermedia", "región de semitonos intermedia", "región genérica intermedia" y "región de refinamiento genérica intermedia" se hace referencia colectivamente como "segmentos de región intermedia".

A los segmentos de tipos "región de texto inmediata", "región de texto sin pérdida inmediata", "región de semitonos inmediata", "región de semitonos sin pérdida inmediata", "región genérica inmediata", "región genérica sin pérdida inmediata", "región de refinamiento genérica inmediata" y "región de refinamiento genérica sin pérdida inmediata" se hace referencia colectivamente como "segmentos de región inmediata".

A los segmentos de tipos "región de refinamiento genérica intermedia", "región de refinamiento genérica inmediata" y "región de refinamiento genérica sin pérdida inmediata" se hace referencia colectivamente como "segmentos de región de refinamiento".

7.3.1 Reglas para referencias a segmentos

Las reglas para las referencias a segmentos son como sigue:

- A un segmento de región intermedia sólo puede hacer referencia otro segmento de no extensión; pueden hacer referencia a él cualquier número de segmentos de extensión.
- A un segmento de tipo "diccionario de símbolos" (tipo 0) pueden hacer referencia cualquier número de segmentos de tipo "diccionario de símbolos" y hasta cuatro segmentos de tipo "tablas".
- A un segmento de tipo "región de texto intermedia", "región de texto inmediata" o "región de texto sin pérdida inmediata" (tipo 4, 6 ó 7) pueden hacer referencia a cualquier número de segmentos de tipo "diccionario de símbolos" y hasta ocho segmentos de tipo "tablas".
- Un segmento de tipo "diccionario de patrones" (tipo 16) no debe hacer referencia a ningún otro segmento.
- Un segmento de tipo "región de semitonos intermedia", "región de semitonos inmediata" o "región de semitonos sin pérdida inmediata" (tipo 20, 22 ó 23) sólo puede hacer referencia a exactamente un segmento, y dicho segmento debe ser del tipo "diccionario de patrones".
- Un segmento de tipo "región genérica intermedia", "región genérica inmediata" o "región genérica sin pérdida inmediata" (tipo 36, 38 ó 39) no debe hacer referencia a ningún otro segmento.
- Un segmento de tipo "región de refinamiento genérica intermedia" (tipo 40) debe hacer referencia a exactamente un segmento distinto. Este otro segmento debe ser un segmento de región intermedia.
- Un segmento de tipo "región de refinamiento genérica inmediata" o "región de refinamiento genérica sin pérdida inmediata" (tipo 42 ó 43) puede hacer referencia a cero segmentos o exactamente a un segmento. Si hace referencia a otro segmento, ese segmento debe ser un segmento de región intermedia.
- Un segmento de tipo "información de página" (tipo 48) no debe hacer referencia a ningún otro segmento.
- Un segmento de tipo "fin de página" (tipo 49) no debe hacer referencia a ningún otro segmento.
- Un segmento de tipo "fin de franja" (tipo 50) no debe hacer referencia a ningún otro segmento.
- Un segmento de tipo "fin de fichero" (tipo 51) no debe hacer referencia a ningún otro segmento.
- Un segmento de tipo "perfiles" (tipo 52) no debe hacer referencia a ningún otro segmento.
- Un segmento de tipo "tablas" (tipo 53) no debe hacer referencia a ningún otro segmento.
- Un segmento de tipo "extensión" (tipo 62) puede hacer referencia a cualquier número de segmentos de cualquier tipo, a menos que el tipo del segmento de extensión imponga alguna restricción.

7.3.2 Reglas para asociaciones de páginas

Todos los segmentos de una región tienen que estar asociados con alguna página (es decir, han de tener un campo de asociación de un número de páginas distinto de cero). Los segmentos "información de página", "fin de página" y "fin de franja" tienen que estar asociados con alguna página. Los segmentos "fin de fichero" no tienen que estar asociados con ninguna página. Los segmentos de otros tipos pueden estar asociados con una página, o pueden no estarlo.

Si un segmento no está asociado con ninguna página, no debe hacer referencia a ningún segmento que sí esté asociado con alguna página.

Si un segmento está asociado con una página, puede hacer referencia a segmentos que no están asociados con ninguna página, y a segmentos que están asociados con la misma página. No debe hacer referencia a ningún segmento que esté asociado con una página diferente.

7.4 Sintaxis de segmentos

En esta subcláusula se describe en detalle la sintaxis de la parte datos de segmento de cada tipo de segmento, y cómo se ha de decodificar.

7.4.1 Campo de información de segmento de región

Todas las partes datos de segmento de región empiezan con un campo de información de segmento de región; su formato se especifica en lo que sigue. Un campo de información de segmento de región contiene los siguientes subcampos, mostrados en la figura 28 y descritos a continuación.

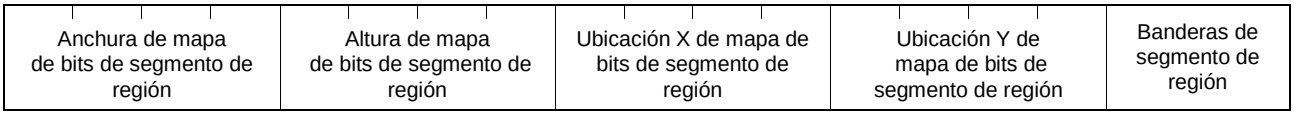


Figura 28 – Estructura de encabezamiento de datos de segmento de región

Anchura de mapa de bits de segmento de región – Véase 7.4.1.1.

Altura de mapa de bits de segmento de región – Véase 7.4.1.2.

Ubicación X de mapa de bits de segmento de región – Véase 7.4.1.3.

Ubicación Y de mapa de bits de segmento de región – Véase 7.4.1.4.

Banderas de segmento de región – Véase 7.4.1.5.

7.4.1.1 Anchura de mapa de bits de segmento de región

Este campo de cuatro bytes da la anchura en píxels del mapa de bits codificado en este segmento.

7.4.1.2 Altura de mapa de bits de segmento de región

Este campo de cuatro bytes da la altura en píxels del mapa de bits codificado en este segmento.

7.4.1.3 Ubicación X de mapa de bits de segmento de región

Este campo de cuatro bytes da el desplazamiento horizontal en píxels del mapa de bits codificado en este segmento con respecto al mapa de bits de la página.

7.4.1.4 Ubicación Y de mapa de bits de segmento de región

Este campo de cuatro bytes da el desplazamiento vertical en píxels del mapa de bits codificado en este segmento con respecto al mapa de bits de la página.

7.4.1.5 Banderas de segmento de región

Este campo de un byte se formatea como se muestra en la figura 29, y se describe a continuación.

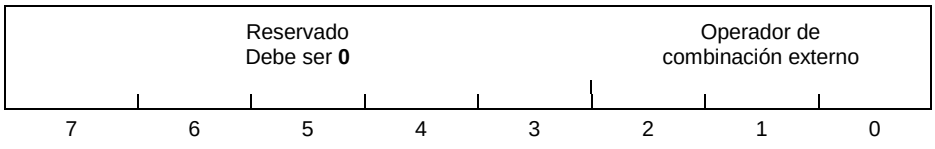


Figura 29 – Estructura de campo de banderas de segmento de región

Bits 0-2 Operador de combinación externo. Este campo de tres bits puede tomar los valores siguientes, cada uno de los cuales representa uno de los cinco posibles operadores de combinación:

0 OR
1 AND
2 XOR
3 XNOR
4 REPLACE

NOTA 1 – Estos operadores describen cómo se ha de combinar el mapa de bits del segmento con el mapa de bits de la página. REPLACE es utilizado por regiones de refinamiento cuando la región refinada sustituye a la región que está refinando. Operadores tales como AND se pueden utilizar para enmascarar, cuando una porción del mapa de bits de la página que ya contiene datos se ha de eliminar para que se pueda escribir ahí otro mapa de bits; considérese la escritura de un mapa de bits a través de una máscara.

NOTA 2 – Los segmentos de región intermedia nunca se combinan directamente con la página, y por ello no se utilizan ni su ubicación ni sus operadores de combinación externos. Sin embargo, estos valores pueden ser aún útiles: si un decodificador desea dibujar una versión de la página antes de que todos los segmentos hayan sido decodificados (para construcción progresiva), quizá le convenga reproducir segmentos intermedios; la fijación de la ubicación y la combinación externa de acuerdo con la manera según la cual se combinará el refinamiento final de ese segmento intermedio con la página puede ayudar al decodificador a producir una secuencia útil de refinamientos progresivos de la página.

Bits 3-7 Reservado; debe ser 0.

En otras palabras, este campo de información de segmento describe el tamaño y la ubicación del mapa de bits codificado en este segmento.

EJEMPLO – Si los valores de tamaño y ubicación son (en orden) 100, 200, 50 y 75, este segmento describe un mapa de bits de 100 píxels de ancho por 200 píxels de alto, cuya esquina superior izquierda está a 50 píxels a la derecha, y 75 píxels por debajo, de la esquina superior izquierda de la página.

7.4.2 Sintaxis de segmento de diccionario de símbolos

7.4.2.1 Encabezamiento de datos de segmento de diccionario de símbolos

La parte datos de un segmento de diccionario de símbolos comienza con un encabezamiento de datos de segmento de diccionario de símbolos, que contiene los campos mostrados en la figura 30 y descritos a continuación.

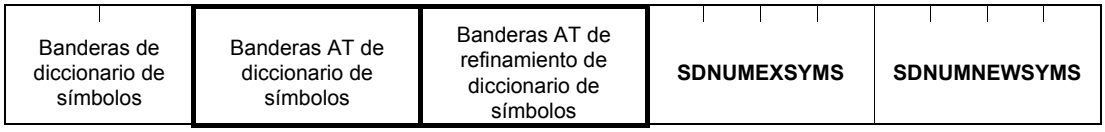


Figura 30 – Estructura de encabezamiento de datos de segmento de diccionario de símbolos

Banderas de diccionario de símbolos – Véase 7.4.2.1.1.

Banderas AT de diccionario de símbolos – Véase 7.4.2.1.2.

Banderas AT de refinamiento de diccionario de símbolos – Véase 7.4.2.1.3.

SDNUMEXSYMS – Véase 7.4.2.1.4.

SDNUMNEWSYMS – Véase 7.4.2.1.5.

7.4.2.1.1 Banderas de diccionario de símbolos

Este campo de dos bytes se formatea como se muestra en la figura 31, y se describe a continuación.

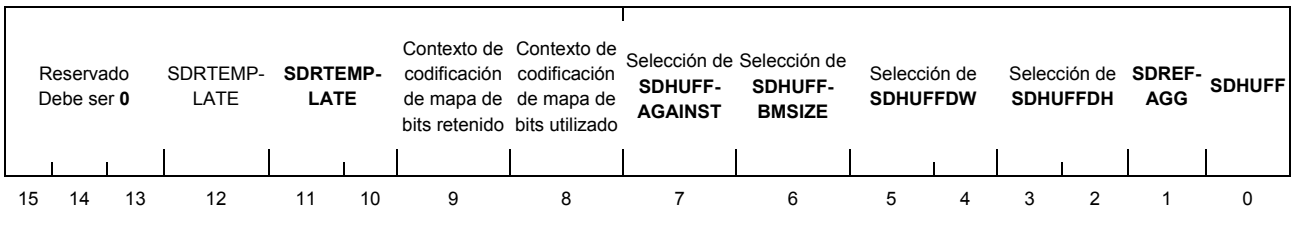


Figura 31 – Estructura de campo de banderas de diccionario de símbolos

Bit 0 **SDHUFF**

Si este bit es **1**, el segmento utiliza la variante de codificación Huffman. Si este bit es **0**, el segmento utiliza la variante de codificación aritmética. La fijación de esta bandera determina la manera según la cual se codifican los datos de este segmento, y además puede modificar el orden en el que se codifican algunos de los datos.

Bit 1 **SDREFAGG**

Si este bit es **0**, no se utiliza codificación de refinamiento o agregación en este segmento. Si este bit es **1**, todos los mapas de bits de símbolos se codifican con refinamiento/agregación.

Bits 2-3 Selección de **SDHUFFDH**. Este campo de dos bits puede tomar uno de tres valores, indicando así que tabla se ha de utilizar para **SDHUFFDH**.

0 La del cuadro B.4
1 La del cuadro B.5
3 Tabla suministrada por el usuario

El valor 2 no está permitido.

Si **SDHUFF** es **0**, este campo debe contener el valor 0.

Bits 4-5 Selección de **SDHUFFDW**. Este campo de dos bits puede tomar uno de tres valores, indicando que tabla se ha de utilizar para **SDHUFFDW**.

0 La del cuadro B.2
1 La del cuadro B.3
3 Tabla suministrada por el usuario

El valor 2 no está permitido.

Si **SDHUFF** es **0**, este campo debe contener el valor 0.

Bit 6 Selección de **SDHUFFBMSIZE**.

Si este campo es **0**, se utiliza la tabla del cuadro B.1 para **SDHUFFBMSIZE**. Si este campo es **1**, se utiliza una tabla suministrada por el usuario para **SDHUFFBMSIZE**.

Si **SDHUFF** es **0**, este campo debe contener el valor **0**.

Bit 7 Selección de **SDHUFFAGGINST**.

Si este campo es **0**, se utiliza la tabla del cuadro B.1 para **SDHUFFAGGINST**. Si este campo es **1**, se utiliza una tabla suministrada por el usuario para **SDHUFFAGGINST**.

Si **SDHUFF** o **SDREFAGG** es **0**, este campo debe contener el valor **0**.

Bit 8 Contexto de codificación de mapa de bits utilizado.

Si **SDHUFF** es **1** y **SDREFAGG** es **0**, este campo debe contener el valor **0**.

Bit 9 Contexto de codificación de mapa de bits retenido.

Si **SDHUFF** es **1** y **SDREFAGG** es **0**, este campo debe contener el valor **0**.

Bits 10-11 **SDTEMPLATE**

Este campo controla la plantilla utilizada para decodificar mapas de bits de símbolos si **SDHUFF** es **0**. Si **SDHUFF** es **1**, este campo debe contener el valor **0**.

Bit 12 **SDRTEMPLATE**

Este campo controla la plantilla utilizada para decodificar mapas de bits de símbolos si **SDREFAGG** es **1**. Si **SDREFAGG** es **0**, este campo debe contener el valor **0**.

Bits 13-15 Reservado; debe ser **0**.

7.4.2.1.2 Banderas AT de diccionario de símbolos

Este campo sólo está presente si **SDHUFF** es **0**. Si **SDTEMPLATE** es **0**, es un campo de ocho bytes, formateado como se muestra en la figura 32, y se describe a continuación.

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

Figura 32 – Estructura de campo de banderas AT de diccionario de símbolos cuando SDTEMPLATE es 0

Byte 0 **$SDATX_1$**

Byte 1 **$SDATY_1$**

Byte 2 **$SDATX_2$**

Byte 3 **$SDATY_2$**

Byte 4 **$SDATX_3$**

Byte 5 **$SDATY_3$**

Byte 6 **$SDATX_4$**

Byte 7 **$SDATY_4$**

Si **SDTEMPLATE** es 1, 2 ó 3, es un campo de dos bytes formateado como se muestra en la figura 33, y descrito a continuación.

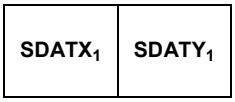


Figura 33 – Estructura de campo de banderas AT de diccionario de símbolos cuando SDTEMPLATE no es 0

Byte 0 **$SDATX_1$**

Byte 1 **$SDATY_1$**

Si **SDTEMPLATE** es 1, 2 ó 3, los valores de **$SDATX_2$** a **$SDATX_4$** y **$SDATY_2$** a **$SDATY_4$** son todos cero.

Los campos de las coordenadas X e Y de AT son valores con signo, y pueden tomar los valores permitidos de acuerdo con la figura 7.

7.4.2.1.3 Banderas AT de refinamiento de diccionario de símbolos

Este campo sólo está presente si **SDREFAGG** es **1** y **SDRTEMPLATE** es **0**. Es un campo de cuatro bytes, formateado como se muestra en la figura 34, y descrito a continuación.

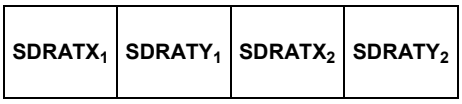


Figura 34 – Estructura de campo de banderas AT de refinamiento de diccionario de símbolos

Byte 0 **$SDRATX_1$**

Byte 1 **$SDRATY_1$**

Byte 2 **$SDRATX_2$**

Byte 3 **$SDRATY_2$**

Los campos de las coordenadas X e Y de AT son valores con signo, y pueden tomar los valores permitidos de acuerdo con 6.3.5.3.

7.4.2.1.4 Número de símbolos exportados (SDNUMEXSYMS)

Este campo de cuatro bytes contiene el número de símbolos exportados desde este diccionario.

Ayuda mucho al decodificador a encontrar fácilmente el número de símbolos presente; podría, por ejemplo, atribuir una formación de estructuras antes de empezar a decodificar el diccionario.

7.4.2.1.5 Número de símbolos nuevos (SDNUMNEWSYMS)

Este campo de cuatro bytes contiene el número de símbolos definidos en este diccionario.

NOTA – **SDNUMEXSYMS** y **SDNUMNEWSYMS** son a menudo, pero no siempre, el mismo valor. Por ejemplo, si un diccionario reexporta parte de los símbolos que ha importado de los diccionarios a los que hace referencia, el diccionario copia efectivamente esos símbolos. Esos símbolos quedan reflejados en **SDNUMEXSYMS** pero no en **SDNUMNEWSYMS**. Otro posible origen de diferencias está en que un diccionario quizás defina algunos símbolos que no exporta.

7.4.2.1.6 Selección de tabla Huffman de segmento de diccionarios de símbolos

Fijar los valores de los parámetros **SDHUFFDH**, **SDHUFFDW**, **SDHUFFBMSIZE** y **SDHUFFAGGINST** de acuerdo con los campos de selección que se muestran en 7.4.2.1.1, y los segmentos de tablas a los que hace referencia este segmento. De manera más precisa, de estas cuatro tablas Huffman, algunas pueden ser especificadas para utilizar alguna tabla normalizada, y algunas pueden ser especificadas para utilizar una tabla suministrada por un usuario. El número de tablas especificadas para utilizar una tabla suministrada por el usuario debe ser igual al número de segmentos de tablas a los que hace referencia este segmento. Estos segmentos de tablas se armonizan con las tablas Huffman utilizando tablas suministradas por el usuario de acuerdo con el orden en que se hace referencia a los segmentos de tablas, y el orden:

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

Si se utiliza una tabla especificada por el usuario para **SDHUFFDW**, esta tabla debe tener la posibilidad de codificar el valor fuera de banda (OOB). Si se utiliza una tabla especificada por el usuario para **SDHUFFDH**, **SDHUFFBMSIZE**, o **SDHUFFAGGINST**, esta tabla no debe tener la posibilidad de codificar el valor fuera de banda (OOB).

EJEMPLO – Si se especifica que **SDHUFFDH** y **SDHUFFAGGINST** utilicen tablas suministradas por el usuario, y se especifica que **SDHUFFDW** y **SDHUFFBMSIZE** utilicen tablas normalizadas (cuadros B.2 y B.1, respectivamente), este segmento debe hacer referencia a exactamente dos segmentos de tablas; el segmento de tablas al que se hace referencia primero se utiliza para **SDHUFFDH** y el segmento de tablas al que se hace referencia en segundo lugar se utiliza para **SDHUFFAGGINST**.

7.4.2.2 Decodificación de un segmento de diccionario de símbolos

Un segmento de diccionario de símbolos se decodifica de acuerdo con los pasos siguientes.

1) Interpretar su encabezamiento, como se describe en 7.4.2.1.
2) Decodificar (o recuperar los resultados de la decodificación) de cualquier diccionario de símbolos referenciados y segmentos de tablas.
3) Si el bit "contexto de codificación de mapa de bits utilizado" del encabezamiento es **1**, fijar como se describe en E.3.8, las estadísticas de codificación aritmética para los procedimientos de decodificación de región genérica y de región de refinamiento genérica en los valores que contenían al final de la decodificación del último segmento de diccionario de símbolos referenciado. El encabezamiento de datos de segmento de diccionario de símbolos de ese segmento de diccionario de símbolos debía tener el bit "contexto de codificación de mapa de bits retenido" igual a **1**. Los valores de **SDHUFF**, **SDREFAGG**, **SDTEMPLATE**, **SDRTEMPLATE**, y todas las ubicaciones AT (tanto directas como de refinamiento) para este diccionario de símbolos deben concordar con los valores correspondientes del diccionario de símbolos cuyos valores de contexto se utilizan.
4) Si el bit "contexto de codificación de mapa de bits utilizado" del encabezamiento es **0**, reponer a cero, como se describe en E.3.7, todas las estadísticas de codificación aritmética para los procedimientos de decodificación de región genérica y de región de refinamiento genérica.
5) Reponer a cero todas las estadísticas de codificación aritmética para todos los contextos de todos los codificadores aritméticos de enteros.
6) Invocar el procedimiento de decodificación de diccionario de símbolos descrito en 6.5, con los parámetros del procedimiento de decodificación de diccionario de símbolos fijados como se muestra en el cuadro 28.
7) Si el bit "contexto de codificación de mapa de bits retenido" del encabezamiento fuese **1**, preservar, como se describe en E.3.8, los contenidos actuales de las estadísticas de codificación aritmética para los procedimientos de decodificación de región genérica y de región de refinamiento genérica.

NOTA – El paso 3) tiene por objeto reducir los costes de la codificación del diccionario de símbolos. Un efecto lateral de la decodificación de un diccionario de símbolos consiste en que las estadísticas de codificación aritmética utilizadas para codificar mapas de bits "aprenden" las estadísticas aproximadas de los símbolos de ese diccionario de símbolos. Estos dos pasos [3) y 7)] permiten un cierto grado de reutilización limitada de esas estadísticas: las estadísticas aprendidas cuando se decodifica el diccionario de símbolos que es el último diccionario de símbolos al que se hace referencia se utilizan como punto de partida para decodificar este diccionario de símbolos.

El paso 7) está presente de manera explícita porque no todas las estadísticas de codificación aritmética de un diccionario de símbolos serán utilizadas por otro diccionario. Sabiendo que no serán utilizadas, el decodificador puede descartarlas, con la consiguiente reducción de la utilización de memoria.

Cuadro 28 – Parámetros utilizados para decodificar un segmento de diccionario de símbolos

Nombre	Valor
SDHUFF	Como se muestra en 7.4.2.1.1
SDREFAGG	Como se muestra en 7.4.2.1.1
SDNUMINSYMS	El número total de símbolos exportados desde todos los segmentos de diccionario de símbolos a los que hace referencia este segmento
SDINSYMS	Concatenar las formaciones de símbolos exportados procedentes de todos los segmentos de diccionario de símbolos a los que hace referencia este segmento, en el orden en el que se hace referencia a los mismos
SDNUMNEWSYMS	Como se muestra en 7.4.2.1.5
SDNUMEXSYMS	Como se muestra en 7.4.2.1.4
SDHUFFDH	Véase 7.4.2.1.6
SDHUFFDW	Véase 7.4.2.1.6
SDHUFFBMSIZE	Véase 7.4.2.1.6
SDHUFFAGGINST	Véase 7.4.2.1.6
SDTEMPLATE	Véase 7.4.2.1.1
$\mathbf{SDATX_1}$	Véase 7.4.2.1.2
$\mathbf{SDATY_1}$	Véase 7.4.2.1.2
$\mathbf{SDATX_2}$	Véase 7.4.2.1.2
$\mathbf{SDATY_2}$	Véase 7.4.2.1.2
$\mathbf{SDATX_3}$	Véase 7.4.2.1.2
$\mathbf{SDATY_3}$	Véase 7.4.2.1.2
$\mathbf{SDATX_4}$	Véase 7.4.2.1.2
$\mathbf{SDATY_4}$	Véase 7.4.2.1.2
SDRTEMPLATE	Véase 7.4.2.1.1
$\mathbf{SDRATX_1}$	Véase 7.4.2.1.3
$\mathbf{SDRATY_1}$	Véase 7.4.2.1.3
$\mathbf{SDRATX_2}$	Véase 7.4.2.1.3
$\mathbf{SDRATY_2}$	Véase 7.4.2.1.3

7.4.3 Sintaxis de segmento de región de texto

Las partes datos de los tres tipos de segmento de región de texto ("región de texto intermedia", "región de texto inmediata" y "región de texto sin pérdida inmediata") se codifican de manera idéntica, pero se actúa en ellas de manera diferente; véase 8.2. Aquí se especifica la sintaxis de la parte de datos de estos tipos de segmento.

7.4.3.1 Encabezamiento de datos de segmento de región de texto

La parte datos de un segmento de región de texto empieza con un encabezamiento de datos de segmento de región de texto. Este encabezamiento contiene los campos mostrados en la figura 35 y descritos a continuación.

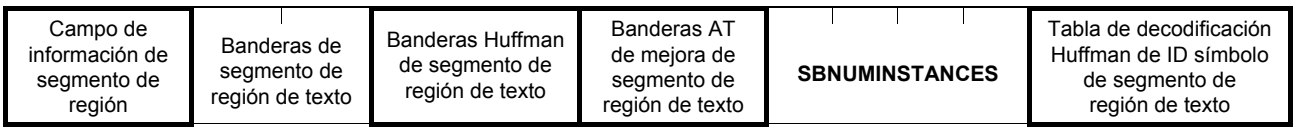


Figura 35 – Estructura de encabezamiento de datos de segmento de región de texto

Campo de información de segmento de región – Véase 7.4.1.

Banderas de segmento de región de texto – Véase 7.4.3.1.1.

Banderas Huffman de segmento de región de texto – Véase 7.4.3.1.2.

Banderas AT de refinamiento de segmento de región de texto – Véase 7.4.3.1.3.

SBNUMINSTANCES – Véase 7.4.3.1.4.

Tabla de decodificación Huffman de ID símbolo de segmento de región de texto – Véase 7.4.3.1.5.

7.4.3.1.1 Banderas de segmento de región de texto

Este campo de dos bytes está formateado como se muestra en la figura 36 y se describen continuación.

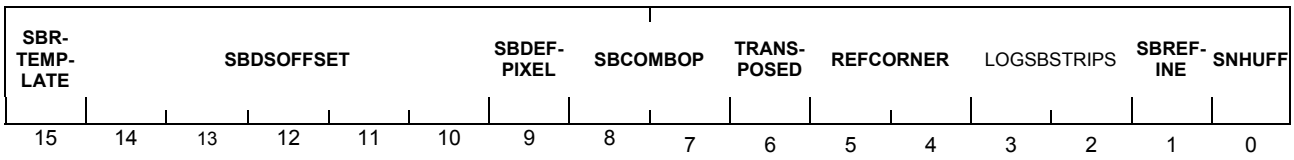


Figura 36 – Estructura de campo de banderas de región de texto

Bit 0 **SBHUFF**

Si este bit es **1**, el segmento utiliza la variante de codificación Huffman. Si este bit es **0**, el segmento utiliza la variante de decodificación aritmética. La fijación de esta bandera determina cómo se codifican los datos en este segmento.

Bit 1 **SBREFINE**

Si este bit es **0**, el segmento no contiene ningún refinamiento de ejemplar de símbolo. Si este bit es **1**, el segmento puede contener refinamientos de instancia de símbolo.

Bits 2-3 LOGSBSTRIPS

Este campo de dos bits codifica el logaritmo en base 2 del tamaño de franja que se utiliza para codificar el segmento. Así pues, se pueden codificar tamaños de franja de 1, 2, 4 y 8.

Bits 4-5 **REFCORNER**. Los cuatro valores que este campo de dos bits puede tomar son:

0 BOTTOMLEFT
1 TOPLEFT
2 BOTTOMRIGHT
3 TOPRIGHT

NOTA – La mejor compresión se consigue normalmente cuando el punto de referencia de cada símbolo se halla en la línea de base del texto. Puesto que el texto puede orientarse en una cualquiera de ocho direcciones, es preciso que exista cierta flexibilidad respecto a qué esquina de un determinado símbolo se utiliza como punto de referencia.

Bit 6 **TRANSPOSED**

Si este bit es **1**, la dirección primaria de la codificación es de arriba abajo. Si este bit es **0**, la dirección primaria de la codificación es de izquierda a derecha. Esto hace posible que el texto se desarrolle hacia arriba y hacia abajo de la página.

Bits 7-8 **SBCOMBOP**. Este campo tiene cuatro valores posibles, representando cada uno de ellos uno de los cuatro posibles operadores de combinación:

0 OR
1 AND
2 XOR
3 XNOR

Bit 9 **SBDEFPIXEL**

Este bit contiene el valor inicial de cada uno de los píxels de la región de texto, antes de que se dibuje cualquier símbolo.

Bits 10-14 SBDSOFFSET

Este campo de cinco bits con signo contiene el valor de **SBDSOFFSET** – Véase 6.4.8.

Bit 15 SBRTEMPLATE

Este campo controla la plantilla utilizada para decodificar refinamiento de ejemplar símbolo si **SBREFINE** es **1**. Si **SBREFINE** es **0**, este campo debe contener el valor **0**.

7.4.3.1.2 Banderas Huffman de segmento de región de texto

Este campo sólo está presente si **SBHUFF** es **1**.

Este campo de dos bytes está formateado como se muestra en la figura 37 y se describe a continuación.

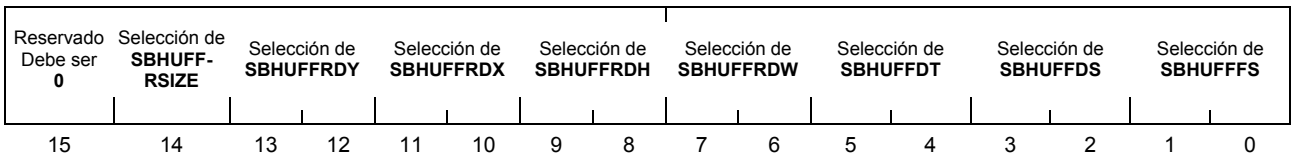


Figura 37 – Estructura de campo de banderas Huffman de región de texto

Bits 0-1 Selección de **SBHUFFFS**. Este campo de dos bits puede tomar uno de tres valores, indicando así que tabla se ha de utilizar para **SBHUFFFS**.

0 La del cuadro B.6
1 La del cuadro B.7
3 Tabla suministrada por el usuario

El valor 2 no está permitido.

Bits 2-3 Selección de **SBHUFFDS**. Este campo de dos bits puede tomar uno de cuatro valores, indicando así que tabla se ha de utilizar para **SBHUFFDS**.

0 La del cuadro B.8
1 La del cuadro B.9
2 La del cuadro B.10
3 Tabla suministrada por el usuario

Bits 4-5 Selección de **SBHUFFDT.** Este campo de dos bits puede tomar uno de cuatro valores, indicando así que tabla se ha de utilizar para **SBHUFFDT.**

0 La del cuadro B.11
1 La del cuadro B.12
2 La del cuadro B.13
3 Tabla suministrada por el usuario

Bits 6-7 Selección de **SBHUFFRDW**. Este campo de dos bits puede tomar uno de tres valores, indicando así que tabla ha de ser utilizada para **SBHUFFRDW**.

0 La del cuadro B.14
1 La del cuadro B.15
3 Tabla suministrada por el usuario

El valor 2 no está permitido.Si **SBREFINE** es **0**, este campo debe contener el valor **0**.

Bits 8-9 Selección de **SBHUFFRDH**. Este campo de dos bits puede tomar uno de tres valores, indicando así que tabla ha de ser utilizada para **SBHUFFRDH**.

0 La del cuadro B.14
1 La del cuadro B.15
3 Tabla suministrada por el usuario

El valor 2 no está permitido. Si **SBREFINE** es **0**, este campo debe contener el valor **0**.

Bits 10-11 Selección de **SBHUFFRDX**. Este campo de dos bits puede tomar uno de tres valores, indicando así que tabla ha de ser utilizada para **SBHUFFRDX**.

0 La del cuadro B.14

1 La del cuadro B.15

3 Tabla suministrada por el usuario

El valor 2 no está permitido. Si **SBREFINE** es **0**, este campo debe contener el valor **0**.

Bits 12-13 Selección de **SBHUFFRDY**. Este campo de dos bits puede tomar uno de tres valores, indicando así que tabla ha de ser utilizada para **SBHUFFRDY**.

0 La del cuadro B.14

1 La del cuadro B.15

3 Tabla suministrada por el usuario

El valor 2 no está permitido. Si **SBREFINE** es **0**, este campo debe contener el valor **0**.

Bit 14 Selección de **SBHUFFRSIZE**. Si este campo es **0**, se utiliza la tabla del cuadro B.1 para **SBHUFFRSIZE**. Si este campo es **1**, se utiliza una tabla suministrada por el usuario para **SBHUFFRSIZE.** Si **SBREFINE** es **0**, este campo debe contener el valor **0**.

Bit 15 **Reservado;** debe ser **0.**

7.4.3.1.3 Banderas AT de refinamiento de región de texto

Este campo sólo está presente si **SBREFINE** es **1** y **SBRTEMPLATE** es **0**. Es un campo de cuatro bytes, formateado como se muestra en la figura 38, y se describe a continuación.

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

Figura 38 – Estructura de campos de banderas AT de refinamiento de región de texto

Byte 0 $SBRATX_1$

Byte 1 $SBRATY_1$

Byte 2 $SBRATX_2$

Byte 3 $SBRATY_2$

Los campos de las coordenadas X e Y de AT son valores con signo, y pueden tomar los valores que están permitidos de acuerdo con 6.3.5.3.

7.4.3.1.4 Número de ejemplares de símbolo (SBNUMINSTANCES)

Este campo de cuatro bytes contiene el número de ejemplares de símbolo codificadas en este segmento.

7.4.3.1.5 Tabla de decodificación Huffman de ID símbolo de segmento de región de texto

Este campo contiene una versión codificada de los códigos Huffman utilizados para decodificar los ID de ejemplar de símbolo en el procedimiento de decodificación de región de texto. Se decodifica como se especifica en 7.4.3.1.7. Sólo está presente si **SBHUFF** es **1**.

7.4.3.1.6 Selección de tabla Huffman de segmento de región de texto

Fijar los valores de los parámetros **SBHUFFFS**, **SBHUFFDS**, **SBHUFFDT**, **SBHUFFRDW**, **SBHUFFRDH**, **SBHUFFRDX**, **SBHUFFRDY** y **SBHUFFRSIZE** de acuerdo con los campos de selección que se muestran en 7.4.3.1.2, y los segmentos de tablas a los que hace referencia este segmento. De manera más precisas, de estas ocho tablas Huffman, algunas pueden ser especificadas para utilizar alguna tabla normalizada, y algunas pueden ser especificadas para utilizar una tabla suministrada por un usuario. El número de tablas especificadas para utilizar una tabla suministrada por el usuario debe ser igual al número de segmentos de tablas a los que hace referencia este segmento. Estos segmentos de tablas se armonizan con las tablas Huffman utilizando tablas suministradas por el usuario de acuerdo con el orden en que se hace referencia a los segmentos de tablas, y el orden:

1) **SBHUFFFS**

2) **SBHUFFDS**

3) **SBHUFFDT**

4) **SBHUFFRDW**

5) **SBHUFFRDH**

6) **SBHUFFRDX**

7) **SBHUFFRDY**

8) **SBHUFFRSIZE**

Si se utiliza una tabla especificada por el usuario para **SBHUFFDS**, esta tabla debe tener la posibilidad de codificar el valor fuera de banda (OOB). Si se utiliza una tabla especificada por el usuario para **SBHUFFFS**, **SBHUFFDT**, **SBHUFFRDW**, **SBHUFFRDH**, **SBHUFFRDX**, **SBHUFFRDY** o **SBHUFFRSIZE**, esta tabla no debe tener la posibilidad de codificar el valor fuera de banda (OOB).

7.4.3.1.7 Decodificación de tabla Huffman de ID símbolo

Esta tabla se codifica como longitudes de código de ID símbolo de **SBNUMSYMS**; los códigos reales en **SBSYMCODES** se asignan a partir de estas longitudes de código de ID símbolo utilizando el algoritmo de B.3.

Las propias longitudes de código de ID símbolo se codifican por el procedimiento de la longitud de pasada, y a las pasadas se les aplica una codificación Huffman. Esto es muy similar al formato codificado "zlib" que se documenta en RFC 1951, aunque no idéntico. La codificación se basa en los códigos que se muestran en el cuadro 29.

La decodificación de una tabla Huffman de ID símbolo procede como sigue:

1) Leer las longitudes de código de RUNCODE0 a RUNCODE34; cada una de ellas se almacena en un valor de cuatro bits.

2) Dadas las longitudes, asignar códigos Huffman de RUNCODE0 a RUNCODE34 utilizando el algoritmo de B.3.

3) Leer un código Huffman utilizando esta asignación. Se decodifica en uno de los códigos de pasada RUNCODE0 a RUNCODE34. Si es RUNCODE32, leer dos bits adicionales. Si es RUNCODE33, leer tres bits adicionales. Si es RUNCODE34, leer siete bits adicionales.

