# INTERNATIONAL STANDARD

ISO/IEC
**11544**

First edition
1993-12-15

# Information technology — Coded representation of picture and audio information — Progressive bi-level image compression

*Technologies de l'information — Représentation codée de l'image et du son — Compression de l'image progressive à deux niveaux*

# Contents

Page

**Annexes**

# Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 11544 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology,* Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information,* in collaboration with CCITT. The identical text is published as CCITT Recommendation T.82.

Annexes A, B, C, D, E and F of this International Standard are for information only.

# Patents

During the preparation of this International Standard, information was gathered on patents upon which application of the standard might depend. Relevant patents were identified as belonging to the patent holders listed in annex E. However, ISO/IEC cannot give authoritative or comprehensive information about evidence, validity or scope of patent and like rights. The patent holders have stated that licences will be granted under reasonable terms. Communications on this subject should be addressed to the patent holders (see annex E).

## Introduction

This Recommendation | International Standard was prepared by the Joint Bi-level Image experts Group (JBIG) of ISO/IEC JTC1/SC29/WG9 and CCITT SGVIII. The JBIG experts group was formed in 1988 to establish a standard for the progressive encoding of bi-level images.

A progressive encoding system transmits a compressed image by first sending the compressed data for a reduced-resolution version of the image and then enhancing it as needed by transmitting additional compressed data, which builds on that already transmitted. This Recommendation | International Standard defines a coding method having progressive, progressive-compatible sequential, and single-progression sequential modes and suggests a method to obtain any needed low-resolution renditions. It has been found possible to effectively use the defined coding and resolution-reduction algorithms for the lossless coding of greyscale and color images as well as bi-level images.

## 0.1 General characteristics

This Specification defines a method for lossless compression encoding of a bi-level image (that is, an image that, like a black-and-white image, has only two colors). The defined method can also be used for coding greyscale and color images. Being adaptive to image characteristics, it is robust over image type. On scanned images of printed characters, observed compression ratios have been from 1,1 to 1,5 times as great as those achieved by the MMR encoding algorithm (which is less complex) described in Recommendations T.4 (G3) and T.6 (G4). On computer generated images of printed characters, observed compression ratios have been as much as 5 times as great. On images with greyscale rendered by halftoning or dithering, observed compression ratios have been from 2 to 30 times as great.

The method is bit-preserving, which means that it, like Recommendations T.4 and T.6, is distortionless and that the final decoded image is identical to the original.

The method also has "progressive" capability. When decoding a progressively coded image, a low-resolution rendition of the original image is made available first with subsequent doublings of resolution as more data is decoded. Note that resolution reduction is performed from the higher to lower resolution layers, while decoding is performed from the lower to higher resolution layers. The lowest resolution image sent in a progressive sequence is a sequentially coded image. In a single-progression sequential coding application, this is the only image sent.

Progressive encodings have two distinct benefits. One is that with them it is possible to design an application with one common database that can efficiently serve output devices with widely different resolution capabilities. Only that portion of the compressed image file required for reconstruction to the resolution capability of the particular output device has to be sent and decoded. Also, if additional resolution enhancement is desired, for say, a paper copy of an image already on a CRT screen, only the needed resolution-enhancing information has to be sent.

The other benefit of progressive encodings is that they can provide subjectively superior image browsing (on a CRT) for an application using low-rate and medium-rate communication links. A low-resolution rendition is transmitted and displayed rapidly, and then followed by as much resolution enhancement as desired. Each stage of resolution enhancement builds on the image already available. Progressive encoding can make it easier for a user to quickly recognize the image as it is being built up, which in turn allows the user to interrupt the transmission of the image.

Let $D$ denote the number of doublings in resolution (called differential layers) provided by the progressive coding. Let $I_D$ denote the highest resolution image and let its horizontal and vertical dimensions in pixels be $X_D$ and $Y_D$. Let $R_D$ denote the sampling resolution of the image $I_D$.

This Specification imposes almost no restrictions on the parameters $R_D$, $X_D$, $Y_D$, or $D$. Choices such as 400 or 200 dpi (dots-per-inch) for the resolution $R_D$ of the highest resolution layer result in a hierarchy of resolutions commensurate with current facsimile standards. Choosing $R_D$ as 600 or 300 dpi gives a progressive hierarchy more compatible with the laser printer resolutions available as of the writing of this Specification.

It is anticipated that $D$ will typically be chosen so that the lowest resolution is roughly 10 to 25 dpi. Typical bi-level images when reduced to such a resolution are not legible, but nonetheless such low-resolution renditions are still quite useful and function as automatically generated icons. Page layout is usually apparent and recognition of particular pages that have been seen before at higher resolution is often possible.

As mentioned above, this Specification does not restrict the number $D$ of resolution doublings. It can be set to 0 if progressive coding is of no utility, as is the case, for example, in hardcopy facsimile. Doing so retains JBIG's compression advantage over MMR (and in fact usually increases it somewhat), while eliminating the need for any buffering and simplifying the algorithm. Single-progression sequential JBIG coding has potential applications identical to those of MMR coding. Images compressed by a single-progression sequential encoder will be readable by decoders capable of progressive decoding, although only the lowest resolution version of a progressively encoded image will be decodable by a single-progression sequential decoder.

It is possible to use this Specification for the lossless coding of greyscale and color images by coding bit-planes independently as though each were itself a bi-level image. This approach to the coding of greyscale and color images can be used as an alternative to the photographic encoding specification CCITT Rec. T.81 | ISO/IEC 10918-1 (JPEG) in its lossless mode. Preliminary experimental results have shown that JBIG has a compression advantage over JPEG in its lossless mode for greyscale images up to 6 bits-per-pixel. For 6 to 8 bits-per-pixel the compression results have been similar for both JBIG and JPEG. This Specification makes provision for images with more than one bit plane, but makes no recommendation on how to map greyscale or color intensities to bit-planes. Experimentally, it has been found that for greyscale images a mapping via Gray-coding of intensity is superior to a mapping via simple weighted-binary coding of intensity.

## 0.2 Stripes and data ordering

When it is necessary to distinguish progressive coding from the more traditional form of image coding in which the image is coded at full resolution from left to right and top to bottom, this older form of coding will be referred to as "sequential". The advantage of sequential coding over progressive coding is that no page (frame) buffer is required. Progressive coding does require a page buffer at the next-to-highest resolution because lower resolution images are used in coding higher resolution images.

It is possible to create a JBIG datastream with only a lowest resolution layer and this can be named single-progression sequential coding. In such coding, a full-resolution image is coded without reference to any differential resolution layers. The parameters $D$ (mentioned in 0.1) is set equal to zero. It should be noted that in a progressive encoding of an image, the lowest resolution layer is actually encoded in single-progression sequential coding. If a full-resolution image is encoded using single-progression sequential coding, it will not be possible to decode the image progressively.

Coding in the progressive-compatible sequential mode is said to be "compatible" with coding in the progressive mode because the datastreams created (encoder) or read (decoder) in either mode carry exactly the same information. All that changes with a switch from progressive to progressive-compatible sequential encoding is the order in which parts of the compressed data are created by the encoder. All that changes with a switch from progressive to progressive-compatible sequential decoding is the order in which these parts are used by the decoder.

This compatibility is achieved by breaking an image into smaller parts before compression. These parts are created by dividing the image in each of its resolution "layers" into horizontal bands called "stripes." Progressive-compatible sequential coding does require a "stripe" buffer (much smaller than a page buffer) and additional individual "state" memory used for adaptive entropy coding of each resolution layer and bit plane.

Figure Intro. 1 shows such a decomposition when there are three resolution layers, three stripes per layer, and only one bit plane. Table Intro. 1 shows defined ways to sequence through the nine stripes.

Notice that in addition to the progressive-versus-sequential distinction that is carried by the **SEQ** bit, there is also a resolution-order distinction that is carried by the **HITOLO** bit. Encoders work from high resolution downward and so most naturally encode the stripes in **HITOLO** order. Decoders must build up the image from low resolution and so most naturally process stripes in the opposite order. When an application uses an encoder that sends progressively coded data directly to a decoder, one or the other must buffer to invert the order. When an application includes a database, the database (with appropriate set-up) can be used to buffer and invert the order (including setting **HITOLO** correctly) thereby removing this requirement from the encoder and decoder.

A stripe has a vertical size that is typically much smaller than that of the entire image. The number $L_0$ of lines per stripe at the lowest layer is another free parameter. As an example, $L_0$ might be chosen so that a stripe is about 8 mm. If such a choice is made, the number $S$ of stripes in an image of a business-letter-sized sheet of paper will be about 35.
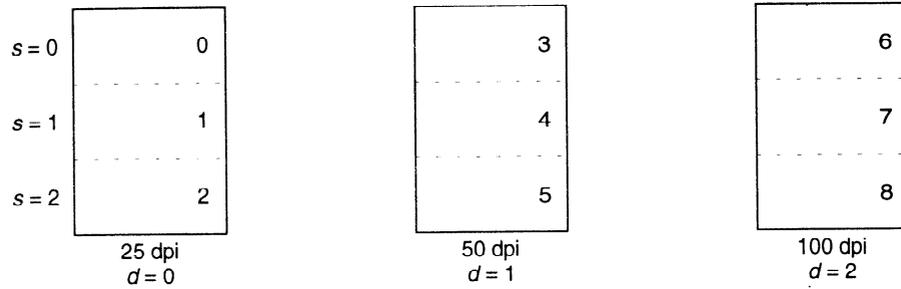
**Figure 0.1 – Decomposition in the special case of 3 layers, 3 stripes, and 1 bit plane**

**Table 0.1 – Possible bi-level data orderings**

| HITOLO | SEQ | Example order |
|--------|-----|---------------|
| 0 | 0 | 0,1,2  3,4,5  6,7,8 |
| 0 | 1 | 0,3,6  1,4,7  2,5,8 |
| 1 | 0 | 6,7,8  3,4,5  0,1,2 |
| 1 | 1 | 6,3,0  7,4,1  8,5,2 |

When there is more than one bit plane, as in Figure 0.2, there are twelve defined stripe orderings. Table 0.2 lists them. As before, the **HITOLO** bit carries the resolution-order distinction, and the **SEQ** bit carries the progressive-versus-sequential distinction. When the **ILEAVE** bit is 1, it indicates the interleaving of multiple bit planes. When the **SMID** bit is 1, it indicates $s$, the index over the stripe, is in the middle as shown more clearly in Table 11 of 6.2.4.
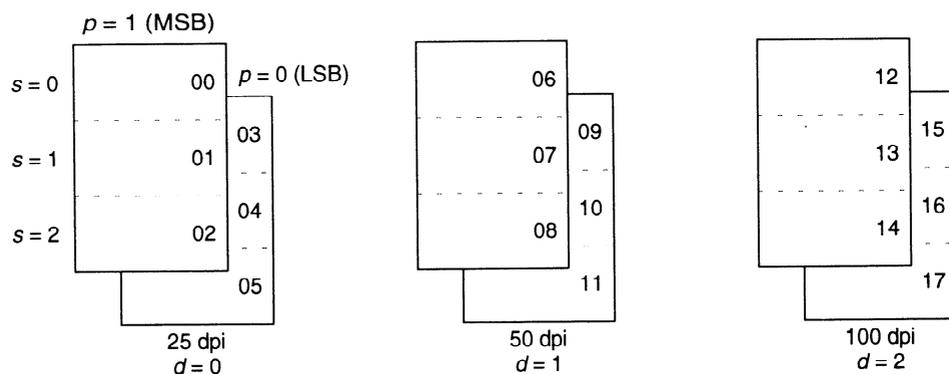


**Figure 0.2 – Decomposition in the special case of 3 layers, 3 stripes, and 2 bit planes**

**Table 0.2 – Possible multi-plane data orderings**

| HITOLO | SEQ | ILEAVE | SMID | Example order |
|--------|-----|--------|------|---------------|
| 0 | 0 | 0 | 0 | (00,01,02 06,07,08 12,13,14) (03,04,05 09,10,11 15,16,17) |
| 0 | 0 | 1 | 0 | (00,01,02 03,04,05) (06,07,08 09,10,11) (12,13,14 15,16,17) |
| 0 | 0 | 1 | 1 | (00,03 01,04 02,05) (06,09 07,10 08,11) (12,15 13,16 14,17) |
| 0 | 1 | 0 | 0 | (00,06,12 03,09,15) (01,07,13 04,10,16) (02,08,14 05,11,17) |
| 0 | 1 | 0 | 1 | (00,06,12 01,07,13 02,08,14) (03,09,15 04,10,16 05,11,17) |
| 0 | 1 | 1 | 0 | (00,03 06,09 12,15) (01,04 07,10 13,16) (02,05 08,11 14,17) |
| 1 | 0 | 0 | 0 | (12,13,14 06,07,08 00,01,02) (15,16,17 09,10,11 03,04,05) |
| 1 | 0 | 1 | 0 | (12,13,14 15,16,17) (06,07,08 09,10,11) (00,01,02 03,04,05) |
| 1 | 0 | 1 | 1 | (12,15 13,16 14,17) (06,09 07,10 08,11) (00,03 01,04 02,05) |
| 1 | 1 | 0 | 0 | (12,06,00 15,09,03) (13,07,01 16,10,04) (14,08,02 17,11,05) |
| 1 | 1 | 0 | 1 | (12,06,00 13,07,01 14,08,02) (15,09,03 16,10,04 17,11,05) |
| 1 | 1 | 1 | 0 | (12,15 06,09 00,03) (13,16 07,10 01,04) (14,17 08,11 02,05) |

The two new variables **ILEAVE** and **SMID** plus the two earlier variables **HITOLO** and **SEQ** make it possible to index all twelve of these orders. The other four of the sixteen possible combinations for these four binary variables have no stripe ordering associated with them. If there is only one plane, stripe order is not dependent on **ILEAVE** and **SMID** and their values are inconsequential.

The compressed data $C_{s, d, p}$ for stripe $s$ of resolution layer $d$ of bit-plane $p$ is independent of stripe ordering. All that changes as **HITOLO**, **SEQ**, **ILEAVE** and **SMID** vary is the order in which the data is concatenated onto a datastream. This is the compatibility feature noted earlier.

For simplicity, the remainder of this introduction will assume there is only one bit plane and the subscript $p$ denoting bit plane will be dropped from $C_{s, d, p}$.

## 0.3 Encoder functional blocks

An encoder can be decomposed as shown in Figure 0.3. (In single-progression sequential coding only the lowest-resolution-layer encoder would be used.)

Although conceptually there are $D$ algorithmically identical differential-layer encoders as shown in Figure 0.3, some implementations may choose to recursively use only one physical differential-layer encoder.
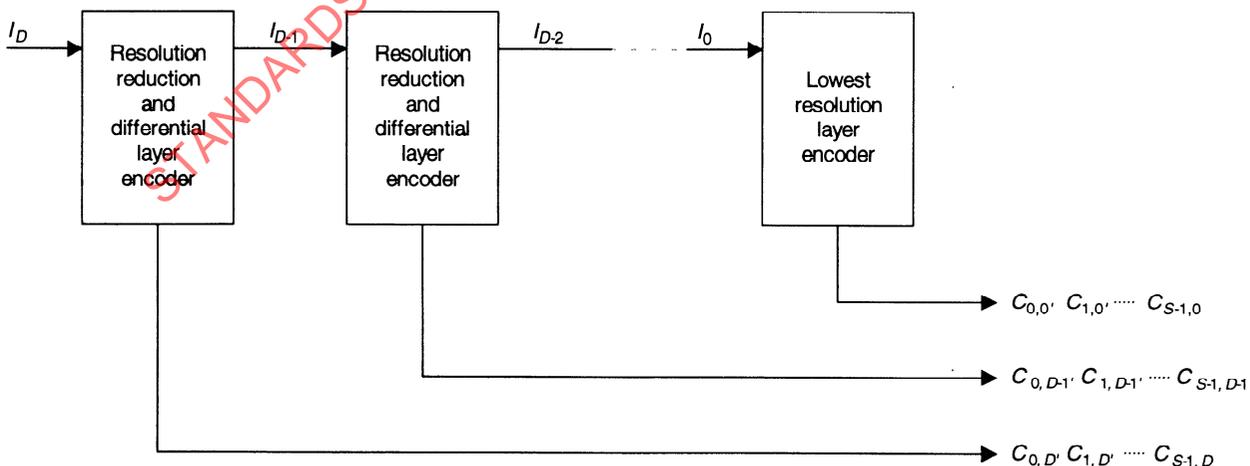


T0806280-90/D03

**Figure 0.3 – Decomposition of encoder**

## 0.3.1 Resolution reduction and differential layer encoder

Each of the resolution-reduction-and-differential-layer-encoder blocks of Figure 0.3 is identical in function, hence only a description of the operation at one layer is needed. For such a description there are only two resolution layers involved. For simplicity in the remainder of this subclause, the incoming image will be referred to as the "high-resolution" image and the outgoing image, as the "low-resolution" image. Note though that the "high" and "low" resolution images of any particular resolution-reduction-and-differential-layer-encoder block in Figure 0.3 are not in general the highest and lowest resolution images of the entire system.

A resolution-reduction-and-differential-layer-encoder block of Figure 0.3 can itself be decomposed into sub-blocks as shown in Figure 0.4. Not all sub-blocks need be used in all systems. Refer to the tables in clause 4 for a definition of signal names.



T0806290-90/D04
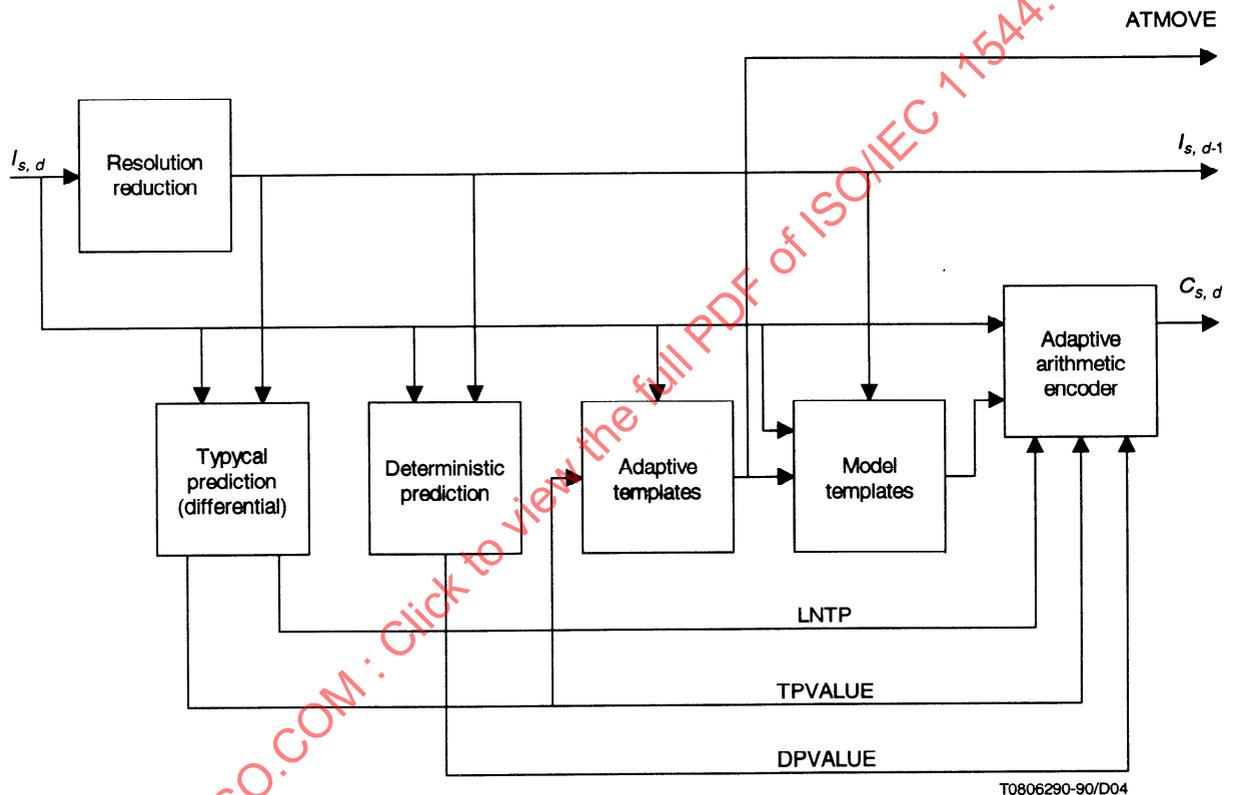
**Figure 0.4 – Resolution reduction and differential layer encoder**

Acronyms for the processing blocks of this figure and some others to be discussed in this introductory clause are given in Table 0.3.

**Table 0.3 – Acronyms for processing blocks**

| Acronym | Meaning |
|---------|---------|
| AAD | Adaptive Arithmetic Decoder |
| AAE | Adaptive Arithmetic Encoder |
| AT | Adaptive Templates |
| DP | Deterministic Prediction |
| MT | Model Templates |
| RR | Resolution Reduction |
| TPB | Typical Prediction (Bottom) |
| TPD | Typical Prediction (Differential) |

### 0.3.1.1 Resolution reduction

The resolution-reduction (RR) block performs resolution reduction. This block accepts a high-resolution image and creates a low-resolution image with, as nearly as possible, half as many rows and half as many columns as the original.

An obvious way to reduce the resolution of a given image by a factor of two in each dimension is to subsample it by taking every other row and every other column. Subsampling is simple, but creates images of poor subjective quality, especially when the input image is bi-level.

For bi-level images containing text and line drawings, subsampling is poor because it frequently deletes thin lines. For bi-level images that contain halftoning or ordered dithering to render greyscale, subsampling is poor because greyness is not well preserved, especially if the dithering period is a power of two, as is frequently the case.

This Specification suggests a resolution reduction method. This particular method has been carefully designed, extensively tested, and found to achieve excellent results for text, line art, dithered greyscale, halftoned greyscale, and error-diffused greyscale.

### 0.3.1.2 Differential layer typical prediction

The differential-layer typical prediction (TP) block provides some coding gain, but its primary purpose is to speed implementations. Differential-layer TP looks for regions of solid color and when it finds that a given current high-resolution pixel for coding is in such a region, none of the processing normally done in the DP, AT, MT, and AAE blocks is needed. On text or line-art images, differential-layer TP usually makes it possible to avoid coding over 95% of the pixels. On bi-level images rendering greyscale, processing savings are significantly smaller.

### 0.3.1.3 Deterministic prediction

The purpose of the deterministic-prediction (DP) block is to provide coding gain. On the set of test images used in the development of this Specification it provided a 7% gain, and such a gain is thought to be typical.

When images are reduced in resolution by a particular resolution reduction algorithm, it sometimes happens that the value of a particular current high-resolution pixel to be coded is inferable from the pixels already known to both the encoder and decoder, that is, all the pixels in the low-resolution image and those in the high-resolution image that are causally related (in a raster sense) to the current pixel. When this occurs, the current pixel is said to be deterministically predictable. The DP block flags any such pixels and inhibits their coding by the arithmetic coder.

DP is a table driven algorithm. The values of particular surrounding pixels in the low-resolution image and causal high-resolution image are used to index into a table to check for determinicity and, when it is present, obtain the deterministic prediction. DP tables are highly dependent on the particular resolution reduction method used. Provision is made for an encoder to download DP tables to a decoder if it is using a private resolution reduction algorithm. If an application requires default DP, decoders need to always have the default DP tables and no DP tables need be sent. Hence, if the suggested resolution reduction algorithm is used, no DP table need ever be sent.

### 0.3.1.4 Model templates

For each high-resolution pixel to be coded, the model-templates (MT) block provides the arithmetic coder with an integer called the context. This integer is determined by the colors (binary levels) of particular pixels in the causal high-resolution image, by particular pixels in the already available low-resolution image, and by the spatial phase of the pixel being coded. "Spatial phase" describes the orientation of the high-resolution pixel with respect to its corresponding low-resolution pixel.

The arithmetic coder maintains for each context an estimate of the conditional probability of the symbol given that context. The greatest coding gain is achieved when this probability estimate is both accurate and close to 0 or 1. Thus, good templates have good predictive value so that when the values of the pixels in the template are known, the value of the pixel to be coded is highly predictable.

### 0.3.1.5 Adaptive templates

The adaptive-templates (AT) block provides substantial coding gain (sometimes as much as 80%) on images rendering greyscale with halftoning. AT looks for periodicity in the image and on finding it changes the template so that the pixel preceding the current pixel by this periodicity is incorporated into the template. Such a pixel has excellent predictive value.

Such changes are infrequent, and when one occurs, a control sequence (indicated symbolically by **ATMOVE** in Figure 0.4) is added to the output datastream. Hence, decoders need not do any processing to search for the correct setting for AT.

### 0.3.1.6 Adaptive arithmetic encoder

The adaptive-arithmetic-encoder (AAE) block is an entropy coder. It notes the outputs of the TP and DP blocks to determine if it is even necessary to code a given pixel. Assuming it is, it then notes the context and uses its internal probability estimator to estimate the conditional probability that the current pixel will be a given color. Often the pixel is highly predictable from the context so that the conditional probability is very close to 0 or 1 and a large entropy coding gain can be realized.

Maintaining probability estimates for each of the contexts is a non-trivial statistical problem. A balance must be struck between obtaining extremely accurate estimates and the conflicting need of adapting quickly to changing underlying statistics.

### 0.3.2 Lowest resolution layer encoder

Figure 0.5 shows a lowest-resolution-layer encoder. It is conceptually simpler than the differential-layer encoder because the RR and DP blocks are not applicable and the PT, AT, and MT blocks are different since there is no lower resolution layer to be used as input. Refer to the tables in clause 4 for a definition of signal names. (Not all sub-blocks need to be used in all systems.)

Lowest-resolution-layer TP like differential-layer TP is primarily intended to speed processing. The algorithms used for the two versions of TP are quite different, however, and it is not possible to skip as high a percentage of pixels with lowest-resolution-layer TP as it is with differential-layer TP. On images with text and line art, lowest-resolution-layer TP allows skipping about 40% of the pixels.
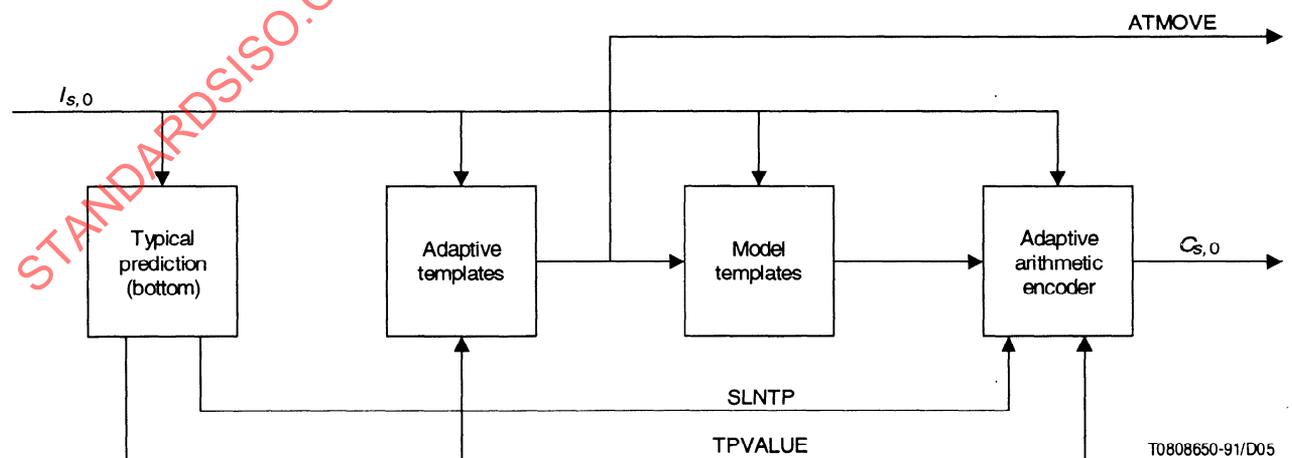


Figure 0.5 – Lowest-resolution-layer encoder

## 0.4  Decoder functional blocks

Figures 0.6, 0.7 and 0.8 are analogous to Figures 0.3, 0.4 and 0.5 but show decoding rather than encoding. Note that the RR and AT blocks do not appear in the decoder. Refer to the tables in clause 4 for a definition of signal names. In single-progression sequential coding only lowest-resolution-layer-decoder block of Figure 0.6 would be used. Not all sub-blocks in Figures 0.7 and 0.8 need be used in all systems.
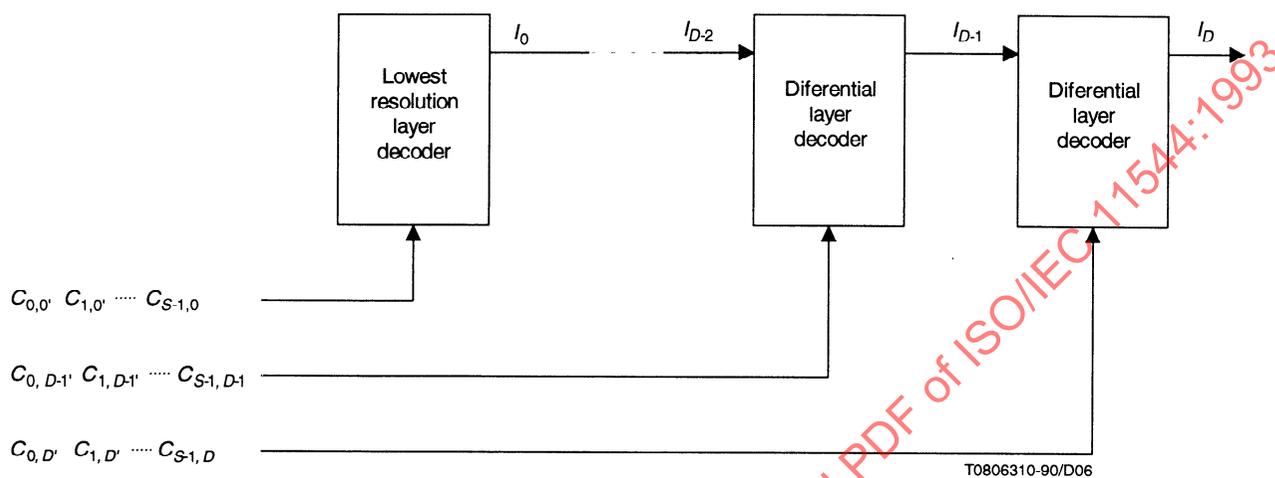


Figure 0.6 – Decomposition of decoder
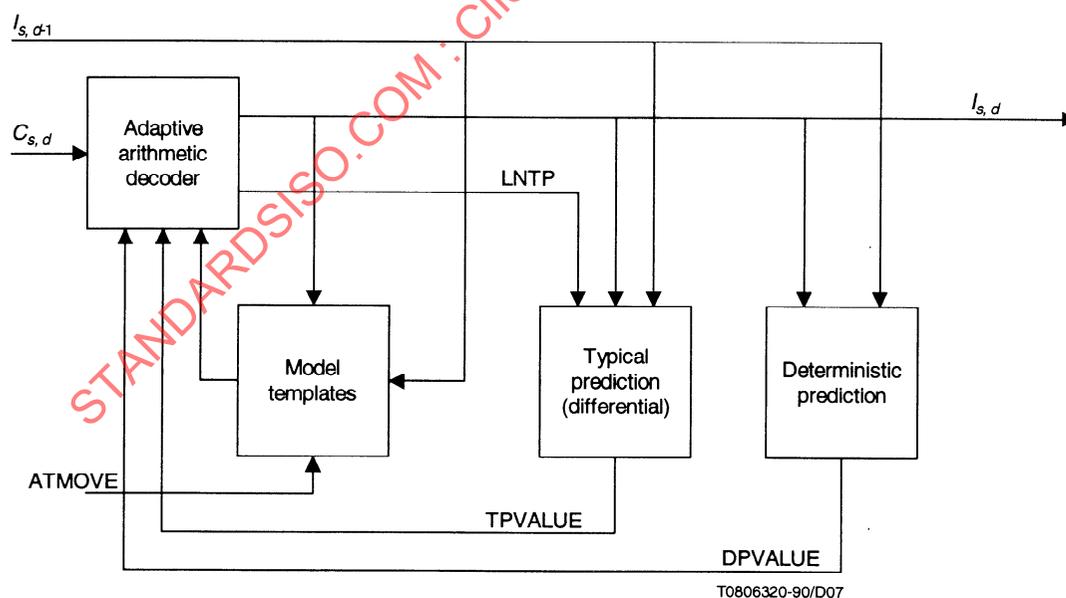


Figure 0.7 – Differential layer decoder

T0808660-91/D08
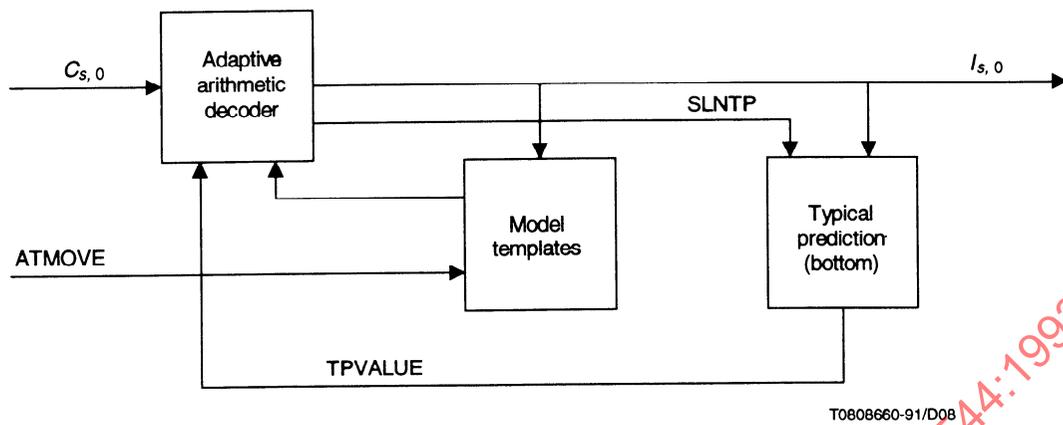
**Figure 0.8 – Lowest-resolution-layer decoder**

This page intentionally left blank

**INTERNATIONAL STANDARD**

**CCITT RECOMMENDATION**

## INFORMATION TECHNOLOGY –
## CODED REPRESENTATION OF PICTURE AND AUDIO INFORMATION –
## PROGRESSIVE BI-LEVEL IMAGE COMPRESSION

# 1 Scope

This Recommendation | International Standard defines a bit-preserving (lossless) compression method for coding image bit-planes and is particularly suitable for bi-level (two-tone, including black-white) images.