4) Interpretar el código RUNCODE y los bits adicionales (si hay alguno) de acuerdo con el cuadro 29. Así se obtienen las longitudes de código de ID símbolo de uno o más símbolos.

5) Repetir los pasos 3 y 4 hasta que se hayan determinado las longitudes de código de ID símbolo de todos los símbolos **SBNUMSYMS**.

6) Saltar los bits restantes en el último byte leído, de forma que el procedimiento de decodificación de región de texto empiece de manera efectiva en una frontera de byte.

7) Asignar un código Huffman a cada símbolo aplicando el algoritmo de B.3 a las longitudes de código de ID símbolo recién decodificadas. El resultado es la tabla Huffman de ID símbolo **SBSYMCODES**.

Cuadro 29 – Significado de los códigos de pasada

RUNCODE0	La longitud de código de ID símbolo es 0
RUNCODE1	La longitud de código de ID símbolo es 1
RUNCODE2	La longitud de código de ID símbolo es 2
RUNCODE3	La longitud de código de ID símbolo es 3
RUNCODE4	La longitud de código de ID símbolo es 4
RUNCODE5	La longitud de código de ID símbolo es 5
RUNCODE6	La longitud de código de ID símbolo es 6
RUNCODE7	La longitud de código de ID símbolo es 7
RUNCODE8	La longitud de código de ID símbolo es 8
RUNCODE9	La longitud de código de ID símbolo es 9
RUNCODE10	La longitud de código de ID símbolo es 10
RUNCODE11	La longitud de código de ID símbolo es 11
RUNCODE12	La longitud de código de ID símbolo es 12
RUNCODE13	La longitud de código de ID símbolo es 13
RUNCODE14	La longitud de código de ID símbolo es 14
RUNCODE15	La longitud de código de ID símbolo es 15
RUNCODE16	La longitud de código de ID símbolo es 16
RUNCODE17	La longitud de código de ID símbolo es 17
RUNCODE18	La longitud de código de ID símbolo es 18
RUNCODE19	La longitud de código de ID símbolo es 19
RUNCODE20	La longitud de código de ID símbolo es 20
RUNCODE21	La longitud de código de ID símbolo es 21
RUNCODE22	La longitud de código de ID símbolo es 22
RUNCODE23	La longitud de código de ID símbolo es 23
RUNCODE24	La longitud de código de ID símbolo es 24
RUNCODE25	La longitud de código de ID símbolo es 25
RUNCODE26	La longitud de código de ID símbolo es 26
RUNCODE27	La longitud de código de ID símbolo es 27
RUNCODE28	La longitud de código de ID símbolo es 28
RUNCODE29	La longitud de código de ID símbolo es 29
RUNCODE30	La longitud de código de ID símbolo es 30
RUNCODE31	La longitud de código de ID símbolo es 31
RUNCODE32	Copiar la longitud de código de ID símbolo previa 3-6 veces. Los dos bits siguientes, más 3, indican esta longitud repetida
RUNCODE33	Repetir la longitud de código de un ID símbolo de 0, 3-10 veces. Los tres bits siguientes, más 3, indican esta longitud repetida
RUNCODE34	Repetir la longitud de código de un ID símbolo de 0, 11-138 veces. Los siete bits siguientes, más 11, indican esta longitud repetida

EJEMPLO 1 – Supóngase que **SBNUMSYMS** es 32 y que las longitudes de código de ID símbolo de estos 32 símbolos son, en orden:

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

Estas longitudes de código de ID símbolo podrían transmitirse como la secuencia de bytes, en hexadecimal:

```
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
```

La interpretación de esta secuencia de bytes se puede llevar a cabo en los tres pasos siguientes:

1) Los primeros 17 bytes más los cuatro primeros bits del decimoctavo byte asignan longitudes de código a los 35 códigos de pasada, como sigue:

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		

Recuérdese que a los códigos que no se utilizan se les asigna una longitud de código de ID símbolo de cero.

2) El algoritmo de B.3 asigna los siguientes códigos Huffman a los códigos de pasada (los códigos de pasada a los que no se les asignan códigos Huffman son omitidos):

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

3) La parte restante de la secuencia de bytes es:

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

en la que la mitad del primer byte ya ha sido consumida. La decodificación de esta secuencia utilizando estos códigos Huffman da los resultados siguientes:

11111 000 RUNCODE 33 (0) – es decir, RUNCODE 33 seguido de tres bits que contienen el valor 0, indicando una pasada de tres longitudes de cero
101 RUNCODE9
100 RUNCODE6
110 00 RUNCODE32(0) – es decir, RUNCODE32 seguido de dos bits que contienen el valor 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 Cuatro bits de relleno del último byte

4) Después de interpretar los códigos de pasada de acuerdo con el cuadro 29, se decodifica la secuencia deseada de longitudes de código de ID símbolo.

EJEMPLO 2 – Este ejemplo describe cómo podría generar un codificador una tabla Huffman de ID símbolo codificados. La tabla de ID símbolo es idéntica a la del ejemplo anterior.

Supóngase que una región de texto hace referencia a un diccionario que contiene 32 símbolos y que cada símbolo se utiliza como sigue:

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

Por ejemplo, el primero, el segundo y el tercer símbolos de un diccionario de símbolos no se utilizan nunca, el cuarto símbolo se utiliza una vez, el quinto símbolo se utiliza ocho veces, y así sucesivamente.

El cuadro 30 muestra, por tanto, de izquierda a derecha, la progresión de la codificación.

Cuadro 30 – Ejemplo de codificación de tabla Huffman de ID símbolo

Símbolo	Cuenta de usos	Longitud de código de ID símbolo	Pasadas	RUNCODEs
Símbolo #1	0	0	Una pasada de 3 longitudes de 0	RUNCODE33(0)
Símbolo #2	0	0		
Símbolo #3	0	0		
Símbolo #4	1	9	Una pasada de 1 longitud de 9	RUNCODE9
Símbolo #5	8	6	Una pasada de 4 longitudes de 6	RUNCODE6
Símbolo #6	8	6		RUNCODE32(0)
Símbolo #7	8	6		
Símbolo #8	8	6		
Símbolo #9	64	3	Una pasada de 1 longitud de 3	RUNCODE3
Símbolo #10	32	4	Una pasada de 6 longitudes de 4	RUNCODE4
Símbolo #11	32	4		RUNCODE32(2)
Símbolo #12	32	4		
Símbolo #13	32	4		
Símbolo #14	32	4		
Símbolo #15	32	4		
Símbolo #16	0	0	Una pasada de 1 longitud de 0	RUNCODE0
Símbolo #17	4	7	Una pasada de 1 longitud de 7	RUNCODE7
Símbolo #18	1	9	Una pasada de 1 longitud de 9	RUNCODE9
Símbolo #19	2	8	Una pasada de 1 longitud de 8	RUNCODE8
Símbolo #20	4	7	Una pasada de 1 longitud de 7	RUNCODE7
Símbolo #21	16	5	Una pasada de 6 longitudes de 5	RUNCODE5
Símbolo #22	16	5		RUNCODE32(2)
Símbolo #23	16	5		
Símbolo #24	16	5		
Símbolo #25	16	5		
Símbolo #26	16	5		
Símbolo #27	64	3	Una pasada de 1 longitud de 3	RUNCODE3
Símbolo #28	8	6	Una pasada de 1 longitud de 6	RUNCODE6
Símbolo #29	4	7	Una pasada de 1 longitud de 7	RUNCODE7
Símbolo #30	32	4	Una pasada de 1 longitud de 4	RUNCODE4
Símbolo #31	4	7	Una pasada de 2 longitudes de 7	RUNCODE7
Símbolo #32	4	7		RUNCODE7

Utilizando un algoritmo arborescente Huffman normalizado, las longitudes de código mostradas en la columna "Longitud de código de ID símbolo" se asignan a los símbolos (en donde una longitud de código de ID símbolo de 0 representa "no utilizado"). A continuación, esas longitudes de código se agrupan en pasadas, como se muestra en la columna "Pasadas". Seguidamente, cada pasada se expresa como uno o más RUNCODE, cada uno de ellas potencialmente con varios bits adicionales. Por ejemplo, RUNCODE32(2) representa RUNCODE32 seguido de dos bits que codifican el valor "2", lo que significa "Copiar la longitud del código de ID símbolo anterior 5 veces".

Una vez hecho esto, se cuenta el número de veces que se ha utilizado cada RUNCODE. Esas cuentas son como sigue (los RUNCODE no utilizados no se muestran):

RUNCODE0	1	RUNCODE3	2	RUNCODE4	2
RUNCODE5	1	RUNCODE6	2	RUNCODE7	5
RUNCODE8	1	RUNCODE9	2	RUNCODE32	3
RUNCODE33	1				

Las cuentas se convierten a continuación en longitudes de código utilizando un algoritmo arborescente Huffman normalizado:

RUNCODE0	5	RUNCODE3	3	RUNCODE4	3
RUNCODE5	5	RUNCODE6	3	RUNCODE7	2
RUNCODE8	5	RUNCODE9	3	RUNCODE32	3
RUNCODE33	5				

El algoritmo de B.3 asigna los siguientes códigos Huffman a los códigos de pasada:

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

y estos códigos Huffman se utilizan seguidamente para codificar la columna "RUNCODEs" del cuadro 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

El codificador emite ahora las longitudes de código RUNCODE codificadas, seguidas de la secuencia de RUNCODEs, más cuatro bits para rellenar el último byte, con lo que se obtiene la secuencia de bytes:

```
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 Decodificación de un segmento de región de texto

Un segmento de región de texto se decodifica de acuerdo con los pasos siguientes:

1) Interpretar su encabezamiento, como se describe en 7.4.3.1.
2) Decodificar (o recuperar los resultados de la decodificación de) cualquier diccionario de símbolos referenciados y segmentos de tablas.
3) Como se describe en E.3.7, reponer a cero todas las estadísticas de codificación aritmética.
4) Invocar el procedimiento de decodificación de región de texto descrito en 6.4, con los parámetros del procedimiento de decodificación de región de texto fijados como se muestra en el cuadro 31.

Cuadro 31 – Parámetros utilizados para decodificar un segmento de región de texto

Nombre	Valor
SBHUFF	Como se muestra en 7.4.3.1.1
SBREFINE	Como se muestra en 7.4.3.1.1
SBDEFPIXEL	Como se muestra en 7.4.3.1.1
SBCOMBOP	Como se muestra en 7.4.3.1.1
TRANSPOSED	Como se muestra en 7.4.3.1.1
REFCORNER	Como se muestra en 7.4.3.1.1
SBDSOFFSET	Como se muestra en 7.4.3.1.1
SBW	Según lo especificado por la anchura de mapa de bits de segmento de región en el encabezamiento de datos de segmento de región de este segmento
SBH	Según lo especificado por la altura de mapa de bits de segmento de región en el encabezamiento de datos de segmento de región de este segmento
SBNUMINSTANCES	Como se muestra en 7.4.3.1.4
SBSTRIPS	$2^{\text{LOGSBSTRIPS}}$
SBNUMSYMS	El número total de símbolos exportados en todos los segmentos de diccionario de símbolos referenciados por este segmento
SBSYMCODES	Como se especifica en 7.4.3.1.7
SBSYMCODELEN	$\lceil \log_2 \textbf{SBNUMSYMS} \rceil$
SBSYMS	Concatenación de las formaciones de símbolos exportados desde todos los segmentos del diccionario de símbolos referenciados por este segmento, en el orden en que están referenciados
SBHUFFFS	Véase 7.4.3.1.6
SBHUFFDS	Véase 7.4.3.1.6
SBHUFFDT	Véase 7.4.3.1.6
SBHUFFRDW	Véase 7.4.3.1.6
SBHUFFRDH	Véase 7.4.3.1.6
SBHUFFRDX	Véase 7.4.3.1.6
SBHUFFRDY	Véase 7.4.3.1.6
SBHUFFRSIZE	Véase 7.4.3.1.6
SBRTEMPLATE	Como se muestra en 7.4.3.1.1
$\textbf{SBRATX}_1$	Véase 7.4.3.1.3
$\textbf{SBRATY}_1$	Véase 7.4.3.1.3
$\textbf{SBRATX}_2$	Véase 7.4.3.1.3
$\textbf{SBRATY}_2$	Véase 7.4.3.1.3

7.4.4 Sintaxis de segmento de diccionario de patrones

7.4.4.1 Encabezamiento de datos de segmento de diccionario de patrones

La parte datos de un segmento de diccionario de patrones se empieza con un encabezamiento de datos de segmento de diccionario de patrones, formateado como se muestra en la figura 39, y se describe a continuación.

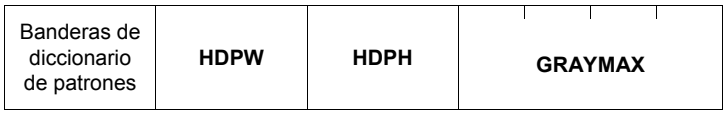


Figura 39 – Estructura de encabezamiento de diccionario de patrones

Banderas de diccionario de patrones – Véase 7.4.4.1.1

HDPW – Véase 7.4.4.1.2

HDPH – Véase 7.4.4.1.3

GRAYMAX – Véase 7.4.4.1.4.

7.4.4.1.1 Banderas de diccionario de patrones

Este campo de un byte se formatea como se muestra en la figura 40 y se describe a continuación.

Bit 0 **HDMMR**

Si este bit es **1**, el segmento utiliza la variante de codificación MMR. Si este bit es **0**, el segmento utiliza la variante de codificación aritmética.

Bit 1-2 **HDTEMPLATE**

Este campo controla la plantilla utilizada para decodificar patrones si **HDMMR** es **0**. Si **HDMMR** es **1**, este campo debe contener el valor 0.

Bits 3-7 Reservado; debe ser **0**.

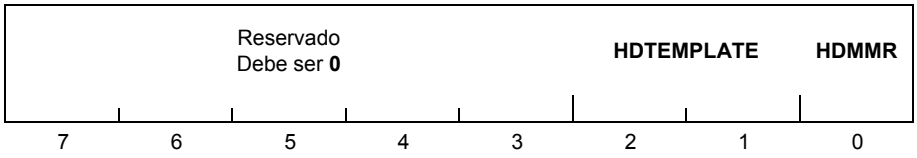


Figura 40 – Estructura de campo de banderas de diccionario de patrones

7.4.4.1.2 Anchura de los patrones en el diccionario de patrones (HDPW)

Este campo de un byte contiene la anchura de los patrones definidos en este diccionario de patrones. Su valor debe ser superior a cero.

7.4.4.1.3 Altura de los patrones en el diccionario de patrones (HDPH)

Este campo de un byte contiene la altura de los patrones definidos en este diccionario de patrones. Su valor debe ser superior a cero.

7.4.4.1.4 Valor mayor de la escala de grises (GRAYMAX)

Este campo de cuatro bytes contiene el número de patrones definidos en este diccionario de patrones menos uno.

7.4.4.2 Decodificación de un segmento de diccionario de patrones

Un segmento de diccionario de patrones se decodifica de acuerdo con los pasos siguientes:

1) Interpretar su encabezamiento, como se describe en 7.4.4.1.
2) Como se describe en E.3.7, reponer a cero todas las estadística de codificación aritmética.
3) Invocar el procedimiento de decodificación de diccionario de patrones descrito en 6.7, con los parámetros del procedimiento de decodificación de diccionario de patrones fijados como se muestra en el cuadro 32.

Cuadro 32 – Parámetros utilizados para decodificar un segmento de diccionario de patrones

Nombre	Valor
HDMMR	Como se muestra en 7.4.4.1.1
HDTEMPLATE	Como se muestra en 7.4.4.1.1
HDPW	Como se muestra en 7.4.4.1.2
HDPH	Como se muestra en 7.4.4.1.3
GRAYMAX	Como se muestra en 7.4.4.1.4

7.4.5 Sintaxis de segmento de región de semitonos

Las partes datos de los tres tipos de segmento de región de semitonos ("región de semitonos intermedias", "región de tonos inmediata" y "región de semitonos sin pérdida inmediata") se codifican de manera idéntica, pero se actúa en ellas de manera diferente; véase 8.2. Aquí se especifica la sintaxis de la parte datos de estos tipos de segmento.

7.4.5.1 Encabezamiento de datos de segmento de región de semitonos

La parte datos de un segmento de región de semitonos empieza con un encabezamiento de datos de segmento de región de semitonos. Este encabezamiento contiene los campos mostrados en la figura 41, y descritos a continuación.

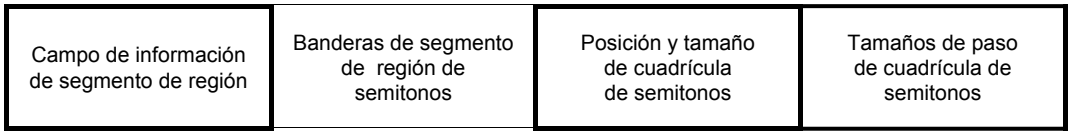


Figura 41 – Estructura de encabezamiento de datos de segmento de región de semitonos

Campo de información de segmento de región – Véase 7.4.1.

Banderas de segmento de región de semitonos – Véase 7.4.5.1.1.

Posición y tamaño de cuadrícula de semitonos – Véase 7.4.5.1.2.

Vector de cuadrícula de semitonos – Véase 7.4.5.1.3.

7.4.5.1.1 Banderas de segmento de región de semitonos

Este campo de un byte está formateado como se muestra en la figura 42, y se describe a continuación.

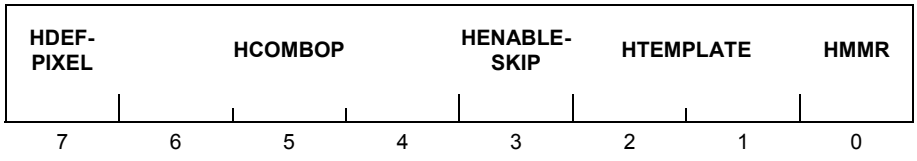


Figura 42 – Estructura de campo de banderas de segmento de región de semitonos

Bit 0 **HMMR**

Si este bit es **1**, el segmento utiliza la variante de codificación MMR. Si este bit es **0**, el segmento utiliza la variante de codificación aritmética.

Bits 1-2 **HTEMPLATE**

Este campo controla la plantilla utilizada para decodificar los planos de bits de valor de escala de grises de semitonos si **HMMR** es **0**. Si **HMMR** es **1**, este campo debe contener el valor **0**.

Bit 3 **HENABLESKIP**

Este campo controla si los valores de escala de grises que no contribuyen al contenido de la región se saltan durante la decodificación. Si **HMMR** es **1**, este campo debe contener el valor **0**.

Bits 4-6 **HCOMBOP**

Este campo tiene cinco valores posibles, representando cada valor uno de los cinco posibles operadores de combinación:

0 OR
1 AND
2 XOR
3 XNOR
4 REPLACE

Bit 7 **HDEFPIXEL**

Este bit contiene el valor inicial de cada uno de los píxels de la región de semitonos, antes de que se dibuje cualquier patrón.

7.4.5.1.2 Posición y tamaño de cuadrícula de semitonos

Este campo describe la posición y el tamaño de la cuadrícula de valores de escala de grises. Para una ilustración de esos valores, véase la figura 24. Está formateado como se muestra en la figura 43 y se describe a continuación.

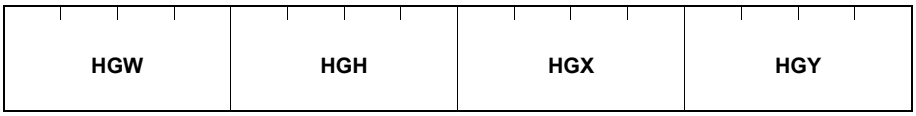


Figura 43 – Estructura de campo de posición de tamaño de cuadrícula de semitonos

HGW – Véase 7.4.5.1.2.1.

HGH – Véase 7.4.5.1.2.2.

HGX – Véase 7.4.5.1.2.3.

HGY – Véase 7.4.5.1.2.4.

7.4.5.1.2.1 Anchura de la imagen de escala de grises (HGW)

Este campo de cuatro bytes contiene la anchura de la formación de valores de escala de grises.

7.4.5.1.2.2 Altura de la imagen de escala de grises (HGH)

Este campo de cuatro bytes contiene la altura de la formación de valores de escala de grises.

7.4.5.1.2.3 Desplazamiento horizontal de la cuadrícula (HGX)

Este campo de cuatro bytes con signo contiene 256 veces el desplazamiento horizontal del origen de la cuadrícula de semitonos.

7.4.5.1.2.4 Desplazamiento vertical de la cuadrícula (HGY)

Este campo de cuatro bytes con signo contiene 256 veces el desplazamiento vertical del origen de la cuadrícula de semitonos.

7.4.5.1.3 Vector de cuadrículas de semitonos

Este campo describe el vector utilizado para dibujar la cuadrícula de valores de escala de grises. Para una ilustración de estos valores véase la figura 24. Está formateado como se muestra en la figura 44 y se describe a continuación.

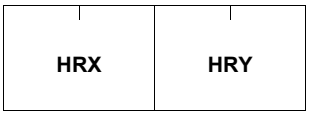


Figura 44 – Estructura de campo de vector de cuadrícula de semitonos

HRX – Véase 7.4.5.1.3.1.

HRY – Véase 7.4.5.1.3.2.

7.4.5.1.3.1 Coordenada horizontal del vector de cuadrícula de semitonos (HRX)

Este campo de dos bytes sin signo contiene 256 veces la coordenada horizontal del vector de cuadrícula de semitonos.

7.4.5.1.3.2 Coordenada vertical del vector de cuadrícula de semitonos (HRY)

Este campo de dos bytes sin signo contiene 256 veces la coordenada vertical del vector de cuadrícula de semitonos.

7.4.5.2 Decodificación de un segmento de región de semitonos

Un segmento de región de semitonos se decodifica de acuerdo con los pasos siguientes:

1) Interpretar su encabezamiento, como se describe en 7.4.5.1.
2) Decodificar (o recuperar los resultados de decodificación de) el segmento referenciado del diccionario de patrones.
3) Como se describe en E.3.7, reponer a cero todas las estadísticas de codificación aritmética a cero.
4) Invocar el procedimiento de codificación de región de semitonos descrito en 6.6, con los parámetros del procedimiento de decodificación de región de semitonos fijados como se muestra en el cuadro 33.

Cuadro 33 – Parámetros utilizados para decodificar un segmento de región de semitonos

Nombre	Valor
HBW	Según lo especificado por la anchura de mapa de bits de segmento de región en el encabezamiento de datos de segmento de región de este segmento
HBH	Según lo especificado por la altura de mapa de bits de segmento de región en el encabezamiento de datos de segmento de región de este segmento
HMMR	Como se muestra en 7.4.5.1.1
HTEMPLATE	Como se muestra en 7.4.5.1.1
HENABLESKIP	Como se muestra en 7.4.5.1.1
HCOMBOP	Como se muestra en 7.4.5.1.1
HDEFPIXEL	Como se muestra en 7.4.5.1.1
HGW	Como se muestra en 7.4.5.1.2.1
HGH	Como se muestra en 7.4.5.1.2.2
HGX	Como se muestra en 7.4.5.1.2.3
HGY	Como se muestra en 7.4.5.1.2.4
HRX	Como se muestra en 7.4.5.1.3.1
HRY	Como se muestra en 7.4.5.1.3.2
HNUMPATS	El número de patrones en el segmento de diccionario de patrones referenciado por este segmento
HPATS	Los patrones del segmento de diccionario de patrones referenciado por este segmento
HPW	La anchura, en píxels, de cada uno de los patrones contenidos en **HPATS**
HPH	La altura, en píxels, de cada uno de los patrones contenidos en **HPATS**

7.4.6 Sintaxis de segmento de región genérica

Las partes datos de los tres tipos de segmento de región genérica ("región genérica intermedia", "región genérica inmediata" y "región genérica sin pérdida inmediata") se codifican de manera idéntica, pero se actúa en ellas de manera diferente; véase 8.2. Aquí se especifica la sintaxis de la parte dato de estos tipos de segmento.

7.4.6.1 Encabezamiento de datos de segmento de región genérica

La parte datos de un segmento de región genérica empieza con un encabezamiento de datos de segmento de región genérica. Este encabezamiento contiene los campos mostrados en el figura 45, y descritos a continuación.

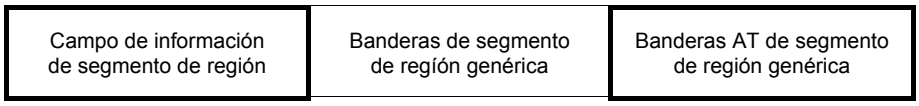


Figura 45 – Estructura de encabezamiento de datos de segmento de región genérica

Campo de información de segmento de región – Véase 7.4.1.

Banderas de segmento de región genérica – Véase 7.4.6.2.

Banderas AT de segmento de región genérica – Véase 7.4.6.3.

7.4.6.2 Banderas de segmento de región genérica

Este campo de un byte está formateado como se muestra en la figura 46 y se describe a continuación.

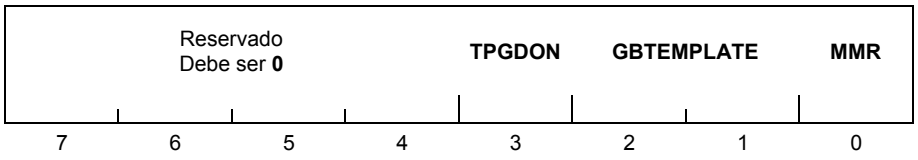


Figura 46 – Estructura de campo de banderas de segmento de región genérica

Bit 0 **MMR**

Bits 1-2 **GBTEMPLATE**

Este campo especifica la plantilla utilizada para la codificación aritmética basada en plantilla. Si **MMR** es **1**, este campo debe contener el valor cero.

Bit 3 **TPGDON**

Este campo especifica si se utiliza predicción típica para codificación directa genérica.

Bits 4-7 Reservado; debe ser cero.

7.4.6.3 Banderas AT de segmento de región genérica

Este campo sólo está presente si **MMR** es **0**. Si **GBTEMPLATE** es **0**, es un campo de ocho bytes, formateado como se muestra en la figura 47 y se describe a continuación.

$GBATX_1$	$GBATY_1$	$GBATX_2$	$GBATY_2$	$GBATX_3$	$GBATY_3$	$GBATX_4$	$GBATY_4$

Figura 47 – Estructura de campo de banderas AT de región genérica cuando GBTEMPLATE es 0

Byte 0 $\mathbf{GBATX_1}$

Byte 1 $\mathbf{GBATY_1}$

Byte 2 $\mathbf{GBATX_2}$

Byte 3 $\mathbf{GBATY_2}$

Byte 4 $\mathbf{GBATX_3}$

Byte 5 $\mathbf{GBATY_3}$

Byte 6 $\mathbf{GBATX_4}$

Byte 7 $\mathbf{GBATY_4}$

Si **GBTEMPLATE** es 1, 2 ó 3, es un campo de dos bytes formateado como se muestra en la figura 48 y se describe más abajo. Si **GBTEMPLATE** es 1, 2 ó 3, los valores de $\mathbf{GBATX_2}$ a $\mathbf{GBATX_2}$ y $\mathbf{GBATY_2}$ a $\mathbf{GBATY_4}$ son todos cero.

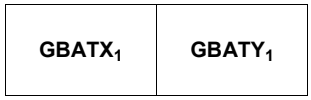


Figura 48 – Estructura de campo de banderas AT de región genérica cuando GBTEMPLATE no es 0

Byte 0 $\mathbf{GBATX_1}$

Byte 1 $\mathbf{GBATY_1}$

Los campos de las coordenadas X e Y de AT son valores con signo, y pueden tomar los valores permitidos de acuerdo con la figura 7.

7.4.6.4 Decodificación de un segmento de región genérica

Un segmento de región genérica se decodifica de acuerdo con los pasos siguientes:

1) Interpretar su encabezamiento, como se describe en 7.4.6.1.
2) Como se describe en E.3.7, reponer a cero todas las estadísticas de codificación aritmética.
3) Invocar el procedimiento de decodificación de región genérica descrito en 6.2, con los parámetros del procedimiento de decodificación de región genérica fijados como se muestra en el cuadro 34.

Cuadro 34 – Parámetros utilizados para decodificar un segmento de región genérica

Nombre	Valor
MMR	Como se muestra en 7.4.6.2
GBTEMPLATE	Como se muestra en 7.4.6.2
TPGDON	Como se muestra en 7.4.6.2
USESKIP	**0**
GBW	Según lo especificado por la anchura de mapa de bits de segmento de región en el encabezamiento de datos de segmento de región de este segmento.
GBH	Según lo especificado por la altura de mapa de bits de segmento de región en el encabezamiento de datos de segmento de región de este segmento
$\mathbf{GBATX_1}$	Véase 7.4.6.3
$\mathbf{GBATY_1}$	Véase 7.4.6.3
$\mathbf{GBATX_2}$	Véase 7.4.6.3
$\mathbf{GBATY_2}$	Véase 7.4.6.3
$\mathbf{GBATX_3}$	Véase 7.4.6.3
$\mathbf{GBATY_3}$	Véase 7.4.6.3
$\mathbf{GBATX_4}$	Véase 7.4.6.3
$\mathbf{GBATY_4}$	Véase 7.4.6.3

Como caso especial, según se indica en 7.2.7, un segmento de región genérica inmediata puede tener una longitud desconocida. En tal caso, también es posible que el segmento contenga menos filas de datos de mapa de bits que lo indicado en el campo de información de segmento de región del segmento.

Para que el decodificador pueda decodificar el segmento correctamente, tiene que leer el campo de cuenta de filas de cuatro bytes, que se almacena en los últimos cuatro bytes de la parte datos del segmento. Estos cuatro bytes pueden ser detectados sin conocer la longitud de los datos por adelantado. Si **MMR** es **1**, van precedidos por la secuencia de dos bytes `0x00 0x00`; si **MMR** es **0**, van precedidos por la secuencia de dos bytes `0xFF 0xAC`. El campo de cuenta de filas contiene el número real de filas contenidas en este segmento; no ha de ser superior al valor de altura de mapa de bits de segmento de región del campo de información de segmento de región del segmento.

NOTA – La secuencia `0x00 0x00` no puede producirse dentro de datos con codificación MMR; la secuencia `0xFF 0xAC` sólo puede producirse al final de datos codificados aritméticamente. Así pues, esas secuencias no pueden producirse por casualidad en los datos que son decodificados para generar el contenido de la región genérica.

7.4.7 Sintaxis de región de refinamiento genérica

Las partes datos de los tres tipos de segmento de región de refinamiento genérica ("región de refinamiento genérica intermedia", "región de refinamiento genérica inmediata" y "región de refinamiento genérica sin pérdida inmediata") se codifican de manera idéntica, pero se actúa en ellas de manera diferente; véase 8.2. Aquí se especifica la sintaxis de las partes datos de estos tipos de segmento.

7.4.7.1 Encabezamiento de datos de segmento de región de refinamiento genérica

La parte datos de un segmento de región de refinamiento genérica empieza con un encabezamiento de datos de segmento de región de refinamiento genérica. Este encabezamiento contiene los campos mostrados en la figura 49 y descritos a continuación.

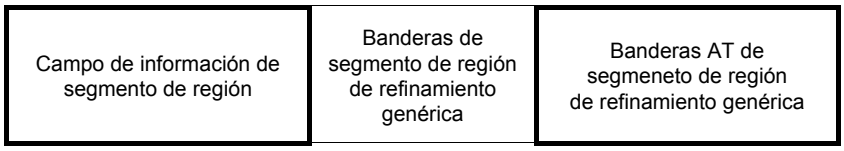


Figura 49 – Estructura de encabezamiento de datos de segmento de región de refinamiento genérica

Campo de información de segmento de región – Véase 7.4.1

Banderas de segmento de región de refinamiento genérica – Véase 7.4.7.2

Banderas AT de segmento de región de refinamiento genérica – Véase 7.4.7.3

7.4.7.2 Banderas de segmento de región de refinamiento genérica

El campo de un byte está formateado como se muestra en la figura 50 y se describe a continuación.

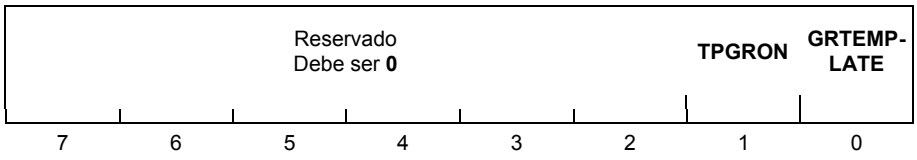


Figura 50 – Estructura de campo de banderas de segmento de región de refinamiento genérica

Bit 0 **GRTEMPLATE**

Este campo especifica la plantilla utilizada para la codificación aritmética basada en plantilla.

Bit 1 **TPGRON**

Este campo especifica si se utiliza predicción típica para refinamiento genérico.

Bits 2-7 Reservado; debe ser cero.

7.4.7.3 Banderas AT de segmento de región de refinamiento genérica

Este campo sólo está presente si **GRTEMPLATE** es **0**. Es un campo de cuatro bytes, formateado como se muestra en la figura 51 y se describe a continuación.

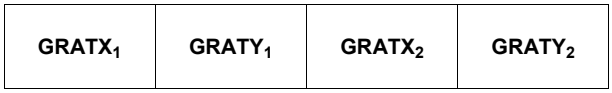


Figura 51 – Estructura de campo de banderas AT de región de refinamiento genérica

Byte 0 **$GRATX_1$**

Byte 1 **$GRATY_1$**

Byte 1 **$GRATY_1$**

Byte 2 **$GRATX_2$**

Byte 3 **$GRATY_2$**

Los campos de las coordenadas X e Y de AT son valores con signo, y pueden tomar uno de los valores permitidos de acuerdo con 6.3.5.3.

7.4.7.4 Selección de mapa de bits de referencia

Si este segmento hace referencia a otro segmento de región, fijar el mapa de bits de referencia **GRREFERENCE** para que sea el contenido actual de la memoria intermedia auxiliar asociada con el segmento de región al que hace referencia este segmento.

Si este segmento no hace referencia a otro segmento de región, fijar **GRREFERENCE** para que sea un mapa de bits que contenga el contenido actual de la memoria intermedia de página (véase la cláusula 8), restringido a la zona de la memoria intermedia de página especificada por el campo de información de segmento de región de este segmento.

7.4.7.5 Decodificación de un segmento de región de refinamiento genérica

Un segmento de región de refinamiento se decodifica de acuerdo con los pasos siguientes:

1) Interpretar su encabezamiento como se describe en 7.4.7.1. Si este segmento no hace referencia a otro segmento de región, su operador de combinación externo debe ser REPLACE. Si hace referencia a otro segmento de región, el tamaño de mapa de bits de región de este segmento, su ubicación y su operador de combinación externo deben ser iguales al tamaño de mapa de bits, la ubicación y el operador de combinación externo de ese otro segmento.

   NOTA – El requisito de que las ubicaciones y los operadores de combinación externos concuerden tiene por objeto ayudar a los decodificadores que desean producir imágenes de una página que está decodificada sólo parcialmente: así se asegura que todos los segmentos intermedios saben cuál es la ubicación final y el operador de combinación externo. Estas imágenes de página parcialmente decodificadas quedan fuera del ámbito de aplicación de la presente Recomendación | Norma Internacional.

2) Como se describe en E.3.7, reponer a cero todas las estadísticas de codificación aritmética.

3) Determinar la memoria intermedia asociada con el segmento de región al que hace referencia este segmento.

4) Invocar el procedimiento de decodificación de región de refinamiento genérica descrito en 6.3, con los parámetros del procedimiento de decodificación de región de refinamiento genérica fijados como se muestra en el cuadro 35.

Cuadro 35 – Parámetros utilizados para decodificar un segmento de región de refinamiento genérica

Nombre	Valor
GRTEMPLATE	Como se muestra en 7.4.6.2
TPGRON	Como se muestra en 7.4.6.2
GRW	Según lo especificado por la anchura de mapa de bits de segmento de región en el encabezamiento de datos de segmento de región de este segmento
GRH	Según lo especificado por la altura de mapa de bits de segmento de región en el encabezamiento de datos de segmento de región de este segmento
GRREFERENCE	Véase 7.4.7.4
GRREFERENCEDX	0
GRREFERENCEDY	0
$GRATX_1$	Véase 7.4.7.3
$GRATX_2$	Véase 7.4.7.3
$GRATY_1$	Véase 7.4.7.3
$GRATY_2$	Véase 7.4.7.3

7.4.8 Sintaxis de segmento de información de página

Un segmento de información de página describe una página. Contiene los campos mostrados en la figura 52 y descritos a continuación.

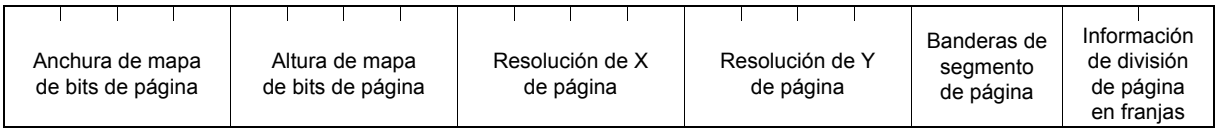


Figura 52 – Estructura de segmento de información de página

Anchura de mapa de bits de página – Véase 7.4.8.1.

Altura de mapa de bits de página – Véase 7.4.8.2.

Resolución de X de página – Véase 7.4.8.3.

Resolución de Y de página – Véase 7.4.8.4.

Banderas de segmento de página – Véase 7.4.8.5.

Información de división de página – véase 7.4.8.6.

El primer segmento asociado con cualquier página debe ser un segmento de información de página.

7.4.8.1 Anchura de mapa de bits de página

Valor de cuatro bytes que contiene la anchura en píxels del mapa de bits de la página.

7.4.8.2 Altura de mapa de bits de página

Valor de cuatro bytes que contiene la altura en píxels del mapa de bits de la página. En algunos casos, este valor puede no ser conocido en el momento en que se escribe el segmento de información de la página. Si tal sucede, el campo debe contener `0xFFFFFFFF`, y la altura de página real puede ser comunicada más tarde, cuando se conozca.

7.4.8.3 Resolución de X de página

Valor de cuatro bytes que contiene la resolución del medio página original, medida en píxels/metro en la dirección horizontal. Si este valor es desconocido, este campo debe contener `0x00000000`.

7.4.8.4 Resolución de Y de página

El valor de cuatro bytes que contiene la resolución del medio de página original, medida en píxels/metro en la dirección vertical. Si este valor es desconocido, este campo debe contener `0x00000000`.

7.4.8.5 Banderas de segmento de página

Éste es un campo de cuatro bytes. Está formateado como se muestra en la figura 53 y se describe a continuación.

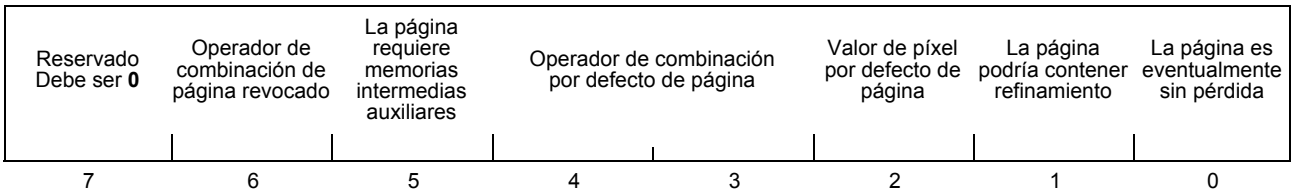


Figura 53 – Estructura de campo de banderas de segmento de página

Bit 0 La página es eventualmente sin pérdida. Si este bit es **0**, el fichero no contiene una representación sin pérdida de la página original (antes de la codificación). Si este bit es **1**, el fichero contiene información suficiente para reconstruir la página original.

Bit 1 La página podría contener refinamientos. Si este bit es **0**, no se puede asociar ningún segmento de región de refinamiento con la página. Si este bit es **1**, se pueden asociar esos segmentos con la página.

Bit 2 Valor de píxel por defecto de página. Este bit contiene el valor inicial de cada uno de los píxels de la página, antes de que se haya decodificado o dibujado cualquier segmento de región.

Bits 3-4 Operador de combinación por defecto de página. Este campo tiene cuatro valores posibles, representando cada valor uno de los cuatro posibles operadores de combinación:

0 OR
1 AND
2 XOR
3 XNOR

Este operador se utiliza para fusionar segmentos de región que se superponen, y también para combinar segmentos de región con el valor de píxel por defecto de página.

Bit 5 La página requiere memorias intermedias auxiliares. Si este bit es **0**, no se puede asociar con la página ningún segmento de región que requiera una memoria intermedia auxiliar. Si este bit es **1**, se pueden asociar con la página esos segmentos.

Bit 6 Operador de combinación de página revocado. Si este bit es **0**, cada uno de los segmentos de región directa asociados con esta página debe utilizar el operador de combinación por defecto de página. Si este bit es **1**, segmentos de región directa asociados con esta página pueden utilizar cualesquiera operadores de combinación.

NOTA 1 – Todos los segmentos de región, excepto los segmentos de región de refinamiento, son segmentos de región directa. Debido a los requisitos de 7.4.7.5 que limitan los operadores de combinación externos de segmentos de región de refinamiento, si este bit es **0**, los segmentos de región de refinamiento asociados con esta página que hacen referencia a segmentos no de región deben tener un operador de combinación externo de REPLACE, y todos los demás segmentos de región asociados con esta página deben tener el operador de combinación externo especificado por el "operador de combinación por defecto de página" de esta página.

NOTA 2 – Si todos los segmentos de región directa asociados con una página utilizan el mismo operador de combinación, se pueden reordenar en cierta medida (no es posible cambiar el orden relativo de cualquier segmento de refinamiento). Si alguno de ellos utiliza operadores de combinación diferentes, el decodificador no podrá efectuar ninguna de esas reordenaciones. Además, el decodificador no puede distinguir a partir de los encabezamientos de segmento si en la página se utilizan algunos de esos operadores de combinación no por defecto, por lo que este bit indica que el reordenamiento es posible, si el decodificador desea efectuarlo.

Bit 7 Reservado; debe ser **0**.

7.4.8.6 Información de división de página en franjas

Éste es un campo de dos bytes. Está formateado como se muestra en la figura 54 y se describe a continuación.

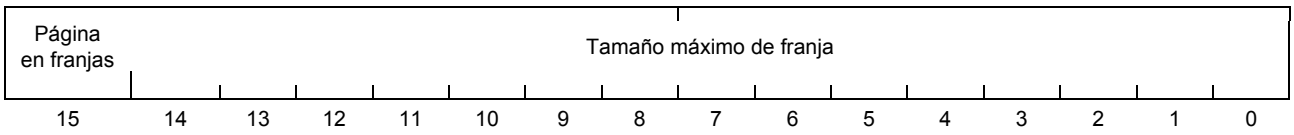


Figura 54 – Estructura de campo de información de división de página en franjas

Bits 0-14 Tamaño máximo de franja.

Bit 15 Página en franjas.

Si el bit "página en franjas" es **1**, la página puede tener segmentos fin de franja asociados con ella. En este caso, el tamaño máximo de cada franja (la distancia entre la fila final de un segmento fin de franja y la fila final del segmento fin de franja previo, o 0 en el caso del primer segmento fin de franja) no debe ser superior al tamaño máximo de franja de la página.

Si la altura del mapa de bits de la página es desconocida (indicada por una altura de mapa de bits de página de `0xFFFFFFFF`) el bit "página en franjas" debe ser **1**.

7.4.9 Sintaxis de segmento fin de página

Un segmento fin de página no tiene datos asociados. Su campo de longitud de datos de segmento debe ser cero.

El último segmento asociado con cualquier página debe ser un segmento fin de página. Cada página debe tener exactamente un segmento fin de página asociado a ella.

Si la altura de una página se desconocía al principio, debe haber por lo menos un segmento fin de página asociado con la página. En este caso, la fila final de esa última franja es la última fila del mapa de bits de la página y no puede haber ningún segmento de región entre el último segmento fin de franja y el segmento fin de página.

7.4.10 Sintaxis de segmento fin de franja

Un segmento fin de franja indica que el codificador ha terminado de codificar una porción de la página a la cual está asociado el segmento y no volverá a visitarla. Especifica la coordenada Y de una fila de la página; ningún segmento que siga al fin de franja puede modificar porción alguna del mapa de bits de página que se halle en esa fila o por encima de la misma; además, ningún segmento que preceda al fin de franja puede modificar porción alguna del mapa de bits de página que se halle por debajo de esa fila. Esta fila se dice que es la "fila final" de la franja.

NOTA 1 – En algunos casos, el decodificador sólo puede tener una cantidad limitada de memoria intermedia para el mapa de bits de la página, más pequeña que el tamaño de la página. Es preciso decir al decodificador cuándo puede sacar el contenido de la memoria intermedia y despejar la memoria para la próxima franja de la página.

La fila final especificada por un segmento fin de franja ha de estar por debajo de cualquier fila final previa de esa página.

Una página cuya altura se desconocía al principio debe contener por lo menos un segmento fin de franja.

NOTA 2 – En este caso se utiliza un segmento fin de franja para comunicar el tamaño de la página.

Los datos de segmento de un segmento fin de franja consisten en un valor de cuatro bytes que especifica la coordenada Y de la fila final.

NOTA 3 – Si el límite de una franja divide una línea de texto en dos partes, una superior y otra inferior, tambien se dividen los caracteres situados en ese límite de franja. Una manera de tratar esos caracteres consiste en codificarlos con una región genérica o agregar las mitades superiores y las mitades inferiores a un diccionario de símbolos y usar esas mitades superiores e inferiores como otros símbolos.

Sin embargo, si el codificador tiene más memoria intermedia que el decodificador, se hace posible un método más eficiente de codificar: el codificador puede (temporalmente) ignorar el límite de franja y generar una lista de símbolos y un diccionario de símbolos. Puede entonces codificar una región de texto en cada franja; la primera región de texto contiene todos los ejemplares de símbolo que afectan a la porción de la página por encima del límite de franja; la segunda región de texto contiene todos los ejemplares de símbolo que afectan a la porción por debajo del límite de franaja.

Esto significa que algunos ejemplares de símbolo aparecen dos veces, utilizando ambas veces el mismo símbolo del diccionario de símbolos. La primera vez que aparece, recodifica la mitad superior del carácter, mientras que la mitad inferior es eliminada por las reglas de dibujo utilizadas en el procedimiento de decodificación de la región de texto. La segunda vez que aparece, codificare, asimismo la mitad inferior del carácter: se codifica todo el carácter, pero la mitad superior es eliminada. Así pues este método de codificación reduce la cantidad de datos de diccionario de símbolos que se necesitan.

7.4.11 Sintaxis de segmento fin de fichero

Si un fichero contiene un segmento fin de fichero, ese segmento debe ser el último.

Un segmento fin de fichero no tiene datos asociados. Su campo de longitud de datos de segmento debe ser cero.

7.4.12 Sintaxis de segmento de perfiles

Un segmento de perfiles contiene una lista de los perfiles con los que está en conformidad un determinado tren de datos JBIG2. Si están presentes cualesquiera segmentos de perfiles, el primer segmento del tren de datos debe ser un segmento de perfiles, y no debe estar asociado con ninguna página. En el anexo F figura la lista de perfiles de la presente Recomendación | Norma Internacional.

Un segmento de perfiles empieza con un campo de cuatro bytes que contiene el número de perfiles listados. Este campo va seguido de muchos campos de cuatro bytes. Cada uno de esos campos contiene un número de identificación de perfil. El tren de datos debe estar en conformidad con cada uno de los perfiles listados.

Pueden estar presentes más de un segmento de perfiles. Si están presentes más de uno, cada uno de los segmentos distintos del primero debe estar asociado con una página. Ninguna página puede tener más de un segmento de perfiles asociado con ella. Además, cada segmento de perfiles después del primero debe ser más restrictivo que éste; es decir, debe tener la lista de todos los números de identificación de perfil indicados en el primer segmento, y posiblemente más. Los segmentos que constituyen cada página deben, colectivamente, estar en conformidad con cada uno de los perfiles listados en cualquier segmento de perfiles asociado con esa página.

NOTA – El segmento de perfiles global permite a un decodificador constatar rápidamente que no puede decodificar un determinado tren de datos. Haciendo que cada página contenga un segmento de perfiles posiblemente diferente (aunque más restrictivo) se facilita el movimiento de las páginas de un fichero a otro.

7.4.13 Sintaxis de segmento de tabla de códigos

En el anexo B se describe la sintaxis de un segmento de tabla de códigos.

7.4.14 Sintaxis de segmento de extensión

Los datos de un segmento de extensión empiezan con un encabezamiento de extensión:

Tipo de extensión: Campo de cuatro bytes que contiene una identificación del tipo de datos presentes en el segmento de extensión.

Los tres bits más significativos de este campo tienen un significado especial:

Bit 29 Reservado. Revisiones futuras de esta Recomendación | Norma Internacional pueden definir tipos de extensión; tipos de extensión pueden ser registrados también por otras partes. Otras partes sólo pueden registrar tipos de extensión con este bit igual a **0**; todos los tipos de extensión que tengan el bit 29 igual a **1** están reservados para revisiones futuras de la presente Recomendación | Norma Internacional.

Bit 30 Dependiente. Si este bit es **1**, la codificación de los datos del segmento de extensión depende de la codificación exacta de los datos del segmento al que hace referencia el segmento de extensión. Cualquier programa de manipulación de fichero que modifique esos segmentos referenciados ha de modificar en consecuencia los datos de este segmento de extensión; si no comprende el segmento de extensión (porque no reconoce su tipo de extensión), y si no es necesario un segmento de extensión, el segmento deberá ser suprimido.

EJEMPLO – Un segmento de extensión que contiene una verificación por redundancia cíclica (CRC, *cyclic redundance check*) del segmento al que hace referencia deberá ser señalado como dependiente.

Bit 31 Necesario. Si este bit es **1**, un decodificador que no sepa cómo descomponer extensiones de este tipo de segmento de extensión no podrá decodificar correctamente el fichero para producir las imágenes de página decodificada pretendidas.

NOTA – Esto tiene por objeto facilitar futuras ampliaciones de la norma JBIG2, tales como mejoras de codificación. Si este bit es **1**, un decodificador que no comprenda la extensión sabe que ha encontrado datos que son necesarios para la decodificación correcta de la página pero que no puede tratar. Por ejemplo, un segmento de extensión que contiene una región codificada con algún método nuevo sería señalado como "necesario", ya que sin esa región la imagen de la página está incompleta. Otro ejemplo podría ser el de un segmento de extensión que contuviera un conjunto de colores que deberían aplicarse a los símbolos en la página a medida que se dibujan.

Si el bit "necesario" es **1**, el bit "reservado" debe ser también **1**.

El resto de los datos del segmento de extensión sigue inmediatamente al campo de tipo de extensión, y se formatea de alguna manera específica del tipo de extensión.

7.4.15 Tipos de extensión definidos

En la actualidad están definidos los siguientes tipos de extensión.

`0x20000000` Comentario codificado en un solo byte. Véase 7.4.15.1.

`0x20000002` Comentario codificado en múltiples bytes. Véase 7.4.15.2

7.4.15.1 Comentario codificado en un solo byte

Un segmento de extensión comentario ASCII contiene información textual sobre algún otro segmento, página, o el tren binario en su conjunto. Si no hace referencia a ningún otro segmento ni está asociado con ninguna página, contiene algún conjunto de comentarios aplicables al tren binario completo. Si no hace referencia a ningún otro segmento, pero está asociado con alguna página, contiene algún conjunto de comentarios aplicables a esa página. Si hace referencia a algunos segmentos, contiene algún conjunto de comentarios aplicables a esos segmentos.

Un segmento comentario codificado en un solo byte contiene un cierto número de pares (de nombres, de valores). Cada elemento de cada par es una cadena de caracteres, terminada por un byte `0x00`. El último par va seguido por un byte `0x00` adicional Los demás bytes se interpretarán según ISO/CEI 8859-1:1998.

NOTA – Un segmento de extensión comentario codificado en un solo byte puede contener cualquier carácter ISO/CEI 8859-1 válido, incluidos los de valor superior a 127.

EJEMPLO – El comentario que contiene los pares siguientes:

```
Título    An Illustrated History of False Teeth
Autor     The Big Cheese
```

se almacena como la secuencia de bytes que se indica más abajo. Los bytes se muestran en números hexadecimales junto con sus equivalentes imprimibles, indicando "." un byte no imprimible. Se señala el tipo de extensión de cuatro bytes al comienzo de los datos del segmento:

```
20 00 00 00 54 69 74 6C 65 00 41 6E 20 49 6C 6C   Title.An Ill
75 73 74 72 61 74 65 64 20 48 69 73 74 6F 72 79 ustrated History
20 6F 66 20 46 61 6C 73 65 20 54 65 65 74 68 00  of False Teeth.
41 75 74 68 6F 72 00 54 68 65 20 42 69 67 20 43 Author.The Big C
68 65 65 73 65 00 00                            Cheese...
```

7.4.15.2 Comentario codificado en múltiples bytes

Un segmento de extensión comentario codificado en múltiples bytes se formatea de la misma manera que un segmento de extensión comentario codificado en un solo byte, con la salvedad de que cada uno de los caracteres individuales ocupa dos bytes, en la codificación (UCS-2) de ISO/CEI 10646-1:1993. Cada elemento de cada par del comentario termina con un `0x0000` y el par final va seguido de un `0x0000` adicional.

8 Composición de página

8.1 Modelo de decodificador

En esta cláusula se describe el resultado que producirá un decodificador conforme a la presente Recomendación | Norma Internacional cuando decodifique una página. Se hace especificando un conjunto de pasos que llevan al resultado correcto; un codificador que cumpla las especificaciones aludidas no necesita ejecutar esos pasos de manera exacta, pero producirá el mismo resultado que si se hubiera atenido al seguimiento de los mismos.

Aquí se describen únicamente las acciones efectuadas para decodificar una sola página. Un decodificador conforme con las especificaciones puede actuar en múltiples páginas a la vez, en tanto en cuanto produzca el resultado final correcto para cada una de ellas.

En la descripción que sigue, se supondrá para mayor sencillez que el decodificador tiene una sola memoria intermedia de página, memorias intermedias auxiliares para utilizar mientras se decodifica esa página y memoria de diccionario adicional. Se permiten decodificadores con otros componentes, siempre que produzcan la misma memoria intermedia de página que este decodificador abstracto.

Al final del proceso de decodificación, la memoria intermedia de página contiene el resultado de la decodificación de la página.

Cada memoria intermedia auxiliar tiene una ubicación asociada con ella; dicha ubicación es la del píxel superior izquierdo de la memoria intermedia, con relación al píxel superior izquierdo de la memoria intermedia de página. Algunos segmentos de región requieren la utilización de memorias intermedias auxiliares; otros pueden decodificar directamente en la memoria intermedia de página. Véase en 8.2 los detalles sobre cómo se han de interpretar combinaciones de segmentos de página.

La memoria de diccionario contiene la información obtenida decodificando segmentos de diccionario.

8.2 Composición de imagen de página

El mapa de bits final de cada página se codifica mediante cero o más segmentos de región asociados con esa página. Cada segmento de región describe parte del contenido de una región rectangular de la página. Puesto que estas regiones de la página pueden superponerse, y puesto que algunas partes de la página podrían ser descritas con diversos niveles de refinamiento, es importante definir cuáles son las reglas de composición de los segmentos de región. Además, un decodificador quizás desee visualizar representaciones intermedias de una página, en base a información parcial, por lo que resulta útil sugerir la interpretación de páginas parciales.

Como se describe en 7.4.8, cada página tiene un valor de píxel por defecto (**0** ó **1**) y uno de cuatro operador de combinación de entre cuatro posibles (OR, AND, XOR, XNOR); ambas cosas se especifican en su segmento de información de página. Cada segmento de región especifica también su propio operador de combinación. El bit bandera "operador de combinación de página revocado" del segmento de información de página especifica si alguno de los segmentos de región directa de la página revoca el operador de combinación de página. Si el bit es **0**, ningún segmento de región directa asociado con esta página revoca el operador de combinación de página. El decodificador puede utilizar esta información para optimizar su decodificación.

El resultado de la decodificación de un segmento de región es un mapa de bits. El tamaño del mismo y su ubicación con respecto a la memoria intermedia de página se indican en el campo de información de segmento de región.

El contenido final de la memoria intermedia de página que el decodificador producirá como resultado final de la decodificación de una página es el que se generaría ejecutando los pasos siguientes:

1) Decodificar el segmento de información de página.

2) Crear la memoria intermedia de página, del tamaño indicado en el segmento de información de página.

   Esto no es posible si la altura de la página es desconocida. Sin embargo, en tal caso la página debe ser dividida en franjas, y la altura máxima de las franjas debe ser especificada, y se puede crear la memoria intermedia de la página inicial con una altura igual al principio a esa altura máxima de franja. A medida que se va encontrando cada segmento fin de franja se puede ir aumentando la altura de la memoria intermedia de página, con lo que la última fila de la nueva memoria intermedia es la altura máxima de franja más la fila final de la franja previa. El segmento fin de página (junto con el último segmento fin de franja) permite determinar la altura real de la página.

   De manera alternativa, cuando se desconoce la altura de la página, el decodificador puede utilizar una memoria intermedia de tamaño fijo cuya altura sea igual a la altura máxima de franja de la página. A medida que se va encontrando cada segmento fin de franja, el decodificador puede imprimir, o copiar en alguna otra ubicación, todas las filas de la memoria intermedia hasta, e incluida, la fila final de la franja, a continuación despeja la memoria intermedia como preparación para la próxima franja. El decodificador puede seguir esta estrategia cuando la página esté dividida en franjas, incluso si la altura de la misma se conoce de antemano.

   NOTA 1 – Los pasos que se indican a continuación se pueden ejecutar con independencia de la estrategia de división en franjas que se siga. Las restricciones impuestas por la división en franjas garantizan que, una vez visto un segmento fin de franja, ninguna parte de la página por encima de, o en, esa fila fin de franja podrá ser modificada, y por ello en la presentación que sigue se hace referencia a una memoria intermedia de página que corresponde al tamaño completo de la página, incluso si la altura de la página se desconoce inicialmente.

3) Llenar la memoria intermedia de página con el valor del píxel por defecto de la página.

4) Buscar el siguiente segmento de región asociado con esa página.

5) Son posibles los casos siguientes:

   a) El segmento de región es un segmento de región directa inmediata. En este caso, decodificar el segmento de región. El resultado de la decodificación del segmento de región es un mapa de bits; combinar este mapa de bits con el contenido actual de la memoria intermedia de página, utilizando el operador de combinación del segmento de región.

   b) El segmento de región es un segmento de región directa intermedia. En este caso, atribuir una nueva memoria intermedia auxiliar, utilizando el tamaño y la ubicación especificados en el campo de información del segmento. Esta memoria intermedia está asociada inicialmente con el segmento de región. Decodificar el segmento de región, situando el mapa de bits resultante en la memoria intermedia auxiliar.

   c) El segmento de región es un segmento de región de refinamiento inmediata que no hace referencia a ningún otro segmento. En este caso, el segmento de región está actuando como un refinamiento de parte de la memoria intermedia de página. Efectuar el refinamiento de acuerdo con el segmento de región en la parte de la memoria intermedia de página especificada en el segmento de región, de acuerdo con los datos contenidos en el segmento de región de refinamiento. Esto sustituye una parte de la memoria intermedia de página por una versión refinada.

   d) El segmento de región es un segmento de región de refinamiento inmediata que hace referencia a otro segmento de región. Este otro segmento de región debe ser un segmento de región intermedia que se haya presentado previamente y al que todavía no ha hecho referencia ningún segmento de región de refinamiento. El otro segmento de región tiene, por tanto, una memoria intermedia auxiliar asociada con él. Se señala que el otro segmento de región puede, él mismo, ser un segmento de región de refinamiento intermedia. Efectuar la operación de refinamiento en la memoria intermedia auxiliar, de acuerdo con los datos contenidos en el segmento de región en curso, y combinar la memoria intermedia resultante con la memoria intermedia de página utilizando el operador de combinación del segmento de región en curso, en la ubicación asociada con la memoria intermedia auxiliar. Descartar la memoria intermedia auxiliar.

   e) El segmento de región es un segmento de región de refinamiento intermedia. Este segmento de región debe hacer referencia a otro segmento de región, que debe ser un segmento de región intermedia que se haya producido previamente y al que todavía no ha hecho referencia ningún segmento de región de refinamiento; el otro segmento de región tiene, por tanto, una memoria intermedia auxiliar asociada con él. Efectuar la operación de refinamiento en la memoria intermedia auxiliar, de acuerdo con los datos contenidos en el segmento de región en curso. Sustituir el contenido previo de la memoria tampón intermedia por el mapa de bits resultante del refinamiento. Cambiar la asociación de la memoria intermedia auxiliar, de manera que esté asociada ahora con el segmento de región en curso, y deje de estarlo con el otro segmento de región.

6) Repetir los pasos 4) y 5) hasta que no haya más segmentos de región asociados con la página. En este punto, todas las memorias intermedias auxiliares que hayan sido atribuidas deberán haber sido refinadas, dibujadas en la página, y descartadas, como se describe en el paso 5 d); no deberá permanecer ninguna memoria intermedia auxiliar.

7) El resultado de descomprimir esa página viene dado por el contenido final de la memoria intermedia de página.

Las reglas descritas en el paso 5) son muy sencillas en principio. Los segmentos de región intermedia se han de dibujar en la memoria intermedia de página, ya sea dibujándolos sencillamente (segmentos directos, paso 5) a), refinando una parte de la memoria intermedia de página (segmentos de refinamiento que no hacen referencia a ningún otro segmento, paso 5) c) o refinando y dibujando a continuación una memoria intermedia auxiliar (segmentos de refinamiento que hacen referencia a algún otro segmento, paso 5) d). Los segmentos de región intermedia conllevan la creación de una memoria intermedia auxiliar que contenga el mapa de bits de la región (segmentos directos, paso 5) b) o la sustitución del contenido actual de una memoria intermedia auxiliar (segmentos de refinamiento, paso 5) e).

Los siguientes son ejemplos de aplicación de esta regla:

EJEMPLO 1 – Si la página no contiene segmentos de región, la memoria intermedia de la página se llena por completo con el valor del píxel por defecto de la página.

EJEMPLO 2 – El segmento de información de página para la página 1 especifica que el operador de combinación por defecto de la página es OR y el valor del píxel por defecto de la página es **0**. Los segmentos de región asociados con la página 1 son, en orden:

- Segmento 3, un segmento de región de texto sin pérdida inmediata cuyo operador de combinación externo es OR.
- Segmento 4, un segmento de región genérica sin pérdida inmediata cuyo operador de combinación externo es OR.
- Segmento 6, un segmento de región de semitonos sin pérdida inmediata cuyo operador de combinación externo es OR.

  El mapa de bits de página resultante se puede obtener decodificando los segmentos 3, 4 y 6, y dibujando cada uno de ellos en su ubicación de región especificada, utilizando OR, en un mapa de bits que contiene inicialmente **0** en todas partes. Se señala que el orden en el que estos tres segmentos se decodifican y dibujan no afecta al mapa de bits de página resultante. Además, si el segmento 3 tiene un operador de combinación interno de OR y un valor de píxel por defecto de **0**, se pueden dibujar simplemente los ejemplares de símbolo directamente en la memoria intermedia de la página; no es necesario decodificarlo en un mapa de bits temporal y dibujar a continuación ese mapa de bits en la memoria intermedia de la página. Una observación similar es aplicable para el segmento 6.

EJEMPLO 3 – El segmento de información de página para la página 2 especifica que el operador de combinación por defecto de la página es OR y el valor de píxel por defecto de la página es **0**. Los segmentos de región asociados con la página 2 son, en orden:

- Segmento 7, un segmento de región de texto intermedia.
- Segmento 8, un segmento de región de mapa de bits genérica intermedia.
- Segmento 13, un segmento de región de refinamiento de mapa de bits genérica inmediata cuyo operador de combinación externo es OR y hace referencia al segmento 8.
- Segmento 14, un segmento de región de refinamiento de mapa de bits genérica inmediata cuyo operador de combinación externo es OR y hace referencia al segmento 7.
- Segmento 19, un segmento de región de texto inmediata cuyo operador de combinación externo es OR.
- Segmento 22, un segmento de región de mapa de bits genérica inmediata cuyo operador de combinación externo es OR.

La memoria intermedia de página resultante es la memoria intermedia que se hubiera obtenido ejecutando los pasos siguientes:

1) Llenar la memora intermedia de página con el valor **0**.

2) Decodificar el segmento 7 en una memoria intermedia auxiliar.

3) Decodificar el segmento 8 en una memoria intermedia auxiliar.

4) Mejorar la memoria intermedia auxiliar del segmento 8, de acuerdo con la información de refinamiento del segmento 13, y dibujar la memoria intermedia refinada en la memoria intermedia de página utilizando OR, descartando la memoria intermedia auxiliar una vez hecho esto.

5) Refinar la memoria intermedia auxiliar del segmento 7, de acuerdo con la información de refinamiento del segmento 14, y dibujar la memoria intermedia mejorada en la memoria intermedia de página utilizando OR, descartando la memoria intermedia auxiliar una vez hecho esto.

6) Decodificar el segmento 19 y dibujar el mapa de bits resultante en la memoria intermedia de página utilizando OR.

7) Decodificar el segmento 22 y dibujar el mapa de bits resultante en la memoria intermedia de página utilizando OR.

El resultado correcto se obtiene también cualquiera que sea el orden en que se ejecuten los pasos 4) a 7); por eso, un decodificador que cumpla las especificaciones tiene la libertad de ejecutar esos pasos en cualquier orden. De hecho, cualquiera que sea el orden de los pasos 2) a 7) se produce el resultado correcto, siempre que el paso 2) se ejecute antes que el paso 5) y el paso 3) antes que el paso 4).

EJEMPLO 4 – Si una página contiene varios segmentos de región con codificación directa inmediata que no revocan el operador de combinación de la página, y un segmento de región de mejora inmediata que no hace referencia a ningún otro segmento, la memoria intermedia de página resultante es la memoria intermedia que se obtendría:

- llenando la memoria intermedia de la página con el valor de píxel por defecto de la página;
- dibujando todos los segmentos de región con codificación directa que preceden al segmento de región de refinamiento;
- refinando la porción de la región cubierta por el segmento de región de refinamiento;
- dibujando todos los segmentos de región con codificación directa que siguen al segmento de región de refinamiento.

En este caso, el orden de dibujo no tiene importancia: todos los segmentos intermedios que preceden al segmento de refinamiento deberán dibujarse antes de dibujar el segmento de refinamiento, y el segmento de refinamiento deberá dibujarse antes que cualquiera de los segmentos inmediatos que le siguen.

NOTA 2 – Quizás, en algunos casos, desee el decodificador visualizar alguna forma intermedia de la página. Por ejemplo, quizás quiera presentar al usuario una visualización progresiva del contenido de la página a medida que los segmentos de página se reciben por algún medio de transmisión. Cualquier mapa de bits de página intermedio que se visualice depende por completo del decodificador, y no es especificado por la presente Recomendación | Norma Internacional.

Una estrategia que podría utilizar un decodificador consiste en tomar los contenidos actuales de la memoria intermedia de página y de cualesquiera memorias intermedia auxiliares activas en ese momento, combinar todas estas memorias intermedias utilizando el operador de combinación por defecto de la página y hacer una presentación visual de todo ello al usuario. Si el operador de combinación de página es XOR o XNOR, esta combinación se puede hacer de manera reversible, es decir, podría hacerse en la memoria intermedia de página efectiva y deshacerse a continuación una vez visualizada para el usuario. Si el operador de combinación de página es OR o AND, la combinación no es reversible y se necesita memoria intermedia adicional para retener los resultados de la combinación.

La anterior descripción paso a paso tiene por objeto especificar solamente los resultados de la descompresión. Un decodificador conforme con las especificaciones puede efectuar los pasos que desee, en tanto en cuanto la memoria intermedia de página final sea la misma que se obtendría siguiendo los pasos indicados.

EJEMPLO 5 – Un decodificador podría observar que un segmento de región intermedio hace referencia a una región de la página a la que no se superpone ningún otro segmento de región y, por ello, no atribuir realmente una memoria intermedia auxiliar a ese segmento de región sino utilizar la memoria intermedia de página inmediatamente. Esto sólo puede hacerlo si está seguro de que no se alterarán los resultados de la decodificación de los segmentos de región de la página.

Anexo A

Procedimiento de decodificación aritmética de enteros

(Este anexo es parte integrante de alta Recomendación | Norma Internacional)

A.1 Descripción general

Esta Recomendación | Norma Internacional utiliza un cierto número de procedimientos de decodificación aritmética para decodificar valores enteros. Son los siguientes:

IAAI Utilizado para decodificar el número de ejemplares de símbolo en una agregación

IADH Utilizado para decodificar la diferencia en altura entre dos clases de altura

IADS Utilizado para decodificar la coordenada S del segundo ejemplar de símbolo y ejemplares de símbolo subsiguientes en una franja

IADT Utilizado para decodificar la coordenada T del segundo ejemplar de símbolo y ejemplares de símbolo subsiguientes en una franja

IADW Utilizado para decodificar la diferencia de anchura entre los símbolos en una clase de altura

IAEX Utilizado para decodificar banderas de exportación

IAFS Utilizado para decodificar la coordenada S del primer ejemplar de símbolo en una franja

IAID Utilizado para decodificar los ID símbolo de ejemplares de símbolo

IAIT Utilizado para decodificar la coordenada T de los ejemplares de símbolo en una franja

IARDH Utilizado para decodificar la altura delta (diferencia de) de refinamientos de ejemplares de símbolo

IARDW Utilizado para decodificar la anchura delta (diferencia de) de refinamientos de ejemplares de símbolo

IARDX Utilizado para decodificar la posición X delta (diferencia de) de refinamientos de ejemplares de símbolo

IARDY Utilizado para decodificar la posición Y delta (diferencia de) de refinamientos de ejemplares de símbolo

IARI Utilizado para decodificar el bit R_I de ejemplares de símbolo

Estos procedimientos se utilizan para decodificar valores enteros (que pueden incluir el valor fuera de banda OOB). La codificación de un entero se basa en un árbol de decisión.

La invocación de un procedimiento de decodificación aritmética de enteros implica la decodificación de una secuencia de bits, cada uno de los cuales se decodifica utilizando un contexto formado por los bits decodificados previamente en esta invocación. Cada contexto de cada procedimiento de decodificación aritmética de enteros tiene su propia estimación de probabilidad adaptativa utilizada por el codificador aritmético subyacente, descrito en el anexo E. La secuencia de bits decodificados es interpretada para formar un valor.

El cuadro A.1 es utilizado por todos los procedimientos de decodificación aritmética de enteros, excepto el IAID.

A.2 Procedimiento de decodificación de valores (excepto IAID)

El diagrama de flujo de la figura A.1 se utiliza como parte del procedimiento de decodificación. Produce dos valores, V y S. El resultado del procedimiento de decodificación aritmética de enteros es igual a:

- V si $S = \mathbf{0}$
- $-V$ si $S = \mathbf{1}$ y $V > 0$
- OOB si $S = \mathbf{1}$ y $V = 0$

Así pues, V representa el valor absoluto del valor entero que se decodifica, y S representa el signo; el valor por otra parte redundante –0 se interpreta como que significa "OOB".

En la figura A.1, cada bit se decodifica en un contexto formado a partir del procedimiento particular de decodificación aritmética de enteros invocado, y los bits decodificados previamente en esta invocación de ese procedimiento de decodificación. Este contexto se forma como sigue.

1) Fijar:

$$PREV = 1$$

2) Seguir el diagrama de flujo de la figura A.1. Decodificar cada bit con CX igual a "IAx + PREV" donde "IAx" representa el identificador del procedimiento de codificación aritmética de enteros que se está utilizando, "+" representa concatenación, y se utilizan los 9 bits de PREV situados más a la derecha.

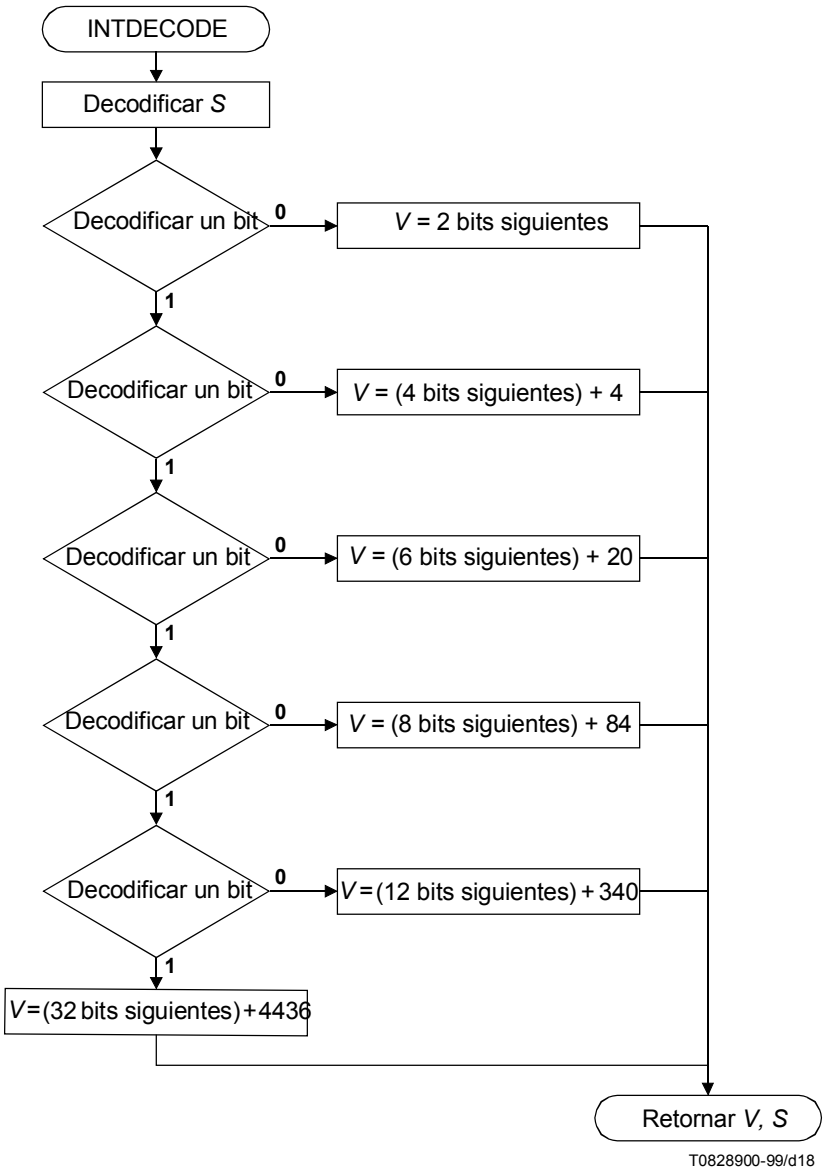


Figura A.1 – Diagrama de flujo de los procedimientos de decodificación aritmética de enteros (excepto IAID**)**

3) Después de codificar cada bit: Si PREV < 256 fijar

$$PREV = (PREV << 1) \text{ OR } D$$

De otro modo, fijar:

$$PREV = (((PREV << 1) \text{ OR } D) \text{ AND } 511) \text{ OR } 256$$

donde D representa el valor del bit que acaba de ser decodificado.

Así pues, PREV contiene siempre los valores de los ocho bits más recientemente decodificados, más un bit **1** delantero, que se utiliza para indicar el número de bits decodificados hasta el momento.

4) La secuencia de bits decodificados, interpretada de acuerdo con el cuadro A.1, da el valor que es el resultado de esta invocación del procedimiento de decodificación aritmética de enteros.

Obsérvese que cada tipo de datos, y cada procedimiento de decodificación aritmética de enteros, utiliza un conjunto distinto de contextos: los contextos utilizados para IAFS son distintos de los contextos utilizados para IADW, por ejemplo.

Cuadro A.1 – Tabla de procedimientos de decodificación aritmética de enteros

VAL	Codificación
0 . . . 3	**00** + VAL codificado como 2 bits
–1	**1001**
–3 . . . –2	**101** + (–VAL – 2) codificado como 1 bit
4 . . . 19	**010** + (VAL – 4) codificado como 4 bits
–19 . . . –4	**110** + (–VAL – 4) codificado como 4 bits
20 . . . 83	**0110** + (VAL – 20) codificado como 6 bits
–83 . . . –20	**1110** + (–VAL – 20) codificado como 6 bits
84 . . . 339	**01110** + (VAL – 84) codificado como 8 bits
–339 . . . –84	**11110** + (–VAL – 84) codificado como 8 bits
340 . . . 4435	**011110** + (VAL – 340) codificado como 12 bits
–4435 . . . –340	**111110** + (–VAL – 340) codificado como 12 bits
4436 . . . ∞	**011111** + (VAL – 4436) codificado como 32 bits
–∞ . . . –4436	**111111** + (–VAL – 4436) codificado como 32 bits
OOB	**1000**

EJEMPLO – Una invocación de IADW podría desarrollarse como sigue.

- Fijar CX en "IADW**000000001**". Esto identifica una determinada estimación de probabilidad adaptativa. Decodificar un bit; suponer que el valor decodificado (D) es **0**.
- Utilizando CX = IADW**000000010**, decodificar un bit; suponer que el valor decodificado es **1**.
- Utilizando CX = IADW**000000101**, decodificar un bit; suponer que el valor decodificado es **0**.
- Utilizando CX = IADW**000001010**, decodificar un bit; suponer que el valor decodificado es **1**.
- Utilizando CX = IADW**000010101**, decodificar un bit; suponer que el valor decodificado es **0**.
- Utilizando CX = IADW**000101010**, decodificar un bit; suponer que el valor decodificado es **0**.
- Utilizando CX = IADW**001010100**, decodificar un bit; suponer que el valor decodificado es **0**.
- La secuencia de bits decodificados hasta ahora es **0101000**. De acuerdo con el cuadro A.1 y la figura A.1, esto corresponde al valor 12 ($S = 0$, $V = 12$), que es el resultado de esta invocación de IADW.

Un contexto es identificado por un nombre de procedimiento de decodificación aritmética de enteros y una secuencia de nueve bits. Cada procedimiento de decodificación aritmética de enteros requiere 512 bytes de almacenamiento para su memoria de contexto.

A.3 Procedimiento de decodificación IAID

Este procedimiento de decodificación es diferente de todos los demás procedimientos de decodificación aritmética de enteros. Utiliza representaciones de longitud fija de los valores que se decodifican, y no limita el número de bits decodificados previamente utilizados como parte del contexto. La longitud es igual a **SBSYMCODELEN**. Este procedimiento de decodificación sólo es invocado desde el procedimiento de decodificación de región de texto, por lo que en el momento de la invocación se conoce **SBSYMCODELEN**.

El procedimiento para decodificar un entero utilizando el procedimiento de decodificación IAID es como sigue.

1) Fijar:

$$PREV = 1$$

2) Decodificar **SBSYMCODELEN** bits como sigue:

   a) Decodificar un bit con CX igual a "IAID + PREV" donde "+" representa concatenación, y se utilizan los **SBSYMCODELEN** + 1 bits situados más a la derecha de PREV.

   b) Después de codificar cada bit, fijar:

$$PREV = (PREV << 1) \text{ OR } D$$

   donde D representa el valor del bit que acaba de ser decodificado.

   Así pues, PREV contiene siempre los valores de todos los bits decodificados hasta entonces, más un bit **1** delantero, que se utiliza para indicar el número de bits decodificados hasta ese momento.

3) Una vez que se han decodificado **SBSYMCODELEN** bits, fijar:

$$PREV = PREV - 2^{\mathbf{SBSYMCODELEN}}$$

   Este paso tiene el efecto de liberar el bit situado más arriba (**1** delantero) de PREV antes de retornarlo.

4) El contenido de PREV es el resultado de esta invocación del procedimiento de decodificación IAID.

El número de contextos requeridos es $2^{\mathbf{SBSYMCODELEN}}$, que es menos de dos veces el ID símbolo máximo. La cantidad de memoria necesaria para contextos se puede calcular, por tanto, a partir del número de símbolos, y normalmente no es de más de dos bytes por símbolo.

EJEMPLO – Supóngase que **SBSYMCODELEN** = 3. Una invocación de IAID podría desarrollarse como sigue.

- Utilizando la estimación de probabilidad adaptativa identificada fijando CX igual a "IAID**0001**", decodificar un bit. Suponer que el valor decodificado es **0**.
- Utilizando CX = IAID**0010**, decodificar un bit; suponer que el valor decodificado es **1**.
- Utilizando CX = IAID**0101**, decodificar un bit; suponer que el valor decodificado es **0**.
- En este punto, PREV = **1010**. Aplicar el paso 3; PREV es ahora **010**. El resultado de esta invocación del procedimiento de decodificación IAID es, por tanto, el valor **010**, o (en base decimal) 2.

La identificación de contexto utilizada aquí depende del valor de **SBSYMCODELEN**. En todos los casos, los contextos de codificador aritméticos se repondrán entre cambios de **SBSYMCODELEN**: **SBSYMCODELEN** nunca cambia durante la decodificación de un solo segmento (pero puede cambiar entre segmentos).

Anexo B

Procedimiento de decodificación de tabla Huffman

(Este anexo es parte integrante de alta Recomendación | Norma Internacional)

B.1 Descripción general

Las tablas de códigos se pueden utilizar para decodificar cualquier tipo de datos numéricos en codificadores de la variante Huffman. En muchos sitios en los que se utiliza una tabla, el decodificador tiene la opción de utilizar una de las tablas normalizadas, o de enviar su propia tabla. Un segmento de tabla de códigos proporciona la manera de enviar esa tabla de usuario. La tabla de códigos es una lista de líneas de tabla de códigos, cada una de las cuales describe cómo se codifica un valor único, o un valor de una gama especificada. Una tabla puede, facultativamente, codificar un OOB, que es una señal fuera de banda del procedimiento de codificación que utiliza la tabla.

B.2 Estructura de tabla de códigos

La figura B.1 muestra la estructura interna de una tabla Huffman codificada. Consta de un conjunto de líneas de tabla, cada una de las cuales describe la codificación de una gama de valores numéricos. Hay también, potencialmente, dos líneas de tabla adicionales especiales que codifican las gamas "de extremo abierto". El valor más pequeño que puede ser codificado en una tabla descrita de acuerdo con esta especificación es –2147483648 (-2^{31}) y el valor más grande es 2147483647 ($2^{31} - 1$), por lo que estas gamas no son realmente de extremo abierto. Hay también, potencialmente, una línea de tabla especial adicional que codifica un valor fuera de banda (OOB).

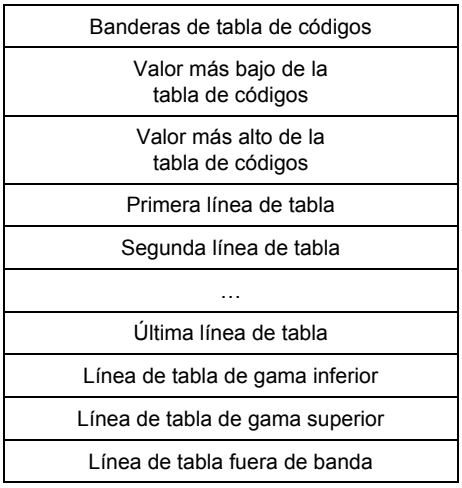


Figura B.1 – Estructura codificada de una tabla Huffman

Cada línea de tabla especifica la longitud del prefijo que está asociado con ella y el número de bits que siguen a ese prefijo para codificar un valor.

Un decodificador que decodifique una tabla Huffman codificada deberá decodificar la tabla que se produce ejecutando los pasos siguientes:

1) Decodificar el campo banderas de tablas de códigos descrito en B.2.1. De esta manera se fijan los valores HTOOB, HTPS y HTRS.

2) Decodificar el campo del valor más bajo de la tabla de códigos descrito en B.2.2. Sea HTLOW el valor decodificado.

3) Decodificar el campo del valor más alto de la tabla de códigos descrito en B.2.3. Sea HTHIGH el valor decodificado.

4) Fijar:

$$\begin{aligned} \text{CURRANGELOW} &= \text{HTLOW} \\ \text{NTEMP} &= 0 \end{aligned}$$

5) Decodificar cada línea de tabla como sigue:

   a) Leer HTPS bits. Fijar PREFLEN[NTEMP] en el valor decodificado.

   b) Leer HTRS bits. Sea RANGELEN[NTEMP] el valor decodificado.

   c) Fijar:

$$\begin{aligned} \text{RANGELOW[NTEMP]} &= \text{CURRANGELOW} \\ \text{CURRANGELOW} &= \text{CURRANGELOW} + 2^{\text{RANGELEN[NTEMP]}} \\ \text{NTEMP} &= \text{NTEMP} + 1 \end{aligned}$$

   d) Si CURRANGELOW $\geq$ HTHIGH, avanzar al paso 6).

6) Leer HTPS bits. Sea LOWPREFLEN el valor leído.

7) Fijar:

$$\begin{aligned} \text{PREFLEN[NTEMP]} &= \text{LOWPREFLEN} \\ \text{RANGELEN[NTEMP]} &= 32 \\ \text{RANGELOW[NTEMP]} &= \text{HTLOW} - 1 \\ \text{NTEMP} &= \text{NTEMP} + 1 \end{aligned}$$

   Ésta es la línea de tabla de gama inferior de esta tabla.

8) Leer HTPS bits. Sea HIGHPREFLEN el valor leído.

9) Fijar:

$$\begin{aligned} \text{PREFLEN[NTEMP]} &= \text{HIGHPREFLEN} \\ \text{RANGELEN[NTEMP]} &= 32 \\ \text{RANGELOW[NTEMP]} &= \text{HTHIGH} \\ \text{NTEMP} &= \text{NTEMP} + 1 \end{aligned}$$

   Ésta es la línea de tabla de gama superior de esta tabla.

10) Si HTOOB es **1**,

   a) Leer HTPS bits. Sea OOBPREFLEN el valor leído.

   b) Fijar:

$$\begin{aligned} \text{PREFLEN[NTEMP]} &= \text{OOBPREFLEN} \\ \text{NTEMP} &= \text{NTEMP} + 1 \end{aligned}$$

   Ésta es la línea de tabla fuera de banda de esta tabla. Se señala que no hay ninguna gama asociada con este valor.

11) Crear los códigos de prefijo utilizando el algoritmo descrito en B.3.

B.2.1 Banderas de tabla de códigos

Este campo de un byte tiene los bits definidos siguientes:

Bit 0 HTOOB. Si este bit es **1**, la tabla puede codificar cualquier valor fuera de banda.

Bits 1-3 Número de bits utilizados en los campos de tamaño de prefijo de línea de tabla de códigos. El valor de HTPS es el valor de este campo más uno.

Bits 4-6 Número de bits utilizados en los campos de tamaño de gama de línea de tabla de códigos. El valor de HTRS es el valor de este campo más uno.

Bit 7 Reservado; debe ser cero.

B.2.2 Valor más bajo de tabla de códigos

Este campo de cuatro bytes con signo es el límite inferior de la primera línea de tabla en la tabla codificada.

B.2.3 Valor más alto de tabla de códigos

Este campo de cuatro bytes con signo es mayor que el límite superior de la última línea de tabla normal en la tabla codificada.

B.3 Asignación de códigos de prefijo

Dadas las longitudes de código de prefijo, PREFLEN, y el número de códigos que se han de asignar, NTEMP, este algoritmo asigna un código de prefijo único a cada línea de tabla, de la longitud dada por PREFLEN para esa línea de tabla.

Se señala que el valor 0 de PREFLEN indica que esa línea de tabla no se utiliza nunca.

1) Construir un histograma con la matriz LENCOUNT contando el número de veces que figura cada valor de longitud de prefijo en PREFLEN: LENCOUNT[*I*] es el número de veces que el valor *I* figura en la matriz PREFLEN.
2) Sea LENMAX el valor mayor para el cual LENCOUNT[LENMAX] > 0. Fijar:

$$\begin{aligned} \text{CURLEN} &= 1 \\ \text{FIRSTCODE}[0] &= 0 \\ \text{LENCOUNT}[0] &= 0 \end{aligned}$$

3) Mientras CURLEN $\leq$ LENMAX, efectuar las operaciones siguientes:
   a) Fijar:

$$\begin{aligned} \text{FIRSTCODE}[\text{CURLEN}] &= (\text{FIRSTCODE}[\text{CURLEN} - 1] + \text{LENCOUNT}[\text{CURLEN} - 1]) \times 2 \\ \text{CURCODE} &= \text{FIRSTCODE}[\text{CURLEN}] \\ \text{CURTEMP} &= 0 \end{aligned}$$

   b) Mientras CURTEMP < NTEMP, efectuar las operaciones siguientes.
      i) Si PREFLEN[CURTEMP] = CURLEN, fijar

$$\begin{aligned} \text{CODES}[\text{CURTEMP}] &= \text{CURCODE} \\ \text{CURCODE} &= \text{CURCODE} + 1 \end{aligned}$$

      ii) Fijar CURTEMP = CURTEMP + 1
   c) Fijar:

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

Después de ejecutar este algoritmo, la línea de tabla *I* tiene asignado un código de longitud PREFLEN[*I*] bits, cuyo valor se almacena en los bits PREFLEN[*I*] de orden bajo de CODES[*I*], a menos que PREFLEN[*I*] sea igual a cero, en cuyo caso no se ha asignado a esa línea de tabla ningún código.

B.4 Utilización de una tabla Huffman

Para decodificar un valor utilizando una tabla Huffman, ejecutar los pasos siguientes:

1) Leer un bit cada vez hasta que la cadena de bits leídos coincida con el código asignado a una de las líneas de tabla. Tal cosa es posible porque ningún código forma un prefijo de ningún otro código. Sea *I* el índice de la línea de tabla cuyo código fue decodificado.

2) Leer RANGELEN[*I*] bits. Sea HTOFFSET el valor leído.

3) Si HTOOB es **1** para esta tabla, y la línea de tabla *I* es la línea de tabla fuera de banda de esta tabla, fijar:

$$\text{HTVAL} = \text{OOB}$$

4) De otro modo, si la línea de tabla *I* es la línea de tabla de gama más baja de esta tabla, fijar:

$$\text{HTVAL} = \text{RANGELOW}[I] - \text{HTOFFSET}$$

5) De otro modo, fijar:

$$\text{HTVAL} = \text{RANGELOW}[I] + \text{HTOFFSET}$$

El valor de HTVAL es el valor decodificado utilizando esta tabla. Se señala que puede ser un valor numérico o el valor especial OOB.

EJEMPLO – La codificación del cuadro B.1 podría ser la secuencia de bytes, en hexadecimal:

```
0x42  0x00  0x00  0x00  0x00  0x00  0x01
0x01  0x10  0x49  0x23  0x81  0x80
```

La decodificación de lo anterior de acuerdo con el algoritmo de B.2 procede como sigue:

- El campo de banderas de tabla de códigos, 0x42. Este campo se descompone en los campos, en binario, **0 100 001 0**, que se decodifican para producir las asignaciones:

$$\begin{aligned} \text{HTOOB} &= \mathbf{0} \\ \text{HTPS} &= 2 \\ \text{HTRS} &= 5 \end{aligned}$$

- El campo del valor más bajo de la tabla de códigos, y el valor de HTLOW, `0x00000000`.
- El campo del valor más alto de la tabla de códigos, y el valor de HTHIGH, `0x00010110` (que, en decimal, es 65808).
- Tres líneas de tabla, la línea de tabla de gama inferior y la línea de tabla de gama superior. Se codifican como la secuencia de bytes 0x49 0x23 0x81 0x80, o en binario, **01001001 00100011 10000001 10000000**. Esta cadena de bits se descompone a continuación en las líneas de tabla como sigue.

01 00100 Los dos primeros bits (HTPS) de esta línea de tabla indican una longitud de prefijo de 1, y los últimos cinco bits (HTRS) de esta línea de tabla indican una longitud de gama de 4.

10 01000 Esta línea de tabla tiene una longitud de prefijo de 2 y una longitud de gama de 8.

11 10000 Esta línea de tabla tiene una longitud de prefijo de 3 y una longitud de gama de 16.

00 La línea de tabla de gama inferior tiene una longitud de prefijo de 0, indicando que esta línea de tabla no se utiliza.

11 La línea de tabla de gama superior tiene una longitud de prefijo de 3.

0000000 Siete bits de relleno, para rellenar el último byte.

Después de codificar estas líneas de tabla, el valor de NTEMP es 5. Las matrices PREFLEN, RANGELEN y RANGELOW son:

PREFLEN	1	2	3	0	3
RANGELEN	4	8	16	32	32
RANGELOW	0	16	272	–1	65808

Aplicando el algoritmo de B.3 a lo anterior se obtiene la siguiente matriz de códigos, en binario,

CODES **0** **10** **110** X **111**

donde X indica que a la línea de tabla de gama inferior no se le ha asignado un código. Así pues, el código de prefijo **0** precede a un campo de 4 bits que codifica un valor de 0 a 15; el código de prefijo **10** precede a un campo de 8 bits que codifica un valor de 16 a 271, y así sucesivamente, como se muestra en el cuadro B.1.

B.5 Tablas Huffman normalizadas

En esta subcláusula se presentan algunas tablas Huffman normalizadas que pueden ser utilizadas en los contextos apropiados sin haber sido transmitidas previamente.

Las tablas Huffman se presentan en una forma similar a la de la transmisión de tablas descrita más arriba. Se da el parámetro de tabla HTOOB (HTPS, HTRS, HTLOW y HTHIGH se pueden obtener a partir de los valores de la tabla), seguido por una lista de líneas de tabla, indicándose la gama a la que aplica esa línea de tabla, la longitud de prefijo de la línea de tabla, la longitud de gama de la línea de tabla, y la codificación efectiva (prefijo y valor básico) de esa línea de tabla. Estas líneas de tabla van seguidas por una línea de tabla de gama inferior y superior y, facultativamente (dependiendo de HTOOB), por una línea de tabla fuera de banda. En algunos casos, las líneas de tabla de gama inferior o superior se omiten en las tablas mostradas, lo que indica que esas líneas de tabla no se utilizan en la tabla (y se les habría asignado un valor PREFLEN de cero).

Cuadro B.1 – Tabla Huffman normalizada A

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
0 . . . 15	1	4	**0** + VAL codificado como 4 bits
16 . . . 271	2	8	**10** + (VAL – 16) codificado como 8 bits
272 . . . 65807	3	16	**110** + (VAL – 272) codificado como 16 bits
65808 . . . ∞	3	32	**111** + (VAL – 65808) codificado como 32 bits

Cuadro B.2 – Tabla Huffman normalizada B

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codificación
0	1	0	**0**
1	2	0	**10**
2	3	0	**110**
3 . . . 10	4	3	**1110** + (VAL – 3) codificado como 3 bits
11 . . . 74	5	6	**11110** + (VAL – 11) codificado como 6 bits
75 . . . ∞	6	32	**111110** + (VAL – 75) codificado como 32 bits
OOB	6		**111111**

Cuadro B.3 – Tabla Huffman normalizada C

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codificación
–256 . . . –1	8	8	**11111110** + (VAL + 256) codificado como 8 bits
0	1	0	**0**
1	2	0	**10**
2	3	0	**110**
3 . . . 10	4	3	**1110** + (VAL – 3) codificado como 3 bits
11 . . . 74	5	6	**11110** + (VAL – 11) codificado como 6 bits
–∞ . . . –257	8	32	**11111111** + (–257 – VAL) codificado como 32 bits
75 . . . ∞	7	32	**1111110** + (VAL – 75) codificado como 32 bits
OOB	6		**111110**

Cuadro B.4 – Tabla Huffman normalizada D

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
1	1	0	**0**
2	2	0	**10**
3	3	0	**110**
4 . . . 11	4	3	**1110** + (VAL – 4) codificado como 3 bits
12 . . . 75	5	6	**11110** + (VAL – 12) codificado como 6 bits
76 . . . ∞	5	32	**11111** + (VAL – 76) codificado como 32 bits

Cuadro B.5 – Tabla Huffman normalizada E

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
–255 . . . 0	7	8	**1111110** + (VAL + 255) codificado como 8 bits
1	1	0	**0**
2	2	0	**10**
3	3	0	**110**
4 . . . 11	4	3	**1110** + (VAL – 4) codificado como 3 bits
12 . . . 75	5	6	**11110** + (VAL – 12) codificado como 6 bits
–∞ . . . –256	7	32	**1111111** + (–256 – VAL) codificado como 32 bits
76 . . . ∞	6	32	**111110** + (VAL – 76) codificado como 32 bits

Cuadro B.6 – Tabla Huffman normalizada F

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
–2048 . . . –1025	5	10	**11100** + (VAL + 2048) codificado como 10 bits
–1024 . . . –513	4	9	**1000** + (VAL + 1024) codificado como 9 bits
–512 . . . –257	4	8	**1001** + (VAL + 512) codificado como 8 bits
–256 . . . –129	4	7	**1010** + (VAL + 256) codificado como 7 bits
–128 . . . –65	5	6	**11101** + (VAL + 128) codificado como 6 bits
–64 . . . –33	5	5	**11110** + (VAL + 64) codificado como 5 bits
–32 . . . –1	4	5	**1011** + (VAL + 32) codificado como 5 bits
0 . . . 127	2	7	**00** + VAL codificado como 7 bits
128 . . . 255	3	7	**010** + (VAL – 128) codificado como 7 bits
256 . . . 511	3	8	**011** + (VAL – 256) codificado como 8 bits
512 . . . 1023	4	9	**1100** + (VAL – 512) codificado como 9 bits
1024 . . . 2047	4	10	**1101** + (VAL – 1024) codificado como 10 bits
–∞ . . . –2049	6	32	**111110** + (–2049 – VAL) codificado como 32 bits
2048 . . . ∞	6	32	**111111** + (VAL – 2048) codificado como 32 bits

Cuadro B.7 – Tabla Huffman normalizada G

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
–1024 . . . –513	4	9	**1000** + (VAL + 1024) codificado como 9 bits
–512 . . . –257	3	8	**000** + (VAL + 512) codificado como 8 bits
–256 . . . –129	4	7	**1001** + (VAL + 256) codificado como 7 bits
–128 . . . –65	5	6	**11010** + (VAL + 128) codificado como 6 bits
–64 . . . –32	5	5	**11011** + (VAL + 64) codificado como 5 bits
–32 . . . –1	4	5	**1010** + (VAL + 32) codificado como 5 bits
0 . . . 31	4	5	**1011** + VAL codificado como 5 bits
32 . . . 63	5	5	**11100** + (VAL – 32) codificado como 5 bits
64 . . . 127	5	6	**11101** + (VAL – 64) codificado como 6 bits
128 . . . 255	4	7	**1100** + (VAL – 128) codificado como 7 bits
256 . . . 511	3	8	**001** + (VAL – 256) codificado como 8 bits
512 . . . 1023	3	9	**010** + (VAL – 512) codificado como 9 bits
1024 . . . 2047	3	10	**011** + (VAL – 1024) codificado como 10 bits
–∞ . . . –1025	5	32	**11110** + (–1025 – VAL) codificado como 32 bits
2048 . . . ∞	5	32	**11111** + (VAL – 2048) codificado como 32 bits

Cuadro B.8 – Tabla Huffman normalizada H

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codificación
–15 . . . –8	8	3	**11111100** + (VAL + 15) codificado como 3 bits
–7 . . . –6	9	1	**111111100** + (VAL + 7) codificado como 1 bit
–5 . . . –4	8	1	**11111101** + (VAL + 5) codificado como 1 bit
–3	9	0	**111111101**
–2	7	0	**1111100**
–1	4	0	**1010**
0 . . . 1	2	1	**00** + VAL codificado como 1 bit
2	5	0	**11010**
3	6	0	**111010**
4 . . . 19	3	4	**100** + (VAL – 4) codificado como 4 bits
20 . . . 21	6	1	**111011** + (VAL – 20) codificado como 1 bit
22 . . . 37	4	4	**1011** + (VAL – 22) codificado como 4 bits
38 . . . 69	4	5	**1100** + (VAL – 38) codificado como 5 bits
70 . . . 133	5	6	**11011** + (VAL – 70) codificado como 6 bits
134 . . . 261	5	7	**11100** + (VAL – 134) codificado como 7 bits
262 . . . 389	6	7	**111100** + (VAL – 262) codificado como 7 bits
390 . . . 645	7	8	**1111101** + (VAL – 390) codificado como 8 bits
646 . . . 1669	6	10	**111101** + (VAL – 646) codificado como 10 bits
–∞ . . . –16	9	32	**111111110** + (–16 – VAL) codificado como 32 bits
1670 . . . ∞	9	32	**111111111** + (VAL – 1670) codificado como 32 bits
OOB	2		**01**

Cuadro B.9 – Tabla Huffman normalizada I

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codificación
–31 . . . –16	8	4	**11111100** + (VAL + 31) codificado como 4 bits
–15 . . . –12	9	2	**111111100** + (VAL + 15) codificado como 2 bits
–11 . . . –8	8	2	**11111101** + (VAL + 11) codificado como 2 bits
–7 . . . –6	9	1	**111111101** + (VAL + 7) codificado como 1 bit
–5 . . . –4	7	1	**1111100** + (VAL + 5) codificado como 1 bit
–3 . . . –2	4	1	**1010** + (VAL + 3) codificado como 1 bit
–1 . . . 0	3	1	**010** + (VAL + 1) codificado como 1 bit
1 . . . 2	3	1	**011** + (VAL – 1) codificado como 1 bit
3 . . . 4	5	1	**11010** + (VAL – 3) codificado como 1 bit
5 . . . 6	6	1	**111010** + (VAL – 5) codificado como 1 bit
7 . . . 38	3	5	**100** + (VAL – 7) codificado como 5 bits
39 . . . 42	6	2	**111011** + (VAL – 39) codificado como 2 bits
43 . . . 74	4	5	**1011** + (VAL – 43) codificado como 5 bits
75 . . . 138	4	6	**1100** + (VAL – 75) codificado como 6 bits
139 . . . 266	5	7	**11011** + (VAL – 139) codificado como 7 bits
267 . . . 522	5	8	**11100** + (VAL – 267) codificado como 8 bits
523 . . . 778	6	8	**111100** + (VAL – 523) codificado como 8 bits
779 . . . 1290	7	9	**1111101** + (VAL – 779) codificado como 9 bits
1291 . . . 3338	6	11	**111101** + (VAL – 1291) codificado como 11 bits
–∞ . . . –32	9	32	**111111110** + (–32 – VAL) codificado como 32 bits
3339 . . . ∞	9	32	**111111111** + (VAL – 3339) codificado como 32 bits
OOB	2		**00**

Cuadro B.10 – Tabla Huffman normalizada J

HTOOB	**1**

VAL	PREFLEN	RANGELEN	Codificación
–21 . . . –6	7	4	**1111010** + (VAL + 21) codificado como 4 bits
–5	8	0	**11111100**
–4	7	0	**1111011**
–3	5	0	**11000**
–2 . . . 1	2	2	**00** + (VAL + 2) codificado como 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) codificado como 6 bits
70 . . . 101	5	5	**11010** + (VAL – 70) codificado como 5 bits
102 . . . 133	6	5	**110111** + (VAL – 102) codificado como 5 bits
134 . . . 197	6	6	**111000** + (VAL – 134) codificado como 6 bits
198 . . . 325	6	7	**111001** + (VAL – 198) codificado como 7 bits
326 . . . 581	6	8	**111010** + (VAL – 326) codificado como 8 bits
582 . . . 1093	6	9	**111011** + (VAL – 582) codificado como 9 bits
1094 . . . 2117	6	10	**111100** + (VAL – 1094) codificado como 10 bits
2118 . . . 4165	7	11	**1111101** + (VAL – 2118) codificado como 11 bits
–∞ . . . –22	8	32	**11111110** + (–22 – VAL) codificado como 32 bits
4166 . . . ∞	8	32	**11111111** + (VAL – 4166) codificado como 32 bits
OOB	2		**10**

Cuadro B.11 – Tabla Huffman normalizada K

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
1	1	0	**0**
2 . . . 3	2	1	**10** + (VAL – 2) codificado como 1 bit
4	4	0	**1100**
5 . . . 6	4	1	**1101** + (VAL – 5) codificado como 1 bit
7 . . . 8	5	1	**11100** + (VAL – 7) codificado como 1 bit
9 . . . 12	5	2	**11101** + (VAL – 9) codificado como 2 bits
13 . . . 16	6	2	**111100** + (VAL – 13) codificado como 2 bits
17 . . . 20	7	2	**1111010** + (VAL – 17) codificado como 2 bits
21 . . . 28	7	3	**1111011** + (VAL – 21) codificado como 3 bits
29 . . . 44	7	4	**1111100** + (VAL – 29) codificado como 4 bits
45 . . . 76	7	5	**1111101** + (VAL – 45) codificado como 5 bits
77 . . . 140	7	6	**1111110** + (VAL – 77) codificado como 6 bits
141 . . . ∞	7	32	**1111111** + (VAL – 141) codificado como 32 bits

Cuadro B.12 – Tabla Huffman normalizada L

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
1	1	0	**0**
2	2	0	**10**
3 . . . 4	3	1	**110** + (VAL – 3) codificado como 1 bit
5	5	0	**11100**
6 . . . 7	5	1	**11101** + (VAL – 6) codificado como 1 bit
8 . . . 9	6	1	**111100** + (VAL – 8) codificado como 1 bit
10	7	0	**1111010**
11 . . . 12	7	1	**1111011** + (VAL – 11) codificado como 1 bit
13 . . . 16	7	2	**1111100** + (VAL – 13) codificado como 2 bits
17 . . . 24	7	3	**1111101** + (VAL – 17) codificado como 3 bits
25 . . . 40	7	4	**1111110** + (VAL – 25) codificado como 4 bits
41 . . . 72	8	5	**11111110** + (VAL – 41) codificado como 5 bits
73 . . . ∞	8	32	**11111111** + (VAL – 73) codificado como 32 bits

Cuadro B.13 – Tabla Huffman normalizada M

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
1	1	0	**0**
2	3	0	**100**
3	4	0	**1100**
4	5	0	**11100**
5 . . . 6	4	1	**1101** + (VAL – 5) codificado como 1 bit
7 . . . 14	3	3	**101** + (VAL – 7) codificado como 3 bits
15 . . . 16	6	1	**111010** + (VAL – 15) codificado como 1 bit
17 . . . 20	6	2	**111011** + (VAL – 17) codificado como 2 bits
21 . . . 28	6	3	**111100** + (VAL – 21) codificado como 3 bits
29 . . . 44	6	4	**111101** + (VAL – 29) codificado como 4 bits
45 . . . 76	6	5	**111110** + (VAL – 45) codificado como 5 bits
77 . . . 140	7	6	**1111110** + (VAL – 77) codificado como 6 bits
141 . . . ∞	7	32	**1111111** + (VAL – 141) codificado como 32 bits

Cuadro B.14 – Tabla Huffman normalizada N

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
–2	3	0	**100**
–1	3	0	**101**
0	1	0	**0**
1	3	0	**110**
2	3	0	**111**

Cuadro B.15 – Tabla Huffman normalizada O

HTOOB	**0**

VAL	PREFLEN	RANGELEN	Codificación
–24 . . . –9	7	4	**1111100** + (VAL + 24) codificado como 4 bits
–8 . . . –5	6	2	**111100** + (VAL + 8) codificado como 2 bits
–4 . . . –3	5	1	**11100** + (VAL + 4) codificado como 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) codificado como 1 bit
5 . . . 8	6	2	**111101** + (VAL – 5) codificado como 2 bits
9 . . . 24	7	4	**1111101** + (VAL – 9) codificado como 4 bits
–∞ . . . –25	7	32	**1111110** + (–25 – VAL) codificado como 32 bits
25 . . . ∞	7	32	**1111111** + (VAL – 25) codificado como 32 bits

Anexo C

Procedimiento de decodificación de imagen de escala de grises

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

C.1 Descripción general

Este procedimiento de decodificación es utilizado por el procedimiento de decodificación de región de semitonos para producir una formación de valores de escala de grises, que a continuación se utilizan como índices en un diccionario de patrones.

C.2 Parámetros de entrada

En el cuadro C.1 se muestran los parámetros de este procedimiento de decodificación.

Cuadro C.1 – Parámetros del procedimiento de decodificación de imagen de escala de grises

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
GSMMR	Entero	1	No	Especifica si se utiliza MMR
GSUSESKIP	Entero	1	No	Especifica si se puede producir un salto de valores de escala de grises
GSBPP	Entero	6	No	El número de bits por valor de escala de grises
GSW	Entero	32	No	La anchura de la imagen de escala de grises
GSH	Entero	32	No	La altura de la imagen de escala de grises
GSTEMPLATE	Entero	2	No	La plantilla utilizada para codificar los planos de bits de la escala de grises[b)]
GSKIP	Mapa de bits			Una máscara que indica qué valores deberán saltarse. **GSW** píxels de ancho, **GSH** píxels de alto[a)]

a) No se utiliza si **GSUSESKIP** = **0**.
b) No se utiliza si **GSMMR** = **1**.

C.3 Valor de retorno

En el cuadro C.2 se muestra la variable cuyo valor es el resultado de este procedimiento de decodificación.

Cuadro C.2 – Valor de retorno del procedimiento de decodificación de imagen de escala de grises

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
GSVALS	Formación			La imagen de escala de grises es decodificada. La formación es **GSW** de ancha y **GSH** de alta

C.4 Variables utilizadas en la decodificación

En el cuadro C.3 se muestran las variables utilizadas por este procedimiento de decodificación.

Cuadro C.3 – Variables utilizadas en el procedimiento de decodificación de imagen de escala de grises

Nombre	Tipo	Tamaño (bits)	¿Con signo?	Descripción y restricciones
GSPLANES	Formación de mapas de bits			Planos de bits de la imagen de escala de grises. Hay **GSBPP** planos de bits en GSPLANES. Cada plano de bits es **GSW** de ancho y **GSH** de alto
J	Entero	32	Sí	Contador de planos de bits

C.5 Decodificación de la imagen de escala de grises

La imagen de escala de grises se obtiene decodificando **GSBPP** planos de bits. Estos planos de bits son designados (del menos significativo al más significativo) por GSPLANES[0], GSPLANES[1], ..., GSPLANES[**GSBPP** – 1]. Los planos de bits tienen codificación Gray, por lo que el valor verdadero de cada plano de bits es igual a su valor codificado al que se aplica el operador XOR con el siguiente plano de bits más significativo.

La imagen de escala de grises se obtiene por el procedimiento siguiente:

1) Decodificar GSPLANES[**GSBPP** – 1], utilizando el procedimiento de decodificación de región genérica. En el cuadro C.4 se muestran los parámetros del procedimiento de decodificación de región genérica.

Cuadro C.4 – Parámetros utilizados para decodificar un plano de bits de la imagen de escala de grises

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) Fijar J = **GSBPP** – 2.

3) Mientras $J \geq 0$, ejecutar los pasos siguientes:

   a) Decodificar GSPLANES[J] utilizando el procedimiento de decodificación de región genérica. En el cuadro C.4 se muestran los parámetros del procedimiento de decodificación de región genérica.

   b) Para cada píxel (x, y) en GSPLANES[J], fijar:

$$\text{GSPLANES}[J][x, y] = \text{GSPLANES}[J + 1][x, y] \text{ XOR } \text{GSPLANES}[J][x, y]$$

   c) Fijar $J = J - 1$.

4) Para cada (x, y), fijar:

$$\text{GSVALS}[x, y] = \sum_{J=0}^{\mathbf{GSBPP}-1} \text{GSPLANES}[J][x, y] \times 2^J$$

Anexo D

Formatos de ficheros

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

Hay dos posibles organizaciones de ficheros autónomos para un tren binario JBIG2. Hay también una tercera organización, de utilización no autónoma, pero que permite en cambio insertar en otro formato de fichero datos codificados según la norma JBIG2.

NOTA – Se recomienda utilizar "`.jbig2`" como extensión de los ficheros JBIG2. En entornos en los que sólo se permiten tres caracteres, se recomienda "`.jb2`". También se recomienda que los decodificadores JBIG2 reconozcan ambas extensiones.

D.1 Organización secuencial

Se trata de una organización de ficheros autónoma. Esta organización está orientada a las aplicaciones inmediatas, en las que se garantiza que el decodificador empieza a actuar al comienzo del tren binario y decodifica todo hasta el final del mismo.

En esta organización, la estructura de un fichero se asemeja a la de la figura D.1. Un encabezamiento de fichero va seguido por una secuencia de segmentos. Las dos partes de cada segmento se almacenan juntas: primero el encabezamiento de segmento y a continuación los datos de segmento.

Los segmentos deben aparecer en orden creciente de sus números de segmento: ningún segmento puede preceder a un segmento que tenga un número inferior al suyo.

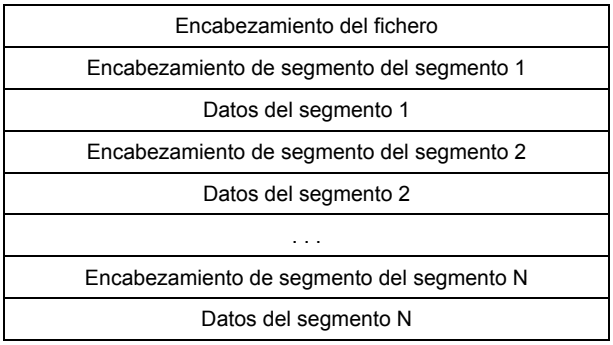


Figura D.1 – Organización secuencial

D.2 Organización de acceso aleatorio

Se trata de una organización de ficheros autónoma. Esta organización está orientada a las aplicaciones de acceso aleatorio, en las que al decodificador quizás le convenga procesar partes del fichero en un orden arbitrario, por ejemplo, decodificando todas las páginas de número impar antes que cualquier página de número par, o decodificar páginas individualmente en respuesta a la indicación de algún usuario. La posibilidad de acceder de forma aleatoria es importante, por consiguiente.

En esta organización, la estructura de un fichero se asemeja a la de la figura D.2. Un encabezamiento de fichero va seguido por una secuencia de encabezamientos de segmento; el encabezamiento del último segmento va seguido por los datos del primer segmento, a continuación por los datos del segundo segmento, y así sucesivamente. El último segmento debe ser un segmento fin de fichero; de no ser así, es imposible que el decodificador determine cuándo ha leído el encabezamiento del último segmento.

Los segmentos deben aparecer en orden creciente de sus números de segmento: ningún segmento puede preceder a un segmento que tenga un número inferior al suyo.

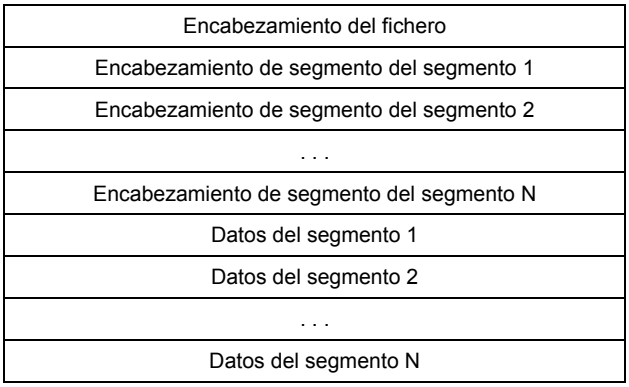


Figura D.2 – Organización de acceso aleatorio

D.3 Organización insertada

No se trata de una organización de ficheros autónoma, sino que depende del formato de algún otro fichero para llevar los segmentos JBIG2. Cada segmento se almacena concatenando sus partes encabezamiento de segmento y datos de segmento, pero no hay un orden de almacenamiento definido para estos segmentos. El formato del fichero en el que se produce la inserción puede almacenar esos segmentos en cualquier orden, y puede separarlos según cualesquiera datos.

A las aplicaciones quizás les convenga que los datos JBIG2 vayan precedidos y seguidos por una combinación de dos bytes (marcador) única manera de que dichos datos puedan ser detectados dentro de otros trenes de datos. Se sugiere utilizar `0xFF 0xAA` para el marcador de principio y `0xFF 0xAB` para el marcador de fin. Estos marcadores no se consideran parte de los datos JBIG2. Se señala que es poco probable que el primer byte de un encabezamiento de segmento tome el valor `0xFF`. Hay que tener en cuenta que las secuencias de dos bytes `0xFF 0xAA` y `0xFF 0xAB` podrían producirse por casualidad dentro de segmentos JBIG2.

NOTA – El objetivo de la organización insertada es que muchos de los sistemas actuales se puedan beneficiar de la incorporación de la compresión de imágenes binivel mejorada. Sin embargo, la manera más conveniente de hacerlo no siempre consiste en incorporar un tren binario JBIG2 entero como una entidad monolítica, ya que esto puede estar en contraposición con otras constricciones. Por ejemplo, el sistema podría tener sus propias ideas sobre cómo se han de dividir las páginas, que quizás no concuerden con las ideas de la norma JBIG2. La JBIG2 es flexible, por tanto, permitiendo que, el sistema en el que se produce la inserción almacene datos JBIG2 de la manera que mejor le convenga.

D.4 Sintaxis de encabezamiento de fichero

Un encabezamiento de fichero contiene los siguientes campos, en orden:

Cadena ID – Véase D.4.1.

Banderas de encabezamiento de fichero – Véase D.4.2.

Número de páginas – Véase D.4.3.

D.4.1 Cadena ID

Es una secuencia de 8 bytes que contiene `0x97 0x4A 0x42 0x32 0x0D 0x0A 0x1A 0x0A`.

NOTA – Esto es algo similar a la cadena ID PNG. El primer carácter no es imprimible, por lo que el fichero no puede ser confundido con ASCII. El bit más significativo del primer carácter se fija para detectar el paso a través de un canal de 7 bits. Los tres bytes siguientes son JB2 y tienen por objeto permitir a una persona que observa el encabezamiento deducir cuál es el tipo de fichero. Los bytes siguientes son `CR LF CONTROL-Z LF`; cualquier corrupción debida a la traducción CR/LF y al truncamiento de fichero DOS puede ser detectada inmediatamente.

D.4.2 Banderas de encabezamiento de fichero

Es un campo de 1 byte. Los bits definidos son:

Bit 0 Tipo de organización de fichero. Si este bit es **0**, el fichero utiliza la organización de acceso aleatorio. Si este bit es **1**, el fichero utiliza la organización secuencial.

NOTA – Se señala que no hay manera de indicar la organización insertada, ya que esa organización no incluye un encabezamiento de fichero JBIG2.

Bit 1 Número de páginas desconocido. Si este bit es **0**, el número de páginas contenidas en el fichero es conocido. Si este bit es **1**, el número de páginas contenidas en el fichero no era conocido en el momento en que se codificó el encabezamiento de fichero.

Bits 2-7 Reservado; debe ser **0**.

D.4.3 Número de páginas

Es un campo de 4 bytes, y no está presente si el bit "número de páginas desconocido" es **1**. Si está presente, debe ser igual al número de páginas contenidas en el fichero.

Anexo E

Codificación aritmética

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

Debe utilizarse un codificador aritmético binario adaptativo, como el codificador de entropía, cuando así lo permiten los modelos. Los modelos utilizados con codificación aritmética binaria adaptativa se definen en 6.2, 6.3 y el anexo A. En este anexo se definen los procedimientos de codificación aritmética básicos.

Los diagramas de flujo y los cuadros del presente anexo y de todas sus subcláusulas son normativos en el sentido de que definen una salida que deberán reproducir las implementaciones alternativas. En H.2 se da un ejemplo sencillo de prueba que servirá para determinar si una determinada implementación es correcta.

E.1 Codificación binaria

La figura E.1 muestra un diagrama de bloques sencillo del codificador aritmético adaptativo binario. Los pares decisión (D) y contexto (CX) se procesan juntos para producir una salida de datos comprimidos (CD). Tanto D como CX son proporcionados por la unidad modelo (no mostrada). CX selecciona la estimación de probabilidad que se ha de utilizar durante la codificación de D. En esta Recomendación | Norma Internacional, CX es la etiqueta de un contexto, formada por una cadena de algunos caracteres seguida por una cadena de bits.

EJEMPLO – Dos valores posibles de CX son "IADW**001010100**" y "GB**1110110010000000**".

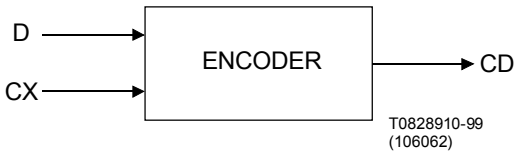


Figura E.1 – Entradas y salidas del codificador aritmético

E.1.1 Subdivisión de intervalo recursiva

La subdivisión de intervalo de probabilidad recursiva de la codificación Elias es la base del proceso de codificación aritmética binaria. Con cada decisión binaria, el intervalo de probabilidad vigente es subdividido en dos subintervalos, y la cadena de códigos es modificada (si es necesario) de manera que apunte hacia la base (el límite inferior) del subintervalo de probabilidad asignado al símbolo que se ha producido.

Al dividir el intervalo vigente en dos subintervalos, el subintervalo del símbolo más probable (MPS, *more probable symbol*) se sitúa por encima del subintervalo del símbolo menos probable (LPS, *less probable symbol*). Por tanto, cuando se codifica el MPS, se añade a la cadena de códigos el subintervalo del LPS. Este convenio de codificación exige que los símbolos se reconozcan como MPS o LPS, en vez de como 0 ó 1. Para codificar cada decisión, deben conocerse, por consiguiente, el tamaño del intervalo del LPS y el sentido del MPS correspondientes a la misma.

Puesto que la cadena de códigos siempre apunta a la base del intervalo vigente, el proceso de decodificación consiste en determinar, para cada decisión, a qué subintervalo apunta la cadena de códigos. Esto se hace también de manera recursiva, utilizando el mismo proceso de subdivisión de intervalo que en el codificador. Cada vez que se decodifica una decisión, el decodificador sustrae cualquier intervalo que el codificador hubiera añadido a la cadena de códigos. La cadena de códigos en el decodificador es, por tanto, un puntero que apunta hacia el intervalo vigente asociado a la base de dicho intervalo. Dado que el proceso de codificación exige la adición de fracciones binarias más que la concatenación de palabras de código enteras, las decisiones binarias más probables pueden codificarse a menudo a un costo mucho menor que el de un bit por decisión.

E.1.2 Convenios de codificación y aproximaciones

Las operaciones de codificación se realizan utilizando aritmética de enteros de precisión fija y una representación entera de valores fraccionarios en la que `0x8000` es equivalente al decimal 0,75. El intervalo A se mantiene en la gama $0{,}75 \leq A < 1{,}5$ duplicándolo cada vez que el valor entero cae por debajo de `0x8000`.

El registro de códigos C se duplica también cada vez que se duplica A. Periódicamente, para evitar el desbordamiento de C, se suprime un byte de datos de los bits de orden alto del registro C y se sitúa en una memoria intermedia de datos comprimidos externa. La transferencia a la memoria intermedia externa se resuelve mediante un procedimiento de relleno de bits.

El mantenimiento de A en la gama de $0{,}75 \leq A < 1{,}5$ permite utilizar una aproximación aritmética sencilla en la subdivisión del intervalo. Si el intervalo es A y la estimación actual de la probabilidad del LPS es Qe, un cálculo preciso de los subintervalos requeriría:

$$A - (Qe \times A) = \text{subintervalo para el MPS}$$
$$Qe \times A = \text{subintervalo para el LPS}$$

Puesto que el valor de A es del orden de una unidad, se aproximan por:

$$A - Qe = \text{subintervalo para el MPS}$$
$$Qe = \text{subintervalo para el LPS}$$

Cuando se codifica el MPS, el valor de Qe se añade al registro de códigos y el intervalo se reduce a A – Qe. Cuando se codifica el LPS, el registro de códigos se deja inalterado y el intervalo se reduce a Qe. A continuación se restablece la gama de precisión requerida para A, si hace falta, renormalizando tanto A como C.

Con el proceso arriba ilustrado, la aproximación en el proceso de subdivisión de intervalos puede hacer a veces que el subintervalo del LPS sea mayor que el subintervalo del MPS. Si, por ejemplo, el valor de Qe es 0,5 y A está en el valor mínimo permitido de 0,75, el reparto aproximado da 1/3 del intervalo al MPS y 2/3 al LPS. Para evitar esta inversión de tamaños, se intercambian los intervalos del MPS y el LPS siempre que el del LPS sea mayor que el del MPS. Este intercambio condicional de MPS/LPS sólo puede producirse cuando se necesite una renormalización.

Cuando se produce una renormalización, se invoca el proceso de estimación de probabilidad que determina una nueva estimación de la probabilidad del contexto que se está codificando en ese momento. Para la estimación no hacen falta cuentas de símbolos explícitas. Las probabilidades relativas de renormalización después de codificar un LPS y un MPS proporcionan un mecanismo de cómputo de símbolos aproximado que se utiliza para estimar directamente las probabilidades.

E.2 Descripción del codificador aritmético

El ENCODER (figura E.2) inicializa el codificador por medio del procedimiento INITENC. Se leen los pares CX y D y se pasan a ENCODE hasta que todos ellos hayan sido leídos. Los procedimientos de estimación de probabilidad que proporcionan estimaciones adaptativas de la probabilidad de cada contexto están incorporados en ENCODE. Se producen como salida bytes de datos comprimidos cuando ya no son modificables. Cuando todos los pares CX y D han sido leídos (¿Terminado?), FLUSH fija el contenido del registro C en tantos bits 1 como sea posible y a continuación genera los bytes finales de salida. FLUSH termina también las operaciones de codificación y emite el marcador de terminación requerido.

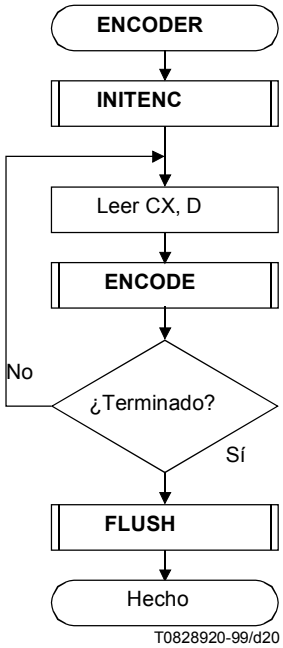


Figura E.2 – Codificación del codificador MQ

E.2.1 Convenios de registro de códigos del codificador

En los diagramas de flujo indicados en este anexo se suponen las siguientes estructuras de registro del codificador:

	MSB			LSB
Registro C	0000cbbb	bbbbbsss	xxxxxxxx	xxxxxxxx
Registro A	00000000	00000000	aaaaaaaa	aaaaaaaa

Los bits "a" son los bits fraccionarios en el registro A (el valor del intervalo vigente) y los bits "x" son los bits fraccionarios en el registro de códigos. Los bits "s" son bits espaciadores que sirven para forzar el arrastre, y los bits "b" indican las posiciones de bits a partir de las cuales los bytes completados de los datos son suprimidos del registro C. El bit "c" es un bit de arrastre.

La descripción detallada del relleno de bits y la forma de efectuar el arrastre se indica más adelante en el presente anexo.

E.2.2 Codificación de una decisión (ENCODE)

El procedimiento ENCODE determina si la decisión D es o no 0. A continuación se llama a un procedimiento CODE0 o CODE1, según convenga. A menudo, las implementaciones no tendrán un procedimiento ENCODE, sino que llamarán a los procedimientos CODE0 o CODE1 directamente para codificar una decisión 0 o una decisión 1.

E.2.3 Codificación de un 1 ó 0 (CODE1 y CODE0)

Cuando se codifica una determinada decisión binaria, se produce una de las dos situaciones siguientes, a saber, que el símbolo sea el más probable o que sea el menos probable. En las figuras E.4 y E.5 se ilustran los procedimientos CODE1 y CODE0. En dichas figuras, CX es el contexto. Para cada contexto se almacena el índice de la estimación de probabilidad que se ha de utilizar en las operaciones de codificación y el valor MPS. MPS(CX) es el sentido (**0** ó **1**) del MPS para el contexto CX.

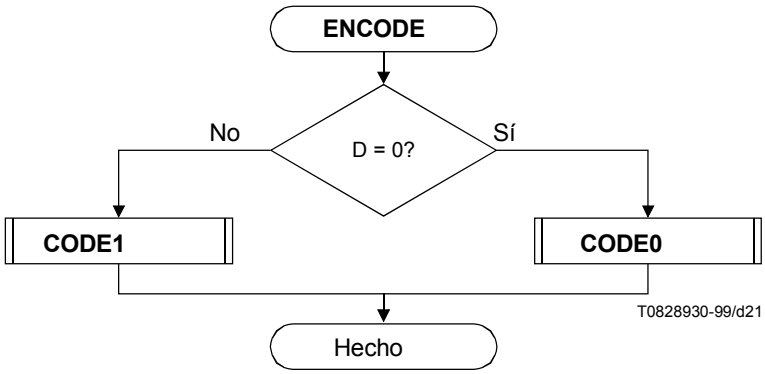


Figura E.3 – Procedimiento ENCODE

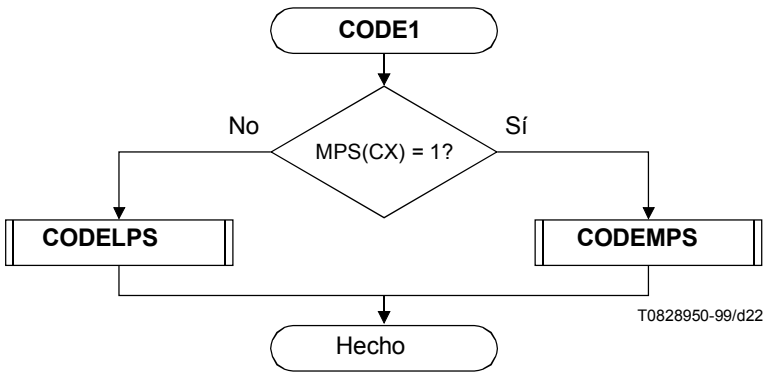


Figura E.4 – Procedimiento CODE1

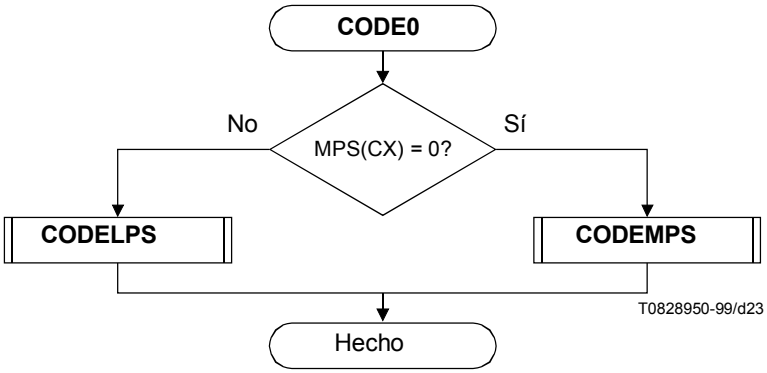


Figure E.5 – Procedimiento CODEO

E.2.4 Codificación de un MPS o LPS (CODEMPS y CODELPS)

El procedimiento CODELPS (figura E.6) consiste normalmente en la aplicación de un factor de escala al intervalo hasta Qe(I(CX)), la estimación de la probabilidad del LPS determinada a partir del índice I almacenado para el contexto CX. Primero se calcula el intervalo superior a fin de compararlo con el intervalo inferior para confirmar que Qe tiene el tamaño menor. Esto va seguido siempre de una renormalización (RENORME). En el caso de inversión de los tamaños de intervalo, no obstante, se produce el intercambio MPS/LPS condicional y se codifica el intervalo superior. De cualquier modo, se actualiza la estimación de la probabilidad. Si se ha fijado la bandera SWITCH para el índice I(CX), se invierte el MPS(CX). Se guarda un nuevo índice I en CX según se determine de acuerdo con la columna de índice de LPS siguiente (NLPS) del cuadro E.1.

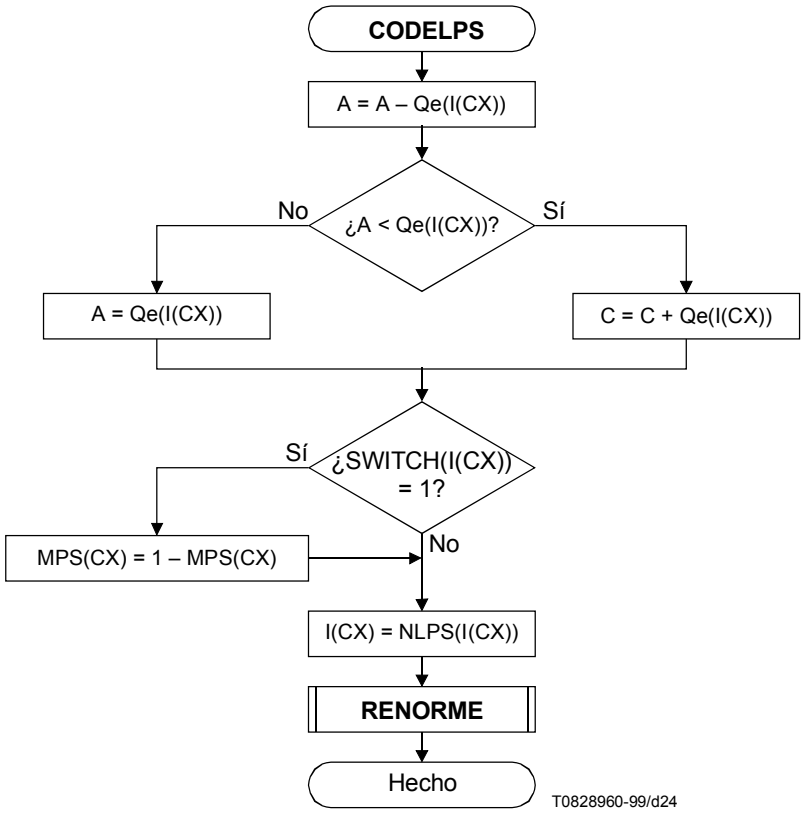


Figura E.6 – Procedimiento CODELPS con intercambio MPS/LPS condicional

El procedimiento CODEMPS (figura E.7) reduce normalmente el tamaño del intervalo al subintervalo del MPS y ajusta el registro de códigos de manera que apunte a la base del subintervalo del MPS. Sin embargo, si se invierten los tamaños de los intervalos, se codifica el subintervalo del LPS. Se señala que la inversión de tamaños no puede ocurrir a menos que se requiera una renormalización (RENORME) después de la codificación del símbolo. La actualización de la estimación de la probabilidad cambia el índice I(CX) de acuerdo con la columna de índice del MPS siguiente (NMPS) del cuadro E.1.

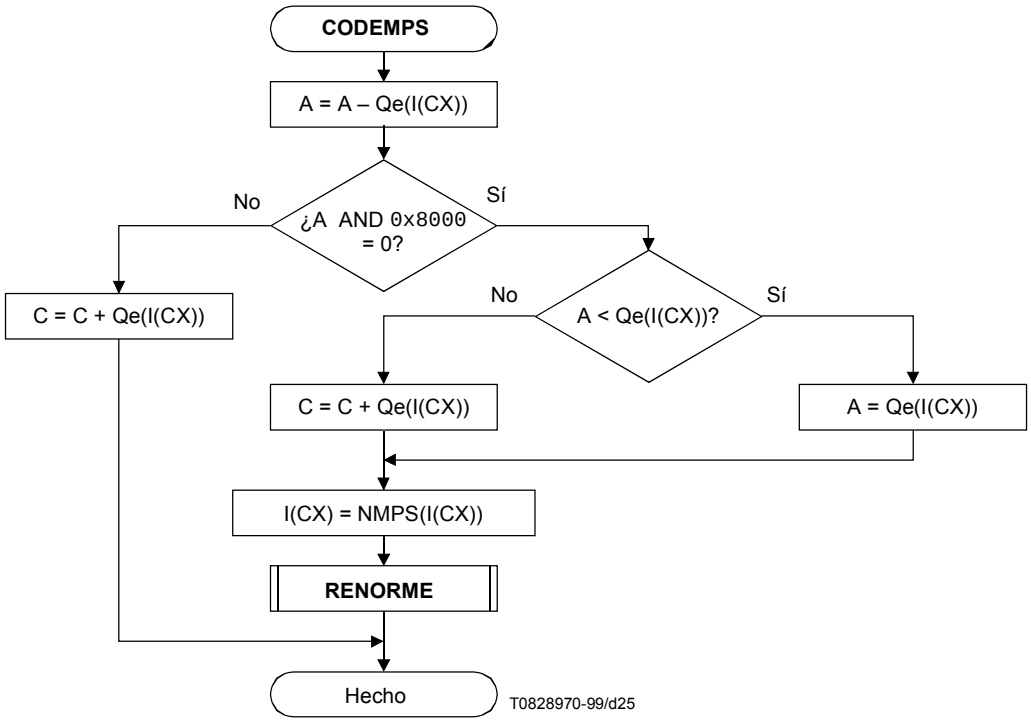


Figura E.7 – Procedimiento CODEMPS con intercambio MPS/LPS condicional

E.2.5 Estimación de la probabilidad

El cuadro E.1 muestra el valor Qe asociado con cada índice Qe. Los valores Qe se expresan como enteros hexadecimales, como enteros binarios y como fracciones decimales. Para convertir la representación entera de Qe de 15 bits en la probabilidad decimal, los valores Qe se dividen por $(4/3) \times (\texttt{0x8000})$.

El estimador se puede definir como una máquina de estados finitos –una tabla de índices Qe y estados siguientes asociados para cada tipo de renormalización (es decir, nuevas posiciones de la tabla)– como se muestra en el cuadro E.1. El cambio de estado se produce solamente cuando se renormaliza el registro de intervalos del codificador aritmético. Esto se hace siempre después de codificar el LPS, y cuando el registro de intervalos es inferior a `0x8000` (0,75 en notación decimal) después de codificar el MPS.

Después de una renormalización de LPS, NLPS da el nuevo índice para la estimación de la probabilidad de LPS, además, si Switch es **1**, se invierte el sentido del símbolo MPS. Después de una renormalización de MPS, NMPS da el nuevo índice para la estimación de la probabilidad de LPS.

El índice de la estimación en curso forma parte de la información almacenada para el contexto CX. Este índice se utiliza como índice de la tabla de valores en MPS, que da el índice siguiente para una renormalización de NMPS. Este índice se guarda en el almacenamiento de contexto en CX. MPS(CX) no cambia.

El procedimiento de estimación de la probabilidad en el trayecto de renormalización de LPS es similar al de una renormalización de MPS, con la salvedad de que cuando Switch(I(CX)) es 1, se invierte el sentido de MPS(CX).

El estado de índice final 46 se puede utilizar para establecer una estimación de probabilidad de 0,5 fija.

Cuadro E.1 – Valores de Qe y proceso de estimación de la probabilidad

Índice	Valor de Qe			NMPS	NLPS	SWITCH
	(hexadecimal)	(binario)	(decimal)			
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 Renormalización en el codificador (RENORME)

La renormalización en el codificador es muy similar a la renormalización en el decodificador. Difieren en que en el codificador genera bits comprimidos y en el decodificador consume bits comprimidos.

En la figura E.8 se ilustra el procedimiento RENORME para la renormalización del codificador. Tanto el registro de intervalos A como el registro de códigos C se desplazan, un bit cada vez. Se cuenta el número de desplazamientos en el contador CT, y cuando la cuenta de CT se reduce a cero, se suprime de C un byte de datos comprimidos por el procedimiento BYTEOUT. La renormalización continúa hasta que A deja de ser menor que `0x8000`.

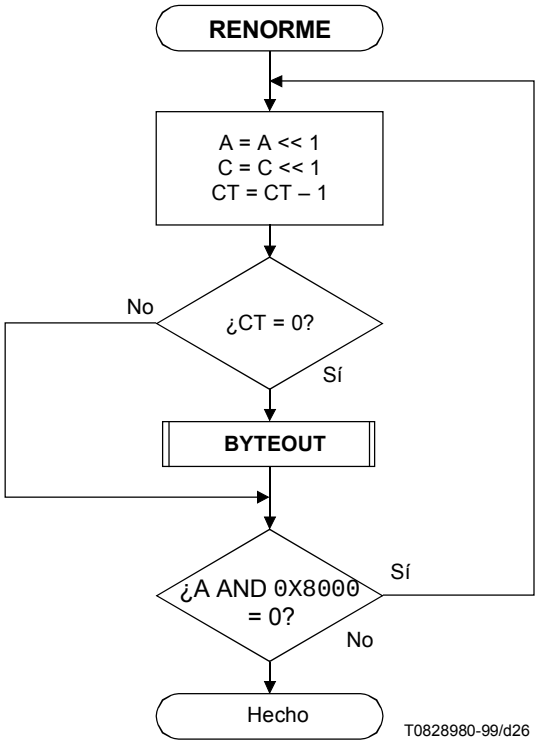


Figura E.8 – Procedimiento de renormalización de codificador

E.2.7 Salida de datos comprimidos (BYTEOUT)

En la figura E.9 se ilustra la rutina BYTEOUT llamada desde RENORME. Esta rutina contiene los procedimientos de relleno de bits necesarios para limitar la propagación del arrastre en los bytes completados de datos comprimidos. Los convenios utilizados hacen imposible que un arrastre se propague a través de más bytes que el escrito más recientemente en la memoria intermedia de datos comprimidos.

El procedimiento del bloque de la sección inferior derecha efectúa el relleno de bits después de un byte `0xFF`; el procedimiento similar de la izquierda queda para el caso en que no se necesita relleno de bits.

B es el byte al que apunta el puntero BP de la memoria intermedia de datos comprimidos. Si B no es un byte `0xFF`, se comprueba el bit de arrastre. Si se ha fijado el bit de arrastre, se añade a B y se comprueba de nuevo B para ver si es preciso un bit de relleno en el byte siguiente. Una vez determinada la necesidad de un bit de relleno, se elige el trayecto adecuado, se incrementa BP y el nuevo valor de B se elimina de los bits "b" del registro de códigos.

E.2.8 Inicialización del codificador (INITENC)

El procedimiento INITENC se utiliza para el arranque del codificador aritmético. En la figura E.10 se muestran sus pasos básicos.

El registro de intervalos y el registro de códigos se fijan a sus valores iniciales, y se fija el contador de bits. La fijación CT = 12 refleja el hecho de que hay tres bits espaciadores en el registro que es preciso rellenar antes de que se alcance el campo del que se eliminan los bytes. Se señala que BP apunta siempre al byte que precede a la posición BPST en la que se sitúa el primer byte. Por consiguiente, si el byte precedente es un byte `0xFF`, se producirá un relleno de bits espúreos, pero se puede compensar incrementando CT. Obsérvese que la inicialización por defecto de los depósitos estadísticos es MPS = 0 e I = 0 (es decir, un Qe = 0x5601 ó 0,503937 en notación decimal).

E.2.9 Terminación de la codificación (FLUSH)

El procedimiento FLUSH mostrado en la figura E.11 se utiliza para terminar las operaciones de codificación y generar el marcador de terminación requerido. El procedimiento garantiza que el prefijo `0xFF` del código marcador se superpone a los bits finales de los datos comprimidos. Así se asegura que cualquier código marcador al final de los datos comprimidos será reconocido e interpretado antes de que se complete la decodificación.

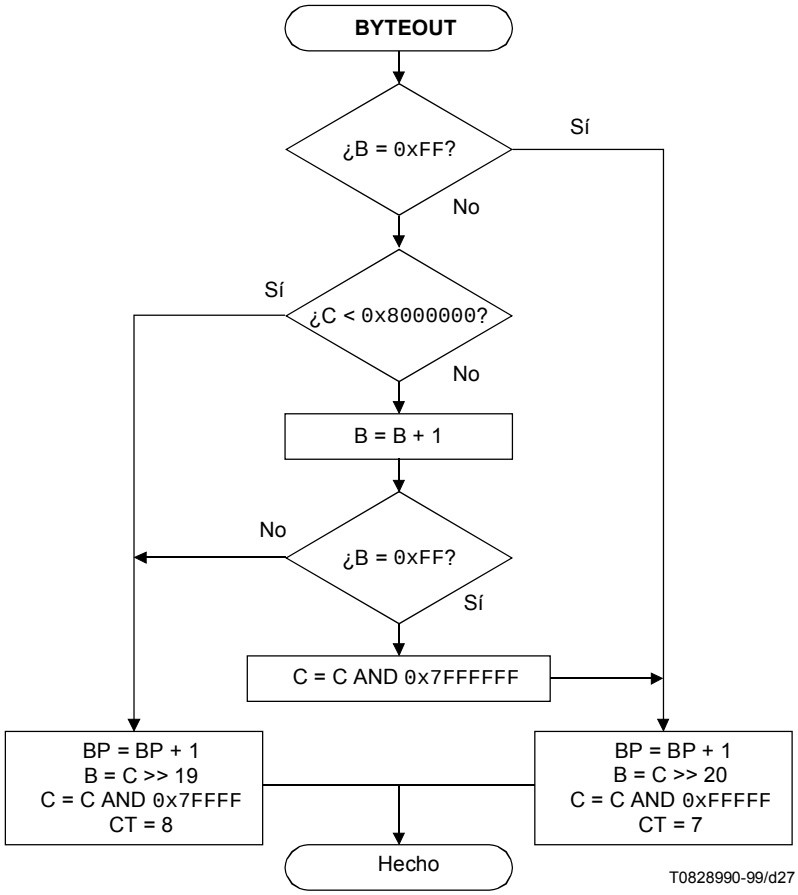


Figura E.9 – Procedimiento BYTEOUT para codificar

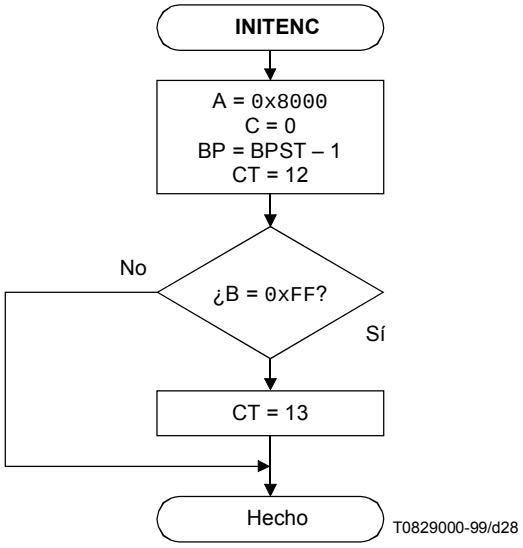


Figura E.10 – Inicialización del codificador

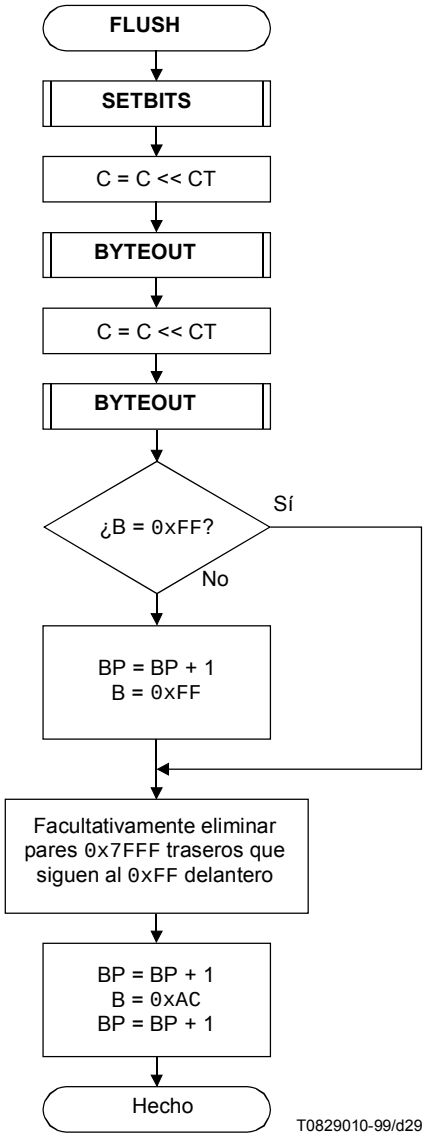


Figura E.11 – Procedimiento FLUSH

La primera parte del procedimiento FLUSH fija en 1 tantos bits del registro C como sea posible, según se muestra en la figura E.12. El límite superior exclusivo para el registro C es la suma del registro C y el registro de intervalos. Se fuerzan a 1 los 16 bits de orden bajo de C, y el resultado se compara con el límite superior. Si C es demasiado grande, se elimina el bit 1 delantero, reduciendo C a un valor que se encuentre dentro del intervalo.

A continuación se completa el byte del registro C desplazando C, y se eliminan seguidamente dos bytes. Si el segundo byte no es `0xFF`, se añade a los datos comprimidos otro byte con la garantía de que es `0xFF`.

E.2.10 Minimación de los datos comprimidos

Si se desea, se pueden truncar los datos comprimidos una vez que se complete el procedimiento FLUSH. Si el codificador aritmético genera una secuencia de bits 1, el relleno de bits producirá pares de bytes `0xFF, 0x7F`. Estos pares de bytes pueden ser recortados desde los datos comprimidos, siempre que el primer byte `0xFF` de la secuencia no se elimine. Este byte `0xFF` restante pasa a ser entonces el prefijo del código marcador que termina los datos comprimidos.

La decodificación no se ve afectada por este proceso de recorte porque en el decodificador se aplica el convenio de que cuando se encuentra un código marcador, se suministran bits 1 (sin relleno de bits) al decodificador hasta completar el intervalo de codificación.

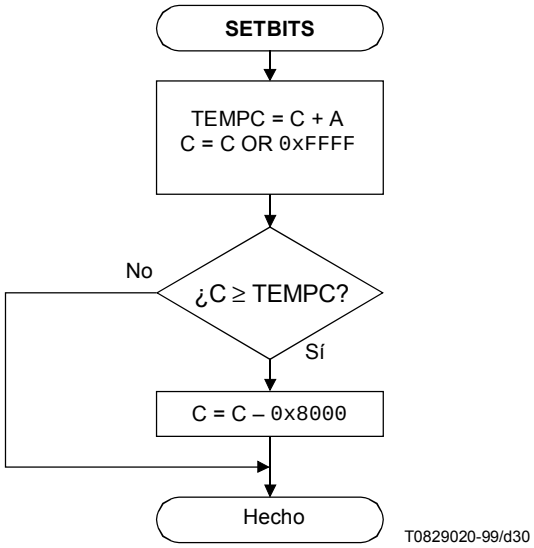


Figura E.12 – Fijación de los bits finales en el registro C

E.3 Procedimiento de decodificación aritmética

La figura E.13 muestra el diagrama de bloques sencillo de un decodificador aritmético adaptativo binario. Los datos comprimidos CD y el contexto CX de la unidad modelo de decodificador (no mostrada) constituyen la entrada al decodificador aritmético. La salida del decodificador es la decisión D. Las unidades modelo de codificador y decodificador han de suministrar exactamente el mismo contexto CX para cada decisión dada.

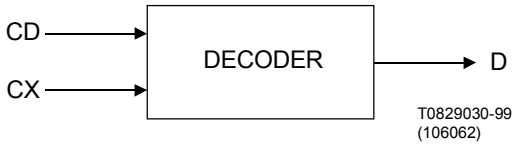


Figura E.1 – Entradas y salidas del codificador aritmético

El DECODER (figura E.14) inicializa el decodificador por medio del procedimiento INITDEC. Contextos, CX y bytes de datos comprimidos (si se necesitan), son leídos y pasados al DECODE hasta que todos los contextos hayan sido leídos. La rutina DECODE decodifica la decisión binaria D y retorna un valor de 0 ó 1. Los procedimientos de estimación de la probabilidad que proporcionan estimaciones adaptativas de la probabilidad para cada contexto están incorporados en DECODE. Cuando todos los contextos han sido leídos (¿Terminados?), los datos comprimidos han sido descomprimidos.

E.3.1 Convenios de registro de códigos del decodificador

En los diagramas de flujo indicados en este anexo se suponen las siguientes estructuras de registro del decodificador:

	15	0
Registro Chigh	xxxxxxxx	xxxxxxxx
Registro Clow	bbbbbbbb	00000000
Registro A	aaaaaaaa	aaaaaaaa

Se puede considerar que Chigh y Clow son un registro C de 32 bits en donde la renormalización de C desplaza un bit de datos nuevos del bit 15 de Clow al bit 0 de Chigh. Sin embargo, las comparaciones de decodificación utilizan solamente Chigh. Los datos nuevos se insertan en los bits "b" de Clow a razón de un byte cada vez.

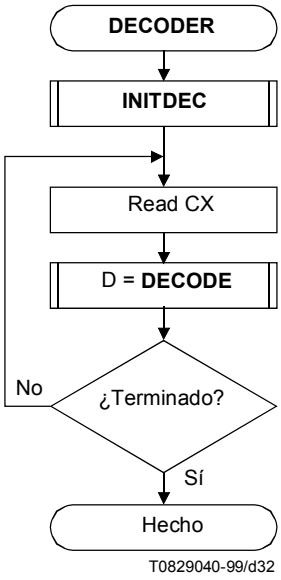


Figura E.14 – Decodificación del codificador MQ

La descripción detallada del tratamiento de los datos con bits de relleno se indicará más adelante en esta subcláusula.

Se señala que, en las comparaciones mostradas en los diversos procedimientos de esta subcláusula, se suponen precisiones superiores a 16 bits. Se pueden utilizar comparaciones lógicas con precisión de 16 bits.

E.3.2 Decodificación de una decisión (DECODE)

El decodificador decodifica una decisión binaria cada vez. Tras decodificar la decisión, sustrae cualquier cantidad de la cadena de códigos que hubiera añadido el codificador. La cantidad que se deja en la cadena de códigos es el desplazamiento de la base del intervalo vigente al subintervalo atribuido a todas las decisiones binarias todavía no decodificadas. En la primera prueba del procedimiento DECODE ilustrado en la figura E.15, el registro Chigh se compara con el tamaño del subintervalo del LPS. A menos que se necesite un intercambio condicional, esta prueba determina si se decodifica un MPS o un LPS. Si Chigh es lógicamente superior o igual a la estimación de probabilidad Qe del MPS para el índice I actual almacenado en CX, Chigh se decrementa en esa cantidad. Si A no es inferior a `0x8000`, se utiliza el sentido del MPS almacenado en CX para fijar la decisión decodificada D.

Cuando se necesita una renormalización, es posible que se haya producido el intercambio condicional MPS/LPS. En la figura E.16 se muestra el procedimiento de intercambio condicional para el trayecto MPS. Mientras que el tamaño A del subintervalo del MPS calculado como el primer paso en la figura 16 no sea lógicamente inferior a la estimación de probabilidad Qe(I(CX)) del LPS, ello significa que se produjo un MPS y la decisión se puede fijar a partir de MPS(CX). A continuación se actualiza el índice I(CX) de acuerdo con la columna de índice de MPS siguiente (NMPS) del cuadro E.1. Si, no obstante, el intervalo del LPS es mayor, ello significa que se produjo el intercambio condicional y un LPS. La actualización de la probabilidad cambia el sentido del MPS si la columna SWITCH tiene un "1" y actualiza el índice I(CX) de la columna (NLPS) de índice de LPS siguiente del cuadro E.1. Se señala que la estimación de la probabilidad en el decodificador ha de ser idéntica a la estimación de la probabilidad en el codificador.

En la figura E.17 se muestra, como procedimiento de intercambio condicional del trayecto LPS del decodificador, el procedimiento LPS_EXCHANGE. La misma comparación lógica entre el subintervalo A del MPS y el subintervalo Qe(I(CX)) del LPS determina si se produjo un intercambio condicional. En ambos trayectos se fija en Qe(I(CX)) el nuevo subintervalo A. En el trayecto de la izquierda se produjo el intercambio condicional por lo que la decisión y la actualización quedan para el caso del MPS. En el trayecto de la derecha, se siguen la decisión de LPS y la actualización.

E.3.3 Renormalización en el decodificador (RENORMD)

En la figura E.18 se ilustra el procedimiento RENORMD para la renormalización del decodificador. Un contador lleva la cuenta del número de bits comprimidos en la sección Clow del registro C. Cuando CT es cero, se inserta un nuevo byte en Clow con el procedimiento BYTEIN.

Se desplazan tanto el registro de intervalos A como el registro de códigos C, un bit cada vez, hasta que A deje de ser inferior a `0x8000`.

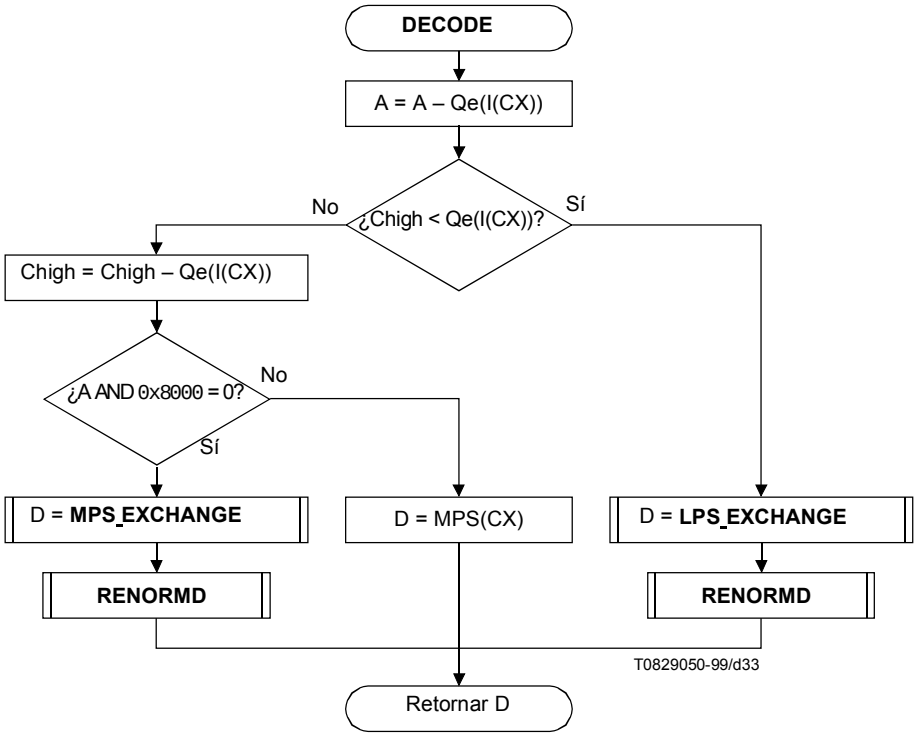


Figura E.15 – Decodificación de un MPS o un LPS

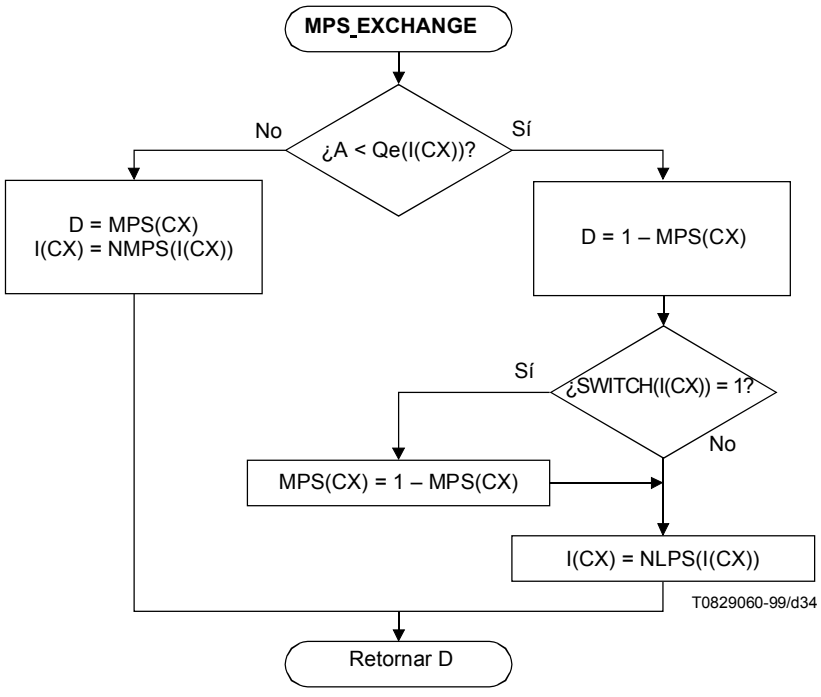


Figura E.16 – Procedimiento de intercambio condicional de trayecto MPS de decodificador

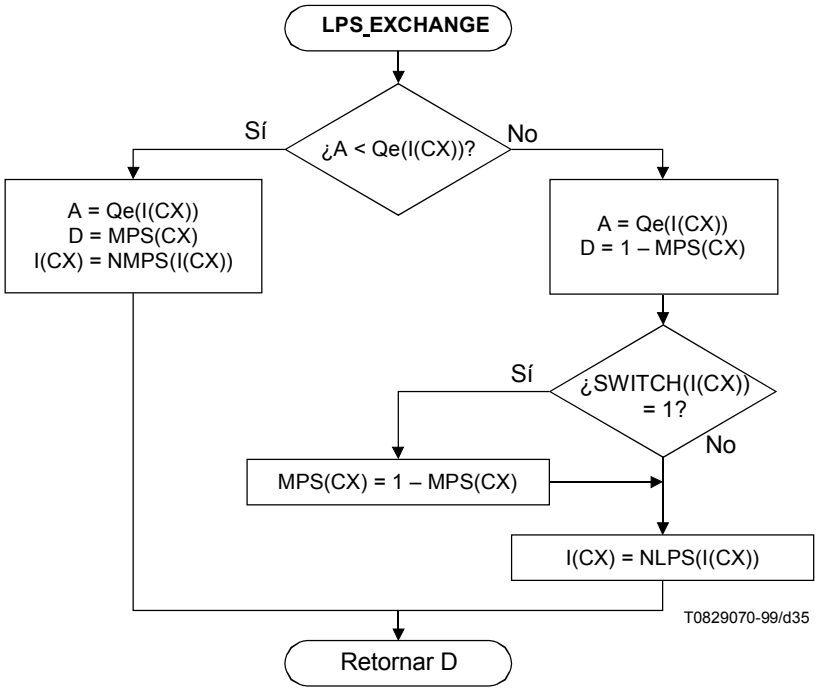


Figura E.17 – Procedimiento de intercambio condicional de trayecto LPS de decodificador

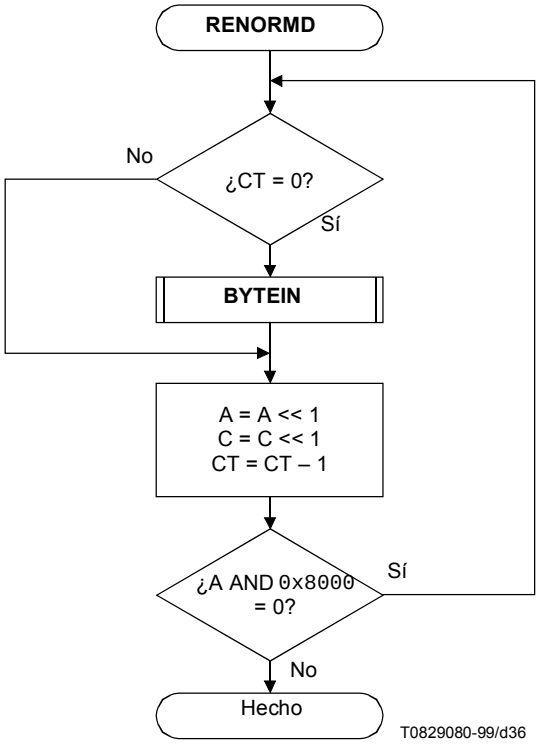


Figura E.18 – Procedimiento de renormalización de decodificador

E.3.4 Entrada de datos comprimidos (BYTEIN)

En la figura E.19 se ilustra el procedimiento BYTEIN llamado desde RENORMD. Este procedimiento lee un byte de datos, compensando al mismo tiempo cualesquiera bits de relleno que sigan al byte `0xFF`. También detecta los códigos marcadores que deben aparecer al final de una exploración o intervalo de resincronización. El registro C es en este procedimiento la concatenación de los registros Chigh y Clow.

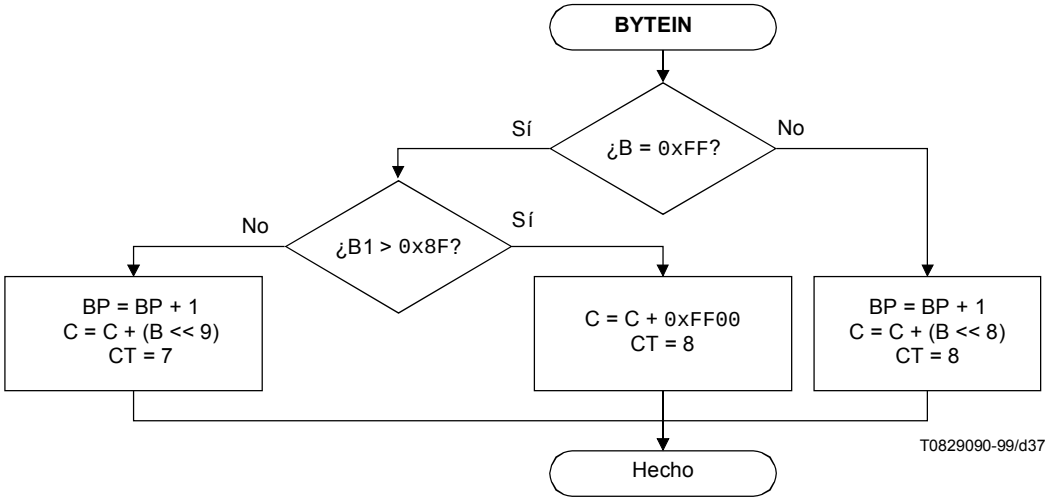


Figura E.19 – Procedimiento BYTEIN para decodificador

B es el byte al que apunta el puntero BP de la memoria intermedia de datos comprimidos. Si B no es un byte `0xFF`, se incrementa BP y el valor nuevo de B se inserta en los bits de orden alto de Clow.

Si B es un byte `0xFF`, se prueba B1 (el byte al que apunta a BP+1). Si B1 es superior a `0x8F`, B1 debe ser uno de los códigos marcadores. El código marcador se interpreta como se requiera, y el puntero de memoria intermedia permanece apuntando al prefijo `0xFF` del código marcador que termina los datos comprimidos aritméticamente. A continuación se introducen bits 1 en el decodificador hasta que se complete la decodificación. Esto se muestra añadiendo `0xFF00` al registro C y fijando el contador de bits CT en 8.

Si B1 no es un código marcador, se incrementa BP para que apunte al byte siguiente que contiene un bit de relleno. El B se añade al registro C con una alineación tal que el bit de relleno (que contiene cualquier arrastre) se añada al bit de orden bajo de Chigh.

E.3.5 Inicialización del decodificador (INITDEC)

El procedimiento INITDEC se utiliza para arrancar el decodificador aritmético. Los pasos básicos se muestran en la figura E.20.

BP, el puntero hacia los datos comprimidos, es inicializado en BPST (apuntando al primer byte comprimido). El primer byte de los datos comprimidos es desplazado al byte de orden bajo de Chigh y a continuación se lee un nuevo byte. El registro C se desplaza entonces 7 bits y C se decrementa en 7, haciendo que el registro C se alinee con el valor de comienzo de A. El registro de intervalos A se fija de modo que concuerde con el valor de comienzo del decodificador.

E.3.6 Resincronización del decodificador

Normalmente, cuando se alcanza el fin de los datos comprimidos aritméticamente, el puntero BP de memoria intermedia de datos comprimidos apunta al byte `0xFF` del código marcador de terminación. Si por cualquier razón el puntero de memoria intermedia de datos comprimidos no está en el byte `0xFF` del marcador, es preciso que un procedimiento de resincronización explore los datos comprimidos hasta encontrar el prefijo del código marcador de terminación. Si hace falta una búsqueda de ese tipo, ello indica una condición de error. El procedimiento de recuperación tras ese error no está normalizado.

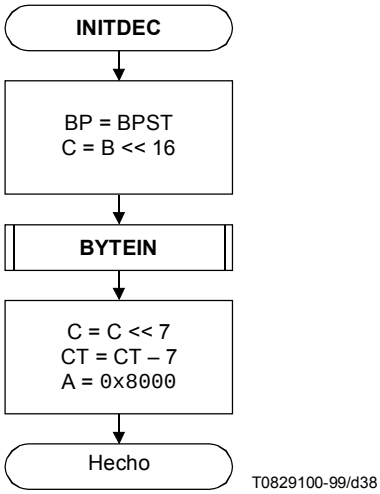


Figura E.20 – Inicialización del decodificador

E.3.7 Reposición de las estadísticas de codificación aritmética

En determinados momentos durante la decodificación, se reponen algunas o la totalidad de las estadísticas de codificación aritmética. Este procedimiento implica la fijación de I(CX) y MSP(CX) igual a cero para algunos de los valores de CX o la totalidad de los mismos.

EJEMPLO – Al comienzo de la codificación de un segmento de región de texto se reponen todas las estadísticas de codificación aritmética.

E.3.8 Conservación de las estadísticas de codificación aritmética

En algunos casos, el decodificador necesita conservar o restablecer algunos valores de I(CX) y MPS(CX). Esto se hace como parte de la decodificación de un segmento de diccionario de símbolos. En tal caso, los valores que se conservan y/o restablecen son todos los valores indizados mediante valores de CX cuya etiqueta inicial "GB" o "GR" (es decir, todos aquellos valores de CX utilizados por el procedimiento de decodificación de región genérica o el procedimiento de decodificación de región de refinamiento genérico).

Anexo F

Perfiles

(Este anexo es parte integrante de esta Recomendación | Norma Internacional)

Se recomienda que un decodificador JBIG2 implemente uno de los perfiles descritos en los cuadros F.1 a F.7. Se señala que el perfil 0×00000001 (cuadro F.1) incluye todas las capacidades de la especificación completa, y es el perfil que se supone cuando no se especifica ninguno de manera explícita.

Véase en 7.4.12 la información sobre cómo se utilizan los números de identificación de perfil.

Los números de identificación de perfil 0×00000000 a 0×00FFFFFF se reservan para aplicaciones ISO/CEI y UIT. De este intervalo se reservan los números de identificación de perfil del 0x01000000 al 0xFFFFFFFF para aplicaciones de facsímil del UIT-T. Las entidades –aparte de ISO/CEI y de la UIT– que deseeen utilizar un número de identificación de perfil no asignado deberán elegir uno de la gama 0×01000000 a 0×FFFFFFFF que probablemente no sea incompatible con la elección de cualquier otra entidad. Se recomienda que los tres primeros bytes del número de identificación de perfil se elijan de modo que concuerden con las tres primeras letras del nombre de la entidad, o que sean una abreviatura adecuada de ese nombre.

Cuadro F.1 – Descripción de perfil para el perfil 0×00000001

Identificación de perfil	0x00000001
Requisitos	Todas las capacidades JBIG2
Codificación de región genérica	Sin restricciones
Codificación de región de refinamiento	Sin restricciones
Codificación de región de semitonos	Sin restricciones
Datos numéricos	Sin restricciones
Recursos requeridos	Procesador de alta velocidad
Ejemplos de aplicación	Impresión para uso general; conversión de formatos

Cuadro F.2 – Descripción de perfil para el perfil 0×00000002

Identificación de perfil	0x00000002
Requisitos	Compresión máxima
Codificación de región genérica	Sólo aritmética; se utiliza cualquier plantilla
Codificación de región de refinamiento	Sin restricciones
Codificación de región de semitonos	Sin restricciones
Datos numéricos	Aritmética solamente
Recursos requeridos	Procesador de alta velocidad
Ejemplos de aplicación	Archivo; WWW inalámbrica

Cuadro F.3 – Descripción de perfil para el perfil 0×00000003

NOTA 1 – Este perfil es un subconjunto del perfil 0x00000002.

Identificación de perfil	0×00000003
Requisitos	Complejidad media y compresión media
Codificación de región genérica	Solamente aritmética; solamente plantillas de 10 píxels y 13 píxels
Codificación de región de refinamiento	Solamente plantilla de 10 píxels (aritmética)
Codificación de región de semitonos	No se utiliza máscara de saltos
Datos numéricos	Solamente aritmética
Recursos requeridos	Procesador de velocidad media
Ejemplos de aplicación	WWW; fax de extremo alto

Cuadro F.4 – Descripción de perfil para el perfil 0×00000004

Identificación de perfil	0x00000004
Requisitos	Baja complejidad con capacidad sin pérdida progresiva
Codificación de región genérica	Solamente MMR
Codificación de región de refinamiento	Solamente plantilla de 10 píxels (aritmética)
Codificación de región de semitonos	No se utiliza máscara de saltos
Datos numéricos	Solamente Huffman
Recursos requeridos	Procesador de velocidad media
Ejemplos de aplicación	WWW

Cuadro F.5 – Descripción de perfil para el perfil 0×00000005

NOTA 2 – Este perfil es un subconjunto del perfil 0x00000004.

Identificación de perfil	0x00000005
Requisitos	Baja complejidad
Codificación de región genérica	Solamente MMR
Codificación de región de refinamiento	No disponible
Codificación de región de semitonos	No se utiliza máscara de saltos
Datos numéricos	Solamente Huffman
Recursos requeridos	Procesador de baja velocidad
Ejemplos de aplicación	Fax de extremo bajo; impresión de alta velocidad procesadores insertados

Cuadro F.6 – Descripción de perfil para el perfil 0x00000006

NOTA 3 – Este perfil es un subconjunto del perfil 0x00000003.

Identificación de perfil	0x00000006
Requisitos	Baja memoria con más compresión
Codificación de región genérica	Solamente aritmética; sólo plantillas de 10 píxels y de 13 píxels
Codificación de región de refinamiento	Solamente plantilla de 10 píxels
Codificación de región de semitonos	No se utiliza máscara de saltos
Datos numéricos	Solamente aritmética
Recursos requeridos	Procesador de velocidad con baja memoria
Ejemplos de aplicación	Dispositivos de múltiples funciones
Restricciones adicionales	• Cada página debe tener al menos dos franjas • Una franja que contiene un segmento de región de texto puede no contener ningún segmento de semitonos de región genérica • Todos los segmentos de región deben ser segmentos de regiones inmediatas (sin memorias intermedias auxiliares) • En cualquier diccionario de simbolos con **SDREFAGG = 1** debe cumplirse REFAGGNINST = 1 para todos los símbolos

Cuadro F.7 – Descripción de perfil para el perfil 0x00000007

NOTA 4 – Este perfil es un subconjunto del perfil 0x00000005.

Identificación de perfil	0x00000007
Requisitos	Baja memoria con más velocidad
Codificación de región genérica	Solamente MMR
Codificación de región de refinamiento	No disponible
Codificación de región de semitonos	No se utiliza máscara de satos
Datos numéricos	Solamente Huffman
Recursos requeridos	Procesador de baja velocidad con baja memoria
Ejemplos de aplicación	Dispositivos de múltiples funciones
Restricciones adicionales	• Cada página debe tener al menos dos franjas • Una franja que contiene un segmento de región de texto puede no contener ningún segmento de semitonos ni de región genérica • Todos los segmentos de región deben ser segmentos de regiones inmediatas (sin memorias intermedias auxiliares) • En cualquier diccionario de símbolos con **SDREFAGG = 1**, REFAGGNINST = 1 para todos los símbolos

Anexo G

Procedimiento de decodificación aritmética (convenios relativos al soporte lógico)

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

Este anexo coantiene algunos diagramas de flujo alternativos para una versión del decodificador de entropía adaptativo. Esta versión alternativa puede ser más eficaz cuando se implemente en soporte lógico, ya que tiene menos operaciones a lo largo del trayecto rápido.

La versión alternativa se obtiene haciendo las siguientes sustituciones:

- El diagrama de flujo de la figura E.20 por el diagrama de flujo de la figura G.1.
- El diagrama de flujo de la figura E.15 por el diagrama de flujo de la figura G.2.
- El diagrama de flujo de la figura E.19 por el diagrama de flujo de la figura G.3.

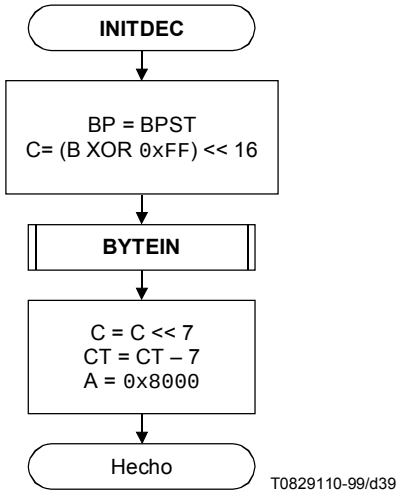


Figura G.1 – Inicialización del decodificador con convenios de soporte lógicos

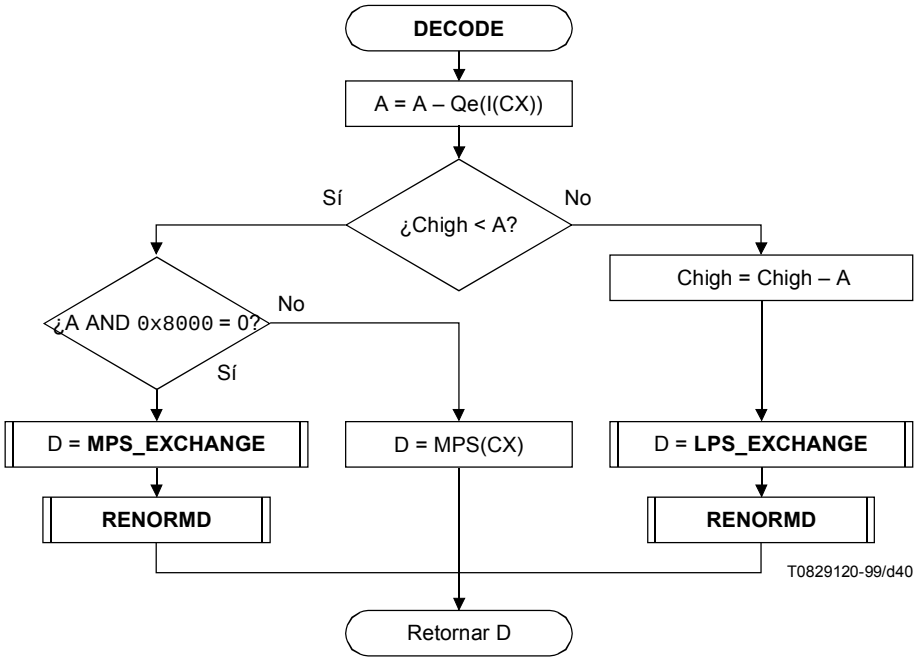


Figura G.2 – Decodificación de un MPS o un LPS en el decodificador con convenios de soporte lógico

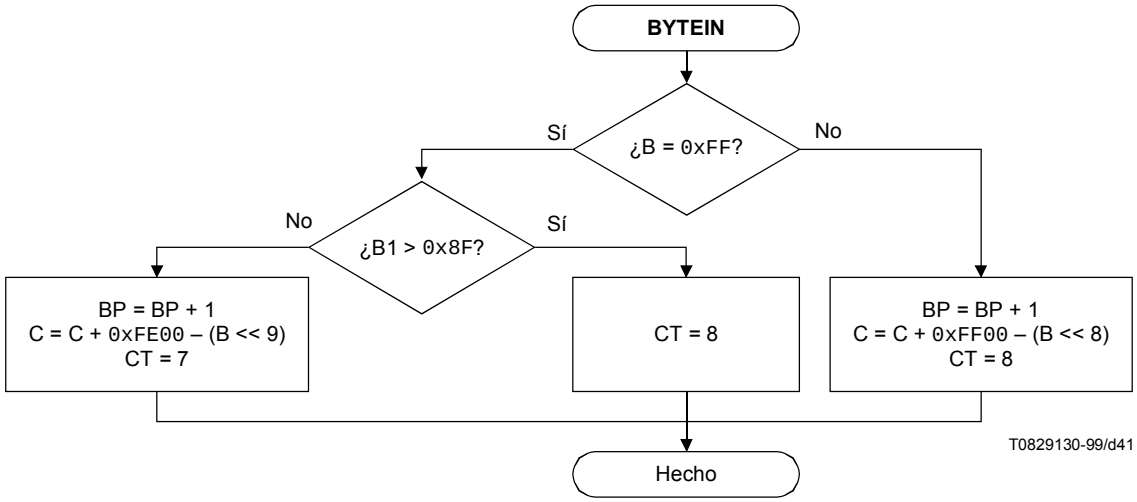


Figura G.3 – Inserción de un nuevo byte en el registro C en el decodificador con convenios de soporte lógico

Anexo H

Ejemplo de tren de datos y secuencia de prueba

(Este anexo no es parte integrante de esta Recomendación | Norma Internacional)

H.1 Ejemplo de tren de datos

En esta subcláusula se presenta un pequeño tren de datos que contiene un gran número de las características funcionales de la norma JBIG2.

Los datos en bruto se representan aquí con el siguiente formato:

```
0023: 01 23 45 67 89 AB CD EF
```

donde el primer campo (antes de los dos puntos) es el desplazamiento de byte en el tren de datos de los datos que se visualizan, y el resto de la línea es una secuencia de bytes que empiezan en ese desplazamiento. Todos estos valores están en hexadecimal.

Por lo general, la decodificación de la primera ocurrencia de cualquier tipo de datos se explica en detalle; posteriores ocurrencias del mismo tipo de datos se explican de manera más somera.

Este tren de datos codifica:

- un diccionario de símbolos que no está asociado con ninguna página, y
- tres páginas.

Es decir, las entidades codificadas en este tren de datos son tres páginas, más un objeto que no está asociado a ninguna página.

Las dos primeras páginas contienen, cada una de ellas, un segmento de información de página, un segmento de diccionario de símbolos, un segmento de región de texto, un segmento de región genérica, un segmento de diccionario de patrones, un segmento de región de semitonos y un segmento fin de página. Los mapas de bits codificados por estas dos páginas son idénticos y son tal como se muestra en la figura H.1. Los datos codificados por los segmentos correspondientes de las dos páginas también son idénticos (por ejemplo, el segmento de región de texto de la página 1 contiene los mismos datos, en el mismo orden, que el segmento de región de texto de la página 2). Sin embargo, los segmentos se codifican de manera diferente en las dos páginas: en la página 1, todos los segmentos utilizan alguna forma de codificación Huffman o MMR; en la página 2, todos los segmentos utilizan alguna forma codificación aritmética. Así, los implementadores pueden verificar la concordancia con sus propias implementaciones para tener la seguridad de que decodifican correctamente.

La tercera página contiene dos diccionarios de símbolos, uno de los cuales define símbolos por refinamiento y agregación de los del otro, y una región de texto que utiliza los símbolos del diccionario que incluye el refinamiento del otro.

A lo largo de esta subcláusula, los píxels que tienen el valor **1** se muestran como píxels negros, mientras que los píxels que tienen el valor **0** se muestran como píxels blancos. Se trata de una interpretación típica del **0** y el **1**, como la que podría hacer una aplicación que utilizara la presente Recomendación | Norma Internacional.

El tren de datos es

```
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
```

El tren de datos se decodifica como sigue.

1) El encabezamiento de fichero:

   `0000: 97 4A 42 32 0D 0A 1A 0A 01 00 00 00 02`

   a) La cadena ID de ocho bytes:

      `0000: 97 4A 42 32 0D 0A 1A 0A`

   b) El campo de banderas de encabezamiento de fichero de un byte:

      `0008: 01`

      Este campo indica que el fichero utiliza la organización secuencial, y el número de páginas es conocido.

   c) El campo de número de páginas de cuatro bytes:

      `0009: 00 00 00 03`

      Este campo indica que el fichero tiene tres páginas.

2) El primer encabezamiento de segmento:

   `000D: 00 00 00 00 00 01 00 00 00 00 18`

   a) El campo de número de segmento de cuatro bytes:

      `000D: 00 00 00 00`

      Este campo indica que el segmento tiene el número de segmento 0.

   b) El campo de banderas de encabezamiento de segmento de un byte:

      `0011: 00`

      Este campo indica que el segmento es de tipo "Diccionario de símbolos" (tipo 0), tiene un campo de asociación de página corto y no tiene fijado el bit de no retención diferida.

   c) El campo de banderas de cuenta de segmentos referenciados y retención de un byte:

      `0012: 01`

      Este campo indica que el segmento no hace referencia a ningún otro segmento, y que deberá ser retenido.

d) El campo de asociación de página de segmento de un byte (forma corta):

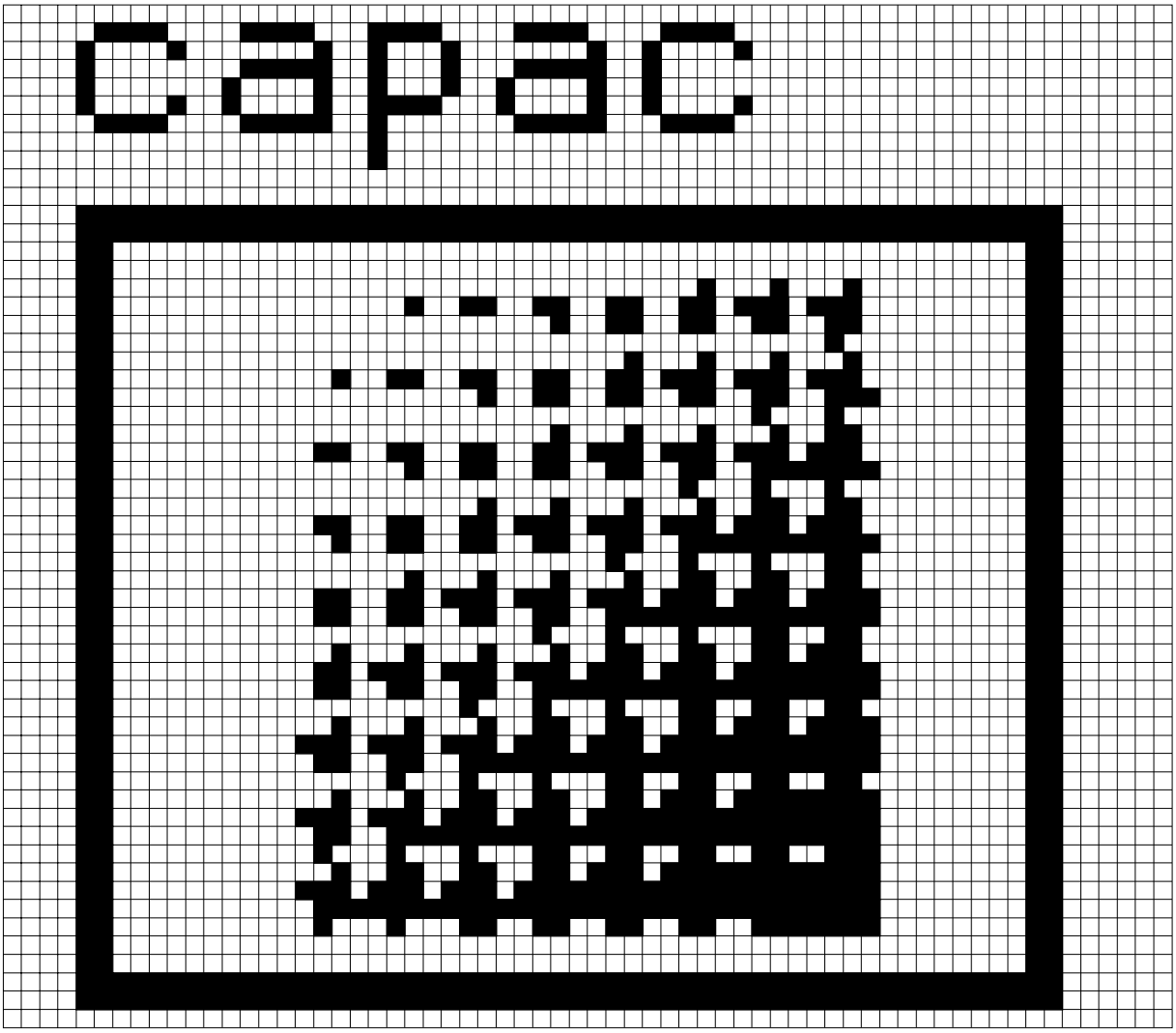
```
0013: 00
```

Este campo indica que este segmento no está asociado con ninguna página.

e) El campo de longitud de datos de segmento de cuatro bytes:

```
0014: 00 00 00 18
```

Este campo indica que la parte datos del segmento tiene una longitud de 24 bytes.



T0829140-99/d42

Figura H.1 – Mapa de bits de página de tren de datos de prueba

3) La parte datos del primer segmento:

```
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) El campo de banderas de diccionario de símbolos de dos bytes:

```
0018: 00 01
```

Este campo indica que el segmento se ha codificado utilizando la variante de codificación Huffman, no utiliza codificación de refinamiento/agregación y emplea el cuadro B.4 para **SDHUFFDH** y el cuadro B.2 para **SDHUFFDW**.

b) El campo **SDNUMEXSYMS** de cuatro bytes:

```
001A: 00 00 00 01
```

Este campo indica que **SDNUMEXSYMS** es 1: este diccionario de símbolos exporta un símbolo.

c) El campo **SDNUMNEWSYMS** de cuatro bytes:

```
001E: 00 00 00 01
```

Este campo indica que **SDNUMNEWSYMS** es 1: este diccionario de símbolos define un símbolo.

d) Los datos del diccionario de símbolos codificados:

```
0022: E9 CB F4 00 26 AF 04 BF F0 78 2F E0 00 40
```

Esto se decodifica como sigue:

i) Utilizando el cuadro B.4, decodificar un valor de altura delta (diferencia de) de clase de altura. Esto consume los bits **1110 100**, indicando una altura delta de clase de altura de 8. La primera clase de altura tiene por tanto una altura de 8 píxels.

ii) Utilizando el cuadro B.2, decodificar un valor de anchura delta (diferencia de). Esto consume los bits **1110 010**, indicando una anchura delta de 5. El primer símbolo tiene por tanto una anchura de 5 píxels.

iii) Utilizando el cuadro B.2, decodificar un valor de delta anchura. Esto consume los bits **111111**, lo que indica una anchura delta de OOB. Esto termina la clase de altura; la clase de altura contiene un símbolo cuya anchura es de 5 píxels y cuya altura es de 8 píxels.

iv) Utilizando el cuadro B.1, decodificar el tamaño en bytes del mapa de bits colectivo de clase de altura. Esto consume los bits **01000**, indicando un tamaño de 8 bytes.

v) Saltar los bits restantes en el último byte leído. Esto consume los bits **0000000**.

vi) Decodificar los ocho bytes siguientes:

```
0026: 26 AF 04 BF F0 78 2F E0
```

utilizando MMR. Esto produce el mapa de bits colectivo de clase de altura, que también es el mapa de bits del único símbolo en la clase de altura, como se muestra en la figura H.2.

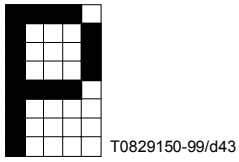


Figura H.2 – Primer símbolo en el primer diccionario de símbolos

vii) Puesto que **SDNUMNEWSYMS** es 1, el último símbolo ha sido ahora decodificado.

viii) Utilizando el cuadro B.1, decodificar una longitud de recorrido de exportación. Esto consume los bits **00000**, indicando que los primeros símbolos 0 no se exportan.

ix) Utilizando el cuadro B.1, decodificar una longitud de recorrido de exportación. Esto consume los bits **00001**, indicando que los siguientes símbolos 1 se exportan. Este diccionario de símbolos define por tanto un símbolo, que es exportado.

x) Saltar los bits restantes en el último byte. Esto consume los bits **000000**.

4) El encabezamiento del segundo segmento:

```
0030: 00 00 00 01 30 00 01 00 00 00 13
```

Este segmento tiene un número de segmento de 1, un tipo de "Información de página" (tipo 48), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento, y no está retenido. Está asociado con la página 1, y tiene una longitud de datos de 19 bytes.

5) La parte datos del segundo segmento:

```
003B: 00 00 00 40 00 00 00 38 00 00 00 00 00 00 00 00
004B: 01 00 00
```

a) El campo de anchura de mapa de bits de página:

```
003B: 00 00 00 40
```

Esto indica que la página tiene una anchura de 64 píxels.

b) El campo de altura de mapa de bits de página:

```
003F: 00 00 00 38
```

Esto indica que la página tiene una altura de 56 píxels.

c) El campo de resolución de X de página:

```
0043: 00 00 00 00
```

Esto indica que la resolución de X de la página es desconocida.

d) El campo de resolución de Y de página:

```
0047: 00 00 00 00
```

Esto indica que la resolución de Y de la página es desconocida.

e) El campo de banderas de segmento de página:

```
004B: 01
```

Esto indica que la página es eventualmente sin pérdida, la página no contiene ningún refinamiento, el valor del píxel por defecto de la página es **0**, el operador de combinación por defecto de la página es OR, la página no requiere ninguna memoria intermedia auxiliar y el operador de combinación por defecto de la página es utilizado por todos los segmentos de región de la página.

f) El campo de información de división en franjas de la página:

```
004C: 00 00
```

Esto indica que la página no está dividida en franjas.

6) El encabezamiento del tercer segmento:

```
004E: 00 00 00 02 00 01 01 00 00 00 1C
```

Este segmento tiene un número de segmento de 2, un tipo de "Diccionario de símbolos" (tipo 0), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y está retenido. Está asociado con la página 1 y tiene una longitud de datos de 28 bytes.

7) La parte datos del tercer segmento:

```
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) El campo de banderas de diccionario de símbolos de dos bytes:

```
0059: 00 01
```

Este campo indica que el segmento se ha codificado utilizando la variante de codificación Huffman, que no utiliza codificación de refinamiento/agregación y que utiliza el cuadro B.4 para **SDHUFFDH** y el cuadro B.2 para **SDHUFFDW**.

b) El campo **SDNUMEXSYMS** de cuatro bytes:

```
005B: 00 00 00 02
```

Este campo indica que **SDNUMEXSYMS** es 2: dos símbolos son exportados por este diccionario de símbolos.

c) El campo **SDNUMNEWSYMS** de cuatro bytes:

```
005F: 00 00 00 02
```

Este campo indica que **SDNUMNEWSYMS** es 2: dos símbolos son definidos por este diccionario de símbolos.

d) Los datos del diccionario de símbolos codificados:

```
0063: E5 CD F8 00 79 E0 84 10 81 F0 82 10 86 10 79 F0
0073: 00 80
```

Esto se decodifica como sigue:

i) Utilizando el cuadro B.4, decodificar un valor de altura delta de clase de altura. Esto consume los bits **1110 010**, indicando una altura delta de clase de altura de 6. La primera clase de altura tiene por tanto una altura de 6 píxels.

ii) Utilizando el cuadro B.2,decodificar un valor de anchura delta. Esto consume los bits **1110 011**, indicando una anchura delta de 6. El primer símbolo tiene por tanto una anchura de 6 píxels.

iii) Utilizando el cuadro B.2, decodificar un valor de anchura delta. Esto consume los bits **0**, indicando una anchura delta de 0. El segundo símbolo tiene por tanto una anchura de 6 píxels.

iv) Utilizando el cuadro B.2, decodificar un valor de anchura delta. Esto consume los bits **111111**, indicando una anchura delta de OOB. Esto termina la clase de altura; la clase de altura contiene dos símbolos, ambos con una anchura de 6 píxels y una altura de 6 píxels.

v) Utilizando el cuadro B.1, decodificar el tamaño en bytes del mapa de bits colectivo de clase de altura. Esto consume los bits **00000**, indicando un tamaño de cero bytes. Esto indica que el mapa de bits colectivo de clase de altura se almacena sin ser comprimido. Puesto que la anchura total de los símbolos en la clase de altura es 12, cada fila de la clase de altura se rellena para que tenga una anchura de 16 bits (2 bytes).

vi) Saltar los bits restantes en el último byte leído. Esto consume los bits **000000**.

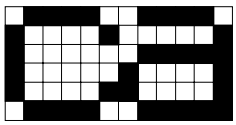
vii) Leer los 12 bytes siguientes (6 filas de 2 bytes cada una), y utilizar los 12 bits situados más a la izquierda de cada fila como mapa de bits colectivo de clase de altura. Estos 12 bytes son:

```
0067: 79 E0 84 10 81 F0 82 10 86 10 79 F0
```

o, en binario:

01111001	**11100000**
10000100	**00010000**
10000001	**11110000**
10000010	**00010000**
10000110	**00010000**
01111001	**11110000**

El mapa de bits colectivo de la clase de altura es por tanto como se muestra en la figura H.3, y los dos símbolos son como se muestran en la figura H.4 a) y b).



T0829160-99/d44

Figura H.3 – Mapa de bits colectivo de clase de altura en el segundo diccionario de símbolos

viii) Puesto que **SDNUMNEWSYMS** es 2, el último símbolo ha sido ahora decodificado.

ix) Utilizando el cuadro B.1, decodificar una longitud de pasada de exportación. Esto consume los bits **00000**, indicando que los primeros símbolos 0 no son exportados.

a) b)

T0829170-99/d45

Figura H.4 – Símbolos en el segundo diccionario de símbolos

x) Utilizando el cuadro B.1, decodificar una longitud de pasada de exportación. Esto consume los bits **00010**, indicando que los dos símbolos siguientes son exportados. El diccionario de símbolos define por tanto dos símbolos, y los dos son exportados.

xi) Saltar los bits restantes en el último byte. Esto consume los bits **000000**.

8) El encabezamiento del cuarto segmento:

```
0075: 00 00 00 03 07 42 00 02 01 00 00 00 31
```

Este segmento tiene un número de segmento 3, un tipo de "Región de texto sin pérdida inmediata" (tipo 7), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. Hace referencia a otros dos segmentos, con número de segmento 0 y 2; el segmento 0 deberá ser retenido, mientras que el segmento 2 y este segmento no deberán ser retenidos. Está asociado con la página 1 y tiene una longitud de datos de 49 bytes.

9) La parte datos del cuarto segmento:

```
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) El campo de anchura de mapa de bits de segmento de región:

```
0082: 00 00 00 25
```

Este campo indica que el mapa de bits de región tiene una anchura de 37 píxels.

b) El campo de altura de mapa de bits de segmento de región:

```
0086: 00 00 00 08
```

Este campo indica que el mapa de bits de región tiene una altura de 8 píxels.

c) El campo ubicación de X de mapa de bits de segmento de región:

```
008A: 00 00 00 04
```

Este campo indica que el borde izquierdo del mapa de bits de región está 4 píxels a la derecha del borde izquierdo de la página.

d) El campo de ubicación de Y de mapa de bits de segmento de región:

```
008E: 00 00 00 01
```

Este campo indica que el borde superior del mapa de bits de la región está 1 píxel por debajo del borde superior de la página.

e) El campo de banderas de segmento de región:

```
0092: 00
```

Este campo indica que la región deberá dibujarse en la página utilizando el operador de combinación OR.

f) El campo de banderas de segmento de región de texto de dos bytes:

```
0093: 0C 09
```

Este campo indica que el segmento se ha codificado utilizando la variante de codificación Huffman, que no contiene ningún refinamiento, tiene un valor **SBSTRIPS** de 4, tiene una esquina de referencia de BOTTOMLEFT, no está transpuesto, combina sus símbolos utilizando OR, tiene un valor de píxel por defecto de **0** y tiene un valor de **SBDSOFFSET** de 3.

g) El campo de banderas Huffman de segmento de región de dos bytes:

```
0095: 00 10
```

Este campo indica que el segmento utiliza el cuadro B.6 para **SBHUFFFS**, el cuadro B.8 para **SBHUFFDS** y el cuadro B.12 para **SBHUFFDT**.

h) El campo **SBNUMINSTANCES** de 4 bytes:

```
0097: 00 00 00 05
```

Este campo indica que **SBNUMINSTANCES** es 5: cinco ejemplares de símbolos son codificados en esta región de texto.

i) La tabla de decodificación Huffman de ID símbolo de segmento de región de texto:

```
009B: 01 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00AB: 00 0C
```

Los dos diccionarios de símbolos a los que hace referencia este segmento tienen un total de 3 símbolos. La decodificación de la tabla Huffman RUNCODE, consumiendo todos los bits de los datos excepto los últimos cuatro, da la siguiente asignación de longitudes de código a los RUNCODE:

RUNCODE1	1	RUNCODE2	1

y por tanto la siguiente tabla Huffman para los RUNCODE:

RUNCODE1	**0**	RUNCODE2	**1**

La decodificación utilizando esta tabla produce la secuencia RUNCODE2, RUNCODE2, RUNCODE1 (consumiendo los bits **110**). Así pues, el primer símbolo (el "p" del primer diccionario de símbolos) tiene una longitud de código Huffman de 2; el segundo símbolo (el "c" del segundo diccionario de símbolos) tiene una longitud de código Huffman de 2, y el tercer símbolo (la "a" del tercer diccionario de símbolos) tiene una longitud de código de 1. La aplicación del algoritmo de asignación de códigos Huffman da la tabla:

p	**10**
c	**11**
a	**0**

En este punto, hay un bit (**0**) restante en el último dato; bit que ahora se salta.

j) Los datos de la región de texto codificados:

```
00AD: 40 07 08 70 41 D0
```

Esto se decodifica como sigue;

i) Utilizando el cuadro B.12, decodificar un valor de T delta. Esto consume el bit **0**, indicando un valor de T delta de 4 (el valor decodificado de la tabla de 1, multiplicado por **SBSTRIPS**). El valor inicial de STRIPT es –4.

ii) Utilizando el cuadro B.12, decodificar un valor de delta T. Esto consume los bits **10**, indicando un valor de T delta de 8 (dos veces **SBSTRIPS**). STRIPT es por tanto ahora 4.

iii) Utilizando el cuadro B.6, decodificar un primer valor de S. Esto consume los bits **00 00000000**, indicando un primer valor de S de 0.

iv) La lectura de dos bits (desde **SBSTRIPS**) consume los bits **01**. La coordenada T del primer ejemplar de símbolo es por tanto 5 (STRIPT más el valor decodificado de 1).

v) Utilizando la tabla Huffman de ID símbolo, decodificar un valor de ID símbolo. Esto consume los bits **11**, indicando el símbolo "c". Así pues, el símbolo "c" deberá dibujarse con su esquina inferior izquierda en (0, 5).

vi) En este punto, CURS es 8 (0 más **SBDSOFFSET** más la anchura del símbolo previo de 6 menos 1).

vii) Utilizando el cuadro B.8, decodificar un valor de delta S. Esto consume los bits **00 0**, indicando un valor de delta S de 0.

viii) La lectura de dos bits consume los bits **01**. La coordenada T del segundo ejemplar de símbolo es por tanto 5.

ix) Utilizando la tabla Huffman de ID símbolo, decodificar un valor de ID símbolo. Esto consume el bit **0**, indicando el símbolo "a". Así pues, el símbolo "a" deberá dibujarse con su esquina inferior izquierda en (8, 5).

x) En este punto, CURS es 16 (8 más **SBDSOFFSET** más la anchura del símbolo previo de 6 menos 1).

xi) Utilizando el cuadro B.8, decodificar un valor de S delta. Esto consume los bits **00 0**, indicando un valor de delta S de 0.

xii) La lectura de dos bits consume los bits **11**. La coordenada T del tercer ejemplar de símbolo es por tanto 7.

xiii) Utilizando la tabla Huffman de ID símbolo, decodificar un valor de ID símbolo. Esto consume los bits **10**, indicando el símbolo "p". Así pues, el símbolo "p" deberá dibujarse con su esquina inferior izquierda en (16, 7).

xiv) En este punto, CURS es 23 (16 más **SBDSOFFSET** más la anchura del símbolo previo de 5 menos 1).

xv) Utilizando el cuadro B.8, decodificar un valor de S delta. Esto consume los bits **00 0**, indicando un valor de delta S de 0.

xvi) La lectura de dos bits consume los bits **01**. La coordenada T del cuarto ejemplar de símbolo es por tanto 5.

xvii) Utilizando la tabla Huffman de ID símbolo, decodificar un valor de ID símbolo. Esto consume el bit **0**, indicando el símbolo "a". Así pues, el símbolo "a" deberá dibujarse con su esquina inferior izquierda en (8, 5).

xviii) En este punto, CURS es 31 (23 más **SBDSOFFSET** más la anchura del símbolo previo de 6 menos 1).

xix) Utilizando el cuadro B.8, decodificar un valor de S delta. Esto consume los bits **00 0**, indicando un valor de S delta de 0.

xx) La lectura de dos bits consume los bits **01**. La coordenada T del quinto ejemplar de símbolo es por tanto 5.

xxi) Utilizando la tabla Huffman de ID símbolo, decodificar un valor de ID símbolo. Esto consume los bits **11**, indicando el símbolo "c". Así pues, el símbolo "c" deberá dibujarse con su esquina inferior izquierda en (31, 5).

xxii) Utilizando el cuadro B.8, decodificar un valor de S delta. Esto consume los bits **01**, indicando un valor de delta S de OOB, lo que indica el fin de esta franja. Puesto que **SBNUMINSTANCES** es 5, y se han decodificado cinco ejemplares de símbolo, no hay más franjas.

xxiii) Saltar los bits restantes en el último byte leído. Esto consume los bits **0000**.

k) La decodificación de los datos produjo la siguiente lista de ejemplares y ubicaciones de símbolo (la ubicación es el lugar en donde deberá situarse la esquina inferior izquierda del símbolo):

Símbolo	Ubicación
c	(0, 5)
a	(8, 5)
p	(16, 7)
a	(23, 5)
c	(31, 5)

El dibujo de estos ejemplares de símbolo produce el mapa de bits de región de 37 por 8 píxels que se muestra en la figura H.5.

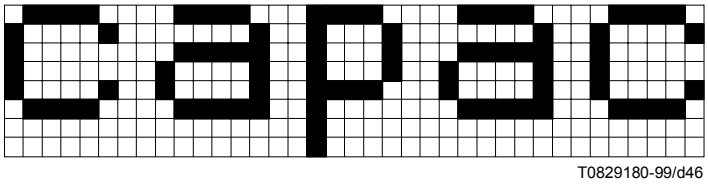


Figura H.5 – Mapa de bits de región de texto

10) El encabezamiento del quinto segmento:

```
00B3: 00 00 00 04 27 00 01 00 00 00 2C
```

Este segmento tiene un número de segmento de 4, un tipo de "Región genérica sin pérdida inmediata" (tipo 39), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y no está retenido. Está asociado con la página 1, y tiene una longitud de datos de 44 bytes.

11) La parte datos del quinto segmento:

```
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) El campo de información de segmento de región:

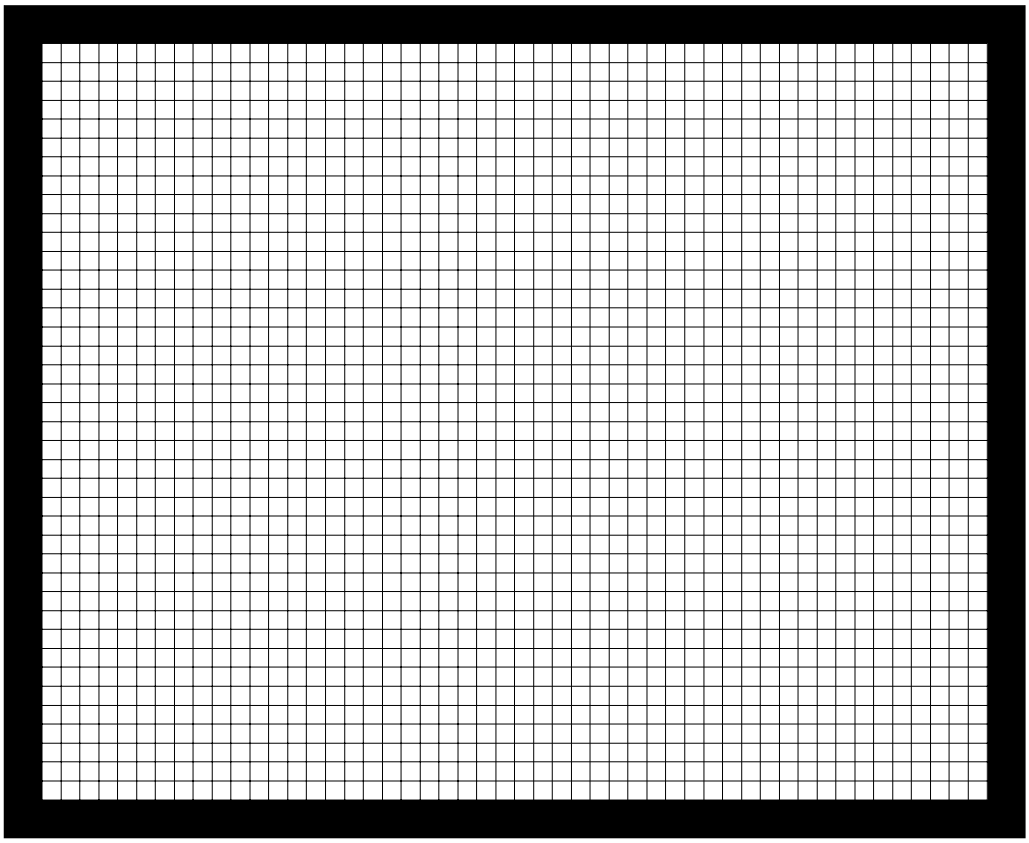
```
00BE: 00 00 00 36 00 00 00 2C 00 00 00 04 00 00 00 0B
00CE: 00
```

Esto indica que el mapa de bits de la región codificada por este segmento tiene 54 píxels de ancho y 44 píxels de alto y su esquina superior izquierda está 4 píxels a la derecha del borde izquierdo de la página y 11 píxels por debajo del borde superior de la página. Deberá dibujarse en la página utilizando OR.

b) Los datos de región:

```
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
```

El primer byte (`01`) es el byte banderas de segmento de región genérica, e indica que la región se ha codificado utilizando MMR. Los bytes restantes son los datos codificados con MMR para el mapa de bits de la región. Estos bytes se descomprimen aplicando MMR para convertirse en el mapa de bits de región de 54 por 44 píxels que se muestra en la figura H.6.



T0829190-99/47

Figura H.6 – Mapa de bits de región genérica

12) El encabezamiento del sexto segmento:

```
00EA: 00 00 00 05 10 01 01 00 00 00 2D
```

Este segmento tiene un número de segmento de 5, un tipo de "Diccionario de patrones" (tipo 16), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento, y está retenido. Está asociado con la página 1 y tiene una longitud de datos de 45 bytes.

13) La parte datos del sexto segmento:

```
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) El campo de banderas de diccionario de patrones de un byte:

```
00F5: 01
```

Este campo indica que el segmento se ha codificado utilizando la variante de codificación MMR.

b) El campo **HDPW** de un byte:

```
00F6: 04
```

Este campo indica que **HDPW**, la anchura de los patrones definidos en este diccionario, es 4.

c) El campo **HDPH** de un byte:

```
00F7: 04
```

Este campo indica que **HPDH**, la altura de los patrones definidos en este diccionario, es 4.

d) El campo **GRAYMAX** de 4 bytes:

```
00F8: 00 00 00 0F
```

Este campo indica que **GRAYMAX** es 15, y por tanto hay 16 patrones en este diccionario de patrones (numerados del 0 al 15).

e) Los 38 bytes restantes en el segmento:

```
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
```

Estos bytes se descomprimen aplicando MMR para convertirse en el mapa de bits colectivo del diccionario de patrones. La anchura del mapa de bits (**GRAYMAX** + 1) × **HDPW**, y la altura es de **HDPH** píxels. En la figura H.7 se muestra el mapa de bits de 64 por 4 píxels que es el resultado de esta descompresión MMR.

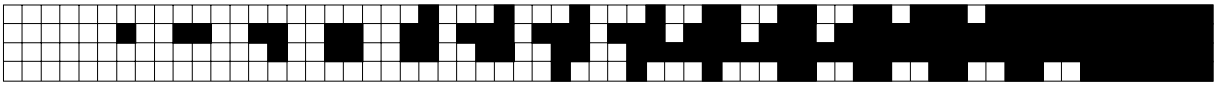


Figura H.7 – Mapa de bits colectivo de diccionario de patrones

Los 16 patrones individuales se obtienen a partir del mapa de bits colectivo descomponiéndolo en partes de 4 píxels de ancho. Se muestran en la figura H.8, en la que el patrón número 0 es el patrón situado arriba a la izquierda, el patrón número 1 está a su derecha, y así sucesivamente.

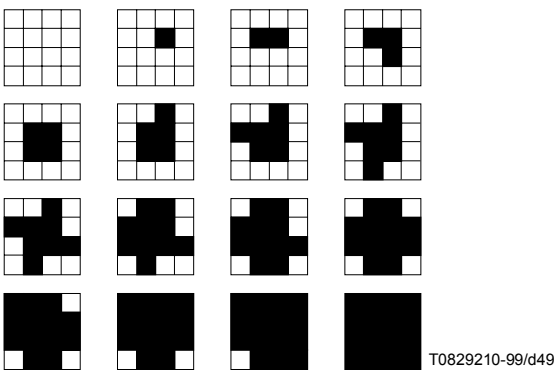


Figura H.8 – Patrones definidos en el diccionario de patrones

14) El encabezamiento del séptimo segmento:

```
0122: 00 00 00 06 17 20 05 01 00 00 00 57
```

Este segmento tiene un número de segmento de 6, un tipo de "Región de semitonos sin pérdida inmediata" (tipo 23), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. Hace referencia a otro segmento, el segmento número 5; ni el segmento número 5 ni este segmento deberán ser retenidos. Está asociado con la página 1 y tiene una longitud de datos de 87 bytes.

15) La parte datos del séptimo segmento:

```
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) El campo de información del segmento de región:

```
012E: 00 00 00 20 00 00 00 24 00 00 00 10 00 00 00 0F
013E: 00
```

Esto indica que el mapa de bits de la región codificada por este segmento tiene una anchura de 32 píxels y una altura de 36 píxels y su esquina superior izquierda está 16 píxels a la derecha del borde izquierdo de la página y 15 píxels por debajo del borde superior de la página. Deberá dibujarse en la página utilizando OR.

b) El campo de banderas del segmento de región de semitonos:

```
013F: 01
```

Este campo indica que la región de semitonos se ha codificado utilizando la variante de codificación MMR. Los patrones deberán combinarse utilizando OR. El valor del píxel por defecto es **0**.

c) El campo **HGW**:

```
0140: 00 00 00 08
```

Este campo indica que la formación de valores de escala de grises tiene una anchura de 8.

d) El campo **HGH**:

```
0144: 00 00 00 09
```

Este campo indica que la formación de valores de escala de grises tiene una altura de 9.

e) El campo **HGX**:

```
0148: 00 00 00 00
```

Este campo indica que el desplazamiento horizontal de la cuadrícula de semitonos es de 0 píxels.

f) El campo **HGY**:

```
014C: 00 00 00 00
```

Este campo indica que el desplazamiento vertical de la cuadrícula de semitonos es de 0 píxels.

g) El campo **HRX**:

```
0150: 04 00
```

Este campo indica que el **HRX** es 1024, y por tanto la coordenada horizontal del vector de la cuadrícula es de 1024/256 píxels, o sea de 4 píxels.

h) El campo **HRY**:

```
0152: 00 00
```

Este campo indica que la coordenada vertical del vector de la cuadrícula es de 0 píxels.

i) Primer plano de bits:

```
0154: AA AA AA AA 80 08 00 80
```

Descomprimiendo esto con MMR se obtiene el mapa de bits mostrado en la figura H.9 a). Se señala que los 7 últimos bits del último byte se saltan una vez que todos los datos codificados con MMR han sido decodificados (es decir, la decodificación del plano de bits está obligada a consumir un número entero de bytes).

j) El segundo plano de bits:

```
015C: 36 D5 55 6B 5A D4 00 40 04
```

Descomprimiendo esto con MMR se obtiene el mapa de bits que se muestra en la figura H.9 b).

k) El tercer plano de bits:

```
0165: 2E E9 52 D2 D2 D2 8A A5 4A 00 20 02
```

Descomprimiendo esto con MMR se obtiene el mapa de bits que se muestra en la figura H.9 c).

l) El cuarto plano de bits:

```
0171: 23 E0 95 24 B4 92 8A 4A 92 54 92 D2 4A 29 2A 49
0181: 40 04 00 40
```

Descomprimiendo esto con MMR se obtiene el mapa de bits que se muestra en la figura H.9 d).

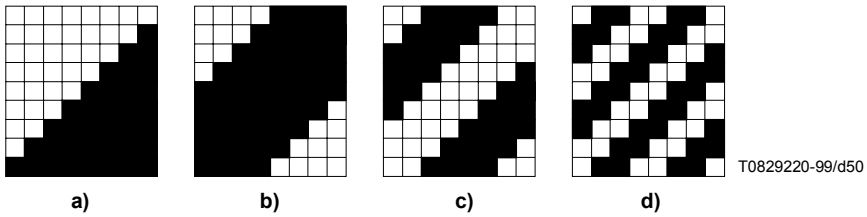


Figura H.9 – Planos de bits en crudo de la región de semitonos

m) A continuación se aplica a estos planos de bits la decodificación Gray descrita en C.5, aplicando el operador XOR entre el primer y el segundo, a continuación entre el plano de bits resultante y tercero, y así sucesivamente. En la figura H.10 se muestran los planos de bits resultantes.

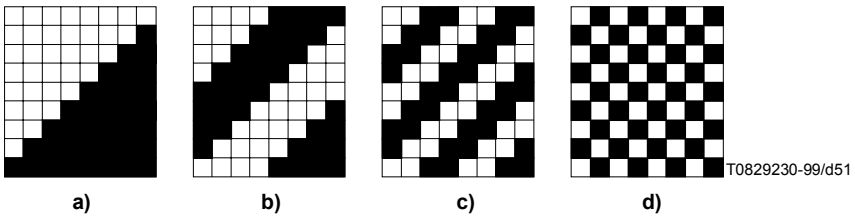


Figura H.10 – Planos de bits de la región de semitonos

n) Apilando estos planos de bits, siendo el primero el más significativo, se obtiene la formación siguiente de valores:

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) El vector de cuadrícula de semitonos y el desplazamiento producen la formación de ubicaciones que se muestra más abajo. Combinándola con la formación de valores se obtiene una lista de operaciones de dibujo que indican que el patrón cuyo índice en el diccionario de patrones del segmento número 5 es ese valor deberá dibujarse con su píxel superior izquierdo en la ubicación indicada.

(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) La realización de esas operaciones de dibujo genera el mapa de bits de región de 32 por 36 que se muestra en la figura H.11.

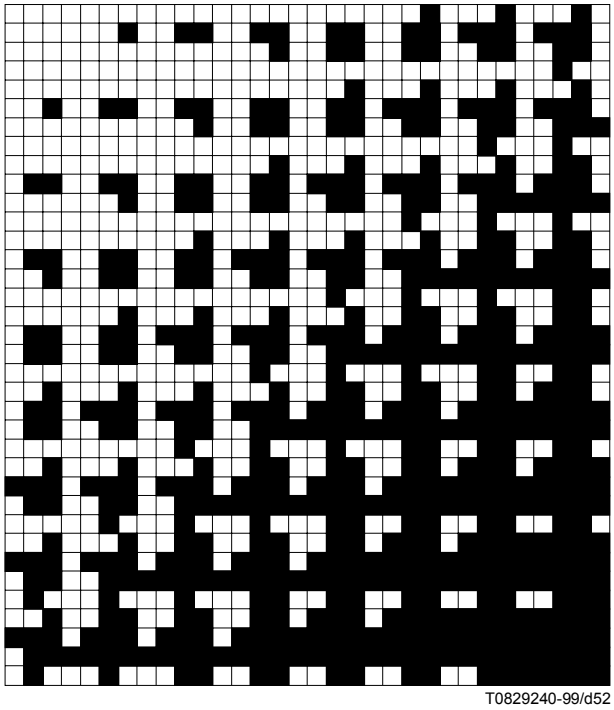


Figura H.11 – Mapa de bits de región de semitonos

16) El encabezamiento del octavo segmento:

```
0185: 00 00 00 07 31 00 01 00 00 00 00
```

Este segmento tiene un número de segmento de 7, un tipo de "Fin de página" (tipo 49), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y no está retenido. Está asociado con la página 1 y tiene una longitud de datos de 0 bytes.

17) Los mapas de bits de segmento de región se combinan como sigue, teniendo en cuenta el valor de píxel por defecto de la página y el operador de combinación de cada segmento de región. Primero, el mapa de bits de página (64 píxels de ancho y 56 píxels de alto) se llena con **0**, el valor de píxel por defecto de la página. A continuación, el mapa de bits mostrado en la figura H.5 se dibuja utilizando OR en el mapa de bits de página con su píxel superior izquierdo en la ubicación (4, 1). Seguidamente, el mapa de bits mostrado en la figura H.6 se dibuja utilizando OR en el mapa de bits de página con su píxel superior izquierdo en la ubicación (4, 11). Finalmente, el mapa de bits mostrado en la figura H.11 se dibuja utilizando OR en el mapa de bits de página con su píxel superior izquierdo en la ubicación (16, 15). Una vez efectuado todo esto, el mapa de bits resultante es el que se muestra en la figura H.1.

18) El encabezamiento del noveno segmento:

```
0190: 00 00 00 08 30 00 02 00 00 00 13
```

Este segmento tiene un número de segmento de 8, un tipo de "Información de página" (tipo 48), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y no está retenido. Está asociado con la página 2 y tiene una longitud de datos de 19 bytes.

19) La parte datos del noveno segmento:

```
019B: 00 00 00 40 00 00 00 38 00 00 00 00 00 00 00 00
01AB: 01 00 00
```

Contiene la misma información que el segmento número 1.

20) El encabezamiento del décimo segmento:

```
01AE: 00 00 00 09 00 01 02 00 00 00 1B
```

Este segmento tiene un número de segmento de 9, un tipo de "Diccionario de símbolos" (tipo 0), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y está retenido. Está asociado con la página 2 y tiene una longitud de datos de 27 bytes.

21) La parte datos del décimo segmento:

```
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) El campo de banderas de diccionario de símbolos de dos bytes:

```
01B9: 08 00
```

Este campo indica que el segmento se ha ido codificando utilizando la variante de codificación aritmética y no utiliza codificación de refinamiento/agregación. **SDTEMPLATE** tiene el valor 2. Los bits "contexto de codificación de mapa de bits utilizado" y "contexto de codificación de mapa de bits retenido" son ambos **0**.

b) El campo de banderas AT de diccionario de símbolos:

```
01BB: 02 FF
```

Este campo indica que $\mathbf{SDATX_1}$ es 2 y $\mathbf{SDATY_1}$ es –1; así pues, el píxel AT A_1 está en la ubicación (2, –1), que es el valor nominal para la plantilla 2.

c) El campo **SDNUMEXSYMS** de cuatro bytes:

```
01BD: 00 00 00 02
```

Este campo indica que **SDNUMEXSYMS** es 2: dos símbolos son exportados por este diccionario de símbolos.

d) El campo **SDNUMNEWSYMS** de cuatro bytes:

```
01C1: 00 00 00 02
```

Este campo indica que **SDNUMNEWSYMS** es 2: dos símbolos son definidos por este diccionario de símbolos.

e) Los datos de diccionario de símbolos codificados:

```
01C5: 4F E7 8C 20 0E 1D C7 CF 01 11 C4 B2 6F FF AC
```

Decodificando esto se obtienen los dos mismos símbolos que se muestran en la figura H.4 a) y b).