# 2 Normative references

There are no normative references. Informative references to standards and to the technical literature are listed in Annex F.

# 3 Definitions

For the purposes of this Recommendation | International Standard, the following definitions apply.

**3.1** **adaptive arithmetic coder**: A mechanism for adaptively compressing or decompressing data by using observed data characteristics to predict and code future data symbols.

**3.2** **adaptive templates (AT)**: Model templates which can be modified by moving an AT pixel during the processing of an image to take advantage of observed patterns in the image.

**3.3** **AT lag**: The distance in pixels between the pixel being encoded and the AT pixel.

**3.4** **AT pixel**: A special pixel in the model template that is allowed to adaptively change its location during the processing of an image.

**3.5** **bit-plane**: An array (or "plane") of bi-level symbols constructed from an image by choosing a particular bit from each pixel.

**3.6** **bit-plane interleaving**: A method used for mixing together two or more bit-planes of data into a single stream.

**3.7** **byte**: Eight bits of data.

**3.8** **byte stuffing**: A mechanism for unambiguously distinguishing between predefined escape bytes indicating the start of a marker segment and bytes identical to the escape byte which naturally occur in a compressed datastream.

**3.9** **context**: An integer corresponding to the specific pattern of the template and spatial phase (if needed) that is used to identify the index of the state of the adaptive arithmetic coder to be used for coding the current pixel.

**3.10** **deterministic prediction (DP)**: A method for exactly predicting (and therefore not coding) individual pixels in an image by using a lower resolution version of the same image along with very specific knowledge of the method of resolution reduction used.

**3.11** **differential layer coder**: A mechanism for encoding or decoding differential-layer images.

**3.12** **differential-layer image**: An image at a given resolution which is described by making reference to pixels in a lower-resolution image.

**3.13** **entropy coder**: Any lossless method for compressing or decompressing data.

**3.14** **escape byte**: A byte in a datastream signifying that information to follow has special marker-code meaning.

**3.15** **high-resolution pixel**: A pixel from the higher resolution image of the two resolution layers under discussion.

**3.16** **line not typical (LNTP)**: A condition which occurs during typical prediction when one or more of the pixels associated with a given low-resolution line would be predicted incorrectly.

**3.17** **lowest-resolution-layer coder**: A mechanism for encoding or decoding lowest-resolution-layer images.

**3.18**　　**lowest-resolution-layer image**: An image at a given resolution which is described without reference to any lower resolution images.

**3.19**　　**low-resolution pixel**: A pixel from the lower resolution image of the two resolution layers under discussion.

**3.20**　　**marker**: A combination of an escape byte and a marker byte that introduces control information.

**3.21**　　**marker byte**: A byte immediately following an escape byte that defines the type of control information being introduced.

**3.22**　　**marker segment**: The combination of a marker and any additional bytes of associated control information.

**3.23**　　**model template (MT)**: A geometric pattern describing the location of pixels relative to a pixel to be coded. It is used to model local image characteristics.

**3.24**　　**pixel**: One picture element of an image which is described by a rectangular array of such elements.

**3.25**　　**progressive behavior**: A coding technique shows progressive behavior if an image is first coded as a lowest resolution layer image and then is successively increased in resolution by means of differential layer images.

**3.26**　　**progressive coding**: A method of coding an image in which the image may be segmented into stripes, and then the entire image is first coded as a lowest resolution layer image and then is successively increased in resolution by means of differential layer images. This is compatible, by stripe/layer data reordering, with progressive-compatible sequential coding.

**3.27**　　**progressive-compatible sequential coding**: A method of coding an image in which the image may be segmented into stripes, the image stripes are coded in sequence, and within each stripe the image is coded to full resolution progressively. This is compatible, by stripe/layer data reordering with progressive coding.

**3.28**　　**protected stripe coded data (PSCD)**: A compressed image datastream which has been modified by stuffing bytes to distinguish between predefined escape bytes which signal special marker segments (which are not a part of the compressed datastream) and bytes identical to the escape byte which occur naturally in the compressed datastream.

**3.29**　　**resolution reduction method (RR)**: A method for transforming an image with a particular resolution into an image describing the same subject, but with a lower resolution.

**3.30**　　**sequential behavior**: A coding technique shows sequential behavior if portions of the image near the top are completely described before portions below have been described at all.

**3.31**　　**single-progression sequential coding**: A method of coding an image such that the image is fully coded in a single resolution layer, line by line, from left to right and top to bottom, without reference to any lower resolution images. This is compatible with progressive coding and progressive-compatible sequential coding if the number of differential layers is zero.

**3.32**　　**spatial phase**: An attribute of a pixel in a differential-layer image that describes its orientation with respect to the low-resolution pixel associated with it.

**3.33**　　**spatial resolution**: The number of pixels used to describe a region of an image of fixed spatial size.

**3.34**　　**stripe**: A fixed vertical size region of an image which encompasses the entire horizontal width of that image.

**3.35**　　**target pixel**: A pixel to be processed.

**3.36**　　**typical prediction (TP)**: A method for exactly predicting (and therefore not coding) blocks of pixels in an image by exploiting large regions of solid color.

# 4　　　Symbols and abbreviations

## 4.1　　　Acronyms

See Table 1.

## 4.2　　　Symbolic constants

See Table 2.

**Table 1 – Acronyms**

| Acronym | Meaning |
|---------|---------|
| AT | Adaptive templates |
| DP | Deterministic prediction |
| LPS | Less probable symbol |
| LSB | Least significant bit |
| MPS | More probable symbol |
| MSB | Most significant bit |
| MT | Model templates |
| RR | Resolution reduction |
| TP | Typical prediction |

**Table 2 – Symbolic constants**

| Constant | Meaning | Value ISO | Value Hexadecimal |
|----------|---------|-----------|-------------------|
| **ABORT** | Abort | 00/04 | 0x04 |
| **ATMOVE** | AT movement | 00/06 | 0x06 |
| **COMMENT** | Private comment | 00/07 | 0x07 |
| **ESC** | Escape | 15/15 | 0xff |
| **NEWLEN** | New length | 00/05 | 0x05 |
| **RESERVE** | Reserve | 00/01 | 0x01 |
| **SDNORM** | Normal stripe data end | 00/02 | 0x02 |
| **SDRST** | Reset at stripe data end | 00/03 | 0x03 |
| **STUFF** | Stuff | 00/00 | 0x00 |

## 4.3 Mathematical symbols, operators, and indicators

See Table 3.

## 4.4 Variables with mnemonic names

See Table 4.

## 5 Conventions

## 5.1 Flow diagram conventions and symbols

All flow diagrams are entered at the top and exited at the bottom. The symbol "<<" denotes a binary left shift with zero fill of low order bits and the symbol ">>" denotes a binary right shift with zero fill of high order bits. For both "<<" and ">>" the quantity on the left is the quantity being shifted and the quantity on the right is the shift amount. The binary logical AND of two numbers is indicated by "&".

## 5.2 Template graphics

It will frequently be necessary to show graphically the relationship of pixels in a high-resolution image to pixels in a low-resolution image. Figure 1 is a three-dimensional graphic showing such a relationship. In the text of this Specification two dimensional drawings like that in Figure 2 are used instead since they are more compact than their three-dimensional equivalent. Note that in the two-dimensional graphic low-resolution pixels are shown as circles and that the corresponding high-resolution pixels are shown as squares, partially hidden by the low-resolution circles.

**Table 3 – Mathematical symbols, operators, and indicators**

| Notation | Meaning |
|---|---|
| $C_{s,d}$ | Coded data for stripe $s$ and layer $d$ |
| $D$ | Number for final resolution layer in **BIE** |
| $D_L$ | Number for initial resolution layer in **BIE** |
| $h$ | High-resolution pixel |
| $I_d$ | Image at layer $d$ |
| $l$ | Low-resolution pixel |
| $L_c$ | Length in bytes of private comment |
| $L_d$ | Lines per stripe at layer $d$ |
| $M_X$ | Maximum horizontal offset allowed for AT pixel |
| $M_Y$ | Maximum vertical offset allowed for AT pixel |
| $p$ | Probability |
| $P$ | Number of bit-planes |
| $R_d$ | Resolution at layer $d$ |
| $S$ | Number of stripes |
| $X_d$ | Horizontal image size at layer $d$ |
| $y_{AT}$ | Line in which an AT switch is to be made |
| $Y_d$ | Vertical image size at layer $d$ |
| $\tau_X$ | Horizontal offset of the AT pixel |
| $\tau_Y$ | Vertical offset of the AT pixel |
| $>>$ | Binary right shift |
| $<<$ | Binary left shift |
| $\&$ | Bitwise AND |
| $!$ | Logical NOT |
| $\oplus$ | Exclusive OR |
| $\lceil \cdot \rceil$ | Ceiling function (smallest integer $\geq$ argument) |
| $0x$ | Hexadecimal indicator |

## 5.3 Spatial phase

A "spatial phase" can be associated with pixels in any resolution layer other than the lowest. This "phase" describes the orientation of the pixel with respect to the lower resolution pixel associated with it. If it is the upper left hand pixel of the four pixels associated with a single low-resolution pixel, it shall be said to have "phase 0". Similarly, the upper right pixel shall have "phase 1," the lower left pixel shall have "phase 2," and the lower right pixel shall have "phase 3" (see Figure 3).

## 5.4 Data structure graphics

Tables 5 to 16 contain graphics illustrating the decomposition of fields into sub-fields. Typographically leftmost sub-fields shall be transmitted earlier. The last row of each of these graphics gives field sizes in bytes. An entry of "1/8" denotes a single bit. An entry of "varies" is used when the field size is variable and depends upon options chosen, parameters chosen, or the particular image being coded.

**Table 4 – Variables with mnemonic names**

| Variable | Meaning |
|---|---|
| **A** | Interval size register |
| **BID** | Bi-level image data |
| **BIE** | Bi-level image entity |
| **BIH** | Bi-level image header |
| **BUFFER** | Buffer |
| **C** | Code register |
| **CE** | Conditional exchange |
| **CHIGH** | Code register, high two bytes |
| **CLOW** | Code register, low two bytes |
| **CT** | Bit counter |
| **CX** | Context |
| **DPLAST** | DP last |
| **DPON** | DP enabled |
| **DPPRIV** | DP private |
| **DPTABLE** | DP table |
| **DPVALUE** | DP value |
| **HITOLO** | High to low |
| **ILEAVE** | Interleave multiple bit-planes |
| **LNTP** | Line not typical |
| **LPIX** | Low-resolution pixel |
| **LRLTWO** | Lowest-resolution-layer two-line template |
| **LSZ** | LPS size on coding interval |
| **MPS** | More probable symbol |
| **NLPS** | Next if LPS |
| **NMPS** | Next if MPS |
| **PIX** | Pixel |
| **PSCD** | Protected stripe coded data |
| **SC** | Stack (of 0xff bytes) counter |
| **SCD** | Stripe coded data |
| **SDE** | Stripe data entity |
| **SEQ** | Sequential |
| **SLNTP** | Same **LNTP** |
| **SMID** | Index over stripe is in middle |
| **SWTCH** | Switch |
| **ST** | State |
| **TPBON** | Lowest-resolution-layer TP enabled |
| **TPDON** | Differential-layer TP enabled |
| **TPVALUE** | TP value |
| **VLENGTH** | Variable length |
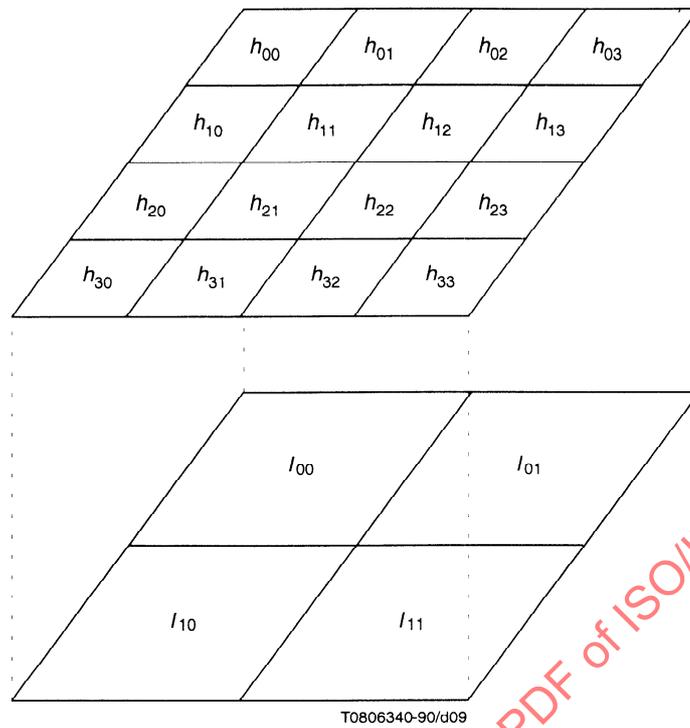
**Figure 1 – High- and low-resolution pixels in three-dimensional graphic**
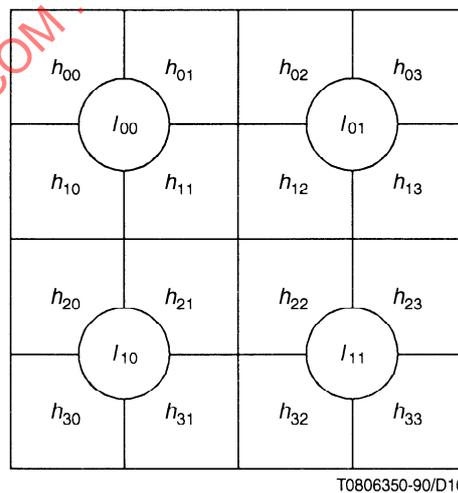


**Figure 2 – High- and low-resolution pixels in two-dimensional graphic**

# 6 Requirements

## 6.1 General rules

### 6.1.1 Color assignment

Each bit of each pixel plane is either 0 or 1. When the image is bi-level, a 1 bit shall indicate the foreground color and a 0 bit shall indicate the background color. If there is more than one bit plane, the mapping of intensity and color to bit-planes is not defined by this Specification.

NOTE – Whether 1 or 0 represents the foreground color is inconsequential for all aspects of this Specification except the described resolution reduction method. This resolution reduction method has a slight asymmetry between foreground and background colors.



T0806360-90/D11
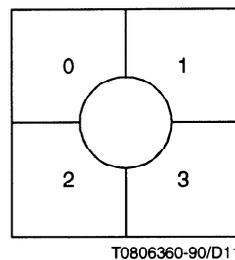
**Figure 3 – Four possible phases for high-resolution pixels**

### 6.1.2 Edge conventions

The resolution reduction, typical prediction, deterministic prediction, and coding algorithms all iterate through the image in the usual raster scan order, that is, from left to right and top to bottom. The processing for a current target pixel will reference the colors of some pixels in fixed spatial relationship to that target pixel. At image edges, these neighbor references may not lie in the actual image. For both high and low-resolution images, the rules to satisfy off-image references shall be as follows:

- a background colored (0) border shall be assumed to lie to the top, left, and right of the actual image;

- the bottom of the image shall be extended downward as far as necessary by pixel replicating the actual last line of the image.

Furthermore, in referencing pixels across stripe boundaries, the following rules shall be used:

- A pixel reference in a stripe above the current one shall return the actual value of the pixel, unless the pixel is above the image, in which case the background-border-rule for the image top shall be applied.

- A pixel reference in a stripe below the current one shall be satisfied by pixel replicating the last line of the current stripe. In particular, actual values shall not be used even if the reference is still within the image.

NOTE – This latter rule is only of consequence for the low-resolution image, as for decodability there can never be any references to high-resolution pixels in the line below. Also, the described resolution reduction algorithm happens to never reference even low-resolution pixels in the line below.

### 6.1.3 Byte alignment

NOTE – Because of the header and marker conventions to be described in 6.2, marker segments are always byte aligned in a datastream.

## 6.2 Data organization

### 6.2.1 Image decomposition

The highest level data structure described in this Specification is known as a bi-level image entity (**BIE**). A given **BIE** may contain data for one or more resolution layers and bit-planes. The data describing a given image in all its available resolutions and bit-planes may, but need not necessarily be, contained in more than one **BIE**.

NOTE – A multiple **BIE** description of an image is needed when images are first made available at low or intermediate resolution or bit-plane precision and there may or may not be a request for enhancement to higher resolution or precision.

### 6.2.2 Bi-level image entity and header (**BIE** and **BIH**) decomposition

As shown in Table 5, a bi-level image entity (**BIE**) shall comprise a bi-level image header (**BIH**) and bi-level image data (**BID**).

**Table 5 – Decomposition of BIE**

| BIE | |
| --- | --- |
| **BIH** | **BID** |
| Varies | Varies |

The bi-level image header shall comprise the fields shown in Tables 6, 7 and 8.

**Table 6 – Decomposition of BIH**

| BIH | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| $D_L$ | $D$ | $P$ | – | $X_D$ | $Y_D$ | $L_0$ | $M_X$ | $M_Y$ | Order | Options | **DPTABLE** |
| 1 | 1 | 1 | 1 | 4 | 4 | 4 | 1 | 1 | 1 | 1 | 0 or 1728 |

**Table 7 – Decomposition of order byte**

| Order | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| MSB | ... | | | | | | LSB |
| – | – | – | – | HITOLO | SEQ | ILEAVE | SMID |
| 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 |

**Table 8 – Decomposition of options byte**

| Options | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| MSB | ... | | | | | | LSB |
| – | LRLTWO | VLENGTH | TPDON | TPBON | DPON | DPPRIV | DPLAST |
| 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 | 1/8 |

The first byte of the **BIH** shall specify $D_L$, the initial resolution layer to be specified in this **BIE**. Most frequently, this number will be zero in which case the data transmitted will allow building up the image without any prior knowledge about it. It will be non-zero if a previous **BIE** has already defined the image to some intermediate layer and only incremental information is to be specified. The second byte specifies $D$, the final resolution layer described in this **BIE**. Note that with such multiple **BIE**'s, when $D_L$ is zero than $D$ equals a number of differential layers, but not the total number of differential layers.

The third byte shall specify $P$, the number of bit-planes. If the image is bi-level, $P$ shall be 1.

The fourth byte is fill. It shall always be written as 0.

The three subsequent four-byte fields specify $X_D$, $Y_D$ and $L_0$, which are respectively, the horizontal dimension at highest resolution, the vertical dimension at highest resolution, and the number of lines per stripe at lowest resolution. These three integers are coded most significant byte first. In other words, $X_D$ is the sum of $256^3$ times the fifth byte in **BIH**, $256^2$ times the sixth byte, 256 times the seventh byte, and the eighth byte.

The seventeenth and eighteenth bytes shall specify $M_X$ and $M_Y$, the maximum horizontal and vertical offsets allowed for the AT pixel. These parameters are discussed further in 6.7.3.

The nineteenth byte of the **BIH** shall carry the binary parameters **HITOLO**, **SEQ**, **ILEAVE** and **SMID**, which together specify the order in which stripe data is concatenated to form **BID**. More detail is provided in 6.2.4. The four most significant bits of this byte are fill and shall always be written as 0.

The twentieth byte of the **BIH** shall specify options. Its most significant bit is fill and shall always be written as 0. The template used for coding the lowest resolution layer shall have 2 or 3 lines as **LRLTWO** is 1 or 0 (see 6.7.1). If the **VLENGTH** bit is 0, there shall be no **NEWLEN** marker segments (see 6.2.6.2). If **VLENGTH** is 1, there may or may not be **NEWLEN** marker segments present. The **TPDON**, **TPBON** and **DPON** bits are 1 when it is desired to enable respectively, differential-layer TP, lowest-resolution-layer TP, and DP. The **DPPRIV** and **DPLAST** bits are only meaningful if **DPON** equals 1. If **DPON** is 1 and **DPPRIV** is 1, a private DP table is to be used. If **DPLAST** is 0, the private DP table (1728 bytes) is to be loaded. Otherwise the DP table last used is to be reused.

The **DPTABLE** field of **BIH** shall only be present if **DPON** equals 1, **DPPRIV** equals 1, and **DPLAST** equals 0. Its size and interpretation are defined in 6.6.

The variables $D_L$, $D$, $P$, $X_D$, $Y_D$, $L_0$, $M_X$, $MY$, **HITOLO**, **SEQ**, **ILEAVE**, **SMID**, **LRLTWO**, **VLENGTH**, **TPDON**, **TPBON**, **DPON**, **DPPRIV** and **DPLAST** are free parameters. Some applications standards may restrict the choices for some or all of them. Table 9 shows the limits on these parameters as set by either their implicit natures or the field sizes allowed for them in a **BIH**.

A JBIG datastream is any datastream created as described in the normative portions of this Specification with parameters in the range of Table 9. In the interests of creating as large as possible a community of applications for which it is possible to share hardware and exchange decodable data, Annex A suggests minimum support for these free parameters. Different applications are encouraged to choose parameter values within the suggested ranges of minimum support whenever it is possible to do so. Implementations desiring to be compatible with a broad range of applications may wish to support all free parameter choices within the suggested ranges.

**Table 9 – Absolute limits on free parameters**

| Parameter | Minimum | Maximum |
|:---:|:---:|:---:|
| $D_L$ | 0 | $D$ |
| $D$ | $D_L$ | 255 |
| $P$ | 1 | 255 |
| $X_D$ | 1 | 4 294 967 295 |
| $Y_D$ | 1 | 4 294 967 295 |
| $L_0$ | 1 | 4 294 967 295 |
| $M_X$ | 0 | 127 |
| $M_Y$ | 0 | 255 |
| **HITOLO** | 0 | 1 |
| **SEQ** | 0 | 1 |
| **ILEAVE** | 0 | 1 |
| **SMID** | 0 | 1 |
| **LRLTWO** | 0 | 1 |
| **VLENGTH** | 0 | 1 |
| **TPDON** | 0 | 1 |
| **TPBON** | 0 | 1 |
| **DPON** | 0 | 1 |
| **DPPRIV** | 0 | 1 |
| **DPLAST** | 0 | 1 |

### 6.2.3 Iteration for parameters dependent on resolution layer

The image dimensions at lower layers (indexed by $d$) shall be defined recursively for $D \geq d \geq 1$ by the iterations.

$$X_{d-1} = \lceil X_d / 2 \rceil \tag{1}$$

$$Y_{d-1} = \lceil Y_d / 2 \rceil \tag{2}$$

where $\lceil \cdot \rceil$ denotes the ceiling function, or, in other words, the smallest integer greater than or equal to the argument.

For $1 \leq d \leq D$ the number of lines per stripe at layer $d$ shall be defined by

$$L_d = 2 \times L_{d-1} \tag{3}$$

At all layers there will necessarily be

$$S = \lceil Y_0 / L_0 \rceil \tag{4}$$

stripes. In many cases, the last stripe in any layer $d$ will have fewer than $L_d$ lines.

### 6.2.4 Bi-level image data (BID) decomposition

The coded data $C_{s, d, p}$ defining a given stripe $s$ at resolution $d$ and bit plane $p$ shall be contained in a stripe data entity or **SDE**. The **BID** shall consist of a concatenation of **SDE**'s and floating marker segments as shown in Table 10.

**Table 10 – Decomposition of BID**

| BID | | | | | | |
|---|---|---|---|---|---|---|
| Floating marker segment(s) | $\textbf{SDE}_{s, d, p}$ | Floating marker segments(s) | $\textbf{SDE}_{s, d, p}$ | $\cdots$ | Floating marker segment(s) | $\textbf{SDE}_{s, d, p}$ |
| Varies | Varies | Varies | Varies | $\cdots$ | Varies | Varies |

The ordering of the **SDE**'s depends on **HITOLO**, **SEQ**, **ILEAVE** and **SMID**. Index nesting shall be as defined by Table 11. Only the six combinations of the three variables **SEQ**, **ILEAVE** and **SMID** shown in Table 11 are allowed. The remaining two combinations shall never occur. The loops over the dummy variables $s$ and $p$ are respectively from 0 to $S-1$ (top to bottom) and $P-1$ to 0 (MSB to LSB). If **HITOLO** is 0, the dummy variable $d$ shall range from $D_L$ to $D$. Otherwise it shall range from $D$ to $D_L$.

For a tutorial example see Table Intro. 2.

**Table 11 – Ordering of stripe encodings in BID**

| SEQ | ILEAVE | SMID | Loops | | |
|---|---|---|---|---|---|
| | | | Outer | Middle | Inner |
| 0 | 0 | 0 | $p$ | $d$ | $s$ |
| 0 | 1 | 0 | $d$ | $p$ | $s$ |
| 0 | 1 | 1 | $d$ | $s$ | $p$ |
| 1 | 0 | 0 | $s$ | $p$ | $d$ |
| 1 | 0 | 1 | $p$ | $s$ | $d$ |
| 1 | 1 | 0 | $s$ | $d$ | $p$ |

### 6.2.5    Stripe data entity (SDE) decomposition

As shown in Table 12, each SDE shall be terminated by an ESC byte and either an SDNORM byte or an SDRST byte.

**Table 12 – Structure of the SDE**

| SDE | | |
|---|---|---|
| PSCD | ESC | SDNORM or SDRST |
| Varies | 1 | 1 |

Normally the terminating byte will be SDNORM, which implies that all "state" information is saved. If instead the terminating byte is SDRST, the "state" for that particular bit-plane and that particular resolution layer shall be reset prior to encoding or decoding the next stripe of that plane and layer. Resetting the state with SDRST requires initializing the adaptive probability estimators as at the top of the image, resetting (if necessary) the AT pixel to its default location, and initializing $LNTP_{y-1}$ to 1 when in the lowest resolution layer. It also requires that all functions including resolution reduction, deterministic prediction, typical prediction, and model templates start the next stripe as they do when at the top of the image, that is, as defined in 6.1.2.

> NOTE – Resetting the state with SDRST should not be done unnecessarily as doing so degrades compression efficiency and may introduce artifacts at stripe borders in lower-resolution images.

Protected stripe coded data (PSCD) is defined as the bytes of SDE that remain after removing the two terminating bytes. A decoder shall create stripe coded data (SCD) from the PSCD by replacing all occurrences of an ESC byte followed by a STUFF byte with a single ESC byte. An encoder shall create PSCD from SCD by replacing all occurrences of an ESC byte with an ESC byte followed by a STUFF byte. The use of SCD is described in 6.8. PSCD is used in defining the SDE rather than SCD so that data for one stripe can be unambiguously located in the BID.

An ESC byte and ABORT byte may be used to prematurely terminate the BID as shown in Table 13.

> NOTE – Without a mechanism like this an encoder encountering a problem would "hang" its associated decoder indefinitely. There would be no way to restart the decoder as it would not reset until after it had received the announced amount of data.

**Table 13 – Marker code to prematurely terminate a BID**

| ESC | ABORT |
|---|---|
| 1 | 1 |

### 6.2.6    Floating marker segments

Floating marker segments provide control information. They shall not occur within an SDE. They may occur between SDE's or before the first SDE. There are three floating marker segments: ATMOVE, NEWLEN, and COMMENT. (See Note 2 in 6.2.6.2.)

### 6.2.6.1    Adaptive-template (AT) movement

The location of the AT pixel may be changed with the ATMOVE marker segment shown in Table 14.

**Table 14 – Structure of the ATMOVE floating marker segment**

| ESC | ATMOVE | $Y_{AT}$ | $\tau_X$ | $\tau_Y$ |
|---|---|---|---|---|
| 1 | 1 | 4 | 1 | 1 |

The third, fourth, fifth, and sixth byte define $y_{AT}$, the line at which the template changes. The seventh and eighth bytes define $\tau_X$ and $\tau_Y$, the horizontal and vertical offsets for the new AT pixel. The line $y_{AT}$ shall be decoded as the sum of $256^3$ times the third byte, $256^2$ times the fourth byte, 256 times the fifth byte, and the sixth byte.

The probability estimator is not re-initialized following an **ATMOVE**. The resolution layer and bit plane for which a given **ATMOVE** marker segment is to be effective shall be that of the first **SDE** to follow the marker segment. The line numbering for $y_{AT}$ restarts at 0 for each stripe so that if, for example, a change is to be effective on the initial line of a stripe, $y_{AT}$ equals 0.

Further discussion of adaptive template pixels and the variables $y_{AT}$, $\tau_X$, and $\tau_Y$ appears in 6.7.3.

### 6.2.6.2 Redefining image length

If **VLENGTH** is 1, it is permissible to change the length $Y_D$ of the image with a new-length marker segment as shown in Table 15.

**Table 15 – Marker segment to indicate a new vertical dimension**

| ESC | NEWLEN | $Y_D$ |
|-----|--------|-------|
| 1   | 1      | 4     |

At most only one new length marker segment shall appear in any **BIE**. However, a marker segment could refer to a line in the immediately preceding stripe due to an unexpected termination of the image or the use of only one stripe. Such a marker segment is followed immediately by **ESC + SDNORM/SDRST**, and the new $Y_D$ given by the newlength marker segment can be less than the line number at the end of the preceding stripe. The encoder shall not code more than the number of lines in each layer corresponding to the new $Y_D$. Within the new length marker segment, $Y_D$ shall be packed into its four byte field exactly as it is packed into its four byte field in **BIH**. The new $Y_D$ shall never be greater than the original.

NOTES

1    The new length marker has been defined so that it is possible to begin coding an image of unknown length. In this case the original dimension $Y_D$ placed into the header serves as an indication of the maximum length that the image might possibly be.

2    Some applications might need the new length marker segment to immediately follow a PSCD just before the **ESC + SDNORM/SDRST** in order to prematurely terminate. This usage is currently under study.

### 6.2.6.3 Comment marker segment

An **ESC** byte followed by a **COMMENT** byte and a four-byte integer $L_c$ shall begin a comment marker segment as shown in Table 16.

The number $L_c$ shall be equal to the sum of $256^3$ times the third byte, $256^2$ times the fourth byte, 256 times the fifth byte, and the sixth byte. This number shall give the length of only the private comment portion of the comment marker segment. In other words, the total length of the comment marker segment shall be $L_c$ + 6 bytes.

**Table 16 – Comment marker segment**

| ESC | COMMENT | $L_c$ |
|-----|---------|-------|
| 1   | 1       | 4     |

### 6.2.7 Reserved marker byte

An **ESC** byte followed by a **RESERVE** byte is a reserved marker. One possible use of this marker is described in 6.8.2.8. No future extensions of this Specification shall use this marker for any purpose. Encoders or decoders may employ it for any private purpose desired. The reserved marker shall never appear in a public datastream.

## 6.3 Resolution reduction

The default deterministic prediction table is matched to the suggested resolution reduction algorithm described in this subclause. It is permissible to use an alternative resolution reduction algorithm in place of the suggested resolution reduction algorithm, but in that case either deterministic prediction must be disabled or a matched deterministic prediction table must be downloaded to the decoder as part of the **BIH**.

The resolution reduction algorithm is identical for all resolution layers and all bit-planes. The processing to create the image at resolution layer $d - 1$ from an image at resolution layer $d$ is described here.

If $X_d$ or $Y_d$ is not even, for purposes of resolution reduction a new layer $d$ image shall be created by adding as necessary either a column of background color (0) at the right side or a pixel replication of the last line at the bottom. Hence for the remainder of this subclause, $X_d$ and $Y_d$ are assumed even.

The original image can be divided up into two by two blocks of pixels, and each of these two by two superpixels shall map to one low-resolution pixel in the reduced-resolution image. These low-resolution pixels shall be determined from left to right and top to bottom in the normal raster scan order. The suggested resolution reduction rule of this Specification is defined by Figure 4 and Table 17. The reasoning behind this particular mapping is explained in Annex B (informative).
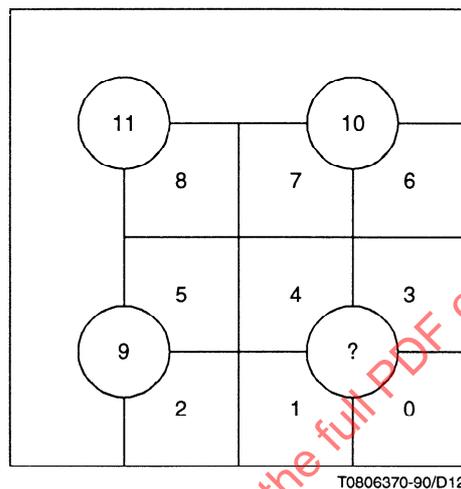


Figure 4 – Pixels used to determine the color of a low-resolution pixel

The circle with a "?" within it represents the low-resolution pixel the color of which is to be determined. The circles and squares with numbers within them correspond to pixels used in making this determination.

The colors of the numbered pixels define an index, with each numbered pixel defining one bit in the index. The pixel labeled "0" determines the least significant bit of the index and each additional numbered pixel determines the bit of the index corresponding to its number. When a pixel takes on the foreground color, its corresponding bit in the index shall take on the value "1." Given this index, the color of the pixel labeled "?" shall be defined by Table 17, which is indexed from left to right. For example, the colors of the pixels corresponding to indices 0 through 7 are 0, 0, 0, 1, 0, 0, 0, 1, respectively.

At the edges of an image, some of the numbered pixels of Figure 4 may not be within the image. For the purpose of defining an index, the general edge rules of 6.1.2 shall be used.

When beginning resolution reduction, the upper-left most pixel of the high resolution image shall be aligned with pixel 4 in Figure 4.

## 6.4    Differential-layer typical prediction

Differential-layer TP shall be enabled or disabled with the **TPDON** bit in the options field of **BIH**. If it is disabled (**TPDON** = 0), the TPD block in both an encoder or decoder shall simply output **TPVALUE** = 2 for all pixels, to indicate to the arithmetic encoding or decoding block that no prediction is being made. Also, when differential-layer TP is disabled, the pseudo-pixel **LNTP** shall be neither encoded nor decoded by the arithmetic coder. The discussion in the remainder of this subclause and its subclauses assumes differential-layer TP is enabled (**TPDON** = 1).

Whenever reference is made to a pixel that because of edge effects is not actually in the current stripe, the value of this pixel shall be determined by the general edge rules of 6.1.2.

**Table 17 – Map to determine low-resolution color**

| Index | Color | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| [0, 63] | 00010001 | 01110011 | 11111111 | 11111111 | 00110011 | 11111111 | 11111111 | 11111111 |
| [64, 127] | 00000001 | 01110111 | 11111111 | 11111111 | 00110111 | 11111111 | 11111111 | 11111111 |
| [128, 191] | 00110111 | 11111111 | 11111111 | 11111111 | 01111101 | 11111111 | 11111111 | 11111111 |
| [192, 255] | 00110111 | 11111111 | 11111111 | 11111111 | 11111111 | 01111101 | 11111111 | 11111111 |
| [256, 319] | 00000001 | 00110111 | 11111101 | 11111111 | 00111111 | 11111111 | 11111111 | 11111111 |
| [320, 383] | 00110111 | 01111111 | 11111111 | 01111111 | 01111111 | 01111111 | 01111111 | 11111111 |
| [384, 447] | 00110101 | 11111111 | 11110111 | 11111111 | 11011111 | 01111111 | 11111111 | 11111111 |
| [448, 511] | 11111111 | 11111111 | 11111111 | 11111111 | 11111111 | 11111111 | 11111111 | 11111111 |
| [512, 575] | 00000001 | 00100011 | 00000101 | 00111011 | 00010001 | 00100011 | 01110001 | 11111111 |
| [576, 639] | 00000001 | 01110101 | 00111011 | 01111111 | 00000000 | 01010011 | 11111110 | 11111111 |
| [640, 703] | 00000001 | 01000001 | 01111111 | 11111111 | 00001001 | 10110111 | 11111111 | 11111111 |
| [704, 767] | 00000000 | 01010011 | 01111111 | 11111011 | 10010011 | 01111001 | 11111111 | 11011111 |
| [768, 831] | 00000001 | 00000000 | 01110011 | 11111111 | 00110001 | 00010011 | 01110101 | 11111111 |
| [832, 895] | 00000000 | 01000001 | 10110111 | 11101110 | 00000001 | 00100001 | 11111100 | 11111111 |
| [896, 959] | 00000000 | 10010011 | 01110101 | 11111111 | 00010001 | 01101011 | 11110111 | 11111111 |
| [960, 1023] | 11101001 | 11110111 | 11111111 | 11111011 | 10110111 | 11111111 | 11111011 | 11111111 |
| [1024, 1087] | 00000001 | 00100011 | 00000001 | 00111111 | 00010001 | 00000001 | 01110111 | 11111111 |
| [1088, 1151] | 00000001 | 01110101 | 01101011 | 01111111 | 00000000 | 01010011 | 11111110 | 11111111 |
| [1152, 1215] | 00000001 | 01100001 | 01111111 | 11111111 | 00101001 | 00110111 | 11111111 | 11111111 |
| [1216, 1279] | 00000000 | 01110011 | 00111111 | 01111011 | 10010010 | 01111101 | 11111111 | 11111111 |
| [1280, 1343] | 00000001 | 00000000 | 01111011 | 11111110 | 00101111 | 00011000 | 01111111 | 11111111 |
| [1344, 1407] | 00000000 | 01000001 | 00110111 | 11111110 | 00001001 | 00110111 | 01111110 | 01111111 |
| [1408, 1471] | 00000000 | 11010010 | 01111111 | 11111111 | 00011011 | 01101111 | 11111111 | 11111111 |
| [1472, 1535] | 00000000 | 01110101 | 01111111 | 01110111 | 00100111 | 01111111 | 01111011 | 01111111 |
| [1536, 1599] | 00000001 | 00000011 | 00000001 | 00001001 | 00010011 | 00000001 | 01000001 | 10010011 |
| [1600, 1663] | 00000001 | 01110101 | 00100001 | 01010101 | 00000001 | 01010001 | 10000000 | 11110111 |
| [1664, 1727] | 00000001 | 01000001 | 01101011 | 00010011 | 00000001 | 00000000 | 11111011 | 11111111 |
| [1728, 1791] | 00000000 | 01010001 | 00000001 | 01110011 | 00000000 | 01000001 | 10110111 | 11111111 |
| [1792, 1855] | 00000001 | 00000000 | 01100001 | 10000001 | 00100111 | 00001001 | 00011110 | 10111111 |
| [1856, 1919] | 00000000 | 01000000 | 00000001 | 01010110 | 00001000 | 00000000 | 00010000 | 01111111 |
| [1920, 1983] | 00000000 | 10000000 | 00100001 | 01110011 | 00000011 | 00000001 | 00111111 | 11111111 |
| [1984, 2047] | 01101000 | 11010000 | 11110011 | 10110011 | 00000000 | 11010011 | 11111011 | 11111111 |
| [2048, 2111] | 00000001 | 00000011 | 00110111 | 11111111 | 00110011 | 00110111 | 01111111 | 11111111 |
| [2112, 2175] | 00000001 | 01110111 | 01111111 | 11111111 | 00010001 | 01111011 | 11111111 | 11111111 |
| [2176, 2239] | 00000001 | 11110111 | 01111111 | 11111111 | 00111111 | 11111111 | 11111101 | 11111111 |
| [2240, 2303] | 00010010 | 11110111 | 11111111 | 11111111 | 11111111 | 11111101 | 11111111 | 01111111 |
| [2304, 2367] | 00000001 | 00010010 | 01101101 | 11111111 | 00111111 | 01111111 | 11111111 | 11111111 |
| [2368, 2431] | 00000000 | 01100010 | 11111111 | 01111111 | 00111111 | 00111111 | 01111111 | 11111111 |
| [2432, 2495] | 00010000 | 11111111 | 11110111 | 11111111 | 01111111 | 11111111 | 01111111 | 11111111 |
| [2496, 2559] | 11111111 | 11111111 | 11111111 | 11111111 | 11111111 | 11111111 | 11111111 | 11111111 |
| [2560, 2623] | 00000001 | 00100011 | 00000001 | 00011011 | 00010001 | 00100011 | 01110111 | 11111111 |
| [2624, 2687] | 00000001 | 01110101 | 00101011 | 01110111 | 00000000 | 01000001 | 10111110 | 11111111 |
| [2688, 2751] | 00000001 | 11000001 | 01011011 | 01111111 | 00001001 | 00110011 | 01111101 | 11111111 |
| [2752, 2815] | 00000000 | 01010001 | 00110111 | 11111011 | 10101001 | 10110001 | 11111111 | 11111111 |
| [2816, 2879] | 00000001 | 00000000 | 01110001 | 10110111 | 00100001 | 00000011 | 01110101 | 11111111 |
| [2880, 2943] | 00000000 | 01000000 | 00010111 | 01101111 | 00000000 | 00000001 | 01111101 | 11111111 |
| [2944, 3007] | 00000000 | 11000001 | 01110101 | 11111111 | 00000001 | 10101011 | 01010001 | 11111111 |
| [3008, 3071] | 11101000 | 11010011 | 11111111 | 11111011 | 10111011 | 11111111 | 11111011 | 11111111 |
| [3072, 3135] | 00000001 | 00100011 | 00000001 | 00011011 | 00110001 | 00000001 | 01010011 | 01111111 |
| [3136, 3199] | 00000001 | 01110101 | 00101001 | 01111111 | 00000000 | 01010001 | 10110110 | 11111111 |
| [3200, 3263] | 00000001 | 11100000 | 01111111 | 11111111 | 00001010 | 00111011 | 01111111 | 11111111 |
| [3264, 3327] | 00000000 | 01110001 | 01111111 | 11111011 | 10001000 | 01110101 | 11111111 | 01111111 |
| [3328, 3391] | 00000001 | 00000000 | 01100001 | 11110110 | 00111111 | 00001001 | 01111111 | 11111111 |
| [3392, 3455] | 00000000 | 01000000 | 00010111 | 01111111 | 00001000 | 00010011 | 01111110 | 01111111 |
| [3456, 3519] | 00000000 | 10000000 | 01110111 | 11111111 | 00101011 | 00101111 | 01111111 | 01111111 |
| [3520, 3583] | 00000000 | 01110001 | 01111111 | 01110111 | 00101011 | 01111111 | 00111011 | 01111111 |
| [3584, 3647] | 00000001 | 00000011 | 00000001 | 00001001 | 00010001 | 00000001 | 01000001 | 00000001 |
| [3648, 3711] | 00000001 | 01110101 | 00100001 | 01010101 | 00000000 | 01010001 | 10000000 | 01010011 |
| [3712, 3775] | 00000001 | 01000001 | 01001001 | 00000001 | 00001001 | 00000000 | 00000001 | 00010011 |
| [3776, 3839] | 00000000 | 01010001 | 00000000 | 01010011 | 10000000 | 01000001 | 00010011 | 11111111 |
| [3840, 3903] | 00000001 | 00000000 | 01100001 | 10000000 | 00100001 | 00000001 | 00000001 | 00010011 |
| [3904, 3967] | 00000001 | 01000000 | 00000000 | 01000000 | 00000000 | 00000000 | 00000000 | 00010011 |
| [3968, 4031] | 00000000 | 10000000 | 00000000 | 00010011 | 00000001 | 00000001 | 01010001 | 01111111 |
| [4032, 4095] | 00000000 | 01010000 | 00000000 | 01110011 | 00000001 | 01010100 | 00110001 | 01110111 |

### 6.4.1    Processing in an encoder

Figure 5 defines an 8-neighborhood. The eight pixels not marked by the "?" are immediately adjacent to it and are its 8-neighborhood.

T0806380-90/D13
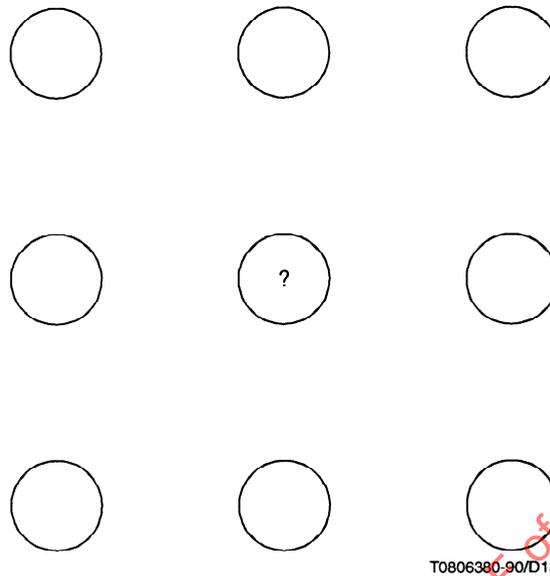
**Figure 5 – Definition of 8-neighborhood**

A given low-resolution pixel is "not typical" if it and all the pixels in its 8-neighborhood are the same color, but one or more of the four high-resolution pixels associated with it differs from this common color. A given low-resolution line is "not typical" (**LNTP**) if it contains any "not typical" pixels. A flow diagram for processing to determine **LNTP** is shown in Figure 6. In this Figure **LPIX** denotes a low-resolution pixel.

> NOTE – Low resolution pixels that are not typical in this sense are possible, but extremely rare.

Figure 7 shows a high-resolution line pair and an associated low-resolution line. Also shown in this figure is the virtual location used for coding the value **LNTP**.

As suggested by this figure the value **LNTP** of the pseudo-pixel shall be coded by the arithmetic coder before any of the regular high-resolution pixels in the line pair are coded. In coding this pseudo-pixel, **TPVALUE** and **DPVALUE** shall always be 2. The context **CX** that shall be used for coding is the same context that is used for coding an ordinary pixel with phase 3 and surrounded by pixels as shown in Figure 8. In this figure "F" denotes foreground and "B" denotes background.

> NOTE – This particular context was chosen for reuse as a context for coding **LNTP** because it occurs infrequently, and for most images the probability of coding foreground within it is small as is the probability that **LNTP** equals one. The more obvious alternative of coding **LNTP** in its own context unfortunately increases the total number of contexts to just over a power of 2.

Figure 9 shows the required processing to produce the output signal **TPVALUE**. In words, if **LNTP** equals 0 and the low-resolution pixel associated with a high-resolution pixel **PIX** is the same color as all of the pixels in its 8-neighborhood, then **TPVALUE** equals that color. Otherwise, it is set equal to 2 to indicate that a prediction can not be made.

### 6.4.2    Processing in a decoder

At the beginning of each high-resolution line pair, the TP pseudo-pixel **LNTP** shall be decoded (see Figure 7). In decoding **LNTP**, the inputs **TPVALUE** and **DPVALUE** shall be 2 and **CX** shall be as described in 6.4.1.

In decoding a given high-resolution pixel **PIX**, **TPVALUE** shall be generated as in an encoder.

T0806390-90/D14
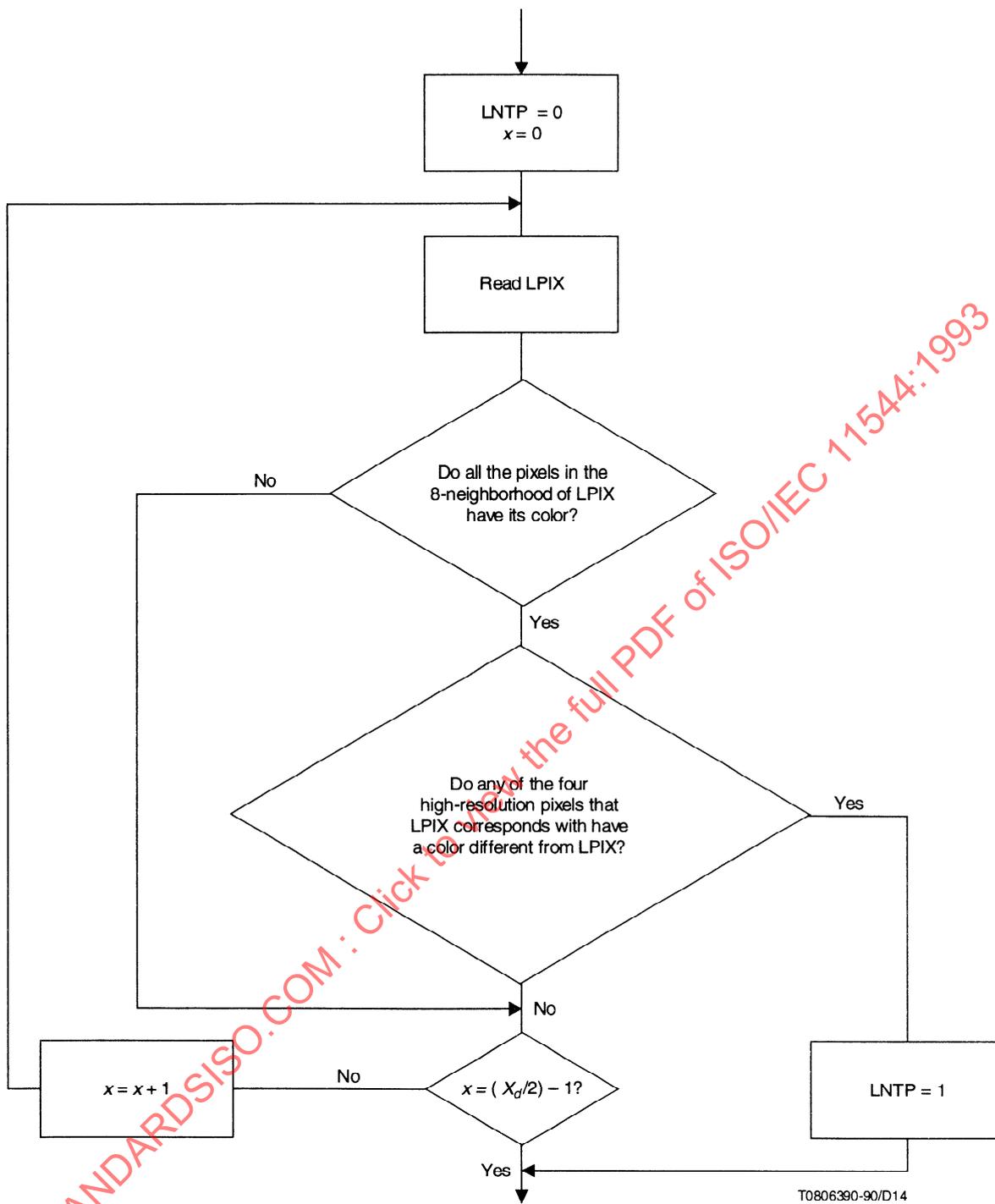
**Figure 6 – Processing to determine LNTP**

## 6.5    Lowest-resolution-layer typical prediction

TP in the lowest-resolution-layer can be enabled or disabled with the **TPBON** bit in the options field of **BIH**. If it is disabled (**TPBON** = 0), the TPB block in both an encoder and decoder shall simply output **TPVALUE** equal to 2 for all pixels thereby indicating to the encoding and decoding blocks that no prediction is being made. Also, when lowest-resolution-layer TP is disabled, the pseudo-pixel **SLNTP** shall be neither encoded nor decoded by the arithmetic coder. The discussion in the remainder of this subclause assumes lowest-resolution-layer TP is enabled (**TPBON** = 1).

T0806400-90/D15
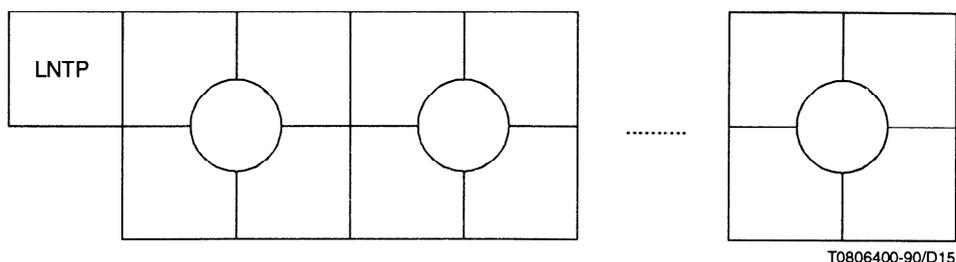
**Figure 7 – Location of pseudo-pixel in relationship to ordinary pixels**



TISO2390-93/D16
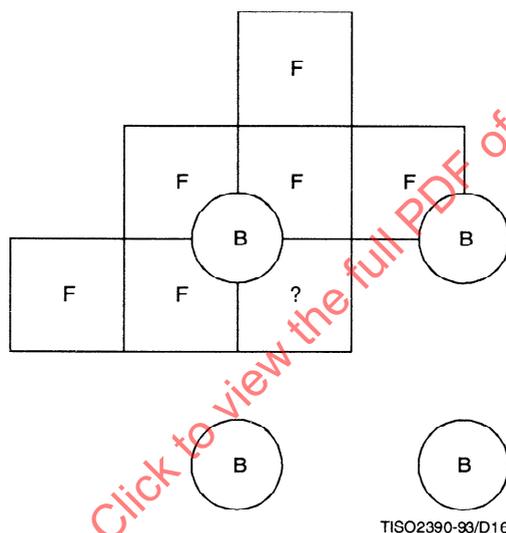
**Figure 8 – Reused context for coding the differential-layer-TP pseudo pixel**

### 6.5.1 Encoder processing

Let $y$ denote the current line. If it differs at any pixel location from the line above, then $\mathbf{LNTP}_y$ shall equal 1 and line $y$ is said to be non-typical. Otherwise, $\mathbf{LNTP}_y$ shall equal 0. In determining whether the very first line of an image is non-typical, the line immediately above the image shall be assumed, as usual, to be the background color.

NOTE – Whereas almost all lines are "typical" in the sense of differential-layer TP, only a modest fraction are "typical" in the sense of lowest-resolution-layer TP.

Define

$$SLNTP_y = !(LNTP_y \oplus LNTP_{y-1}) \tag{5}$$

where the symbol $\oplus$ denotes the exclusive-or operation and the symbol ! denotes logical negation. In words, $\mathbf{SLNTP}_y$ will be 1 if and only if $\mathbf{LNTP}_y$ is the same as $\mathbf{LNTP}_{y-1}$. For the top line of an image, $\mathbf{LNTP}_{y-1}$ shall be set equal to 1.

When lowest-resolution-layer TP is enabled, a pseudo-pixel equal in value to $\mathbf{SLNTP}$ shall be coded in the virtual location shown before any pixels of line y are coded (see Figure 10).
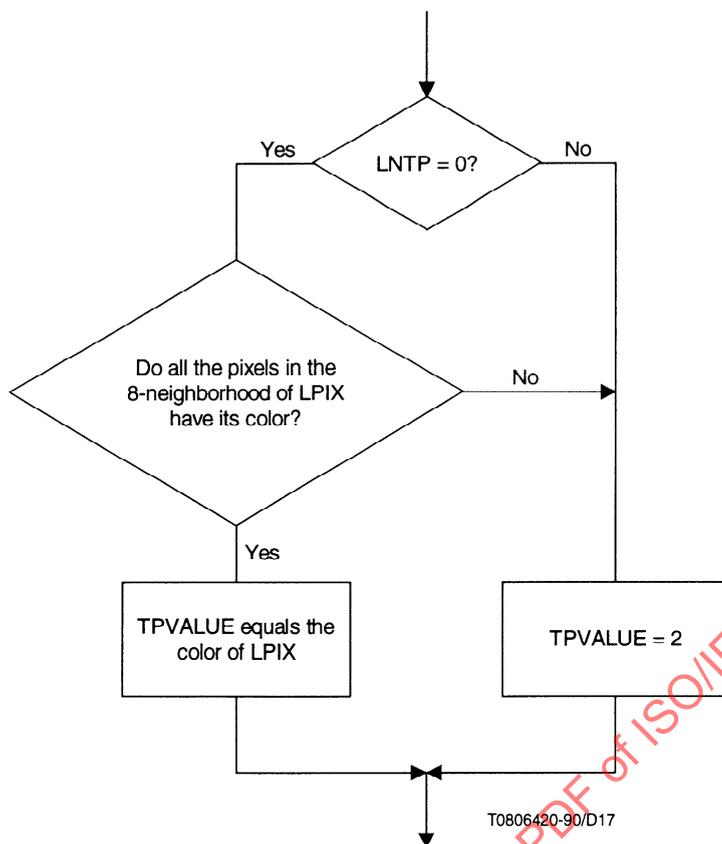
Figure 9 – Processing to determine TPVALUE
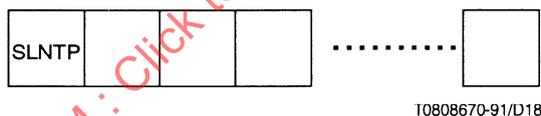


Figure 10 – Location of pseudo-pixel in relationship to ordinary pixels

When **SLNTP** is coded, it shall be coded in the context shown in Figure 11 if **LRLTWO** is 0 and in the context shown in Figure 12 if **LRLTWO** is 1 (see 6.7.1). In this figure "F" denotes foreground and "B" denotes background. When coding **SLNTP**, **TPVALUE** shall always equal 2. In other words, **SLNTP** can never be predicted with TP and must always be arithmetically encoded.

NOTE – Arithmetically encoding changes in **LNTP** is more efficient than arithmetically encoding **LNTP**. In lowest-resolution-layer TP, **LNTP** does not take on either the value 1 or the value 0 with high probability, and can not be entropy coded with high efficiency.

If **LNTP**$_y$ is 0, the TPB block shall output the value common to the current pixel and the pixel above as **TPVALUE**. Otherwise, it shall output 2 to indicate that no prediction can be made.

### 6.5.2 Decoder processing

If **TPBON** is one, the sameness indicator **SLNTP**$_y$ shall be decoded (see Figure 10). In decoding **SLNTP**, **TPVALUE** shall be 2 and **CX** shall be as in Figure 11 or Figure 12 as appropriate.

TISO2400-93/D19

**Figure 11 – Reused context for coding the lowest-resolution-layer-TP pseudo-pixel (three-line template)**



TISO2410-93/D20

**Figure 12 – Reused context for coding the lowest-resolution-layer-TP pseudo pixel (two-line template)**

The decoder shall recreate $\text{LNTP}_y$ by

$$\text{LNTP}_y = !(\text{SLNTP}_y \oplus \text{LNTP}_{y-1}) \tag{6}$$

As in the encoder, this iteration shall be initialized with **LNTP** set equal to 1 for the line immediately above the top line of the real image.