22) El encabezamiento del undécimo segmento:

```
01D4: 00 00 00 0A 07 40 00 09 02 00 00 00 1F
```

Este segmento tiene un número de segmento de 10, un tipo de "Región de textos sin pérdida inmediata" (tipo 7), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. Hace referencia a otros dos segmentos, segmento 0 y 9; el segmento 0, el segmento 9 y este segmento no deberán estar retenidos. Está asociado con la página 2 y tiene una longitud de datos de 31 bytes.

23) La parte datos del undécimo segmento:

```
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) El campo de información de segmento de región:

```
01E1: 00 00 00 25 00 00 00 08 00 00 00 04 00 00 00 01
01F1: 00
```

Esto indica que el mapa de bits de región codificado por este segmento tiene 37 píxels de ancho y 8 píxels de alto y su esquina superior izquierda está 4 píxels a la derecha del borde izquierdo de la página y 1 píxel por debajo del borde superior de la página. Deberá dibujarse en la página utilizando OR.

b) El campo de banderas de segmento de región de texto de dos bytes:

```
01F2: 0C 08
```

Este campo indica que el segmento ha sido codificado utilizando la variante de codificación aritmética, no contiene ningún refinamiento, tiene un valor **SBTRIPS** de 4, tiene una esquina de referencia de BOTTOMLEFT, no está transpuesto, combina sus símbolos utilizando OR, tiene un valor de píxel por defecto de **0** y tiene un valor **SBDSOFFSET** de 3.

c) El campo **SBNUMINSTANCES** de cuatro bytes:

```
01F4: 00 00 00 05
```

Este campo indica que **SBNUMINSTANCES** es 5: cinco ejemplares de símbolo están codificados en esta región de texto.

d) Los datos de región de texto codificados:

```
01F8: 8D 6E 5A 12 40 85 FF AC
```

Al decodificar se sigue exactamente la misma secuencia que se vio en la decodificación del segmento número 3 (segmento de región de texto de la página 1), y se obtiene como resultado el mapa de bits de región que se muestra en la figura H.5.

24) El encabezamiento del decimosegundo segmento:

```
0200: 00 00 00 0B 27 00 02 00 00 00 23
```

Este segmento tiene un número de segmento de 11, un tipo de "Región genérica sin pérdida inmediata" (tipo 39), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y no está retenido. Está asociado con la página 2 y tiene una longitud de datos de 35 bytes.

25) La parte datos del decimosegundo segmento:

```
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) El campo de información de segmento de región:

```
020B: 00 00 00 36 00 00 00 2C 00 00 00 04 00 00 00 0B
021B: 00
```

Esto indica que el mapa de bits de la región codificado por este segmento tiene 54 píxels de ancho y 44 píxels de alto, y su esquina superior izquierda está 4 píxels a la derecha del borde izquierdo de la página y 11 píxels por debajo del borde superior de la página. Deberá dibujarse en la página utilizando OR.

b) El campo de banderas de región genérica de un byte:

```
021C: 08
```

Esto indica que la región se ha codificado utilizando codificación aritmética, que **GBTEMPLATE** es 0 y que **TPGDON** es **1**.

c) El campo de banderas AT de segmento de región genérica:

```
021D: 03 FF FD FF 02 FE FE FE
```

Este campo tiene una longitud de ocho bytes porque **GBTEMPLATE** es 0, y hay por tanto cuatro píxels AT cuyas posiciones deben ser determinadas. Los píxels AT están situados con A_1 en (3, –1); A_2 en (–3, –1); A_3 en (2, –2); y A_4 en (–2, –2). Éstas son las posiciones nominales de esos píxels para esta plantilla.

d) Los datos de región:

```
0225: 04 EE ED 87 FB CB 2B FF AC
```

Al decodificar esto utilizando los valores decodificados de **GBTEMPLATE** y **TPGDON** y las ubicaciones de píxels AT se genera el mapa de bits de región de 54 por 44 que se muestra en la figura H.6.

26) El encabezamiento del decimotercer segmento:

```
022E: 00 00 00 0C 10 01 02 00 00 00 1C
```

Este segmento tiene un número de segmento de 12, un tipo de "Diccionario de patrones" (tipo 16), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y no está retenido. Está asociado con la página 2 y tiene una longitud de datos de 28 bytes.

27) La parte datos del decimotercer segmento:

```
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) El campo de banderas de diccionario de patrones de un byte:

```
0239: 06
```

Este campo indica que el segmento se ha codificado utilizando la variante de codificación aritmética, y que **HDTEMPLATE** es 3.

b) El campo **HDPW** de un byte:

```
023A: 04
```

Este campo indica que **HDPW** es 4.

c) El campo **HDPH** de un byte:

```
023B: 04
```

Este campo indica que **HDPH** es 4.

d) El campo **GRAYMAX** de cuatro bytes:

```
023C: 00 00 00 0F
```

Este campo indica que **GRAYMAX** es 15, y por tanto hay 16 patrones en este diccionario de patrones.

e) Los restantes 21 bytes del segmento:

```
0240: 90 71 6B 6D 99 A7 AA 49 7D F2 E5 48 1F DC 68 BC
0250: 6E 40 BB FF AC
```

Estos bytes se descomprimen, utilizando el procedimiento de decodificación de diccionario de patrones, convirtiéndose en el mapa de bits colectivo que se muestra en la figura H.7, y por tanto los 16 patrones definidos por este diccionario de patrones son tal como se muestra en la figura H.8.

28) El encabezamiento del decimocuarto segmento:

```
0255: 00 00 00 0D 17 20 0C 02 00 00 00 3E
```

Este segmento tiene un número de segmento de 13, un tipo de "Región de semitonos sin pérdida inmediata" (tipo 23), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. Hace referencia a otro segmento, el segmento número 12; ni el segmento número 12 ni este segmento deberán ser retenidos. Está asociado con la página 2 y tiene una longitud de datos de 62 bytes.

29) La parte de datos del decimocuarto segmento:

```
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) El campo de información de segmento de región:

```
0261: 00 00 00 20 00 00 00 24 00 00 00 10 00 00 00 0F
0271: 00
```

Esto indica que el mapa de bits de región codificado por este segmento tiene 32 píxels de ancho y 36 píxels de alto y su esquina superior izquierda está 16 píxels a la derecha del borde izquierdo de la página y 15 píxels por debajo del borde superior de la página. Deberá dibujarse en la página utilizando OR.

b) El campo de banderas de segmento de región de semitonos:

```
0272: 02
```

Este campo indica que la región de semitonos se ha codificado utilizando la variante de codificación aritmética, y que **HTEMPLATE** es 1. Los patrones deberán combinarse utilizando OR. El valor de píxel por defecto es **0**.

c) Los otros parámetros:

```
0273: 00 00 00 08 00 00 00 09 00 00 00 00 00 00 00 00
0283: 04 00 00 00
```

Los campos siguientes indican que **HGW** es 8, **HGH** es 9, **HGX** es 0, **HGY** es 0, **HRX** es 1024 y **HRY** es 0.

d) Los cuatro planos de bits:

```
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
```

La decodificación de cuatro planos de bits de 8 por 9 a partir de estos datos genera los cuatro planos de bits que se muestran en la figura H.9. Al igual que en el segmento número 6, la decodificación Gray de estos planos, combinándolos en una formación de valores, y el dibujo de los patrones del diccionario de patrones de acuerdo con esa formación y los parámetros de la cuadrícula de semitonos da como resultado el mapa de bits de región que se muestra en la figura H.11.

30) El encabezamiento del decimoquinto segmento:

```
029F: 00 00 00 0E 31 00 02 00 00 00 00
```

Este segmento tiene un número de segmento de 14, un tipo de "Fin de página" (tipo 49), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y no está retenido. Está asociado con la página 2 y tiene una longitud de datos de 0 bytes.

31) El mapa de bits de página se forma combinando los tres mapas de bits de región del mismo modo en que fueron combinados en la página 1, dando como resultado el mismo mapa de bits de página.

32) El encabezamiento del decimosexto segmento:

```
02AA: 00 00 00 0F 30 00 03 00 00 00 13
```

Este segmento tiene un número de segmento de 15, un tipo de "Información de página" (tipo 48), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y no está retenido. Está asociado con la página 3 y tiene una longitud de datos de 19 bytes.

33) La parte datos del decimosexto segmento:

```
02B5: 00 00 00 25 00 00 00 08 00 00 00 00 00 00 00 00
02C5: 01 00 00
```

Esto indica que la página tiene 37 píxels de ancho, 8 píxels de alto y resolución de X e Y desconocida, es eventualmente sin pérdidas, no contiene ningún refinamiento, tiene un valor de píxel por defecto de **0**, un operador de combinación por defecto OR, no requiere ninguna memoria intermedia auxiliar y utiliza el operador de combinación por defecto de página en todas las regiones de una página.

34) El encabezamiento del decimoséptimo segmento:

```
02C8: 00 00 00 10 00 01 00 00 00 00 16
```

Este segmento tiene un número de segmento de 16, un tipo de "Diccionario de símbolos" (tipo 0), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y está retenido. No está asociado con ninguna página y tiene una longitud de datos de 22 bytes.

35) La parte datos del decimoséptimo segmento:

```
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) El campo de banderas de diccionario de símbolos de dos bytes:

```
02D3: 08 00
```

Este campo indica que el segmento se ha codificado utilizando la variante de codificación aritmética y no utiliza codificación de refinamiento/agregación. **SDTEMPLATE** tiene el valor 2. Los bits "contexto de codificación de mapa de bits utilizado" y "contexto de codificación de mapa de bits retenido" son ambos **0**.

b) El campo de banderas AT de diccionario de símbolos:

```
02D5: 02 FF
```

Este campo indica que $\mathbf{SDATX_1}$ es 2 y $\mathbf{SDATY_1}$ es –1; así pues, el píxel AT A_1 está en la ubicación (2, –1), que es el valor nominal para la plantilla 2.

c) El campo **SDNUMEXSYMS** de cuatro bytes:

```
02D7: 00 00 00 01
```

Este campo indica que **SDNUMEXSYMS** es 1: un símbolo es exportado por este diccionario de símbolos.

d) El campo **SDNUMNEWSYMS** de cuatro bytes:

```
02DB: 00 00 00 01
```

Este campo indica que **SDNUMNEWSYMS** es 1: un símbolo es definido por este diccionario de símbolos.

e) Los datos de diccionario de símbolos codificados:

```
02DF: 4F E7 8D 68 1B 14 2F 3F FF AC
```

Esto se decodifica como sigue:

i) Reponer a cero todas las estadísticas de codificación aritmética.

ii) Utilizando el procedimiento de decodificación aritmética de enteros IADH, decodificar un valor de delta altura de clase de altura. El valor decodificado es 6, lo que indica que la primera clase de altura es de 6 píxels de alto.

iii) Utilizando el procedimiento de decodificación aritmética de enteros IADW, decodificar un valor de delta anchura. El valor decodificado es 6. El primer símbolo tiene por tanto una anchura de 6 píxels.

iv) Utilizando el procedimiento de decodificación de región genérica, con **GBTEMPLATE** y el píxel AT A_1 fijados como se describe en el encabezamiento de datos de diccionario de símbolos, decodificar un mapa de bits de 6×6. Así se genera el mapa de bits mostrado en la figura H.12 a).

v) Utilizando el procedimiento de decodificación aritmética de enteros IADW, decodificar un valor de anchura delta. El valor decodificado es OOB, indicando el fin de la clase de altura.

vi) Puesto que **SDNUMNEWSYMS** es 1, el último símbolo ha sido ahora decodificado.

vii) Utilizando el procedimiento de decodificación aritmética de enteros IAEX, decodificar una longitud de recorrido de exportación. El valor decodificado es 0, indicando que los primeros símbolos 0 no son exportados.

viii) Utilizando el procedimiento de decodificación aritmética de enteros IAEX, decodificar una longitud de recorrido de exportación. El valor decodificado es 1, indicando que los siguientes símbolos 1 son exportados. Este diccionario de símbolos define por tanto un símbolo, que es exportado.

Figura H.12 – Símbolos en los diccionarios de símbolos de la tercera página

36) El encabezamiento del decimoctavo segmento:

```
02E9: 00 00 00 11 00 21 10 03 00 00 00 20
```

Este segmento tiene un número de segmentos de 17, un tipo de "Diccionario de símbolos" (tipo 0), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. Hace referencia a otro segmento, el segmento 16. El segmento 16 no está retenido, pero este segmento sí lo está. Está asociado con la página 3 y tiene una longitud de datos de 32 bytes.

37) La parte datos del decimoctavo segmento:

```
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) El campo de banderas de diccionario de símbolos de 2 bytes:

```
02F5: 08 02
```

Este campo indica que el segmento se ha codificado utilizando la variante de codificación aritmética y utiliza codificación de refinamiento/agregación. **SDTEMPLATE** tiene el valor 2. **SDRTEMPLATE** tiene el valor 0. Los bits "contexto de codificación de mapa de bits utilizado" y "contexto de codificación de mapa de bits retenido" son ambos **0**.

b) El campo de banderas AT de diccionario de símbolos:

```
02F7: 02 FF
```

Este campo indica que $\mathbf{SDATX_1}$ es 2 y $\mathbf{SDATY_1}$ es –1; así pues, el píxel AT A_1 está en la ubicación (2, –1), que es el valor nominal para la plantilla 2.

c) Las banderas AT de refinamiento de diccionario de símbolos:

```
02F9: FF FF FF FF
```

Este campo indica que $\mathbf{SDRATX_1}$ es –1, $\mathbf{SDRATY_1}$ es –1, $\mathbf{SDRATX_2}$ es –1 y $\mathbf{SDRATY_2}$ es –1. Así pues, el píxel AT RA_1 está en la ubicación (–1, –1) y el píxel AT RA_2 está en la ubicación (–1, –1), que son las ubicaciones nominales para la plantilla de refinamiento 0.

d) El campo **SDNUMEXSYMS** de cuatro bytes:

```
02FD: 00 00 00 03
```

Este campo indica que **SDNUMEXSYMS** es 3: tres símbolos son exportados por este diccionario de símbolos.

e) El campo **SDNUMNEWSYMS** de cuatro bytes:

```
0301: 00 00 00 02
```

Este campo indica que **SDNUMNEWSYMS** es 2: dos símbolos son definidos por este diccionario de símbolos.

f) Los datos del diccionario de símbolos codificados:

```
0305: 4F E9 D7 D5 90 C3 B5 26 A7 FB 6D 14 98 3F FF AC
```

Esto se decodifica como sigue:

i) Reponer a cero todas las estadísticas de codificación aritmética.

ii) Utilizando el procedimiento de decodificación aritmética de enteros IADH, decodificar un valor de altura delta de clase de altura. El valor decodificado es 6, indicando que la primera clase de altura es de 6 píxels de alto.

iii) Utilizando el procedimiento de decodificación aritmética de entero IADW, decodificar un valor de anchura delta. El valor decodificado es 6. El primer símbolo tiene por tanto una anchura de 6 píxels.

iv) Utilizando el procedimiento de decodificación aritmética de entero IAAI, decodificar una cuenta de ejemplares de símbolo en la agregación que forma el primer símbolo. El valor decodificado es 1.

v) Utilizando el procedimiento de decodificación aritmética de enteros IAID, decodificar un ID símbolo. El valor decodificado es 0. El primer símbolo es por tanto el identificado por ID símbolo 0, que es el primer (y único) símbolo en el diccionario (segmento 16) referenciado por este segmento.

vi) Utilizando el procedimiento de decodificación aritmética de enteros IARDX, decodificar un valor de X delta de mejora de ejemplar de símbolo. El valor decodificado es 0.

vii) Utilizando el procedimiento de decodificación aritmética de enteros IARDY, decodificar un valor de Y delta de refinamiento de ejemplar de símbolo. El valor decodificado es 0. La mejora se efectúa por tanto con el símbolo refinado alineado con el símbolo de referencia (es decir, **GRREFERENCEDX** y **GRREFERENCEDY** son ambos 0).

viii) Utilizando el procedimiento de decodificación de región de refinamiento genérica, decodificar el mapa de bits de ejemplar de símbolo refinado. El mapa de bits de referencia es el mapa de bits mostrado en la figura H.12 a) y el mapa de bits refinado, que es el mapa de bits del primer símbolo definido en este diccionario de símbolos, es el mapa de bits mostrado en la figura H.12 b).

ix) Utilizando el procedimiento de decodificación aritmética de enteros IADW, decodificar un valor de anchura delta. El valor decodificado es 8. El segundo símbolo tiene por tanto una anchura de 14 píxels.

x) Utilizando el procedimiento de decodificación aritmética de enteros IAAI, decodificar una cuenta de ejemplares de símbolo en la agregación que forma el segundo símbolo. El valor decodificado es 2.

xi) Utilizando el procedimiento de decodificación de región de texto, con los parámetros fijados como se describe en 6.5.8.2, decodificar una región de texto de 14 × 6 píxels.

- Utilizando el procedimiento de decodificación aritmética de enteros IADT, decodificar el valor inicial de SPRIPT. El valor decodificado es 0.
- Utilizando el procedimiento de decodificación aritmética de enteros IADT, decodificar un valor de T delta. El valor decodificado es 0.
- Utilizando el procedimiento de decodificación aritmética de enteros IAFS, decodificar un primer valor de S. El valor decodificado es 0. La esquina de referencia (esquina superior izquierda en este caso) del primer ejemplar de símbolo en la agregación está por tanto en (0, 0).
- Utilizando el procedimiento de decodificación aritmética de enteros IAID, decodificar un ID símbolo. El valor decodificado es 0. El primer símbolo es por tanto el identificado por ID símbolo 0, que es el primer (y único) símbolo en el diccionario (segmento 16) referenciado por este segmento.
- Utilizando el procedimiento de decodificación aritmética de enteros IARI, decodificar una bandera de refinamiento. El valor decodificado es **0**, indicando que el primer ejemplar de símbolo no es refinado.
- En este punto CURS es 5. Utilizando el procedimiento de decodificación aritmética de entero IADS, decodificar un valor de S delta. El valor decodificado es 3. La esquina de referencia del segundo ejemplar de símbolo en la agregación está por tanto en (8, 0).
- Utilizando el procedimiento de decodificación aritmética de enteros IAID, decodificar un ID símbolo. El valor decodificado es 1. El segundo símbolo es por tanto el identificado por ID símbolo 1, que es el primer (y hasta ahora único) símbolo definido en este diccionario de símbolos.
- Utilizando el procedimiento de decodificación aritmética de enteros IARI, decodificar una bandera de refinamiento. El valor decodificado es **0**, indicando que el primer ejemplar de símbolo no es refinado.
- Utilizando el procedimiento de decodificación aritmética de enteros IADS, decodificar un valor de S delta. El valor decodificado es OOB, indicando el fin de la franja.
- La decodificación de la región de texto está ahora completa. El mapa de bits de la región de texto, que es el mapa de bits del segundo símbolo definido en el diccionario de símbolos, se muestra en la figura H.12 c). Se obtiene dibujando la figura H.12 a) en (0, 0) y la figura H.12 b) en (8, 0).

xii) Utilizando el procedimiento de decodificación aritmética de enteros IADW, decodificar un valor de anchura delta. El valor decodificado es OOB, indicando el fin de la clase de altura.

xiii) Puesto que **SDNUMNEWSYMS** es 2, el último símbolo ha sido ahora decodificado.

xiv) Utilizando el procedimiento de decodificación aritmética de enteros IAEX, decodificar una longitud de recorrido de exportación. El valor decodificado es 0, lo que indica que los primeros símbolos 0 no son exportados.

xv) Utilizando el procedimiento de decodificación aritmética de enteros IAEX, decodificar una longitud de recorrido de exportación. El valor decodificado es 3, indicando que los 3 símbolos siguientes son exportados. Así pues, este diccionario de símbolos importa un símbolo y define dos símbolos, y los tres son exportados.

Se señala que el símbolo del segmento 16 es reexportado por este diccionario. Los tres símbolos exportados por este diccionario son por consiguiente los tres símbolos mostrados en la figura H.12. Una región de texto puede utilizar por tanto el símbolo mostrado en la figura H.12 a) (definido al principio en el segmento 16) haciendo referencia al segmento 17, incluso aunque el segmento 16 no sea retenido tras el fin del segmento 17.

38) El encabezamiento del decimonoveno segmento:

```
0315: 00 00 00 12 07 20 11 03 00 00 00 25
```

Este segmento tiene un número de segmentos de 18, un tipo de "Región de texto sin pérdida inmediata" (tipo 7), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. Hace referencia a otro segmento, el segmento 17. El segmento 17 y este segmento no deberán ser retenidos. Está asociado con la página 3 y tiene una longitud de datos de 37 bytes.

39) La parte de datos del decimonoveno segmento:

```
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) El campo de información de segmento de región:

```
0321: 00 00 00 25 00 00 00 08 00 00 00 00 00 00 00 00
0331: 00
```

Esto indica que el mapa de bits de la región codificado por este segmento tiene 37 píxels de ancho y 8 píxels de alto y su esquina superior izquierda está 0 píxels a la derecha del borde izquierdo de la página y 0 píxels por debajo del borde superior de la página. Deberá dibujarse en la página utilizando OR.

b) El campo de banderas de segmento de región de texto de dos bytes:

```
0332: 8C 12
```

Este campo indica que el segmento se ha codificado utilizando la variante de codificación aritmética, contiene refinamientos, tiene un valor de **SBSTRIPS** de 1, tiene una esquina de referencia de TOPLEFT, no está transpuesto, combina sus símbolos utilizando OR, tiene un valor de píxel por defecto de **0**, tiene un valor **SBDSOFFSET** de 3 y tiene un valor **SBRTEMPLATE** de 1.

c) El campo **SBNUMINSTANCES** de 4 bytes:

```
0334: 00 00 00 04
```

Este campo indica que **SBNUMINSTANCES** es 4: ejemplares instancias de símbolos están codificadas en esta región de texto.

d) Los datos de región de texto codificados:

```
0338: A9 5C 8B F4 C3 7D 96 6A 28 E5 76 8F FF AC
```

Esto se decodifica como sigue:

i) Reponer a cero todas las estadísticas de codificación aritmética.

ii) Utilizando el procedimiento de decodificación aritmética de enteros IADT, decodificar el valor STRIPT inicial. El valor decodificado es 0.

iii) Utilizando el procedimiento de decodificación aritmética de enteros IADT, decodificar un valor de T delta. El valor decodificado es 0.

iv) Utilizando el procedimiento de decodificación aritmética de enteros IAFS, decodificar un primer valor de S. El valor decodificado es 0. La esquina de referencia (esquina superior izquierda en este caso) del primer ejemplar de símbolo en la región de texto está por tanto en (0, 0).

v) Utilizando el procedimiento de decodificación aritmética de enteros IAID, decodificar un ID símbolo. El valor decodificado es 1. El primer ejemplar de símbolo utiliza por tanto el símbolo mostrado en la figura H.12 b).

vi) Utilizando el procedimiento de decodificación aritmética de enteros IARI, decodificar una bandera de refinamiento. El valor decodificado es **0**, indicando que el primer ejemplar de símbolo no es refinado.

vii) En este punto CURS es 5. Utilizando el procedimiento de decodificación aritmética de enteros IADS, decodificar un valor de S delta. El valor decodificado es 0. Tras añadir en **SBDSOFFSET**, la esquina de referencia del segundo ejemplar de símbolo en la agregación está por tanto en (8, 0).

viii) Utilizando el procedimiento de decodificación aritmética de enteros IAID, decodificar un ID símbolo. El valor decodificado es 0. El segundo ejemplar de símbolo utiliza por tanto el símbolo mostrado en la figura H.12 a).

ix) Utilizando el procedimiento de decodificación aritmética de enteros IARI, decodificar una bandera de refinamiento. El valor decodificado es **0**, indicando que el primer ejemplar de símbolo no es refinado.

x) En este punto CURS es 13. Utilizando el procedimiento de decodificación aritmética de enteros IADS, decodificar un valor de S delta. El valor decodificado es 0. Tras añadir en **SBDSOFFSET**, la esquina de referencia del tercer ejemplar de símbolo en la agregación está por tanto en (16, 0).

xi) Utilizando el procedimiento de decodificación aritmética de enteros IAID, decodificar un ID símbolo. El valor decodificado es 1. El tercer ejemplar de símbolo utiliza por tanto el símbolo mostrado en la figura H.12 b).

xii) Utilizando el procedimiento de decodificación aritmética de enteros IARI, decodificar una bandera de refinamiento. El valor decodificado es 1, indicando que el tercer ejemplar de símbolo está refinado.

xiii) Utilizando el procedimiento de decodificación aritmética de enteros IARDW, decodificar un valor de anchura delta de mejora de ejemplar de símbolo. El valor decodificado es –1.

xiv) Utilizando el procedimiento de decodificación aritmética de enteros IARDH, decodificar un valor de altura delta de mejora de ejemplar de símbolo. El valor decodificado es 2. Puesto que el símbolo de referencia es de 6 por 6 píxels, el ejemplar de símbolo refinado es de 5×8 píxels.

xv) Utilizando el procedimiento de decodificación aritmética de enteros IARDX, decodificar un valor de X delta de refinamiento de ejemplar de símbolo. El valor decodificado es 1.

xvi) Utilizando el procedimiento de decodificación aritmética de enteros IARDY, decodificar un valor de Y delta de mejora de ejemplar de símbolo. El valor decodificado es –2. **GRREFERENCEDX** y **GRREFERENCEDY** se fijan por tanto como:

$$\mathbf{GRREFERENCEDX} = \lfloor -1/2 \rfloor + 1 = -1 + 1 = 0$$

$$\mathbf{GRREFERENCEDY} = \lfloor 2/2 \rfloor - 2 = 1 - 2 = -1$$

con lo que la mejora se hace con el símbolo refinado situado de manera que el píxel superior izquierdo del símbolo refinado esté alineado con el píxel (0, 1) del símbolo de referencia.

xvii) Utilizando el procedimiento de decodificación de región de refinamiento genérica, decodificar el mapa de bits del ejemplar de símbolo refinado. El mapa de bits de referencia es el mapa de bits mostrado en la figura H.12 b) y el mapa de bits mejorado, que es el mapa de bits para el tercer ejemplar de símbolo, es el mapa de bits mostrado en la figura H.2.

xviii) En este punto CURS es 18. Utilizando el procedimiento de decodificación aritmética de enteros IADS, decodificar un valor de S delta. El valor decodificado es 0. Tras añadir en **SBDSOFFSET**, la esquina de referencia del tercer ejemplar de símbolo en la agregación está por tanto en (23, 0).

xix) Utilizando el procedimiento de decodificación aritmética de enteros IAID, decodificar un ID símbolo. El valor decodificado es 2. El cuarto ejemplar de símbolo utiliza por tanto el símbolo mostrado en la figura H.12 c).

xx) Utilizando el procedimiento de decodificación aritmética de enteros IADS, decodificar un valor de S delta. El valor decodificado es OOB, indicando el fin de la franja.

xxi) La decodificación de la región de texto está ahora completa. El mapa de bits de la región de texto es el mapa de bits mostrado en la figura H.5. Se obtiene dibujando la figura H.12 b) con su esquina superior izquierda en (0, 0), la figura H.12 a) con su esquina superior izquierda en (8, 0), la figura H.2 con su esquina superior izquierda en (16, 0) y la figura H.12 c) con su esquina superior izquierda en (23, 0).

40) El encabezamiento del vigésimo segmento:

```
0346: 00 00 00 13 31 00 03 00 00 00 00
```

Este segmento tiene un número de segmento de 19, un tipo de "Fin de página" (tipo 49), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y no está retenido. Está asociado con la página 3 y tiene una longitud de datos de 0 bytes.

41) El mapa de bits de página se forma a partir del único segmento de región de la página, el segmento número 18, y es igual al mapa de bits de ese segmento de región (figura H.5).

42) El encabezamiento del vigésimo primer segmento:

```
0351: 00 00 00 14 33 00 00 00 00 00 00
```

Este segmento tiene un número de segmento de 20, un tipo de "Fin de ficheros" (tipo 51), un campo de asociación de página corto y no tiene fijado el bit de no retención diferida. No hace referencia a ningún otro segmento y no está retenido. No está asociado con ninguna página y tiene una longitud de datos de 0 bytes.

H.2 Secuencia de prueba para codificador aritmético

En esta subcláusula se da un pequeño conjunto de datos para la prueba del codificador y el decodificador aritméticos. La prueba está estructurada de modo que puedan probarse muchos trayectos de codificador y decodificador, pero es imposible verificarlos todos ellos en una secuencia de prueba corta, por lo que la concordancia con los resultados de esta prueba no garantiza, desgraciadamente, una implementación totalmente correcta.

Las decisiones que se han de codificar, empaquetadas en 32 bytes y en notación hexadecimal, son:

```
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
```

Los datos codificados que deberán obtenerse codificando esa secuencia, mostrados como 30 bytes hexadecimales, son:

```
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
```

La decodificación de estos 30 bytes deberá producir los 32 bytes originales.

El cuadro H.1 expone, bit por bit, el funcionamiento del codificador y el decodificador aritméticos. La primera línea del cuadro corresponde a las operaciones INITENC e INITDEC. El valor del byte antes del primer byte en la memoria intermedia de salida se supone que es `0x00`, con lo que el valor inicial de B es `0x00`. La última línea del cuadro corresponde a la operación FLUSH.

Se utiliza un solo valor de CX para toda la prueba. I(CX) es inicialmente 0 y MPS(CX) es inicialmente **0**.

La primera columna es el contador de eventos EC. La segunda columna es la de decisión D que se ha de codificar. Las columnas tercera y cuarta dan los valores de I(CX) y MPS(CX). La quinta columna muestra una "X" que indica que el intercambio condicional se producirá cuando se codifique (decodifique) la decisión en curso. La sexta columna muestra el valor de Qe correspondiente a I(CX) (véase el cuadro E.1). La séptima columna muestra el valor del registro A antes de que se codifique (decodifique) la decisión. Se señala que el registro A es siempre mayor o igual que `0x8000`.

Las variables hasta este punto son comunes para el codificador y el decodificador. Las cuatro columnas que siguen (C, CT, B, OUT) son sólo para el codificador. Las cuatro columnas que figuran a continuación (C, CT, B, IN) son sólo para el decodificador. La columna final (C) muestra el registro C para el decodificador con convenios de soporte lógico, ya que es el único valor que difiere para dicho decodificador. Todos los valores mostrados para el registro C se dan antes de que se codifique (decodifique) la decisión en curso.

Para el codificador, CT es un contador que indica cuándo está un byte listo para salir del registro C. La columna situada debajo de B muestra el byte en B variable en espera de ser enviado. Este byte puede algunas veces cambiar a causa de un arrastre. Por último, los bytes comprimidos para el codificador se indican en la columna situada debajo de OUT. Se considera que un byte es "salida" cuando el puntero BP de datos comprimidos avanza apuntando más allá de ese byte.

El contador CT del decodificador indica cuándo hay que introducir el byte siguiente de los datos comprimidos. La columna debajo de B muestra el valor del registro B, que se utiliza para determinar cuándo se ha producido un relleno de bits. La columna debajo de IN muestra los bytes que se han consumido. Se considera que un byte se ha "consumido" cuando el puntero BP de datos comprimidos avanza apuntando a ese byte. Se señala que el byte final `0xAC` nunca se consume, de acuerdo con esta definición, aunque se lee como parte de BYTEIN.

Cuadro H.1 – Datos de seguimiento del codificador y el decodificador

Común							Codificador				Decodificador				Decodificador con convenios de soporte lógico
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

Cuadro H.1 – Datos de seguimiento del codificador y el decodificador *(cont.)*

Común							Codificador				Decodificador				Decodificador con convenios de soporte lógico
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

Cuadro H.1 – Datos de seguimiento del codificador y el decodificador *(cont.)*

Común							Codificador				Decodificador				Decodificador con convenios de soporte lógico
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

Cuadro H.1 – Datos de seguimiento del codificador y el decodificador *(fin)*

Común							Codificador				Decodificador				Decodificador con convenios de soporte lógico
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					

Bibliografía

[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, noviembre de 1974.

[2] BILGIN (A.), SEMENTILLI (P.), MARCELLIN (M.): Progressive image coding using trellis coded quantization, *IEEE Transactions on Image Processing*, noviembre de 1997. (presentada).

[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, febrero-abril de 1994.

[5] HOWARD (P. G.): Lossless and lossy compression of text images by soft pattern matching, in *Proc. of Data Compression Conference*, páginas 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, noviembre de 1998.

[8] HUNTER (R.), ROBINSON (A. H.): International digital facsimile coding standards, in *Proceedings of IEEE*, Volume 68, pages 854-867, julio de 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, abril de 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, páginas 38-49, 1997.

[12] MARTINS (B.), FORCHHAMMER (S.): Halftone Coding with JBIG2, submitted to Journal of Electronic Imaging, septiembre de 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, mayo de 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, páginas 786-796, julio de 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, junio de 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.

SERIES DE RECOMENDACIONES DEL UIT-T

Serie A	Organización del trabajo del UIT-T
Serie B	Medios de expresión: definiciones, símbolos, clasificación
Serie C	Estadísticas generales de telecomunicaciones
Serie D	Principios generales de tarificación
Serie E	Explotación general de la red, servicio telefónico, explotación del servicio y factores humanos
Serie F	Servicios de telecomunicación no telefónicos
Serie G	Sistemas y medios de transmisión, sistemas y redes digitales
Serie H	Sistemas audiovisuales y multimedios
Serie I	Red digital de servicios integrados
Serie J	Redes de cable y transmisión de programas radiofónicos y televisivos, y de otras señales multimedios
Serie K	Protección contra las interferencias
Serie L	Construcción, instalación y protección de los cables y otros elementos de planta exterior
Serie M	RGT y mantenimiento de redes: sistemas de transmisión, circuitos telefónicos, telegrafía, facsímil y circuitos arrendados internacionales
Serie N	Mantenimiento: circuitos internacionales para transmisiones radiofónicas y de televisión
Serie O	Especificaciones de los aparatos de medida
Serie P	Calidad de transmisión telefónica, instalaciones telefónicas y redes locales
Serie Q	Conmutación y señalización
Serie R	Transmisión telegráfica
Serie S	Equipos terminales para servicios de telegrafía
Serie T	**Terminales para servicios de telemática**
Serie U	Conmutación telegráfica
Serie V	Comunicación de datos por la red telefónica
Serie X	Redes de datos y comunicación entre sistemas abiertos
Serie Y	Infraestructura mundial de la información y aspectos del protocolo Internet
Serie Z	Lenguajes y aspectos generales de soporte lógico para sistemas de telecomunicación

Ginebra, 2002