If $\text{LNTP}_y$ is 0, the block **TPB** shall output the value of the pixel immediately above the current one as **TPVALUE**. Otherwise, it shall output 2 to indicate no prediction can be made.

## 6.6 Deterministic prediction (DP)

DP shall be enabled or disabled with the **DPON** bit in the options field of **BIH**. If DP is disabled (**DPON** = 0), the DP block in both an encoder or decoder shall simply output **DPVALUE** = 2 for all pixels. The discussion in the remainder of this subclause and its subclauses assumes DP is enabled (**DPON** = 1).

If DP is used when encoding an image it shall be assumed that the DP tables described in this subclause were used to make predictions unless the use of private DP tables has been signaled as described in 6.2.

### 6.6.1 Definition of associated pixels

For the purposes of describing the deterministic prediction algorithm, Figure 13 shows the labeling that will be used to refer to needed pixels from both the low-resolution and high-resolution images. Whenever reference is made to a pixel that because of edge effects is not actually in the current stripe, the value of this pixel shall be determined by the general edge rules of 6.1.2.

T0806430-90/D21
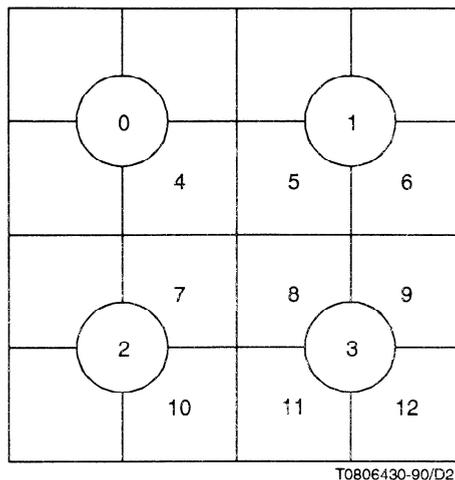
**Figure 13 – Labeling of pixels used by DP**

### 6.6.2 Default DP tables

The neighboring, or "reference", pixels which are used to make predictions for each particular spatial phase shall be as listed in Table 18. Note that for each of the four possible spatial phases a different set of pixels is used for making DP predictions. The pixels used for each possible phase are those that are labeled in Figure 13 and are known to both an encoder and a decoder at the time the particular spatial phase is to be coded. Also in this table is a number indicating how many combinations of reference pixels actually result in a DP prediction (or hit) when using the DP rules that follow.

**Table 18 – DP pixels for each spatial phase**

| Phase | Target pixel | Reference pixels | Number of hits with default resolution reduction |
|---|---|---|---|
| 0 | 8 | 0,1,2,3,4,5,6,7, | 20 |
| 1 | 9 | 0,1,2,3,4,5,6,7,8 | 108 |
| 2 | 11 | 0,1,2,3,4,5,6,7,8,9,10 | 526 |
| 3 | 12 | 0,1,2,3,4,5,6,7,8,9,10,11 | 1044 |

Private DP tables shall not use any pixels for any of the four phases other than those indicated in Table 18. The number of reference pixel patterns they will "hit" will in general be different from the numbers listed in Table 18 for the default resolution reduction algorithm.

Tables 19, 20, 21 and 22 define DP for the default resolution reduction algorithm. These four tables are to be used respectively for determining **DPVALUE** in each of the four spatial phases 0, 1, 2, and 3. The index into the table is created in the same way as the index into the resolution reduction Table 17 except that the bit significance shall be as defined by the pixel numbers given in Figure 13 rather than Figure 4.

The entries in these tables give **DPVALUE** and are all 0, 1, or 2. A "2" indicates that it is not possible to make a deterministic prediction. A "0" indicates that there is a DP "hit" and that the target pixel must be background (0). A "1" indicates that there is a DP "hit" and that the target pixel must be foreground (1). As in Table 17 the entries are read from left to right with increasing index.

**Table 19 – DP table for predicting spatial phase 0**

| Index | DPVALUE | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| [0, 63] | 02222222 | 22222222 | 22222222 | 22222222 | 02222222 | 22222222 | 22222222 | 22222222 |
| [64, 127] | 02222222 | 22222222 | 22222222 | 22222222 | 00222222 | 22222222 | 22222222 | 22222222 |
| [128, 191] | 02222222 | 22222222 | 00222222 | 22222222 | 02020222 | 22222222 | 02022222 | 22222222 |
| [192, 255] | 00222222 | 22222222 | 22222222 | 22222221 | 02020022 | 22222222 | 22222222 | 22222222 |

**Table 20 – DP table for predicting spatial phase 1**

| Index | DPVALUE | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| [0, 63] | 22222222 | 22222222 | 22222222 | 22000000 | 02222222 | 22222222 | 00222222 | 22111111 |
| [64, 127] | 22222222 | 22222222 | 22222222 | 21111111 | 02222222 | 22111111 | 22222222 | 22112221 |
| [128, 191] | 02222222 | 22222222 | 02222222 | 22222222 | 00222222 | 22222200 | 20222222 | 22222222 |
| [192, 255] | 02222222 | 22111111 | 22222222 | 22222102 | 11222222 | 22222212 | 22220022 | 22222212 |
| [256, 319] | 20222222 | 22222222 | 00220222 | 22222222 | 20000222 | 22222222 | 00000022 | 22222221 |
| [320, 383] | 20222222 | 22222222 | 11222222 | 22222221 | 22222222 | 22222221 | 22221122 | 22222221 |
| [384, 447] | 20020022 | 22222222 | 22000022 | 22222222 | 20202002 | 22222222 | 20220002 | 22222222 |
| [448, 511] | 22000022 | 22222222 | 00220022 | 22222221 | 21212202 | 22222222 | 22220002 | 22222222 |

**Table 21 – DP table for predicting spatial phase 2**

| Index | DPVALUE | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| [0, 63] | 22222222 | 12222222 | 22222222 | 22222222 | 02222222 | 12222222 | 02222222 | 11222222 |
| [64, 127] | 22222222 | 22222222 | 02222222 | 12222222 | 02222222 | 11222222 | 22221122 | 22222222 |
| [128, 191] | 00202222 | 11111111 | 00200222 | 11111111 | 00222222 | 21122222 | 10222222 | 22111222 |
| [192, 255] | 02222222 | 11222222 | 00222222 | 21222222 | 22222222 | 22202220 | 22220022 | 22112222 |
| [256, 319] | 20222222 | 21222222 | 20020222 | 22222222 | 20000222 | 22222220 | 20000022 | 22222212 |
| [320, 383] | 20220222 | 22211111 | 22020222 | 22112122 | 20000022 | 22122122 | 22002222 | 22222222 |
| [384, 447] | 20020022 | 22222200 | 22000022 | 22222212 | 22202022 | 22222222 | 20202200 | 22222212 |
| [448, 511] | 22202022 | 22222200 | 00002022 | 22222212 | 22222202 | 22222221 | 22002220 | 22212221 |
| [512, 575] | 02222222 | 22111122 | 02222222 | 11222222 | 22212122 | 22220000 | 22122122 | 22202000 |
| [576, 639] | 00000000 | 22222222 | 02222222 | 22000000 | 22002222 | 22222222 | 22002112 | 22222222 |
| [640, 703] | 20222222 | 21221122 | 20222222 | 22121222 | 22000022 | 22112122 | 02222222 | 22212222 |
| [704, 767] | 20222222 | 22222022 | 00022222 | 21111222 | 02000022 | 22222200 | 22002212 | 22222222 |
| [768, 831] | 22020222 | 22111122 | 22222022 | 22222200 | 22222022 | 22222212 | 22222202 | 22222221 |
| [832, 895] | 22000022 | 22222222 | 00201222 | 22222220 | 22022202 | 22222222 | 22002200 | 22222222 |
| [896, 959] | 22202222 | 22222222 | 22222202 | 22222221 | 22222222 | 22222221 | 22202220 | 22222222 |
| [960, 1023] | 22222222 | 22222222 | 22002202 | 22222221 | 20202220 | 22222222 | 22002220 | 22222222 |
| [1024, 1087] | 22222222 | 11111111 | 22222222 | 11111111 | 02222222 | 21111111 | 22222222 | 22222222 |
| [1088, 1151] | 22222222 | 11111111 | 02222222 | 22222222 | 02222222 | 21222222 | 22222222 | 22222002 |
| [1152, 1215] | 00222222 | 11111111 | 22222202 | 22221121 | 12222222 | 22222212 | 22002202 | 22221111 |
| [1216, 1279] | 02222222 | 21222222 | 22222222 | 22201202 | 22220222 | 22101222 | 22000022 | 22222221 |
| [1280, 1343] | 20222222 | 22112111 | 11020222 | 22211111 | 22202002 | 22222222 | 00202222 | 22222212 |
| [1344, 1407] | 22020022 | 22222211 | 22000022 | 22222212 | 22222022 | 22222212 | 22222202 | 22222212 |
| [1408, 1471] | 22002022 | 22211211 | 22222212 | 22222221 | 21222102 | 22222221 | 22222222 | 22222121 |
| [1472, 1535] | 22121122 | 22222222 | 22111122 | 22220222 | 22222200 | 22222221 | 22000002 | 22222221 |
| [1536, 1599] | 00000000 | 11111111 | 02222222 | 21222222 | 20222222 | 22121111 | 22220222 | 22121122 |
| [1600, 1663] | 00222222 | 22222222 | 22222222 | 21121222 | 20020222 | 22111111 | 22220022 | 22212122 |
| [1664, 1727] | 20220022 | 22111111 | 22020022 | 22221121 | 22000022 | 22222122 | 22220022 | 22222211 |
| [1728, 1791] | 20020222 | 22111111 | 22002222 | 22211122 | 11122222 | 22222111 | 22222202 | 22222210 |
| [1792, 1855] | 22200022 | 22222222 | 22122022 | 22222212 | 22222202 | 22222211 | 22222200 | 22222221 |
| [1856, 1919] | 22222022 | 22222222 | 22121222 | 22222222 | 22000000 | 22222211 | 22000000 | 22222211 |
| [1920, 1983] | 22222202 | 22222211 | 22222220 | 22222221 | 22222220 | 22222221 | 22222222 | 22222222 |
| [1984, 2047] | 22222200 | 22222221 | 22222220 | 22222221 | 22222222 | 22222222 | 22222220 | 22222222 |

## Table 22 – DP table for predicting spatial phase 3

| Index | DPVALUE | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| [0, 63] | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 |
| [64, 127] | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222202 | 22222212 |
| [128, 191] | 22222222 | 22222222 | 22222222 | 22222222 | 02222222 | 12222222 | 20222222 | 21222222 |
| [192, 255] | 22222222 | 22222222 | 02222222 | 12222222 | 22111121 | 22000020 | 22221122 | 22220022 |
| [256, 319] | 22222222 | 22222222 | 20000000 | 21111111 | 20000000 | 21111111 | 22000022 | 22111122 |
| [320, 383] | 20022222 | 21122222 | 22221222 | 22220222 | 22200222 | 22211222 | 22002222 | 22112222 |
| [384, 447] | 20000000 | 21111111 | 22000022 | 22111122 | 22202022 | 22212122 | 20202020 | 21212121 |
| [448, 511] | 22212111 | 22202000 | 00002022 | 11112122 | 22222212 | 22222202 | 22022222 | 22122222 |
| [512, 575] | 02222222 | 12222222 | 22222222 | 22222222 | 22020200 | 22121211 | 22211211 | 22200200 |
| [576, 639] | 00000000 | 11111111 | 00000000 | 11111111 | 20000000 | 22111111 | 22002220 | 22112221 |
| [640, 703] | 22222222 | 22222222 | 20222222 | 21222222 | 22221222 | 22220222 | 02020122 | 12101022 |
| [704, 767] | 20000000 | 21111111 | 02222222 | 12222222 | 02000100 | 12111011 | 22002220 | 22112221 |
| [768, 831] | 22222222 | 22222222 | 22222111 | 22222000 | 22222022 | 22222122 | 22222202 | 22222212 |
| [832, 895] | 22000000 | 22111111 | 00202000 | 11212111 | 22022200 | 22122211 | 22002210 | 22112201 |
| [896, 959] | 22202212 | 22212202 | 22222212 | 22222202 | 22222222 | 22222222 | 22202220 | 22212221 |
| [960, 1023] | 22222220 | 22222221 | 22002202 | 22112212 | 20202220 | 21212221 | 22002220 | 22112221 |
| [1024, 1087] | 22222222 | 22222222 | 22222222 | 22222222 | 02222222 | 12222222 | 02222222 | 12222222 |
| [1088, 1151] | 22222222 | 22222222 | 02222222 | 12222222 | 02222222 | 12222222 | 22221112 | 22220002 |
| [1152, 1215] | 22222222 | 22222222 | 22222202 | 22222212 | 22111121 | 22000022 | 22112222 | 22002222 |
| [1216, 1279] | 02222222 | 12222222 | 22112212 | 22002202 | 22212122 | 22202022 | 22010122 | 22101022 |
| [1280, 1343] | 20220222 | 21221222 | 22122222 | 22022222 | 22212111 | 22202000 | 00200022 | 11211122 |
| [1344, 1407] | 22111122 | 22000022 | 22000022 | 22111122 | 22222022 | 22222122 | 22222222 | 22222222 |
| [1408, 1471] | 22022022 | 22122122 | 22220022 | 22221122 | 22222212 | 22222202 | 22220222 | 22221222 |
| [1472, 1535] | 22202222 | 22212222 | 22222222 | 22222222 | 22222202 | 22222212 | 22111112 | 22000002 |
| [1536, 1599] | 22222222 | 22222222 | 02222222 | 12222222 | 20222222 | 21222222 | 22222222 | 22222222 |
| [1600, 1663] | 00000000 | 11111111 | 22222222 | 22222222 | 20222222 | 21222222 | 22020222 | 22121222 |
| [1664, 1727] | 20222222 | 21222222 | 22112212 | 22002202 | 22010222 | 22101222 | 22221122 | 22220022 |
| [1728, 1791] | 21222222 | 20222222 | 22022222 | 22122222 | 22201222 | 22210222 | 22222220 | 22222221 |
| [1792, 1855] | 22211111 | 22200000 | 22202022 | 22212122 | 22222222 | 22222222 | 22222202 | 22222212 |
| [1856, 1919] | 22222000 | 22222111 | 22222202 | 22222212 | 22111122 | 22000022 | 22001122 | 22110022 |
| [1920, 1983] | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 |
| [1984, 2047] | 22222202 | 22222212 | 22222222 | 22222222 | 22222222 | 22222222 | 22222220 | 22222221 |
| [2048, 2111] | 02222222 | 12222222 | 22222222 | 22222222 | 22222222 | 22222222 | 20222222 | 21222222 |
| [2112, 2175] | 22222222 | 22222222 | 22222222 | 22222222 | 20222222 | 21222222 | 22221112 | 22220002 |
| [2176, 2239] | 22020000 | 22121111 | 22122111 | 22022000 | 22122222 | 22022222 | 02010222 | 12101222 |
| [2240, 2303] | 20222222 | 21222222 | 22222222 | 22222222 | 22020122 | 22121022 | 22112222 | 22002222 |
| [2304, 2367] | 22222222 | 22222222 | 22212211 | 22202200 | 22222012 | 22222102 | 22222212 | 22222202 |
| [2368, 2431] | 22112111 | 22002000 | 22202022 | 22212122 | 22222222 | 22222222 | 22222222 | 22222222 |
| [2432, 2495] | 22202222 | 22212222 | 22222202 | 22222212 | 22222222 | 22222222 | 22222020 | 22222121 |
| [2496, 2559] | 22222222 | 22222222 | 22222202 | 22222212 | 22222220 | 22222221 | 22222222 | 22222222 |
| [2560, 2623] | 22111122 | 22000022 | 20222222 | 21222222 | 22112222 | 22002222 | 22020222 | 22121222 |
| [2624, 2687] | 22222222 | 22222222 | 21222222 | 20222222 | 22220000 | 22221111 | 22222000 | 22222111 |
| [2688, 2751] | 22221122 | 22220022 | 22020222 | 22121222 | 22222222 | 22222222 | 22111122 | 22000022 |
| [2752, 2815] | 22000200 | 22111211 | 22201222 | 22210222 | 22222222 | 22222222 | 22222200 | 22222211 |
| [2816, 2879] | 22202222 | 22212122 | 22222222 | 22222222 | 22222202 | 22222212 | 22222220 | 22222221 |
| [2880, 2943] | 22222200 | 22222211 | 22221102 | 22220012 | 22222220 | 22222221 | 22222222 | 22222222 |
| [2944, 3007] | 22222202 | 22222212 | 22222220 | 22222221 | 22222220 | 22222221 | 22222222 | 22222222 |
| [3008, 3071] | 22222220 | 22222221 | 22222220 | 22222221 | 22222222 | 22222222 | 22222222 | 22222222 |
| [3072, 3135] | 00000000 | 11111111 | 00000000 | 11111111 | 20000000 | 21111111 | 02222222 | 12222222 |
| [3136, 3199] | 00000000 | 11111111 | 22222222 | 22222222 | 21222222 | 20222222 | 22220222 | 22221222 |
| [3200, 3263] | 22000000 | 22111111 | 22220020 | 22221121 | 02010000 | 12101111 | 22220020 | 22221121 |
| [3264, 3327] | 20222222 | 21222222 | 22020222 | 22121222 | 22122022 | 22022122 | 22222220 | 22222221 |
| [3328, 3391] | 22000000 | 22111111 | 00202000 | 11212111 | 22222220 | 22222221 | 22220002 | 22221112 |
| [3392, 3455] | 22202200 | 22212211 | 22222202 | 22222212 | 22222202 | 22222212 | 22222222 | 22222222 |
| [3456, 3519] | 22220200 | 22221211 | 22220010 | 22221101 | 20222020 | 21222121 | 22220020 | 22221121 |
| [3520, 3583] | 22111122 | 22000022 | 22112022 | 22002122 | 22222222 | 22222222 | 22222220 | 22222221 |
| [3584, 3647] | 22222222 | 22222222 | 21222222 | 20222222 | 22020000 | 22121111 | 22122022 | 22022122 |
| [3648, 3711] | 22000000 | 22111111 | 21020222 | 20121222 | 22202000 | 22212111 | 22002222 | 22112222 |
| [3712, 3775] | 22002200 | 22112211 | 22202200 | 22212211 | 22222222 | 22222222 | 22222200 | 22222211 |
| [3776, 3839] | 22202000 | 22212111 | 22220022 | 22221122 | 00000000 | 11111111 | 22222222 | 22222222 |
| [3840, 3903] | 22222200 | 22222211 | 22112202 | 22002212 | 22222220 | 22222221 | 22222222 | 22222222 |
| [3904, 3967] | 22222200 | 22222211 | 22121212 | 22020202 | 22222222 | 22222222 | 22222222 | 22222222 |
| [3968, 4031] | 22222220 | 22222221 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 |
| [4032, 4095] | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 | 22222222 |

### 6.6.3    DP table format

If **DPON** = 1, **DPPRIV** = 1 and **DPLAST** = 0, the private DP table shall be encoded into the **DPTABLE** field of **BIH** (see 6.2). The **DPTABLE** field shall be defined by paralleling the structure of the four tables above that define DP for the default resolution reduction algorithm. In particular, it shall be a concatenation of the four tables with two bits allocated to each table entry so that four entries pack into one byte. Typographically leftmost and uppermost entries in the tables shall pack into higher order bits in bytes and earlier bytes in the **DPTABLE** field. When it is not possible to make a DP prediction, 2 shall be coded into the two-bit field.

NOTE – This permission is granted so that the most significant bit of the two-bit field becomes by itself an indication that a DP prediction can not be made.

Because there are 8, 9, 11, and 12 reference pixels for predicting respectively pixels in spatial phases 0, 1, 2, and 3, the **DPTABLE** field will be

$$1728 = 2 \times (256 + 512 + 2048 + 4096)/8 \tag{7}$$

bytes in length.

## 6.7    Model templates and adaptive templates

Model templates define a neighborhood around a pixel to be coded. The values of the pixels in these neighborhoods, plus spatial phase in differential layers, define a context, with a separate arithmetic coding adapter used for each different context (see 6.8). Although a template is a geometric pattern of pixels, the pixels in a template are said to take on values when the template is aligned to a particular part of the image.

### 6.7.1    Lowest resolution layer

Figure 14 shows the template which shall be used when encoding the lowest resolution layer when **LRLTWO** is 0.



T0806440-90/D22
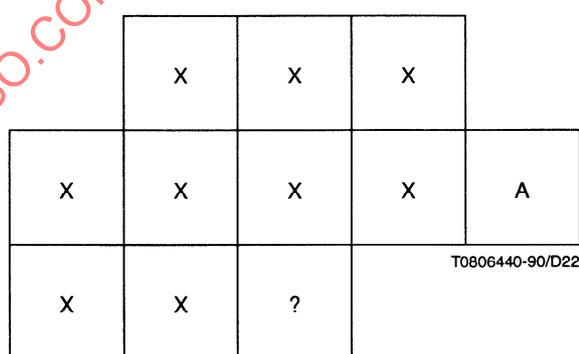
Figure 14 – Three-line model template for lowest resolution layer

The pixel denoted by a "?" corresponds to the pixel to be coded and is not part of the template. The pixels denoted by "X" correspond to ordinary pixels in the template, and the pixel denoted by "A" is a special pixel in the template that is called an "adaptive" or "AT" pixel. This pixel is special in that its position is allowed to change during the process of encoding an image. See 6.7.3 for a description of AT pixels. The "A" indicates the initial location of the AT pixel.

The values of the pixels in the template shall be combined to form a context. Each pixel in the model template (including the adaptive pixel) shall correspond to a specific bit in the context, although the pixels in the template may be assigned to bits in the context in any order. Because there are 10 pixels in this template, contexts associated with the lowest resolution layer can take on 1024 different values. This context shall be used to identify which arithmetic coder adapter is to be used for encoding the pixel to be coded, as described in 6.8.

If **LRLTWO** is 1, the lowest-resolution-layer model-template shall be that shown in Figure 15.

| | X | X | X | X | X | A |
|---|---|---|---|---|---|---|
| X | X | X | X | ? | | |

T0808700-91/D23
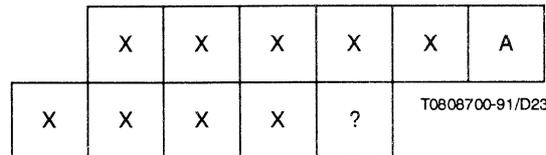
**Figure 15 – Two-line model template for lowest resolution layer**

The meaning of the labels "X", "A", and "?" is as before.

NOTE – Software execution speed in the lowest resolution layer will be somewhat faster with the two-line template than the three-line template. The penalty in using the two-line template is about a 5% loss in compression efficiency.

Whenever any of the pixels in the templates of Figures 14 or 15 (as dictated by **LRLTWO**) lie outside the boundaries of the image or stripe, the general edge rules of 6.1.2 shall be used.

### 6.7.2 Differential layer

Figure 16 shows the templates that shall be used when encoding differential-layer images. Notice that these templates contain references to pixels in the next lower resolution image as well as to pixels in the image being encoded, and that the model template is different for different phases. The symbols "?", "X", and "A" have the same meaning as in the previous subclause.

Contexts shall be formed from these templates in a way similar to what was described for the lowest-resolution-layer template. Each pixel in a template shall contribute one bit to the context. In addition, when encoding differential-layer images, two additional bits shall be added to the context to describe the phase of the pixel being encoded. As before, any particular bits may be used to describe the phase information, although the assignment of pixels and phase information to bits in the context shall remain fixed while encoding an image. Because there are 10 pixels in the differential-layer templates and because 2 bits are used to describe phase information, there are 4096 different possible contexts while processing differential images. This context shall be used to identify which arithmetic coder adapter is to be used for encoding the pixel to be coded.

### 6.7.3 Adaptive template pixels

In coding the differential layers as well as the lowest-resolution layer, the model template shall be allowed to change in the restricted way described in this subclause.

The single pixel that is allowed to change shall be called the AT pixel. Its initial (or default) location is indicated by "A" in Figures 14, 15, and 16 for, respectively, lowest-resolution-layer coding with a three-line template, lowest-resolution-layer coding with a two-line template, and differential-layer coding. In general, the AT pixel can be moved independently for all layers to anywhere in the field shown in Figure 17.

However, the new AT location shall not overlap any regular pixels in the template. (Hence permissible movements for two-line lowest-resolution-layer coding, three-line lowest-resolution-layer coding, and differential-layer coding are slightly different.)

A differential-layer AT movement is effective for all four phases simultaneously. If there is more than one differential layer, the movements in each are independent. However, in any one layer, once the AT pixel is moved for a particular stripe, subsequent stripes shall continue to use the new location.

The parameters $M_X$ and $M_Y$ define the size of the rectangle in Figure 17. Absolute limits and suggested minimum support are as in Tables 9 and A.1.
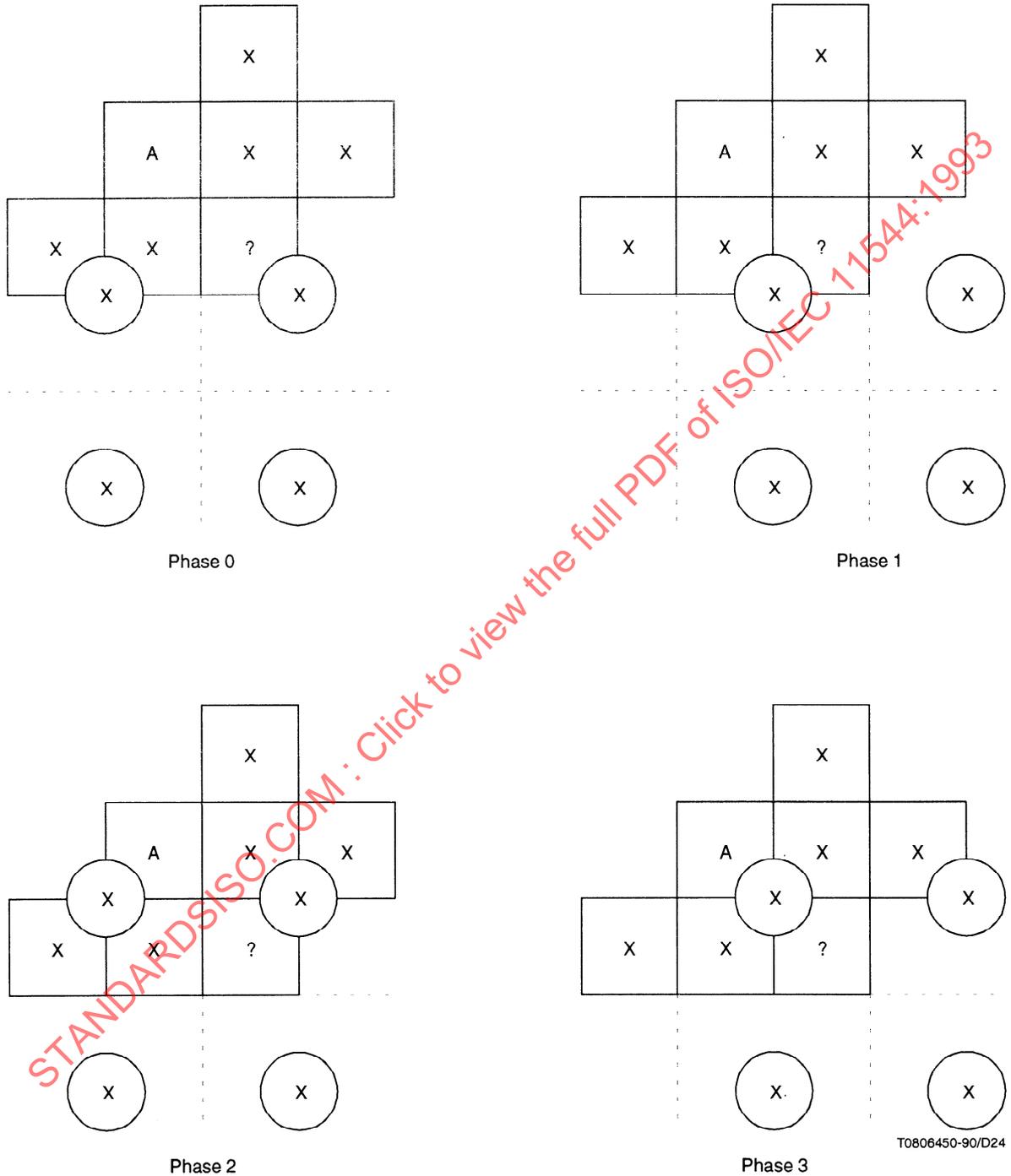
**Figure 16 – Model templates for differential-layer coding**

| $M_X, M_Y$ | ........ | 2, $M_Y$ | 1, $M_Y$ | 0, $M_Y$ | -1, $M_Y$ | -2, $M_Y$ | ........ | $-M_X, M_Y$ |

| $M_X, 1$ | ........ | 2, 1 | 1, 1 | 0, 1 | -1, 1 | -2, 1 | ........ | $-M_X, 1$ |
| $M_X, 0$ | ........ | 2, 0 | 1, 0 | ? | | | | |

T0808710-91/D25
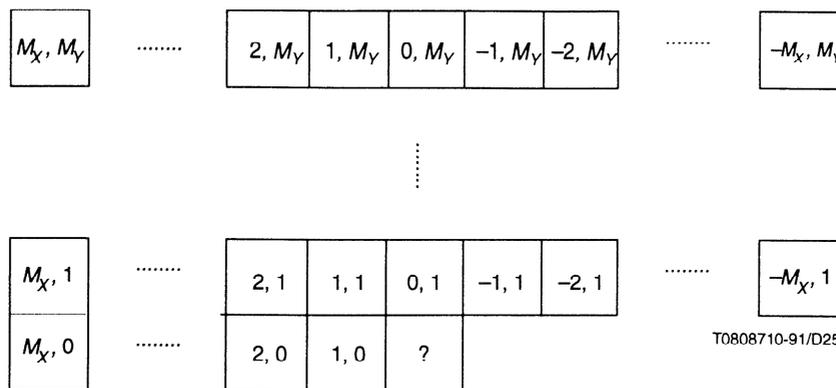
**Figure 17 – Field to which AT movements are restricted**

NOTE – Because it is more costly, in general, for hardware to support vertical movement of the AT pixel, the suggested minimum support for $M_Y$ is restricted to zero so that AT pixel movement to anywhere but the default location is restricted to the current line being coded.

If an encoder wishes to change the location of the AT pixel, it shall inform the decoder of the change by coding $\tau_X$, $\tau_Y$, and $y_{AT}$ as indicated in Table 14. The numbers coded into $\tau_X$ and $\tau_Y$ shall be, respectively, the horizontal and vertical offsets from the target pixel as shown in Figure 17. The possibly negative number $\tau_X$ shall be encoded in twos-complement form. The number coded into $y_{AT}$ shall be the number of the high-resolution line at the beginning of which the change shall be made. The line numbering used shall restart with 0 at the top of each stripe.

It is permissible to move the AT pixel back to its initial (or default) location after having once moved it away. The default location for the AT pixel shall always be coded by $\tau_X = 0$ and $\tau_Y = 0$ rather than the true X and Y coordinates.

NOTE – This convention is convenient because the true coordinates are different for lowest-resolution-layer coding and differential-layer coding. It also makes it possible for an encoder to inform a decoder that there will never be any AT movements by setting $M_X$ and $M_Y$ both equal to 0.

Annex C describes a computationally simple technique for making determinations of when an AT pixel change is desirable and where it should be moved.

## 6.8 Arithmetic coding

The entropy coder used in this Specification is an adaptive arithmetic compression coder. In this subclause and all its subclauses, the flow diagrams and Table 24 are normative only in the sense that they are defining an output that alternative implementations must duplicate. All background information and discussion in the subclauses of this subclause is informative.

NOTE – It is intended that the arithmetic coding operations described in this subclause be identical to the arithmetic coding operations described in CCITT Rec. T.81 I ISO/IEC 10918-1. However, should there be any unintentional difference in the descriptions, the procedure described in this Specification shall be used.

It is permissible to have as many as four **ATMOVE** marker segments in any one stripe. When there are multiple **ATMOVE** marker segments in one stripe, their effective lines $Y_{AT}$ shall be distinct and ordered with earlier **ATMOVE** markers having numerically lower $Y_{AT}$ values

For each stripe of each resolution-layer, the arithmetic encoder shall produce a byte stream **SCD**. It shall have four streams as inputs, each of them providing one value for each pixel in the stripe being coded. As shown in Figure 18 these four inputs shall be a pixel **PIX**, a context value **CX**, a TP indication **TPVALUE**, and a DP indication **DPVALUE**.

PIX ——
CX ——  Arithmetic encoder  —— SCD
TPVALUE ——
DPVALUE ——

T0806480-90/D26
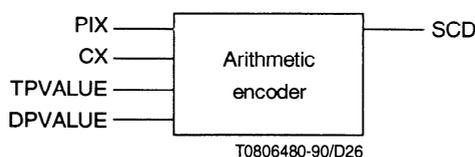
**Figure 18 – Encoder inputs and outputs**

The pixel to be coded is generally just a pixel from the image, but if lowest-resolution-layer TP or differential-layer TP is enabled, it will occasionally be a pseudo-pixel, **LNTP** (differential-layer) or **SLNTP** (lowest-resolution-layer). The inputs **CX**, **TPVALUE**, and **DPVALUE** are generated as described in 6.7, 6.4, 6.5 and 6.6.

For each stripe of each resolution layer the arithmetic decoder shall read a byte stream **SCD**. As shown in Figure 19 this stream along with the per-pixel inputs streams **CX**, **TPVALUE**, and **DPVALUE** shall be processed to recreate the stream **PIX**.
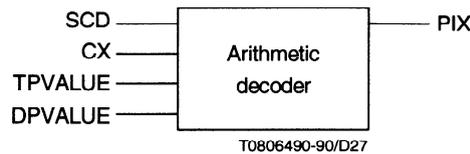


Figure 19 – Decoder inputs and outputs

The inputs **CX**, **TPVALUE**, and **DPVALUE** are identical to those used in the encoder.

It is simplest to specify the requirements for the encoder and decoder by describing sample procedures. The sample procedures are defined by flow diagrams and a table. Many equivalent procedures exist. Some will have speed, memory-usage, or simplicity advantages over others. Some are more suitable for hardware implementation and others more suitable for software implementation. The choice here was weighted in favor of greatest simplicity and conciseness. Any encoding or decoding procedures producing the same outputs as the sample procedures may be used. This output equivalence shall be the **only** requirement.

### 6.8.1 Fundamental arithmetic coding concepts (informative)

### 6.8.1.1 Interval subdivision

Recursive probability interval subdivision is the basis for arithmetic coding. Conceptually, an input sequence of symbols is mapped into a real number $x$ on the interval [0,1) where a square bracket on an interval end denotes equality being allowed and a curved bracket denotes it being disallowed. What is transmitted or stored instead of the original sequence of symbols is the binary expansion of $x$.

Figure 20 shows an example of such interval division through an initial sequence 0, 1, 0, 0 to be coded.
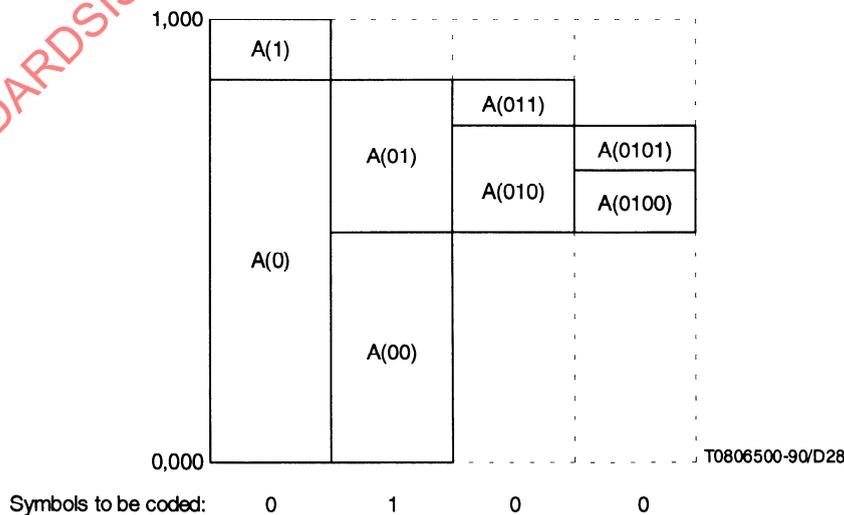


Figure 20 – Interval subdivision

The portion of [0,1) on which x is known to lie after coding an initial sequence of symbols is known as the current coding interval. For each binary input the current coding interval is divided into two sub-intervals with sizes proportional to the relative probabilities of symbol value occurrences. The new current coding interval becomes that associated with the symbol value that actually occurred. In an encoder, knowledge of the current coding interval is maintained in a variable giving its size and a second variable giving its base (lower bound). The output stream is obtained from the variable pointing to the base.

In the partitioning of the current interval into two sub-intervals, the sub-interval for the less probable symbol (LPS) is ordered above the sub-interval for the more probable symbol (MPS). Therefore, when the LPS is coded, the MPS sub-interval is added to the base. This coding convention requires that symbols be recognized as either MPS or LPS, rather than 0 or 1. Consequently, the size of the LPS interval and the sense of the MPS for each symbol must be known in order to code that symbol.

Since the code stream always points to a real number in the current coding interval, the decoding process is a matter of determining, for each decision, which sub-interval is pointed to by the code string. This is also done recursively, using the same interval sub-division process as in the encoder. Each time a decision is decoded, the decoder subtracts any interval the encoder added to the code stream. Therefore, the code stream in the decoder is a pointer into the current interval relative to the base of the current interval.

Since the coding process involves addition of binary fractions rather than concatenation of integer code words, the more probable binary decisions can often be coded at a cost of much less than one bit per decision.

### 6.8.1.2 Coding conventions and approximations

It is possible to perform these coding operations using fixed precision integer arithmetic. A register **A** contains the size of the current coding interval normalized to always lie in the range [0x8000,0x10000] where an "0x" prefix denotes a hexadecimal integer. Whenever as a result of coding a symbol **A** temporarily falls below 0x8000, it is doubled recursively until it is greater than or equal to 0x8000. Such doublings are termed "renormalizations".

A second register, **C**, contains the trailing bits of the code stream. The register **C** is also doubled each time **A** is doubled. Periodically (to keep **C** from overflowing) a byte of data is removed from the high order bits of the **C** register and placed in an external code string buffer. Possible carry-over must be resolved before the contents of this buffer is committed to output.

A simple arithmetic approximation is used in the interval subdivision. For an interval **A** and a current estimate p of the LPS probability, a precise calculation of the LPS sub-interval would require a multiplication $p \times $ **A**. Instead, the approximation

$$p \times A \approx p \times \overline{A} = \text{LSZ} \tag{8}$$

is used where the overscore denotes an average over the probability density of **A** and **LSZ** is a stored quantity equal to the size of the approximated interval for the LPS. Because **A** is kept in the range [0x8000,0x10000], replacing **A** by its statistical average does not introduce too great an error. Empirically, **A** is found to have a probability density inversely proportional to **A**.

Whenever the LPS is coded, the value of the MPS sub-interval **A-LSZ** is added to the code register and the coding interval is reduced to the value **LSZ** of the LPS sub-interval. Whenever the MPS is coded, the code register is left unchanged and the interval is reduced to **A-LSZ**. If **A** falls below 0x8000 in performing these operations, it is restored to the proper range by renormalizing both **A** and **C**.

With the process sketched above, the approximation in the interval subdivision process can sometimes make the LPS sub-interval larger than the MPS sub-interval. If, for example, the value of **LSZ** is 0,33 × 0x10000 and **A** is at the minimum allowed value of 0x8000, the approximate scaling gives 1/3 of the interval to the MPS and 2/3 to the LPS. To avoid this size inversion, the interval is subdivided using this simple approximation, but the MPS and LPS interval assignments are exchanged whenever the LPS interval is larger than the MPS interval. This MPS/LPS "conditional exchange" can only occur when a renormalization will be needed.

Whenever a renormalization occurs, a probability estimation process is invoked which determines a new probability estimate for the context currently being coded.

### 6.8.2    Encoder

### 6.8.2.1    Encoder flow diagram

This flow diagram is executed for each stripe of each resolution layer. Pixels that are not typically predictable and are not deterministically predictable are coded with the procedure **ENCODE**. The initialization procedure **INITENC** is called on entry, and the termination procedure **FLUSH**, on exit (see Figure 21).
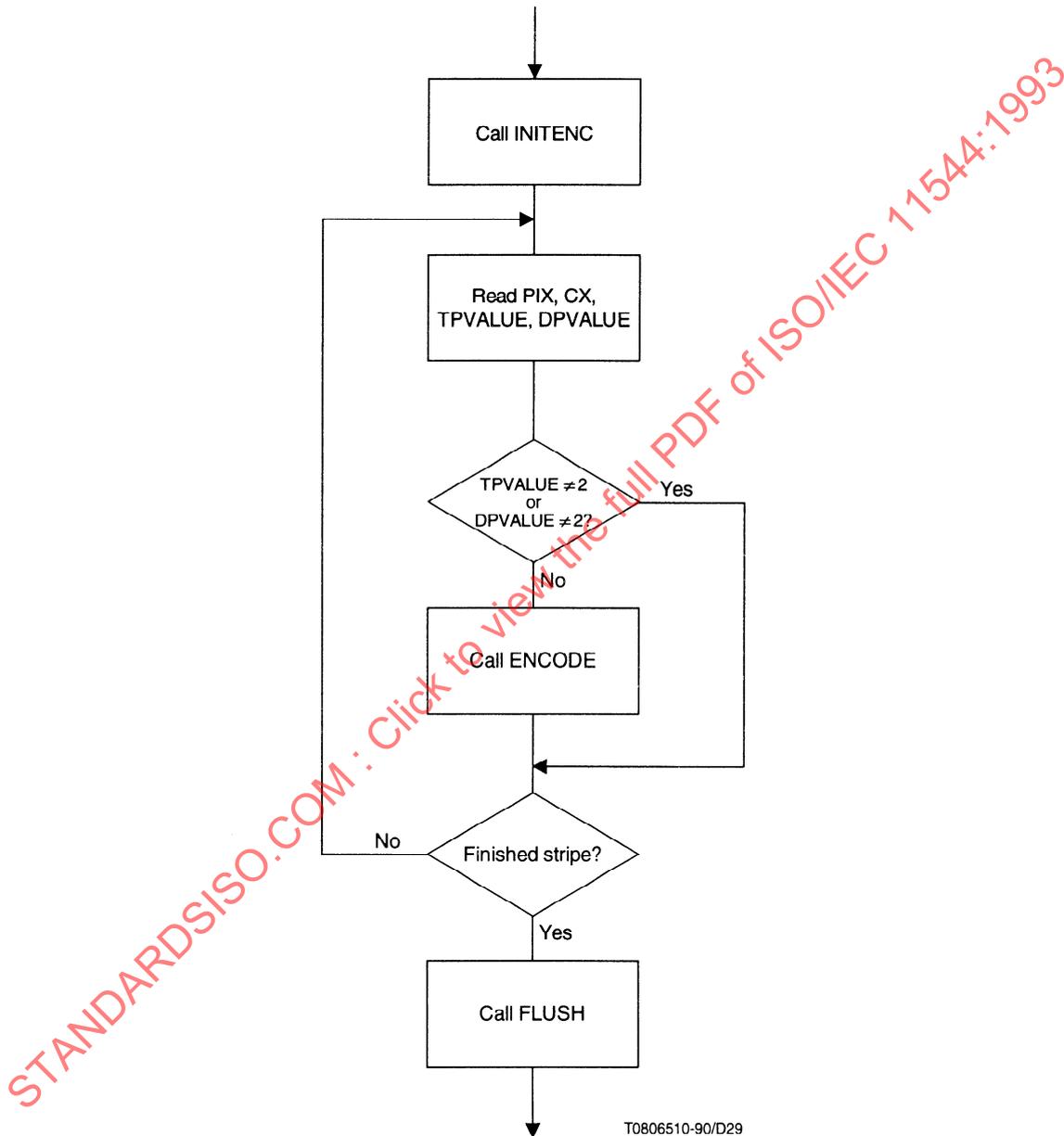


Figure 21 – Encoder flow diagram

### 6.8.2.2 Encoder code register conventions

The flow diagrams given in this subclause assume the register structure shown in Table 23.

**Table 23 – Encoder register structure**

|  | msb |  |  | lsb |
|---|---|---|---|---|
| **C** register | 0000cbbb, | bbbbbsss, | xxxxxxxx, | xxxxxxxx |
| **A** register | 00000000, | 0000000a, | aaaaaaaa, | aaaaaaaa |

The "a" bits are the fractional bits in the current interval value and the "x" bits are the fractional bits in the code register. The "s" bits are spacer bits, at least one of which is needed to constrain carry-over, and the "b" bits indicate the bit positions from which the completed bytes of data are removed from the **C** register. The "c" bit is a carry bit. The seventeenth **A** register bit is conceptually present and hence shown here, but it can easily be avoided if a 16-bit implementation is desired. In this case, initializing to 0x0000 instead of 0x10000 works properly as long as underflow in the underlying hardware or software produces the same low order 16 bits on subtracting from 0x0000 as on subtracting from 0x10000. Such behavior is the usual.

These register conventions illustrate one possible implementation. Here especially, there are many other possibilities.

### 6.8.2.3 Probability estimation tables

For each possible value of the context **CX** there is stored a one-bit value **MPS[CX]** and a seven-bit value **ST[CX]**, which together completely capture the adaptive probability estimate associated with that particular context. Four arrays indexed by **ST[CX]** are shown in Table 24.

The color **MPS** is the (estimated) most likely color for **PIX**. **LSZ** is the LPS interval size, which can be interpreted to a probability via equation 9, although no such interpretation need be made as only **LSZ** ever enters subsequent calculations.

The arrays **NLPS** and **NMPS** give, respectively, the next probability-estimation state for an observation of the LPS and the MPS. The movement given by **NMPS** only occurs if in addition to observing the MPS, a renormalization also occurs. When the movement given by **NLPS** occurs, there will also be an inversion of **MPS[CX]** if **SWTCH[CX]** is 1.

Annex D (informative) explains why the entries in Table 24 are the way they are.

### 6.8.2.4 Flow diagram for the procedure ENCODE

If the current symbol **PIX** equals the value currently thought to be most probable, the routine **CODEMPS** is called. Otherwise, **CODELPS** is called (see Figure 22).

### 6.8.2.5 Flow diagram for the procedure CODELPS

The **CODELPS** procedure normally consists of the addition of the MPS sub-interval **A-LSZ[ST[CX]]** to the code stream and a scaling of the interval to the sub-interval **LSZ[ST[CX]]**. It is always followed by a renormalization. If **SWTCH[ST[CX]]** is 1, **MPS[CX]** is inverted.

However, in the event that the LPS sub-interval is larger than the MPS sub-interval, the conditional MPS/LPS exchange occurs and the MPS sub-interval is coded (see Figure 23).

### 6.8.2.6 Flow diagram for the procedure CODEMPS

The **CODEMPS** procedure normally reduces the size of the interval to the MPS sub-interval. However, if the LPS sub-interval is larger than the MPS sub-interval, the conditional exchange occurs and the LPS sub-interval is coded instead. Note that this interval size inversion cannot occur unless a renormalization is required after the coding of the symbol (see Figure 24).

## Table 24 – Probability estimation table

| ST | LSZ | NLPS | NMPS | SWTCH | ST | LSZ | NLPS | NMPS | SWTCH |
|----|-----|------|------|-------|----|-----|------|------|-------|
| 0 | 0x5a1d | 1 | 1 | 1 | 57 | 0x01a4 | 55 | 58 | 0 |
| 1 | 0x2586 | 14 | 2 | 0 | 58 | 0x0160 | 56 | 59 | 0 |
| 2 | 0x1114 | 16 | 3 | 0 | 59 | 0x0125 | 57 | 60 | 0 |
| 3 | 0x080b | 18 | 4 | 0 | 60 | 0x00f6 | 58 | 61 | 0 |
| 4 | 0x03d8 | 20 | 5 | 0 | 61 | 0x00cb | 59 | 62 | 0 |
| 5 | 0x01da | 23 | 6 | 0 | 62 | 0x00ab | 61 | 63 | 0 |
| 6 | 0x00e5 | 25 | 7 | 0 | 63 | 0x008f | 61 | 32 | 0 |
| 7 | 0x006f | 28 | 8 | 0 | 64 | 0x5b12 | 65 | 65 | 1 |
| 8 | 0x0036 | 30 | 9 | 0 | 65 | 0x4d04 | 80 | 66 | 0 |
| 9 | 0x001a | 33 | 10 | 0 | 66 | 0x412c | 81 | 67 | 0 |
| 10 | 0x000d | 35 | 11 | 0 | 67 | 0x37d8 | 82 | 68 | 0 |
| 11 | 0x0006 | 9 | 12 | 0 | 68 | 0x2fe8 | 83 | 69 | 0 |
| 12 | 0x0003 | 10 | 13 | 0 | 69 | 0x293c | 84 | 70 | 0 |
| 13 | 0x0001 | 12 | 13 | 0 | 70 | 0x2379 | 86 | 71 | 0 |
| 14 | 0x5a7f | 15 | 15 | 1 | 71 | 0x1edf | 87 | 72 | 0 |
| 15 | 0x3f25 | 36 | 16 | 0 | 72 | 0x1aa9 | 87 | 73 | 0 |
| 16 | 0x2cf2 | 38 | 17 | 0 | 73 | 0x174e | 72 | 74 | 0 |
| 17 | 0x207c | 39 | 18 | 0 | 74 | 0x1424 | 72 | 75 | 0 |
| 18 | 0x17b9 | 40 | 19 | 0 | 75 | 0x119c | 74 | 76 | 0 |
| 19 | 0x1182 | 42 | 20 | 0 | 76 | 0x0f6b | 74 | 77 | 0 |
| 20 | 0x0cef | 43 | 21 | 0 | 77 | 0x0d51 | 75 | 78 | 0 |
| 21 | 0x09a1 | 45 | 22 | 0 | 78 | 0x0bb6 | 77 | 79 | 0 |
| 22 | 0x072f | 46 | 23 | 0 | 79 | 0x0a40 | 77 | 48 | 0 |
| 23 | 0x055c | 48 | 24 | 0 | 80 | 0x5832 | 80 | 81 | 1 |
| 24 | 0x0406 | 49 | 25 | 0 | 81 | 0x4d1c | 88 | 82 | 0 |
| 25 | 0x0303 | 51 | 26 | 0 | 82 | 0x438e | 89 | 83 | 0 |
| 26 | 0x0240 | 52 | 27 | 0 | 83 | 0x3bdd | 90 | 84 | 0 |
| 27 | 0x01b1 | 54 | 28 | 0 | 84 | 0x34ee | 91 | 85 | 0 |
| 28 | 0x0144 | 56 | 29 | 0 | 85 | 0x2eae | 92 | 86 | 0 |
| 29 | 0x00f5 | 57 | 30 | 0 | 86 | 0x299a | 93 | 87 | 0 |
| 30 | 0x00b7 | 59 | 31 | 0 | 87 | 0x2516 | 86 | 71 | 0 |
| 31 | 0x008a | 60 | 32 | 0 | 88 | 0x5570 | 88 | 89 | 1 |
| 32 | 0x0068 | 62 | 33 | 0 | 89 | 0x4ca9 | 95 | 90 | 0 |
| 33 | 0x004e | 63 | 34 | 0 | 90 | 0x44d9 | 96 | 91 | 0 |
| 34 | 0x003b | 32 | 35 | 0 | 91 | 0x3e22 | 97 | 92 | 0 |
| 35 | 0x002c | 33 | 9 | 0 | 92 | 0x3824 | 99 | 93 | 0 |
| 36 | 0x5ae1 | 37 | 37 | 1 | 93 | 0x32b4 | 99 | 94 | 0 |
| 37 | 0x484c | 64 | 38 | 0 | 94 | 0x2e17 | 93 | 86 | 0 |
| 38 | 0x3a0d | 65 | 39 | 0 | 95 | 0x56a8 | 95 | 96 | 1 |
| 39 | 0x2ef1 | 67 | 40 | 0 | 96 | 0x4f46 | 101 | 97 | 0 |
| 40 | 0x261f | 68 | 41 | 0 | 97 | 0x47e5 | 102 | 98 | 0 |
| 41 | 0x1f33 | 69 | 42 | 0 | 98 | 0x41cf | 103 | 99 | 0 |
| 42 | 0x19a8 | 70 | 43 | 0 | 99 | 0x3c3d | 104 | 100 | 0 |
| 43 | 0x1518 | 72 | 44 | 0 | 100 | 0x375e | 99 | 93 | 0 |
| 44 | 0x1177 | 73 | 45 | 0 | 101 | 0x5231 | 105 | 102 | 0 |
| 45 | 0x0e74 | 74 | 46 | 0 | 102 | 0x4c0f | 106 | 103 | 0 |
| 46 | 0x0bfb | 75 | 47 | 0 | 103 | 0x4639 | 107 | 104 | 0 |
| 47 | 0x09f8 | 77 | 48 | 0 | 104 | 0x415e | 103 | 99 | 0 |
| 48 | 0x0861 | 78 | 49 | 0 | 105 | 0x5627 | 105 | 106 | 1 |
| 49 | 0x0706 | 79 | 50 | 0 | 106 | 0x50e7 | 108 | 107 | 0 |
| 50 | 0x05cd | 48 | 51 | 0 | 107 | 0x4b85 | 109 | 103 | 0 |
| 51 | 0x04de | 50 | 52 | 0 | 108 | 0x5597 | 110 | 109 | 0 |
| 52 | 0x040f | 50 | 53 | 0 | 109 | 0x504f | 111 | 107 | 0 |
| 53 | 0x0363 | 51 | 54 | 0 | 110 | 0x5a10 | 110 | 111 | 1 |
| 54 | 0x02d4 | 52 | 55 | 0 | 111 | 0x5522 | 112 | 109 | 0 |
| 55 | 0x025c | 53 | 56 | 0 | 112 | 0x59eb | 112 | 111 | 1 |
| 56 | 0x01f8 | 54 | 57 | 0 | | | | | |

T0806520-90/D30
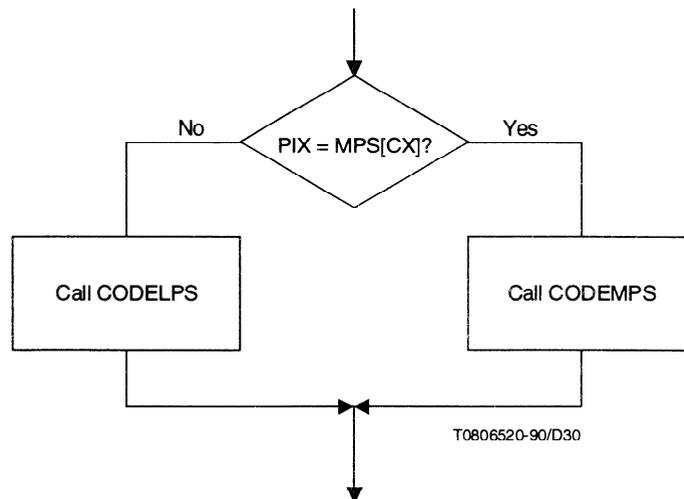
**Figure 22 – Flow diagram for the procedure ENCODE**

### 6.8.2.7    Flow diagram for the procedure RENORME

Both the interval register **A** and the code register **C** are shifted, one bit at a time. The number of shifts is counted in the counter **CT**, and when **CT** is counted down to zero, a byte of compressed data is removed from **C** by the procedure **BYTEOUT**. Renormalization continues until **A** is no longer less than 0x8000 (see Figure 25).

### 6.8.2.8    Flow diagram for the procedure BYTEOUT

The procedure **BYTEOUT** is called from **RENORME**. The variable **TEMP** is a temporary variable that holds the byte at the top of the **C** register that is to be output plus a carry indication. The variable **BUFFER** holds the most recent tentative output that was unequal to 0xff. The counter **SC** holds the number of 0xff bytes there have been since the byte in **BUFFER** was tentatively output (see Figure 26).

The shift of the code register by 19 bits aligns the output bits "b" with the low order bits of **TEMP**. The first test then determines if a carry-over has occurred. If so, the carry must be added to the tentative output byte in **BUFFER** before it is finally committed to output. Any stacked output bytes (converted to zeros by the carry) are then output. Finally the new tentative output byte **BUFFER** is set equal to **TEMP** less any carry.

If a carry has not occurred, the output byte is checked to see if it is 0xff. If so, the stack count **SC** is incremented, as the output must be delayed until the carry is resolved. If not, the carry has been resolved, and any stacked 0xff bytes may be output.

NOTE – The probability that the counter **SC** will reach a given integer $n$ falls off rapidly as $2^{-8n}$ so that in practice values of **SC** beyond 3 or 4 are rarely seen in coding an image. However, in principle **SC** can become as large as the number of bytes in the output file **SCD**. Whenever carry is resolved, the input image can not be processed while **SC** 0x00 or 0xff bytes are output. Since **SC** can become in principle quite large, this halt can also in principle become quite long.

If it is important for a particular implementation, the reserved marker can be used to finitely bound any halt of the processing of the input image. One way is to insert the reserved marker whenever **SC** reaches some small number, say 8, and then decrement **SC** by 8. Then, in a postprocessing step within the encoder each of these markers is replaced by either eight 0xff bytes or eight 0x00 bytes as appropriate. This postprocessing must be done by the encoder as no decoder is expected to know anything about such an application of the reserved marker. Such a use of the reserved marker byte only works with **PSCD** data dynamically created from **SCD** data as it is generated.

As a second way to use the reserved marker to this same end, let **SC** build up to arbitrarily large values. If **SC** is larger than a given number (e.g. eight) when carry is eventually resolved, output the reserved marker followed by an 0x00 or 0xff to indicate the carry resolution value. Then encode the actual **SC** with additional bytes. Again, encoder postprocessing is required to replace this marker by the proper number of 0xff or 0x00 bytes.

T0806530-90/D31
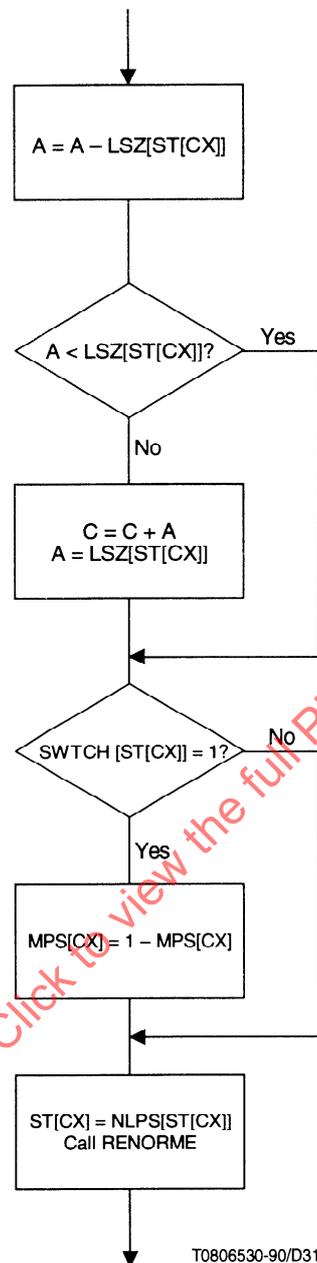
**Figure 23 – Flow diagram for the procedure CODELPS**

### 6.8.2.9 Flow diagram for the procedure INITENC

If this stripe is at the top of the image, the probability-estimation states for all possible values of **CX** are set to 0 (that is, the equiprobable state). Otherwise, they are reset to their values at the end of the last stripe at this resolution. The stack count **SC** and the code register **C** are cleared. The counter **CT** is set to 11 (a byte plus the 3 spacer bits). The coding interval register **A** is set to 0x10000. Alternatively, for 16-bit implementation, it can be set to 0x0000 as long as the hardware or software produces the same 16 bits on subtracting a 16-bit quantity from 0 as is obtained in mathematically subtracting from 0x10000. This will almost always be the case (see Figure 27).
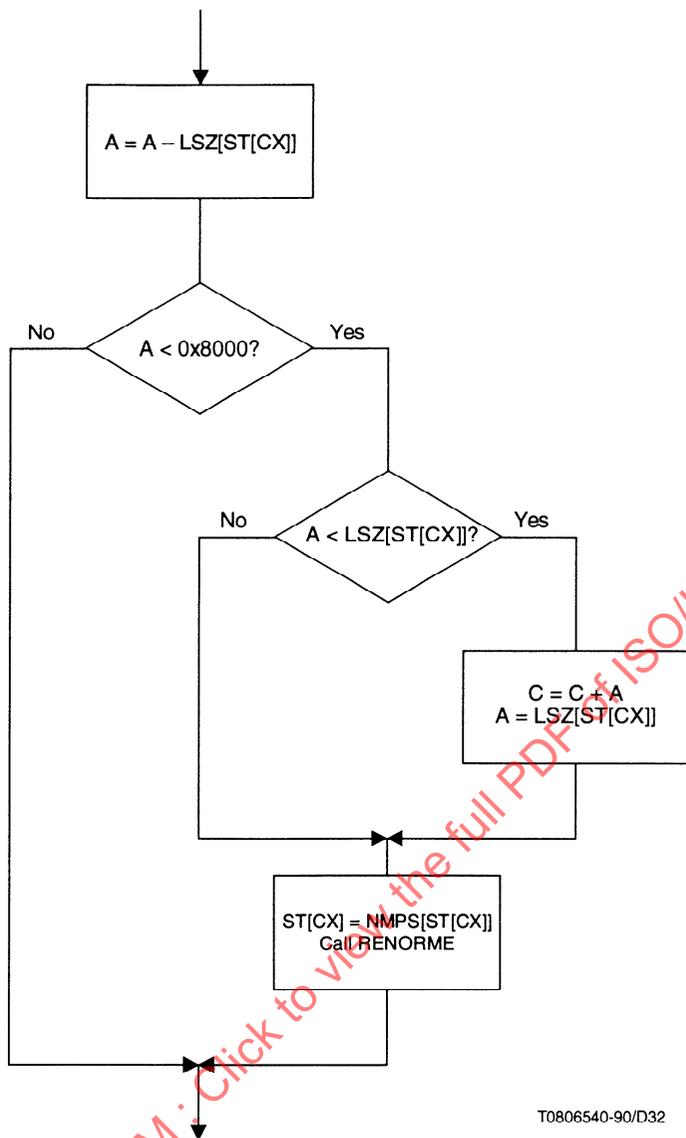
Figure 24 – Flow diagram for the procedure CODEMPS

### 6.8.2.10 Flow diagram for the procedure FLUSH

Two subprocedures are called first. Then the first byte that was written into the stream SCD is removed and if desired some or all of any 0x00 bytes at the end of SCD are also removed until finally coming to a byte unequal to 0x00. Good software and hardware implementations will set up auxiliary variables so that these bytes are never written in the first place. The implementation described here was chosen for simplicity and conciseness (see Figure 28).

### 6.8.2.11 Flow diagram for the procedure CLEARBITS

The code register C is set to the value in [C, C+A-1] that ends with the greatest possible number of zero bits (see Figure 29).

### 6.8.2.12 Flow diagram for the procedure FINALWRITES

The final carry resolution is performed and two bytes from C are written (see Figure 30).

T0806550-90/D33
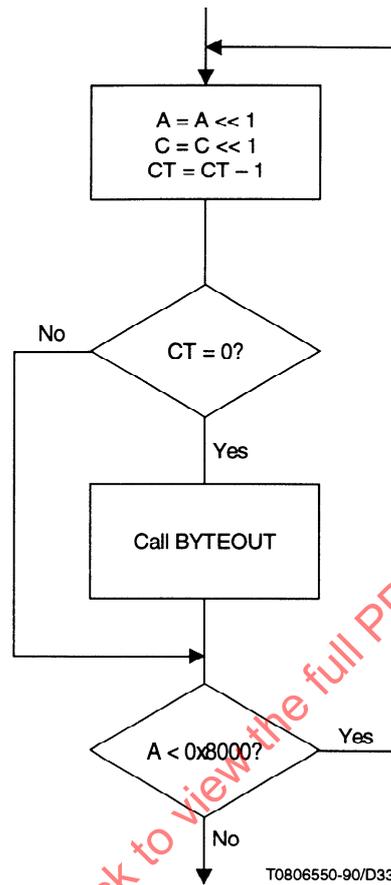
**Figure 25 – Flow diagram for the procedure RENORME**

### 6.8.3 Decoder

#### 6.8.3.1 Decoder flow diagram

This flow diagram is executed for each stripe of each resolution layer. Pixels that are not typically predictable and are not deterministically predictable are decoded by the procedure **DECODE**. The initialization procedure **INITDEC** is called on entry (see Figure 31).

#### 6.8.3.2 Decoder code register conventions

The flow diagrams given in this subclause assume the register structure shown in Table 25.
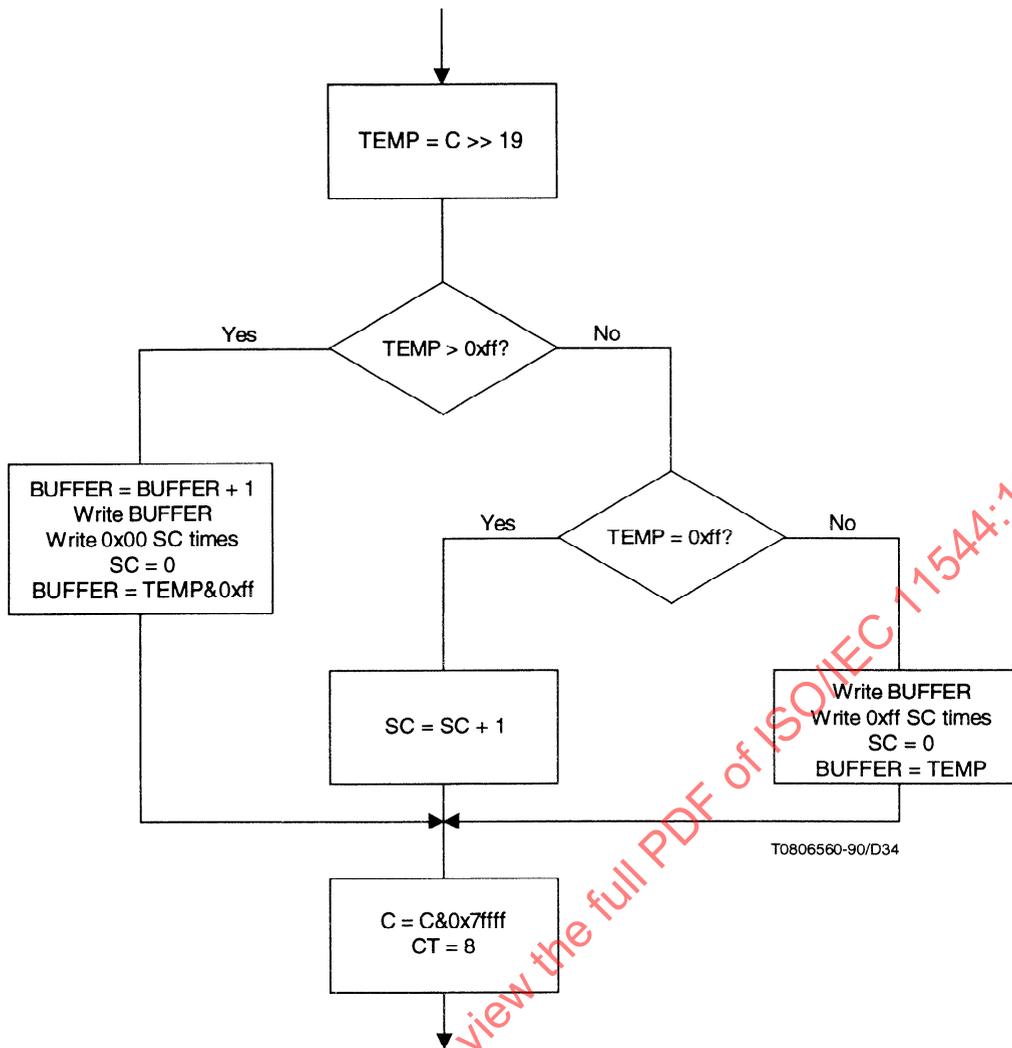
**Figure 26 – Flow diagram for the procedure BYTEOUT**
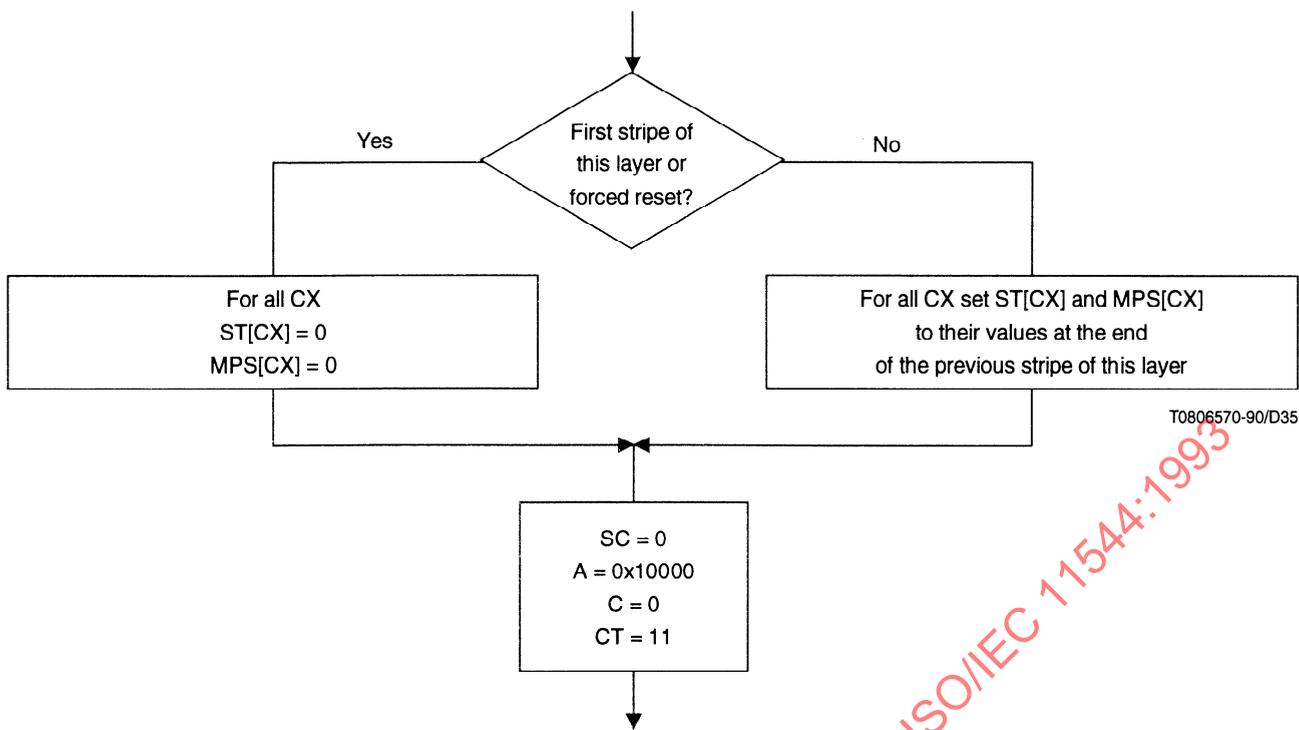
**Figure 27 – Flow diagram for the procedure INITENC**
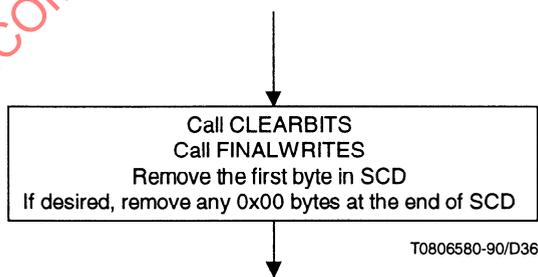


**Figure 28 – Flow diagram for the procedure FLUSH**
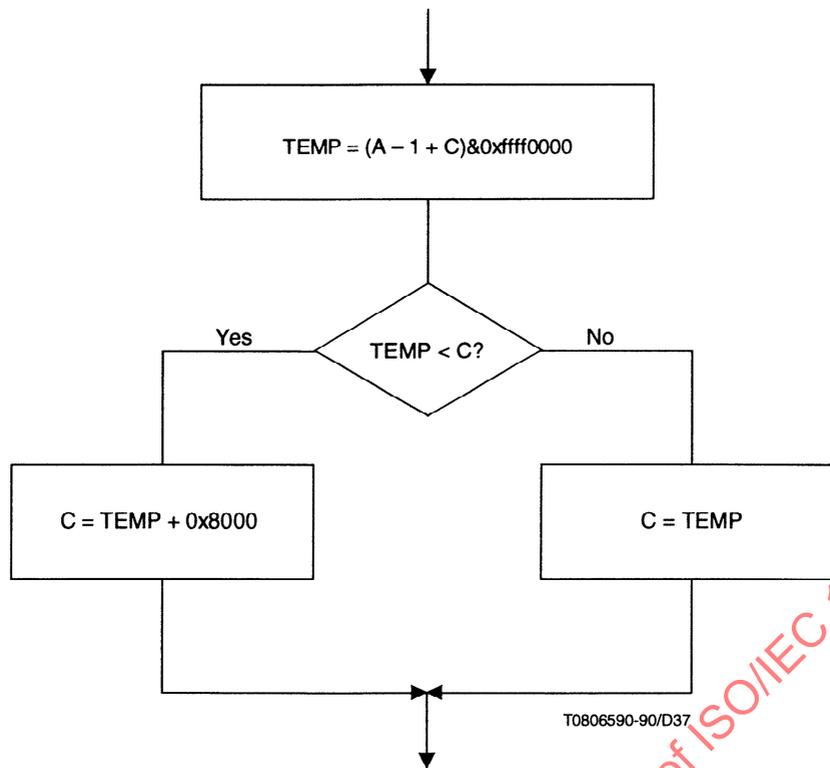
**Figure 29 – Flow diagram for the procedure CLEARBITS**
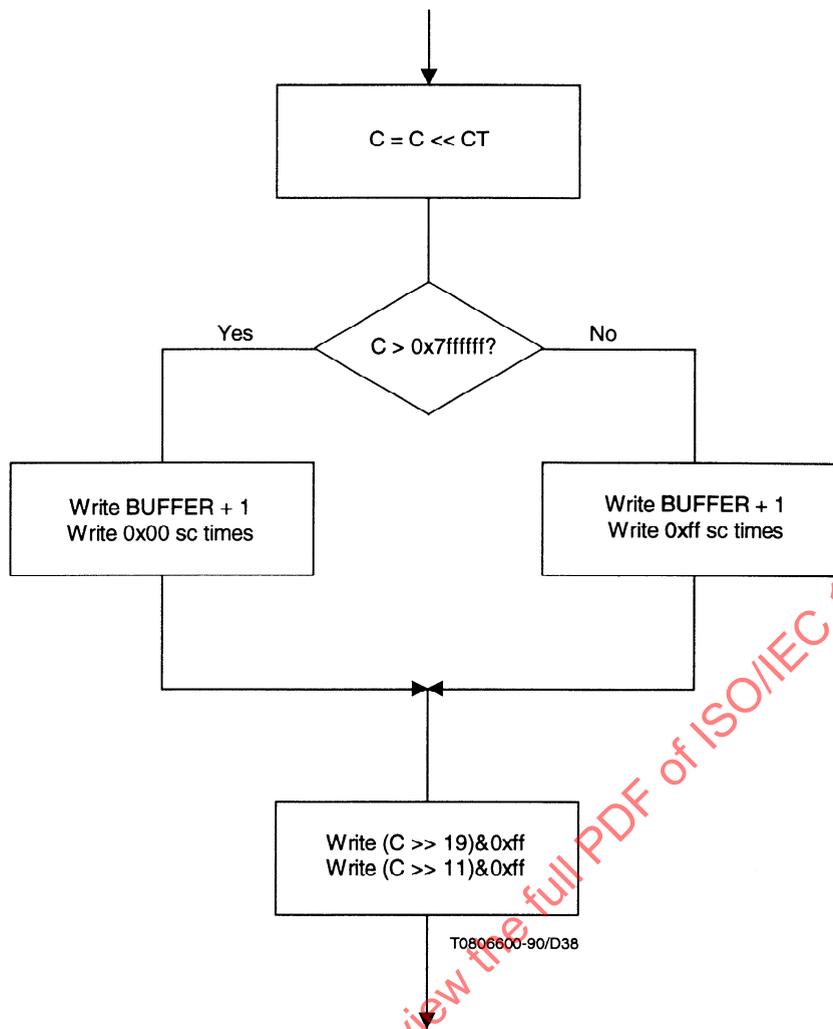
Figure 30 – Flow diagram for the procedure FINALWRITES

T0806600-90/D38
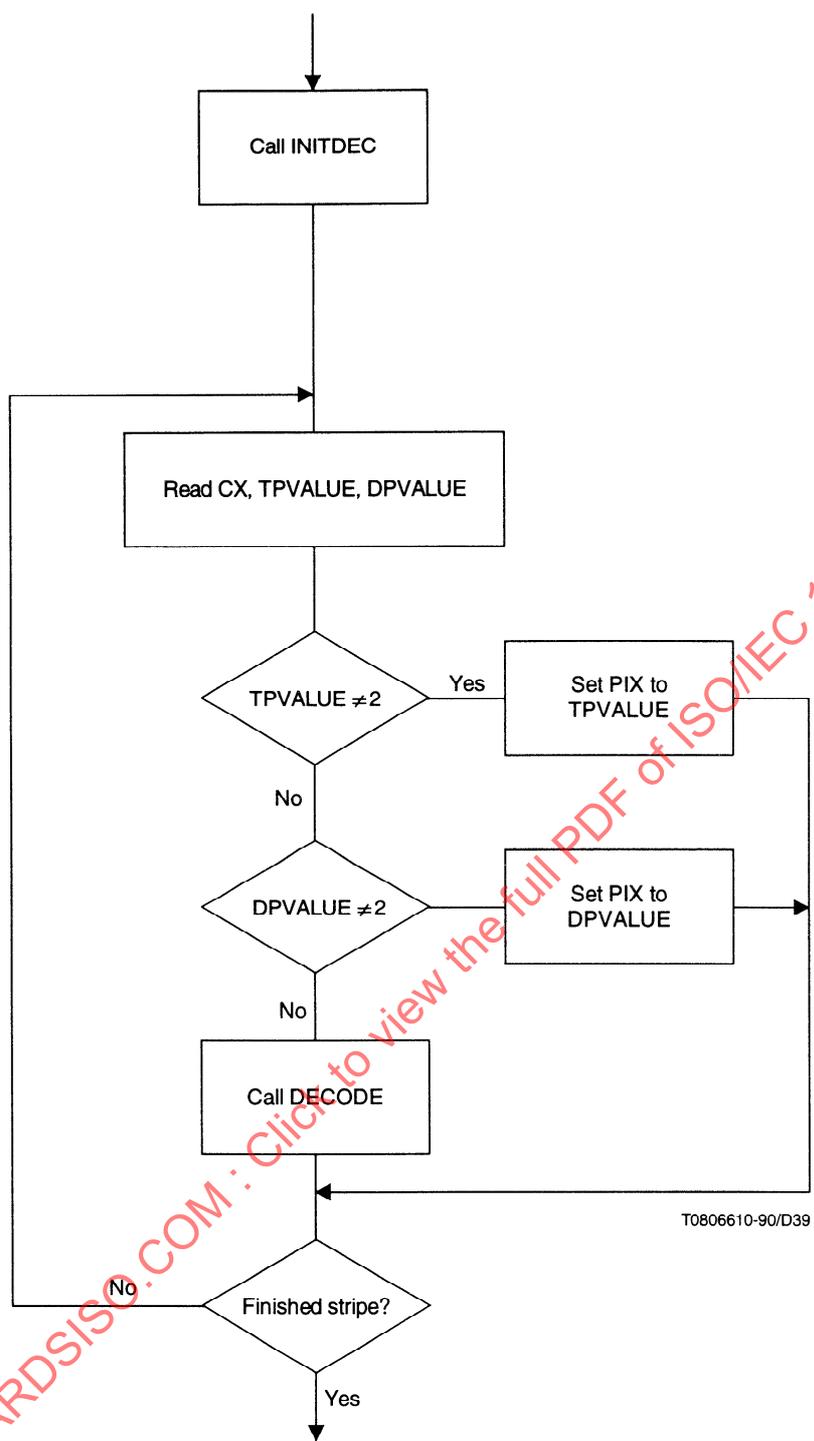
**Figure 31 – Decoder flow diagram**

**Table 25 – Decor register structure**

|  | msb | lsb |
|---|---|---|
| **CHIGH** register | xxxxxxxx, | xxxxxxxx |
| **CLOW** register | bbbbbbbb, | 00000000 |
| **A** register | a, aaaaaaaa, | aaaaaaaa |

**CHIGH** and **CLOW** can be thought of as one 32 bit **C** register, in that renormalization of **C** shifts a bit of new data from bit 15 (leftmost) of **CLOW** to bit 0 (rightmost) of **CHIGH**. However, the decoding comparisons use **CHIGH** alone. New data is inserted into the "b" bits of **CLOW** one byte at a time. As in the encoder, the seventeenth **A** register bit is conceptually present, but easily avoided in implementations.

### 6.8.3.3 Probability estimation tables

The probability-estimation tables used in decoding are identical to those used in encoding.

### 6.8.3.4 Flow diagram for the procedure DECODE

Only when a renormalization is needed is it possible that the MPS/LPS conditional exchange may have occurred (see Figure 32).



T0806620-90/D40
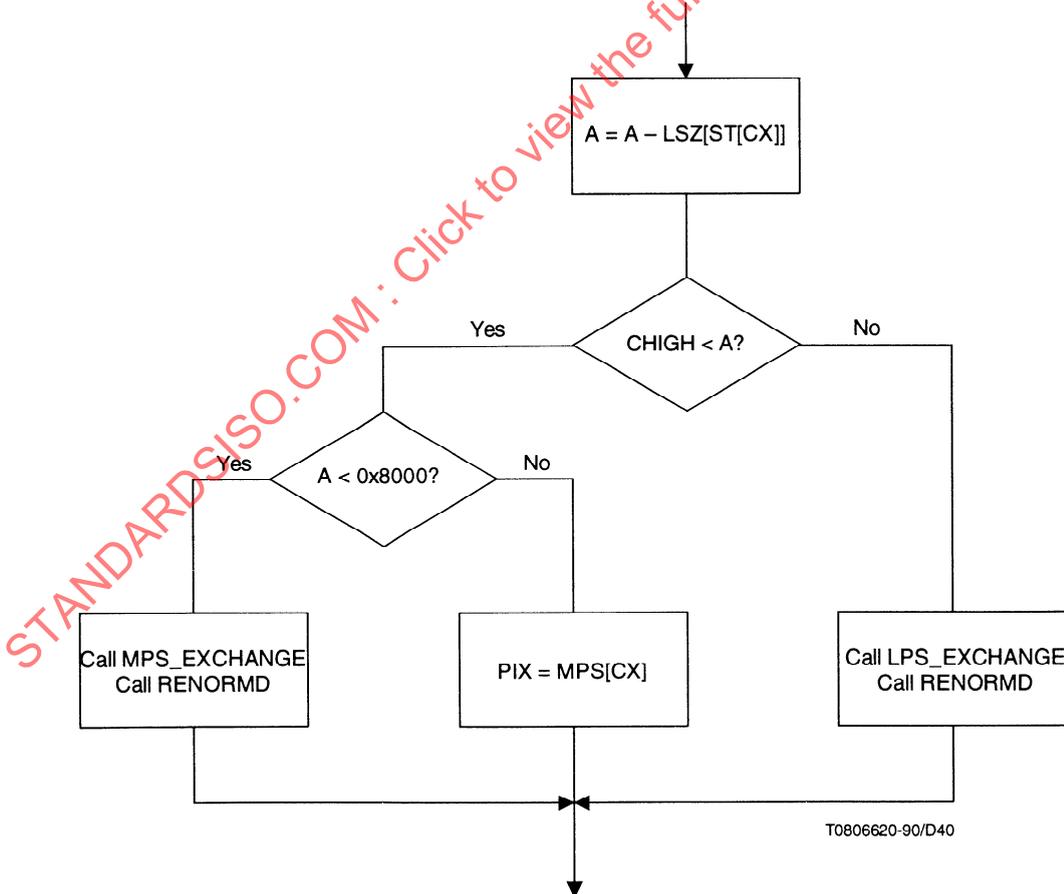
**Figure 32 – Flow diagram for the procedure DECODE**

### 6.8.3.5 Flow diagram for the procedure LPS_EXCHANGE

See Figure 33.
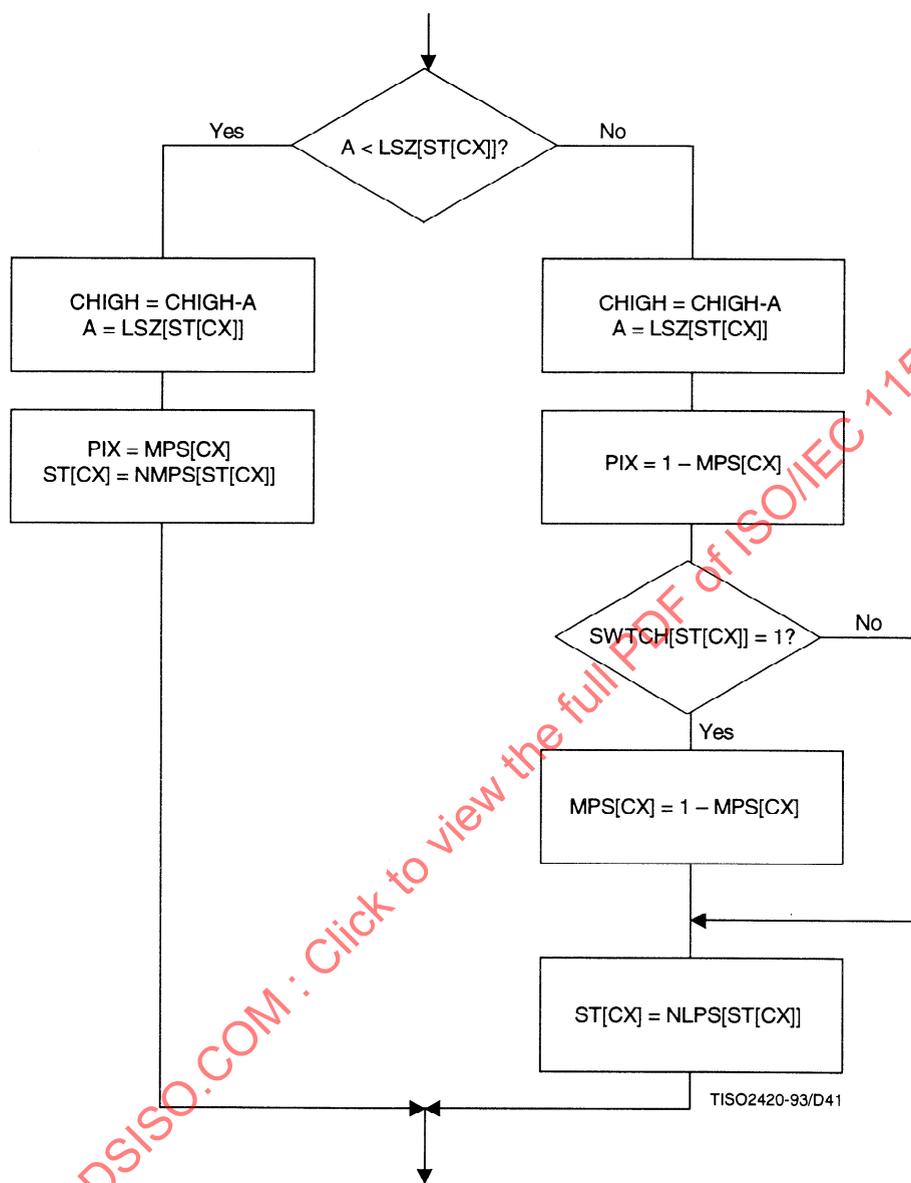


**Figure 33 – Flow diagram for the procedure LPS_EXCHANGE**

### 6.8.3.6 Flow diagram for the procedure MPS_EXCHANGE

See Figure 34.

### 6.8.3.7 Flow diagram for the procedure RENORMD

CT is a counter which keeps track of the number of compressed bits in the CLOW section of the C register. When CT is zero, a new byte is inserted into CLOW.

Both the interval register A and the code register C are shifted, one bit at a time, until A is no longer less than 0x8000. (See Figure 35).
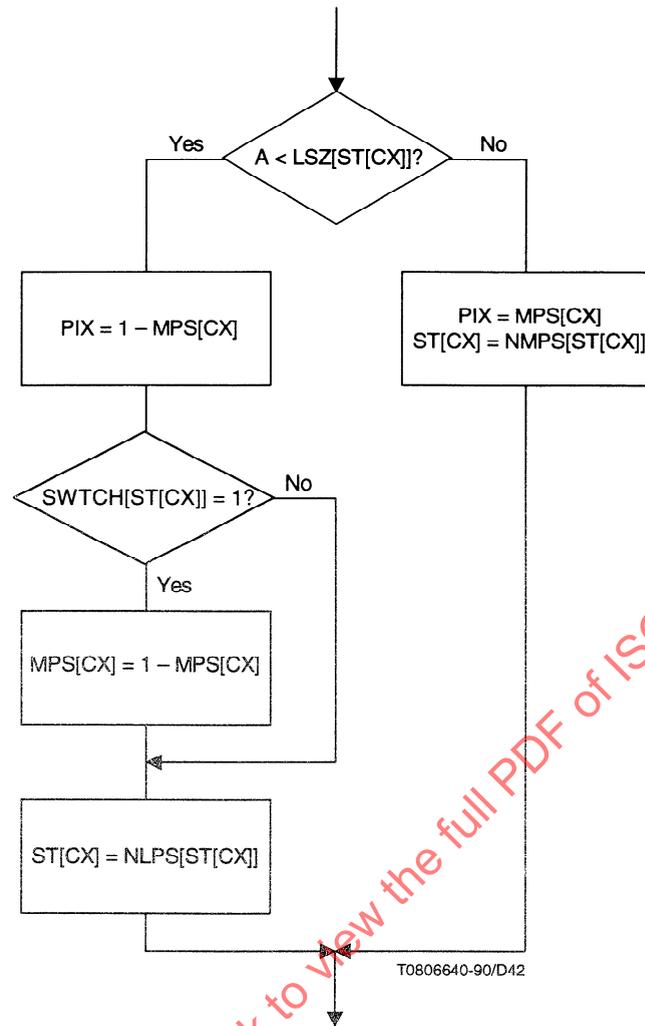
Figure 34 – Flow diagram for the procedure MPS_EXCHANGE

### 6.8.3.8 Flow diagram for the procedure BYTEIN

Bytes are read from **SCD** until it exhausts, after which further reads are satisfied by returning 0x00. The bytes read are inserted into the upper 8 bits of **CLOW**. The counter **CT** is reset to 8 (see Figure 36).

### 6.8.3.9 Flow diagram for the procedure INITDEC

If this stripe is at the top of the image, the probability-estimation states for all possible values of **CX** are set to 0. Otherwise, they are reset to their values at the end of the last stripe at this resolution. Three bytes are read into the **C** register (see Figure 37).

# 7 Test methods and datastream examples

This normative clause describes test methods for the algorithm described in earlier clauses of this Specification. There are many possible parameterizations, and this clause will document ways to test the accuracy of some parameterizations thought to be helpful in debugging implementations.
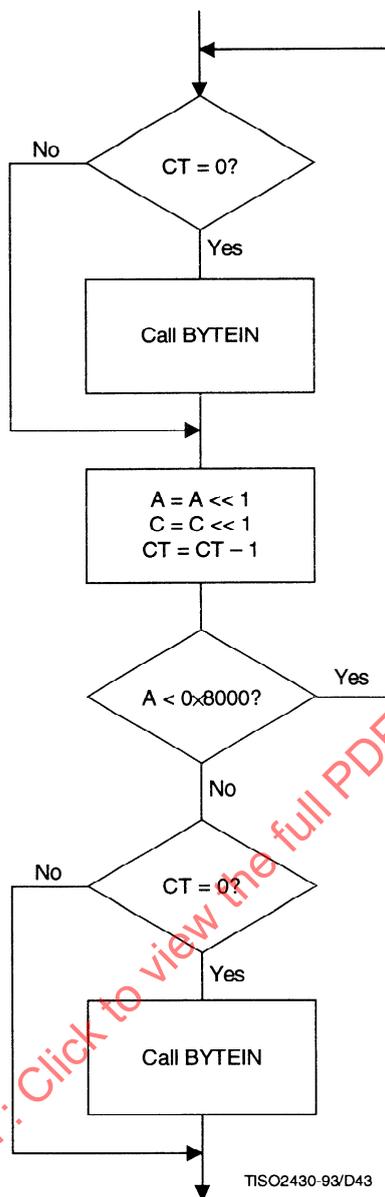
**Figure 35 – Flow diagram for the procedure RENORMD**

## 7.1 Arithmetic coding

In this subclause a small data set is provided for testing the arithmetic encoder and decoder. It will be assumed that this data set represents the raw data of a stripe in raster scan order and from MSB to LSB. The test is structured to test many of the encoder and decoder paths, but it is impossible in a short test sequence to check all of them so agreement with the results of this test unfortunately does not guarantee a completely correct implementation.

**PIX:**  05e0  0000  8b00  01c4  1700  0034  7fff  1a3f  951b  05d8  1d17  e770  0000  0000  0656  0e6a

**CX:**  0fe0  0000  0f00  00f0  ff00  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000  0000

T0806660-90/D44
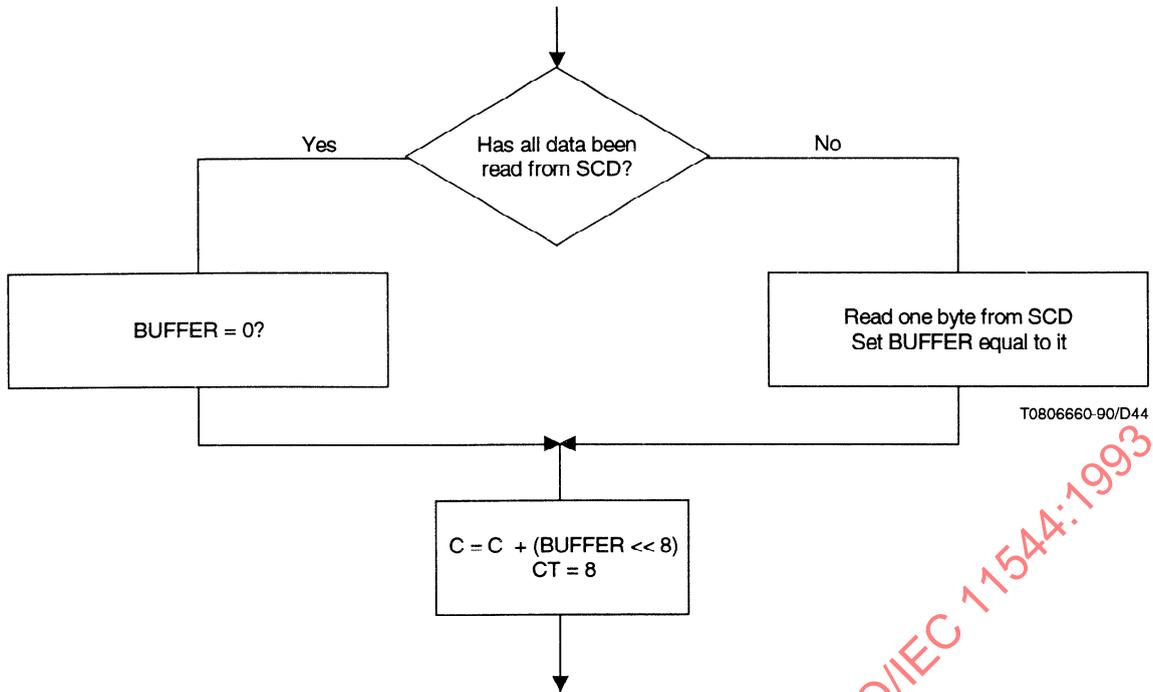
**Figure 36 – Flow diagram for the procedure BYTEIN**



T0806670-90/D45
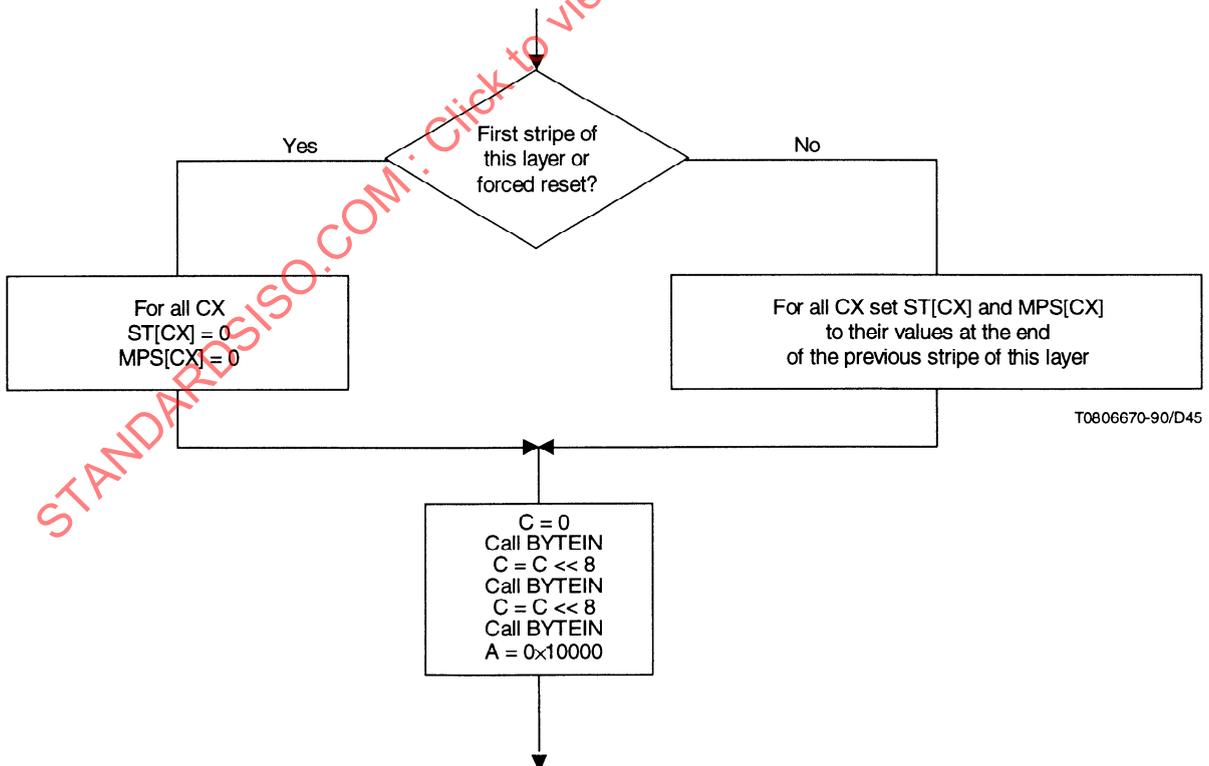
**Figure 37 – Flow diagram for the procedure INITDEC**

The encoder, see Figure 18, has four inputs. The first sequence describes the **PIX** input and the second one describes the corresponding **CX** input. For simplicity, the input variable **CX** for this test takes only the two values 0 and 1, rather than the 4096 values it takes on for the JBIG encoder/decoder. The other two inputs **TPVALUE** and **DPVALUE** are assumed to be always 2 (for the encoder and decoder). The encoder's output (**SCD**) consists of 200 bits (25 bytes) and is shown below in hex form (hexadecimal).

```
SCD:    6989 995c 32ea faa0 d5ff 527f ffff ffc0 0000 003f ff2d 2082 91
```

For the decoder, see Figure 19, the inputs **SCD** and **CX** are given from the sequences above while **PIX** is now the output.

Table 26 provides a symbol by symbol list of the arithmetic encoder and decoder operation. The first line in this table corresponds to the **INITENC** and **INITDEC** procedures with **BUFFER** initialized to 0x00. The last line in Table 26 corresponds to the encoder's **FINALWRITES** procedure. The first column is the event counter (**EC**), the second is the value of the binary event **PIX** to be encoded and decoded and the third is the corresponding value of the input variable **CX**. The **MPS** column indicates the sense of **MPS**, and the **CE** column indicates that the conditional exchange (see 6.8.2.5, 6.8.2.6 and 6.8.3.4) will occur when encoding (decoding) the current binary event **PIX**. The current state (see Table 24) and its corresponding **LPS** size are shown in the columns labeled **ST** and **LSZ**. Next is listed the value of the register **A** before the event is encoded (decoded). Note that the **A** register is always greater than or equal to 0x8000.

The variables up to this point were common for the encoder and decoder. The next five columns (**C**, **CT**, **SC**, **BUF**, **OUT**) are only for the encoder and the last 3 columns (**C**, **CT**, **IN**) are for the decoder. For the encoder the inputs are (**PIX**, **CX**) and its output is given in column **OUT**, while for the decoder the inputs are (**CX**, **IN**) and the output is **PIX**. The values of the register **C** listed under column **C** are given before the current event is encoded (decoded). For the encoder, **CT** is a counter indicating when a byte is ready for output from register **C**. **SC** is the number of 0xff bytes stacked in the encoder waiting for resolution of carry-over. The column under **BUF** shows the byte in variable **BUFFER** waiting to be sent out. This byte can sometimes change from a carry-over. Finally, for the encoder the code bytes are listed under column **OUT**. These bytes are sent to the output during the coding of the current event. If more than one byte is listed, these bytes were also output during the current event and they were generated by clearing the **SC** counter.

For the decoder, the values of the register **C** are given before the event is decoded and they are listed under column **C**. The decoder's counter **CT** is shown in the next column and indicates when to input the next byte from the code stream. Finally in the last column, the code bytes are listed if they were read into the code register at the end of the current event.

The last row, shows the output generated by the **FINALWRITES** procedure. This procedure generates also five additional 0x00 bytes, two from clearing the counter **SC** and three from flushing the **C** register. These final 0x00 bytes are not included in the code bytes **SCD**, since the option to remove all the 0x00 bytes from the end of the code stream of a stripe (see FLUSH procedure in Figure 28) was exercised. Notice, that it is allowable to leave any of these final 0x00 at the end of the code stream **SCD**. The decoder, upon reaching the end of the coded stream, reads 0x00's in its **C** register until it decodes the desired number (256) of pixels.

In order to generate the **PSCD** code stream, see Table 12, the **STUFF** (0x00) byte must be inserted after each **ESC** (0xff) byte in **SCD**.

```
PSCD:   6989 995c 32ea faa0 d5ff 0052 7fff 00ff 00ff 00c0 0000 003f ff00 2d20 8291
```

Finally, because of the assumption that the data (**CX**, **PIX**) are the raw data of a stripe, the stripe data entity (**SDE**) can be generated by appending the bytes **ESC** (0xff) and **SDNORM** (0x02).

```
SDE:    6989 995c 32ea faa0 d5ff 0052 7fff 00ff 00ff 00c0 0000 003f ff00 2d20 8291 ff02
```

## Table 26 – Encoder and decoder trace data

| EC | PIX | CX | MPS | CE | ST | LSZ hex | A hex | ENCODER | | | | | DECODER | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | C hex | CT | CS | BUF hex | OUT hex | C hex | CT | IN hex |
| 0 | | | | | | | | | | | | | 00000000 | 0 | 698999 |
| 1 | 0 | 0 | 0 | 0 | 0 | 5a1d | 10000 | 00000000 | 11 | 0 | 00 | | 69899900 | 8 | |
| 2 | 0 | 0 | 0 | 1 | 0 | 5a1d | 0a5e3 | 00000000 | 11 | 0 | 00 | | 69899900 | 8 | |
| 3 | 0 | 0 | 0 | 0 | 1 | 2586 | 0b43a | 0000978c | 10 | 0 | 00 | | 3b873200 | 7 | |
| 4 | 0 | 0 | 0 | 0 | 1 | 2586 | 08cb4 | 0000978c | 10 | 0 | 00 | | 3b873200 | 7 | |
| 5 | 0 | 1 | 0 | 0 | 0 | 5a1d | 0d25c | 00012f18 | 9 | 0 | 00 | | 770e6400 | 6 | |
| 6 | 1 | 1 | 0 | 0 | 1 | 2586 | 0f07e | 00025e30 | 8 | 0 | 00 | | ee1cc800 | 5 | |
| 7 | 0 | 1 | 0 | 1 | 14 | 5a7f | 09618 | 000ca4a0 | 6 | 0 | 00 | | 8c932000 | 3 | |
| 8 | 1 | 1 | 0 | 0 | 15 | 3f25 | 0b4fe | 0019c072 | 5 | 0 | 00 | | a1f44000 | 2 | 5c |
| 9 | 1 | 1 | 0 | 0 | 36 | 5ae1 | 0fc94 | 0068d92c | 3 | 0 | 00 | | b06d5c00 | 8 | |
| 10 | 1 | 1 | 1 | 0 | 37 | 484c | 0b5c2 | 00d2f5be | 2 | 0 | 00 | | 1d74b800 | 7 | |
| 11 | 1 | 1 | 1 | 0 | 38 | 3a0d | 0daec | 01a5eb7c | 1 | 0 | 00 | | 3ae97000 | 6 | |
| 12 | 0 | 0 | 0 | 0 | 2 | 1114 | 0a0df | 01a5eb7c | 1 | 0 | 00 | | 3ae97000 | 6 | |
| 13 | 0 | 0 | 0 | 0 | 2 | 1114 | 08fcb | 01a5eb7c | 1 | 0 | 00 | | 3ae97000 | 6 | |
| 14 | 0 | 0 | 0 | 0 | 3 | 080b | 0fd6e | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 15 | 0 | 0 | 0 | 0 | 3 | 080b | 0f563 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 16 | 0 | 0 | 0 | 0 | 3 | 080b | 0ed58 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 17 | 0 | 0 | 0 | 0 | 3 | 080b | 0e54d | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 18 | 0 | 0 | 0 | 0 | 3 | 080b | 0dd42 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 19 | 0 | 0 | 0 | 0 | 3 | 080b | 0d537 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 20 | 0 | 0 | 0 | 0 | 3 | 080b | 0cd2c | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 21 | 0 | 0 | 0 | 0 | 3 | 080b | 0c521 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 22 | 0 | 0 | 0 | 0 | 3 | 080b | 0bd16 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 23 | 0 | 0 | 0 | 0 | 3 | 080b | 0b50b | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 24 | 0 | 0 | 0 | 0 | 3 | 080b | 0ad00 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 25 | 0 | 0 | 0 | 0 | 3 | 080b | 0a4f5 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 26 | 0 | 0 | 0 | 0 | 3 | 080b | 09cea | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 27 | 0 | 0 | 0 | 0 | 3 | 080b | 094df | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 28 | 0 | 0 | 0 | 0 | 3 | 080b | 08cd4 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 29 | 0 | 0 | 0 | 0 | 3 | 080b | 084c9 | 0003d6f8 | 8 | 0 | 69 | | 75d2e000 | 5 | |
| 30 | 0 | 0 | 0 | 0 | 4 | 03d8 | 0f97c | 0007adf0 | 7 | 0 | 69 | | eba5c000 | 4 | |
| 31 | 0 | 0 | 0 | 0 | 4 | 03d8 | 0f5a4 | 0007adf0 | 7 | 0 | 69 | | eba5c000 | 4 | |
| 32 | 0 | 0 | 0 | 0 | 4 | 03d8 | 0f1cc | 0007adf0 | 7 | 0 | 69 | | eba5c000 | 4 | |
| 33 | 1 | 0 | 0 | 0 | 4 | 03d8 | 0cdf4 | 0007adf0 | 7 | 0 | 69 | | eba5c000 | 4 | 32 |
| 34 | 0 | 0 | 0 | 0 | 20 | 0cef | 0f600 | 02260300 | 1 | 0 | 69 | | 6270c800 | 6 | |
| 35 | 0 | 0 | 0 | 0 | 20 | 0cef | 0e911 | 02260300 | 1 | 0 | 69 | | 6270c800 | 6 | |
| 36 | 0 | 0 | 0 | 0 | 20 | 0cef | 0dc22 | 02260300 | 1 | 0 | 69 | | 6270c800 | 6 | |
| 37 | 1 | 1 | 1 | 0 | 38 | 3a0d | 0cf33 | 02260300 | 1 | 0 | 69 | | 6270c800 | 6 | |
| 38 | 0 | 1 | 1 | 0 | 38 | 3a0d | 09526 | 02260300 | 1 | 0 | 69 | 69 | 6270c800 | 6 | |
| 39 | 1 | 1 | 1 | 0 | 65 | 4d04 | 0e834 | 00097864 | 7 | 0 | 89 | | 1d5f2000 | 4 | |
| 40 | 1 | 1 | 1 | 0 | 65 | 4d04 | 09b30 | 00097864 | 7 | 0 | 89 | | 1d5f2000 | 4 | |
| 41 | 0 | 0 | 0 | 0 | 20 | 0cef | 09c58 | 0012f0c8 | 6 | 0 | 89 | | 3abe4000 | 3 | |
| 42 | 0 | 0 | 0 | 0 | 20 | 0cef | 08f69 | 0012f0c8 | 6 | 0 | 89 | | 3abe4000 | 3 | |
| 43 | 0 | 0 | 0 | 0 | 20 | 0cef | 0827a | 0012f0c8 | 6 | 0 | 89 | | 3abe4000 | 3 | |
| 44 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0eb16 | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 45 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0e175 | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 46 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0d7d4 | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 47 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0ce33 | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 48 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0c492 | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 49 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0baf1 | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 50 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0b150 | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 51 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0a7af | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 52 | 0 | 0 | 0 | 0 | 21 | 09a1 | 09e0e | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 53 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0946d | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 54 | 0 | 0 | 0 | 0 | 21 | 09a1 | 08acc | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 55 | 0 | 0 | 0 | 0 | 21 | 09a1 | 0812b | 0025e190 | 5 | 0 | 89 | | 757c8000 | 2 | |
| 56 | 1 | 0 | 0 | 0 | 22 | 072f | 0ef14 | 004bc320 | 4 | 0 | 89 | 89 | eaf90000 | 1 | ea |

**Table 26** – (*continued*)

| EC | PIX | CX | MPS | CE | ST | LSZ hex | A hex | ENCODER C hex | CT | CS | BUF hex | OUT hex | DECODER C hex | CT | IN hex |
|----|-----|-----|-----|-----|-----|---------|-------|---------------|----|----|---------|---------|----------------|----|--------|
| 57 | 1 | 1 | 1 | 0 | 66 | 412c | 0e5e0 | 000560a0 | 7 | 0 | 99 | | 628ea000 | 4 | |
| 58 | 1 | 1 | 1 | 0 | 66 | 412c | 0a4b4 | 000560a0 | 7 | 0 | 99 | | 628ea000 | 4 | |
| 59 | 0 | 1 | 1 | 0 | 67 | 37d8 | 0c710 | 000ac140 | 6 | 0 | 99 | | c51d4000 | 3 | |
| 60 | 0 | 1 | 1 | 0 | 82 | 438e | 0df60 | 002d41e0 | 4 | 0 | 99 | | d7950000 | 1 | fa |
| 61 | 0 | 0 | 0 | 0 | 46 | 0bfb | 0871c | 005bbb64 | 3 | 0 | 99 | | 7786fa00 | 8 | |
| 62 | 1 | 0 | 0 | 0 | 47 | 09f8 | 0f642 | 00b776c8 | 2 | 0 | 99 | 99 | ef0df400 | 7 | |
| 63 | 0 | 0 | 0 | 0 | 77 | 0d51 | 09f80 | 00063120 | 6 | 0 | 5c | | 2c3f4000 | 3 | |
| 64 | 0 | 0 | 0 | 0 | 77 | 0d51 | 0922f | 00063120 | 6 | 0 | 5c | | 2c3f4000 | 3 | |
| 65 | 0 | 1 | 1 | 1 | 89 | 4ca9 | 084de | 00063120 | 6 | 0 | 5c | | 2c3f4000 | 3 | |
| 66 | 0 | 1 | 1 | 0 | 95 | 56a8 | 0e0d4 | 0018c480 | 4 | 0 | 5c | | b0fd0000 | 1 | a0 |
| 67 | 0 | 1 | 0 | 0 | 95 | 56a8 | 0ad50 | 00329d58 | 3 | 0 | 5c | | 4da2a000 | 8 | |
| 68 | 1 | 1 | 0 | 0 | 96 | 4f46 | 0ad50 | 00653ab0 | 2 | 0 | 5c | | 9b454000 | 7 | |
| 69 | 0 | 1 | 0 | 1 | 101 | 5231 | 09e8c | 00cb3174 | 1 | 0 | 5c | 5c | 7a768000 | 6 | |
| 70 | 1 | 1 | 0 | 0 | 102 | 4c0f | 0a462 | 0006fb9e | 8 | 0 | 32 | | 5c370000 | 5 | |
| 71 | 1 | 1 | 0 | 1 | 106 | 50e7 | 0981e | 000ea7e2 | 7 | 0 | 32 | | 07c80000 | 4 | |
| 72 | 1 | 1 | 0 | 1 | 108 | 5597 | 08e6e | 001d4fc4 | 6 | 0 | 32 | | 0f900000 | 3 | |
| 73 | 0 | 0 | 0 | 0 | 77 | 0d51 | 0e35c | 00753f10 | 4 | 0 | 32 | | 3e400000 | 1 | |
| 74 | 0 | 0 | 0 | 0 | 77 | 0d51 | 0d60b | 00753f10 | 4 | 0 | 32 | | 3e400000 | 1 | |
| 75 | 0 | 0 | 0 | 0 | 77 | 0d51 | 0c8ba | 00753f10 | 4 | 0 | 32 | | 3e400000 | 1 | |
| 76 | 0 | 0 | 0 | 0 | 77 | 0d51 | 0bb69 | 00753f10 | 4 | 0 | 32 | | 3c400000 | 1 | |
| 77 | 0 | 0 | 0 | 0 | 77 | 0d51 | 0ae18 | 00753f10 | 4 | 0 | 32 | | 3e400000 | 1 | |
| 78 | 0 | 0 | 0 | 0 | 77 | 0d51 | 0a0c7 | 00753f10 | 4 | 0 | 32 | | 3e400000 | 1 | |
| 79 | 0 | 0 | 0 | 0 | 77 | 0d51 | 09376 | 00753f10 | 4 | 0 | 32 | | 3e400000 | 1 | |
| 80 | 0 | 0 | 0 | 0 | 77 | 0d51 | 08625 | 00753f10 | 4 | 0 | 32 | | 3e400000 | 1 | d5 |
| 81 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 0f1a8 | 00ea7e20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 82 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 0e5f2 | 00ea7e20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 83 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 0da3c | 00ea7e20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 84 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 0cc86 | 00ea7c20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 85 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 0c2d0 | 00ea7e20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 86 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 0b71a | 00ea7e20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 87 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 0ab64 | 00ea7e20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 88 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 09fae | 00ea7e20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 89 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 093f8 | 00ea7e20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 90 | 0 | 0 | 0 | 0 | 78 | 0bb6 | 08842 | 00ea7e20 | 3 | 0 | 32 | | 7c80d500 | 8 | |
| 91 | 1 | 0 | 0 | 0 | 79 | 0a40 | 0f918 | 01d4fc40 | 2 | 0 | 32 | 32 | f901aa00 | 7 | |
| 92 | 1 | 0 | 0 | 0 | 77 | 0d51 | 0a400 | 001eb180 | 6 | 0 | ea | | a29aa000 | 3 | ff |
| 93 | 0 | 0 | 0 | 0 | 75 | 119c | 0d510 | 01f482f0 | 2 | 0 | ea | | bebbfe00 | 7 | |
| 94 | 1 | 0 | 0 | 0 | 75 | 119c | 0c374 | 01f482f0 | 2 | 0 | ea | ea | bebbfe00 | 7 | |
| 95 | 0 | 0 | 0 | 0 | 74 | 1424 | 08ce0 | 0009a640 | 7 | 0 | fa | | 671ff000 | 4 | |
| 96 | 0 | 0 | 0 | 0 | 75 | 119c | 0f178 | 00134c80 | 6 | 0 | fa | | ce3fe000 | 3 | |
| 97 | 0 | 0 | 0 | 0 | 75 | 119c | 0dfdc | 00134c80 | 6 | 0 | fa | | ce3fe000 | 3 | |
| 98 | 1 | 0 | 0 | 0 | 75 | 119c | 0ce40 | 00134c80 | 6 | 0 | fa | | ce3fe000 | 3 | 52 |
| 99 | 1 | 0 | 0 | 0 | 74 | 1424 | 08ce0 | 00a04920 | 3 | 0 | fa | fa | 8cdf5200 | 8 | |
| 100 | 1 | 0 | 0 | 0 | 72 | 1aa9 | 0a120 | 00060ee0 | 8 | 0 | a0 | | a11a9000 | 5 | |
| 101 | 1 | 0 | 0 | 0 | 87 | 2516 | 0d548 | 0034aab8 | 5 | 0 | a0 | | d51c8000 | 2 | 7f |
| 102 | 1 | 0 | 0 | 0 | 86 | 299a | 09458 | 00d56ba8 | 3 | 0 | a0 | | 93aa7f00 | 8 | |
| 103 | 1 | 0 | 0 | 0 | 93 | 32b4 | 0a668 | 03575998 | 1 | 0 | a0 | a0 | a3b1fc00 | 6 | |
| 104 | 1 | 0 | 0 | 0 | 99 | 3c3d | 0cad0 | 000f3530 | 7 | 0 | d5 | | bff7f000 | 4 | |
| 105 | 1 | 0 | 0 | 0 | 104 | 415e | 0f0f4 | 003f0f0c | 5 | 0 | d5 | | c593c000 | 2 | |
| 106 | 1 | 0 | 0 | 1 | 103 | 4639 | 082bc | 007f7d44 | 4 | 0 | d5 | | 2bfb8000 | 1 | ff |
| 107 | 1 | 0 | 0 | 0 | 107 | 4b85 | 0f20c | 01fdf510 | 2 | 0 | d5 | | afeffe00 | 7 | |
| 108 | 1 | 0 | 0 | 1 | 109 | 504f | 0970a | 03fd372e | 1 | 0 | d5 | | 12d1fc00 | 6 | |
| 109 | 1 | 0 | 0 | 1 | 111 | 5522 | 08d76 | 00026e5c | 8 | 1 | d5 | | 25a3f800 | 5 | |
| 110 | 1 | 0 | 0 | 0 | 112 | 59eb | 0e150 | 0009b970 | 6 | 1 | d5 | | 968fe000 | 3 | |
| 111 | 1 | 0 | 1 | 0 | 112 | 59eb | 0b3d6 | 001481aa | 5 | 1 | d5 | | 1e55c000 | 2 | |
| 112 | 1 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 00290354 | 4 | 1 | d5 | | 3cab8000 | 1 | ff |

**Table 26** – (*continued*)

| EC | PIX | CX | MPS | CE | ST | LSZ hex | A hex | ENCODER C hex | CT | CS | BUF hex | OUT hex | DECODER C hex | CT | IN hex |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 113 | 0 | 0 | 1 | 0 | 109 | 504f | 0bd68 | 005206a8 | 3 | 1 | d5 | | 7957ff00 | 8 | |
| 114 | 0 | 0 | 1 | 1 | 111 | 5522 | 0a09e | 00a4e782 | 2 | 1 | d5 | | 187dfe00 | 7 | |
| 115 | 0 | 0 | 1 | 1 | 112 | 59eb | 096f8 | 0149cf04 | 1 | 1 | d5 | d5ff | 30fbfc00 | 6 | |
| 116 | 1 | 0 | 0 | 0 | 112 | 59eb | 0f434 | 00073c10 | 7 | 0 | 52 | | c3eff000 | 4 | |
| 117 | 1 | 0 | 1 | 0 | 112 | 59eb | 0b3d6 | 000facb2 | 6 | 0 | 52 | | 534de000 | 3 | |
| 118 | 0 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 001f5964 | 5 | 0 | 52 | | a69bc000 | 2 | |
| 119 | 1 | 0 | 1 | 1 | 112 | 59eb | 0aa44 | 003f7030 | 4 | 0 | 52 | | 8fcf8000 | 1 | ff |
| 120 | 0 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 007f8112 | 3 | 0 | 52 | | 7eedff00 | 8 | |
| 121 | 0 | 0 | 1 | 1 | 112 | 59eb | 0aa44 | 00ffbf8c | 2 | 0 | 52 | | 4073fe00 | 7 | |
| 122 | 0 | 0 | 0 | 1 | 112 | 59eb | 0a0b2 | 01ff7f18 | 1 | 0 | 52 | 52 | 80e7fc00 | 6 | |
| 123 | 1 | 0 | 0 | 0 | 111 | 5522 | 0b3d6 | 00078bbe | 8 | 0 | 7f | | 7441f800 | 5 | |
| 124 | 1 | 0 | 0 | 1 | 112 | 59eb | 0aa44 | 000fd4e4 | 7 | 0 | 7f | | 2b1bf000 | 4 | |
| 125 | 1 | 0 | 1 | 1 | 112 | 59eb | 0a0b2 | 001fa9c8 | 6 | 0 | 7f | | 5637e000 | 3 | |
| 126 | 1 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 003fe11e | 5 | 0 | 7f | | 1ee1c000 | 2 | |
| 127 | 1 | 0 | 1 | 0 | 109 | 504f | 0bd68 | 007fc23c | 4 | 0 | 7f | | 3dc38000 | 1 | c0 |
| 128 | 1 | 0 | 1 | 0 | 107 | 4b85 | 0da32 | 00ff8478 | 3 | 0 | 7f | | 7b87c000 | 8 | |
| 129 | 1 | 0 | 1 | 1 | 107 | 4b85 | 08cad | 00ff8478 | 3 | 0 | 7f | | 7b87c000 | 8 | |
| 130 | 0 | 0 | 1 | 0 | 103 | 4639 | 0970a | 01ff8f40 | 2 | 0 | 7f | | 70bf8000 | 7 | |
| 131 | 0 | 0 | 1 | 1 | 107 | 4b85 | 08c72 | 03ffc022 | 1 | 0 | 7f | | 3fdd0000 | 6 | |
| 132 | 1 | 0 | 1 | 1 | 109 | 504f | 081da | 00078044 | 8 | 1 | 7f | | 7fba0000 | 5 | |
| 133 | 0 | 0 | 1 | 0 | 107 | 4b85 | 0a09e | 000f639c | 7 | 1 | 7f | | 9c5e0000 | 4 | |
| 134 | 1 | 0 | 1 | 1 | 109 | 504f | 0970a | 001f716e | 6 | 1 | 7f | | 8e8a0000 | 3 | |
| 135 | 0 | 0 | 1 | 0 | 107 | 4b85 | 0a09e | 003f7052 | 5 | 1 | 7f | | 8f9e0000 | 2 | |
| 136 | 1 | 0 | 1 | 1 | 109 | 504f | 0970a | 007f8ad6 | 4 | 1 | 7f | | 750a0000 | 1 | 00 |
| 137 | 0 | 0 | 1 | 0 | 107 | 4b85 | 0a09e | 00ffa322 | 3 | 1 | 7f | | 5c9e0000 | 8 | |
| 138 | 0 | 0 | 1 | 1 | 109 | 504f | 0970a | 01fff076 | 2 | 1 | 7f | | 0f0a0000 | 7 | |
| 139 | 0 | 0 | 1 | 1 | 111 | 5522 | 08d76 | 03ffe0ec | 1 | 1 | 7f | | 1e140000 | 6 | |
| 140 | 1 | 0 | 1 | 0 | 112 | 59eb | 0e150 | 000f83b0 | 7 | 2 | 7f | | 78500000 | 4 | |
| 141 | 1 | 0 | 1 | 1 | 112 | 59eb | 08765 | 000f83b0 | 7 | 2 | 7f | | 78500000 | 4 | |
| 142 | 0 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 001f6254 | 6 | 2 | 7f | | 95ac0000 | 3 | |
| 143 | 1 | 0 | 1 | 1 | 112 | 59eb | 0aa44 | 003f8210 | 5 | 2 | 7f | | 6df00000 | 2 | |
| 144 | 1 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 007fa4d2 | 4 | 2 | 7f | | 3b2e0000 | 1 | 00 |
| 145 | 0 | 0 | 1 | 0 | 109 | 504f | 0bd68 | 00ff49a4 | 3 | 2 | 7f | | 765c0000 | 8 | |
| 146 | 0 | 0 | 1 | 1 | 111 | 5522 | 0a09e | 01ff6d7a | 2 | 2 | 7f | | 12860000 | 7 | |
| 147 | 0 | 0 | 1 | 1 | 112 | 59eb | 096f8 | 03fedaf4 | 1 | 2 | 7f | | 250c0000 | 6 | |
| 148 | 0 | 0 | 0 | 0 | 112 | 59eb | 0f434 | 000b6bd0 | 7 | 3 | 7f | | 94300000 | 4 | |
| 149 | 0 | 0 | 0 | 1 | 112 | 59eb | 09a49 | 000b6bd0 | 7 | 3 | 7f | | 94300000 | 4 | |
| 150 | 1 | 0 | 0 | 0 | 111 | 5522 | 0b3d6 | 0017585c | 6 | 3 | 7f | | a7a40000 | 3 | |
| 151 | 0 | 0 | 0 | 1 | 112 | 59eb | 0aa44 | 002f6e20 | 5 | 3 | 7f | | 91e00000 | 2 | |
| 152 | 1 | 0 | 0 | 0 | 111 | 5522 | 0b3d6 | 005f7cf2 | 4 | 3 | 7f | | 830e0000 | 1 | 00 |
| 153 | 1 | 0 | 0 | 1 | 112 | 59eb | 0aa44 | 00bfb74c | 3 | 3 | 7f | | 48b40000 | 8 | |
| 154 | 1 | 0 | 1 | 1 | 112 | 59eb | 0a0b2 | 017f6e98 | 2 | 3 | 7f | | 91680000 | 7 | |
| 155 | 0 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 02ff6abe | 1 | 3 | 7f | 7fffffff | 95420000 | 6 | |
| 156 | 1 | 0 | 1 | 1 | 112 | 59eb | 0aa44 | 000792e4 | 8 | 0 | bf | | 6d1c0000 | 5 | |
| 157 | 1 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 000fc67a | 7 | 0 | bf | | 39860000 | 4 | |
| 158 | 1 | 0 | 1 | 0 | 109 | 504f | 0bd68 | 001f8cf4 | 6 | 0 | bf | | 730c0000 | 3 | |
| 159 | 0 | 0 | 1 | 1 | 111 | 5522 | 0a09e | 003ff41a | 5 | 0 | bf | | 0be60000 | 2 | |
| 160 | 0 | 0 | 1 | 1 | 112 | 59eb | 096f8 | 007fe834 | 4 | 0 | bf | | 17cc0000 | 1 | 3f |
| 161 | 0 | 0 | 0 | 0 | 112 | 59eb | 0f434 | 01ffa0d0 | 2 | 0 | bf | | 5f307e00 | 7 | |
| 162 | 0 | 0 | 0 | 1 | 112 | 59eb | 09a49 | 01ffa0d0 | 2 | 0 | bf | | 5f307e00 | 7 | |
| 163 | 0 | 0 | 0 | 0 | 111 | 5522 | 0b3d6 | 03ffc25c | 1 | 0 | bf | | 3da4fc00 | 6 | |
| 164 | 1 | 0 | 0 | 0 | 109 | 504f | 0bd68 | 000784b8 | 8 | 1 | bf | | 7b49f800 | 5 | |
| 165 | 1 | 0 | 0 | 1 | 111 | 5522 | 0a09e | 000fe3a2 | 7 | 1 | bf | | 1c61f000 | 4 | |
| 166 | 1 | 0 | 0 | 1 | 112 | 59eb | 096f8 | 001fc744 | 6 | 1 | bf | | 38c3c000 | 3 | |
| 167 | 0 | 0 | 1 | 0 | 112 | 59eb | 0f434 | 007f1d10 | 4 | 1 | bf | | e30f8000 | 1 | ff |
| 168 | 1 | 0 | 0 | 0 | 112 | 59eb | 0b3d6 | 00ff6eb2 | 3 | 1 | bf | | 918dff00 | 8 | |

**Table 26 –** (*continued*)

| EC | PIX | CX | MPS | CE | ST | LSZ hex | A hex | ENCODER C hex | CT | CS | BUF hex | OUT hex | DECODER C hex | CT | IN hex |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 169 | 0 | 0 | 1 | 0 | 112 | 59eb | 0b3d6 | 01ff913a | 2 | 1 | bf | | 6f45fc00 | 7 | |
| 170 | 0 | 0 | 0 | 0 | 112 | 59eb | 0b3d6 | 03ffd64a | 1 | 1 | bf | | 2ab5fc00 | 6 | |
| 171 | 0 | 0 | 0 | 0 | 111 | 5522 | 0b3d6 | 0007ac94 | 8 | 2 | bf | | 556bf800 | 5 | |
| 172 | 1 | 0 | 0 | 0 | 109 | 504f | 0bd68 | 000f5928 | 7 | 2 | bf | | aad7f000 | 4 | |
| 173 | 0 | 0 | 0 | 1 | 111 | 5522 | 0a09c | 001f8c82 | 6 | 2 | bf | | 7b7dc000 | 3 | |
| 174 | 1 | 0 | 0 | 0 | 109 | 504f | 0aa44 | 003faffc | 5 | 2 | bf | | 6003c000 | 2 | |
| 175 | 1 | 0 | 0 | 1 | 111 | 5522 | 0a09e | 008013e2 | 4 | 2 | bf | | 0c1d8000 | 1 | 2d |
| 176 | 1 | 0 | 0 | 1 | 112 | 59eb | 096f8 | 010027c4 | 3 | 2 | bf | | 183b2d00 | 8 | |
| 177 | 1 | 0 | 1 | 0 | 112 | 59eb | 0f434 | 04009f10 | 1 | 2 | bf | | 60ecb400 | 6 | |
| 178 | 1 | 0 | 1 | 1 | 112 | 59eb | 09a49 | 04009f10 | 1 | 2 | bf | c00000 | 60ecb400 | 6 | |
| 179 | 1 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 0001bedc | 8 | 0 | 00 | | 411d6800 | 5 | |
| 180 | 0 | 0 | 1 | 0 | 109 | 504f | 0bd68 | 00037db8 | 7 | 0 | 00 | | 823ad000 | 4 | |
| 181 | 0 | 0 | 1 | 1 | 111 | 5522 | 0a09e | 0007d5a2 | 6 | 0 | 00 | | 2a43a000 | 3 | |
| 182 | 1 | 0 | 1 | 1 | 112 | 59eb | 096f8 | 000fab44 | 5 | 0 | 00 | | 54874000 | 2 | |
| 183 | 1 | 0 | 1 | 0 | 111 | 5522 | 0b3d6 | 001fd0a2 | 4 | 0 | 00 | | 2ef48000 | 1 | 20 |
| 184 | 1 | 0 | 1 | 0 | 109 | 504f | 0bd68 | 003fa144 | 3 | 0 | 00 | | 5de92000 | 8 | |
| 185 | 0 | 0 | 1 | 0 | 107 | 4b85 | 0da32 | 007f4288 | 2 | 0 | 00 | | bbd24000 | 7 | |
| 186 | 1 | 0 | 1 | 1 | 109 | 504f | 0970a | 00ffa26a | 1 | 0 | 00 | 00 | 5a4a8000 | 6 | |
| 187 | 1 | 0 | 1 | 0 | 107 | 4b85 | 0a09e | 0007d24a | 8 | 0 | 3f | | 271f0000 | 5 | |
| 188 | 1 | 0 | 1 | 0 | 103 | 4639 | 0aa32 | 000fa494 | 7 | 0 | 3f | | 4e3e0000 | 4 | |
| 189 | 0 | 0 | 1 | 0 | 104 | 415e | 0c7f2 | 001f4928 | 6 | 0 | 3f | | 9c7c0000 | 3 | |
| 190 | 0 | 0 | 1 | 1 | 103 | 4639 | 082bc | 003f9f78 | 5 | 0 | 3f | | 2bd00000 | 2 | 82 |
| 191 | 0 | 0 | 1 | 0 | 107 | 4b85 | 0f20c | 00fe7de0 | 3 | 0 | 3f | | af408200 | 8 | |
| 192 | 0 | 0 | 1 | 1 | 109 | 504f | 0970a | 01fe48ce | 2 | 0 | 3f | | 11730400 | 7 | |
| 193 | 0 | 0 | 1 | 1 | 111 | 5522 | 08d76 | 03fc919c | 1 | 0 | 3f | | 22e60800 | 6 | |
| 194 | 0 | 0 | 1 | 0 | 112 | 59eb | 0e150 | 00024670 | 7 | 1 | 3f | | 8b982000 | 4 | |
| 195 | 0 | 0 | 0 | 0 | 112 | 59eb | 0b3d6 | 00059baa | 6 | 1 | 3f | | 08664000 | 3 | |
| 196 | 0 | 0 | 0 | 0 | 111 | 5522 | 0b3d6 | 000b3754 | 5 | 1 | 3f | | 10cc8000 | 2 | |
| 197 | 0 | 0 | 0 | 0 | 109 | 504f | 0bd68 | 00166ea8 | 4 | 1 | 3f | | 21990000 | 1 | 91 |
| 198 | 0 | 0 | 0 | 0 | 107 | 4b85 | 0da32 | 002cdd50 | 3 | 1 | 3f | | 43329100 | 8 | |
| 199 | 0 | 0 | 0 | 1 | 107 | 4b85 | 08ead | 002cdd50 | 3 | 1 | 3f | | 43329100 | 8 | |
| 200 | 0 | 0 | 0 | 0 | 103 | 4639 | 0970a | 005a40f0 | 2 | 1 | 3f | | 00152200 | 7 | |
| 201 | 0 | 0 | 0 | 0 | 104 | 415e | 0a1a2 | 00b481e0 | 1 | 1 | 3f | 3fff | 002a4400 | 6 | |
| 202 | 0 | 0 | 0 | 0 | 99 | 3c3d | 0c088 | 000103c0 | 8 | 0 | 2d | | 00548800 | 5 | |
| 203 | 0 | 0 | 0 | 0 | 99 | 3c3d | 0844b | 000103c0 | 8 | 0 | 2d | | 00548800 | 5 | |
| 204 | 0 | 0 | 0 | 0 | 100 | 375e | 0901c | 00020780 | 7 | 0 | 2d | | 00a91000 | 4 | |
| 205 | 0 | 0 | 0 | 0 | 93 | 32b4 | 0b17c | 00040f00 | 6 | 0 | 2d | | 01522000 | 3 | |
| 206 | 0 | 0 | 0 | 0 | 94 | 2c17 | 0fd90 | 00081c00 | 5 | 0 | 2d | | 02a44000 | 2 | |
| 207 | 0 | 0 | 0 | 0 | 94 | 2e17 | 0cf79 | 00081e00 | 5 | 0 | 2d | | 02a44000 | 2 | |
| 208 | 0 | 0 | 0 | 0 | 94 | 2e17 | 0a162 | 00081e00 | 5 | 0 | 2d | | 02a44000 | 2 | |
| 209 | 0 | 0 | 0 | 0 | 86 | 299a | 0e696 | 00103c00 | 4 | 0 | 2d | | 05488000 | 1 | |
| 210 | 0 | 0 | 0 | 0 | 86 | 299a | 0bcfc | 00103c00 | 4 | 0 | 2d | | 05488000 | 1 | |
| 211 | 0 | 0 | 0 | 0 | 86 | 299a | 09362 | 00103c00 | 4 | 0 | 2d | | 05488000 | 1 | 00 |
| 212 | 0 | 0 | 0 | 0 | 87 | 2516 | 0d390 | 00207800 | 3 | 0 | 2d | | 0a910000 | 8 | |
| 213 | 0 | 0 | 0 | 0 | 87 | 2516 | 0ae7a | 00207800 | 3 | 0 | 2d | | 0a910000 | 8 | |
| 214 | 0 | 0 | 0 | 0 | 87 | 2516 | 08964 | 00207800 | 3 | 0 | 2d | | 0a910000 | 8 | |
| 215 | 0 | 0 | 0 | 0 | 71 | 1edf | 0c89c | 0040f000 | 2 | 0 | 2d | | 15220000 | 7 | |
| 216 | 0 | 0 | 0 | 0 | 71 | 1edf | 0a9bd | 0040f000 | 2 | 0 | 2d | | 15220000 | 7 | |
| 217 | 0 | 0 | 0 | 0 | 71 | 1edf | 08ade | 0040f000 | 2 | 0 | 2d | | 15220000 | 7 | |
| 218 | 0 | 0 | 0 | 0 | 72 | 1aa9 | 0d7fe | 0081e000 | 1 | 0 | 2d | | 2a440000 | 6 | |
| 219 | 0 | 0 | 0 | 0 | 72 | 1aa9 | 0bd55 | 0081e000 | 1 | 0 | 2d | | 2a440000 | 6 | |
| 220 | 0 | 0 | 0 | 0 | 72 | 1aa9 | 0a2ac | 0081e000 | 1 | 0 | 2d | | 2a440000 | 6 | |
| 221 | 0 | 0 | 0 | 0 | 72 | 1aa9 | 08803 | 0081e000 | 1 | 0 | 2d | 2d | 2a440000 | 6 | |
| 222 | 0 | 0 | 0 | 0 | 73 | 174e | 0dab4 | 0003c000 | 8 | 0 | 20 | | 54880000 | 5 | |
| 223 | 0 | 0 | 0 | 0 | 73 | 174e | 0c366 | 0003c000 | 8 | 0 | 20 | | 54880000 | 5 | |

**Table 26** – (*concluded*)

| EC | PIX | CX | MPS | CE | ST | LSZ hex | A hex | ENCODER C hex | CT | CS | BUF hex | OUT hex | DECODER C hex | CT | IN hex |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 224 | 0 | 0 | 0 | 0 | 73 | 174e | 0ac18 | 0003c000 | 8 | 0 | 20 | | 54880000 | 5 | |
| 225 | 0 | 0 | 0 | 0 | 73 | 174e | 094ca | 0003c000 | 8 | 0 | 20 | | 54880000 | 5 | |
| 226 | 0 | 0 | 0 | 0 | 74 | 1424 | 0faf8 | 00078000 | 7 | 0 | 20 | | a9100000 | 4 | |
| 227 | 0 | 0 | 0 | 0 | 74 | 1424 | 0e6d4 | 00078000 | 7 | 0 | 20 | | a9100000 | 4 | |
| 228 | 0 | 0 | 0 | 0 | 74 | 1424 | 0d2b0 | 00078000 | 7 | 0 | 20 | | a9100000 | 4 | |
| 229 | 0 | 0 | 0 | 0 | 74 | 1424 | 0be8c | 00078000 | 7 | 0 | 20 | | a9100000 | 4 | |
| 230 | 1 | 0 | 0 | 0 | 74 | 1424 | 0aa68 | 00078000 | 7 | 0 | 20 | | a9100000 | 4 | |
| 231 | 1 | 0 | 0 | 0 | 72 | 1aa9 | 0a120 | 0040b220 | 4 | 0 | 20 | | 96600000 | 1 | 00 |
| 232 | 0 | 0 | 0 | 0 | 87 | 2516 | 0d548 | 0209c4b8 | 1 | 0 | 20 | | 7f480000 | 6 | |
| 233 | 0 | 0 | 0 | 0 | 87 | 2516 | 0b032 | 0209c4b8 | 1 | 0 | 20 | | 7f480000 | 6 | |
| 234 | 1 | 0 | 0 | 0 | 87 | 2516 | 08b1c | 0209c4b8 | 1 | 0 | 20 | 20 | 7f480000 | 6 | |
| 235 | 0 | 0 | 0 | 0 | 86 | 299a | 09458 | 0008aaf8 | 7 | 0 | 82 | | 65080000 | 4 | |
| 236 | 1 | 0 | 0 | 0 | 87 | 2516 | 0d57c | 001155f0 | 6 | 0 | 82 | | ca100000 | 3 | |
| 237 | 0 | 0 | 0 | 0 | 86 | 299a | 09458 | 00481958 | 4 | 0 | 82 | | 66a80000 | 1 | 00 |
| 238 | 1 | 0 | 0 | 0 | 87 | 2516 | 0d57c | 009032b0 | 3 | 0 | 82 | | cd500000 | 8 | |
| 239 | 1 | 0 | 0 | 0 | 86 | 299a | 09458 | 02438c58 | 1 | 0 | 82 | 82 | 73a80000 | 6 | |
| 240 | 0 | 0 | 0 | 0 | 93 | 32b4 | 0a668 | 000fdc58 | 7 | 0 | 90 | | 23a80000 | 4 | |
| 241 | 0 | 0 | 0 | 0 | 94 | 2e17 | 0e768 | 001fb8b0 | 6 | 0 | 90 | | 47500000 | 3 | |
| 242 | 0 | 0 | 0 | 0 | 94 | 2e17 | 0b951 | 001fb8b0 | 6 | 0 | 90 | | 47500000 | 3 | |
| 243 | 0 | 0 | 0 | 0 | 94 | 2e17 | 08b3a | 001fb8b0 | 6 | 0 | 90 | | 47500000 | 3 | |
| 244 | 0 | 0 | 0 | 0 | 86 | 299a | 0ba46 | 003f7160 | 5 | 0 | 90 | | 8ea00000 | 2 | |
| 245 | 1 | 0 | 0 | 0 | 86 | 299a | 090ac | 003f7160 | 5 | 0 | 90 | | 8ea00000 | 2 | 00 |
| 246 | 1 | 0 | 0 | 0 | 93 | 32b4 | 0a668 | 00ff61c8 | 3 | 0 | 90 | | 9e380000 | 8 | |
| 247 | 1 | 0 | 0 | 0 | 99 | 3c3d | 0cad0 | 03ff55f0 | 1 | 0 | 90 | | aa100000 | 6 | |
| 248 | 0 | 0 | 0 | 0 | 104 | 415e | 0f0f4 | 000f920c | 7 | 1 | 90 | | 6df40000 | 4 | |
| 249 | 0 | 0 | 0 | 0 | 104 | 415e | 0af96 | 000f920c | 7 | 1 | 90 | | 6df40000 | 4 | |
| 250 | 1 | 0 | 0 | 0 | 99 | 3c3d | 0dc70 | 001f2418 | 6 | 1 | 90 | | dbe80000 | 3 | |
| 251 | 1 | 0 | 0 | 0 | 104 | 415e | 0f0f4 | 007ff12c | 4 | 1 | 90 | | eed40000 | 1 | 00 |
| 252 | 0 | 0 | 0 | 1 | 103 | 4639 | 082bc | 00ff8184 | 3 | 1 | 90 | | 7e7c0000 | 8 | |
| 253 | 1 | 0 | 0 | 0 | 104 | 415e | 08c72 | 01ff7c0e | 2 | 1 | 90 | | 83f20000 | 7 | |
| 254 | 0 | 0 | 0 | 1 | 103 | 4639 | 082bc | 03ff8e44 | 1 | 1 | 90 | | 71bc0000 | 6 | |
| 255 | 1 | 0 | 0 | 0 | 104 | 415e | 08c72 | 0007958e | 8 | 2 | 90 | | 6a720000 | 5 | |
| 256 | 0 | 0 | 0 | 1 | 103 | 4639 | 082bc | 000fc144 | 7 | 2 | 90 | | 3ebc0000 | 4 | |
| 257 | | | | | | | 08c72 | 08000000 | 6 | 2 | 90 | 91 | | | |

## 7.2 Parameterized algorithm

This normative subclause describes test methods for the algorithm described in earlier clauses of this Specification. There are many possible parameterizations, and this subclause will document ways to test the accuracy of some parameterizations thought to be helpful in debugging implementations. If an encoder implementation claims to support a parameterization broader than or as broad as any configuration for which test data is supplied in this subclause, then that implementation must generate exactly the byte counts shown for those test data. If a decoder implementation claims to support a parameterization broader than or as broad as any configuration for which test data is supplied in this subclause, then that implementation must decode those test data (generated by an encoder implementation which satisfies the encoder implementation requirements outlined above) and exactly generate the artificial image described in 7.2.1. An encoder supporting AT but not using the suggested algorithm of Annex C to determine AT pixel movement shall artificially force AT movements identical to those to be described here. Also, an encoder not choosing to remove all possible 0x00 bytes from the end of all **SDE** will need to temporarily postprocess to do so in order to duplicate the byte counts given.

### 7.2.1 Artificial image

The various tests of the full algorithm use an artificially generated image. This image is generated by the flow chart in Figure 38. It has 1960 pixels/line and 1951 lines and contains 861 965 foreground pixels and 2 961 995 background pixels.

```
                              ┌──────────────────┐
                              │      J = 0       │
                              │    PRSG = 1      │
                              └──────────────────┘
                                       │
                              ┌──────────────────┐
                              │      I = 0       │
                              └──────────────────┘
                                       │
                                    ◇ J < 192? ◇ ──── Yes ────────┐
                                       │ No                       │
                                ◇ J < 1023        ◇ ── No ──┐      │
                                   or                       │      │
                                 (I >> 3)&3 = 0?            │  ┌──────────┐
                                       │ Yes               │  │ Write 0  │
                              ┌──────────────────────────┐ │  └──────────┘
                              │ SUM = (PRSG&1) +         │ │      │
                              │ [(PRSG >> 2)&1] +         │ │      │
                              │ (PRSG >> 11)&1] +         │ ┌──────────────┐
                              │ [(PRSG >> 15)&1]          │ │ Write        │
                              │ SUM = SUM&1               │ │ REPEAT[I&7]  │
                              │ PRSG = (PRSG << 1) + SUM  │ └──────────────┘
                              └──────────────────────────┘
                                       │
                        Yes ──◇ PRSG&3 = 0? ◇── No
                          ┌──────────────┐  ┌──────────────┐
                          │ Write 1      │  │ Write 0      │
                          │ REPEAT[I&7]=1│  │ REPEAT[I&7]=0│
                          └──────────────┘  └──────────────┘
```

TISO2440-93/D46

```
                              ┌──────────────────┐
                              │    I = I + 1     │
                              └──────────────────┘
                                       │
                    Yes ──────◇ I < 1960 ◇
                                       │ No
                              ┌──────────────────┐
                              │    J = J + 1     │
                              └──────────────────┘
                                       │
                    Yes ──────◇ J < 1951 ◇
                                       │ No
                                       ▼
```
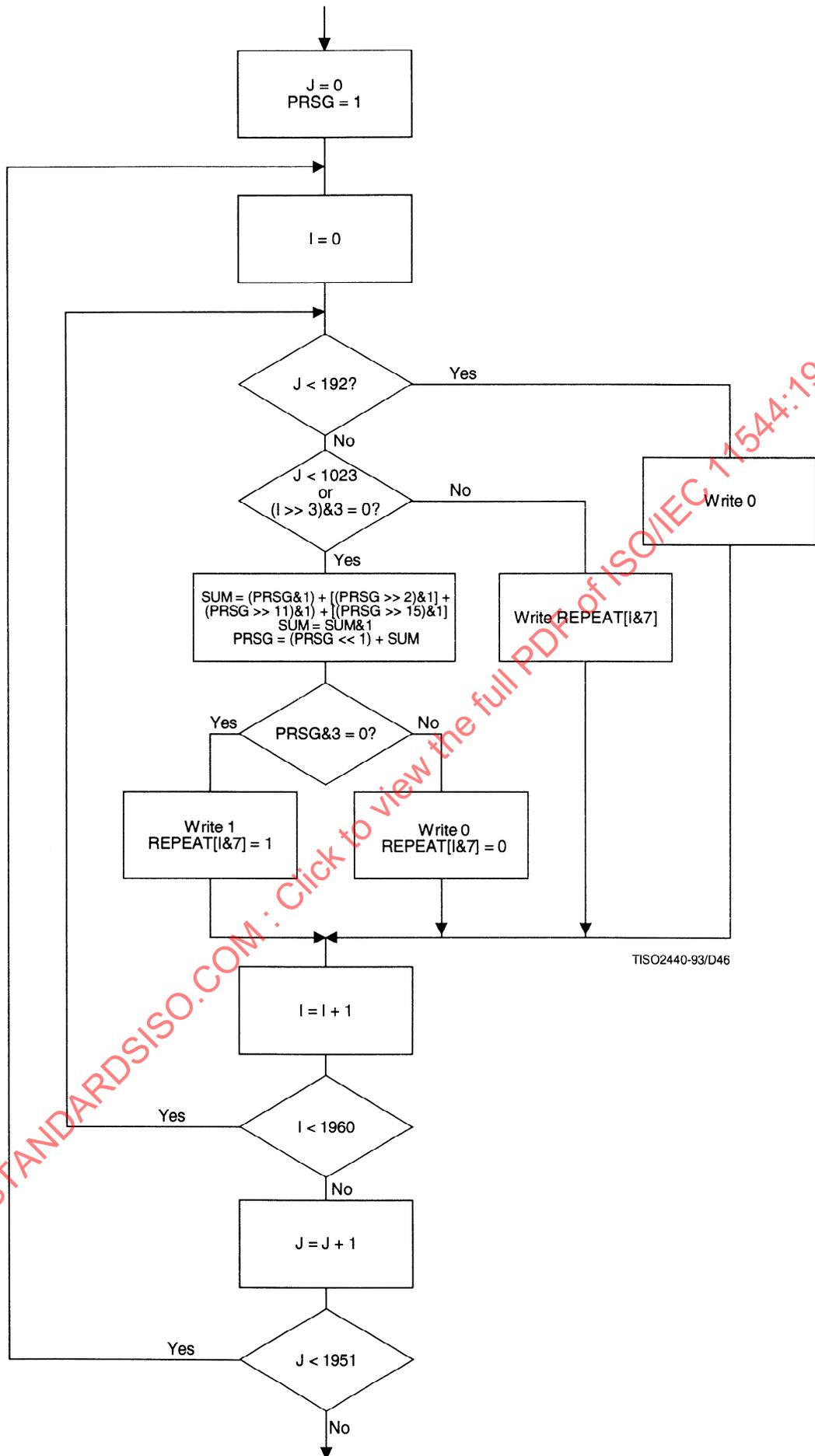
Figure 38 – Procedure for generating testing image

This image has been constructed so as to exercise as many features as possible. It is in no sense a typical image and compression results with it are not representative.

### 7.2.2 Single-progression sequential tests

For all three tests of this subclause, $D_L = 0$, $D = 0$, $P = 1$, $X_0 = 1960$, $Y_0 = 1951$, and $M_Y = 0$. The values of **HITOLO, SEQ, ILEAVE, SMID, VLENGTH, TPDON, DPON, DPPRIV, DPLAST** are immaterial. The four remaining parameters, $L_0$, $M_X$, **LRLTWO**, and **TPBON** vary as shown in Table 27. The remaining columns of this table provide trace data when the input image is the artificial image described in the previous subclause. For each of the first two tests there is just one **SCD** and the indicated byte count is its size. The final test, having $L_0 = 128$, produces 16 **SCD** and the indicated byte count is the sum of their sizes. In all cases all possible trailing 0x00 bytes are removed from the end of each **SCD** (see 6.8.2.10 and Figure 28).

**Table 27 – Trace parameters for tests of single-progression sequential coding**

| TPBON | $M_X$ | LRLTWO | $L_0$ | TP pixels | Encoded pixels | Coded bytes |
|-------|-------|--------|-------|-----------|----------------|-------------|
| 0 | 0 | 0 | 1951 | 0 | 3 823 960 | 316 094 |
| 0 | 0 | 1 | 1951 | 0 | 3 823 960 | 315 887 |
| 1 | 8 | 0 | 128 | 376 320 | 3 447 640 | 252 557 |

In the first two tests, AT is effectively disabled by having $M_X$ set equal to zero. The final test turns on AT as well as lowest-resolution-layer TP. AT was implemented as described in Annex C and the suggestion to defer any AT switches until the beginning of the next stripe was followed. The data in the last line of Table 28 were obtained by using **SDNORM**, which means that the probability estimator is not re-initialized. Table 29 provides information helpful in debugging AT. The first two columns give the stripe and line at which the switch becomes effective (both line and stripe numbering starts with zero), and the third column gives the lag $\tau_X$ for the new AT pixel location. The final 8 columns give the values of the counters described in Annex C when the AT movement is triggered.

**Table 28 – AT change information for third test of single-progression sequential coding**

| Stripe | Line | $\tau_X$ | $C_{all}$ | $C_0$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ | $C_8$ |
|--------|------|----------|-----------|-------|-------|-------|-------|-------|-------|-------|
| 9 | 0 | 8 | 3900 | 2336 | 2456 | 2472 | 2446 | 2442 | 2730 | 3534 |

**Table 29 – Byte counts for tests of single-progession sequential conding**

| TPBON | $M_X$ | LRLTWO | $L_0$ | SCD | PSCD | SDE | BID | BIE |
|-------|-------|--------|-------|-----|------|-----|-----|-----|
| 0 | 0 | 0 | 1951 | 316 094 | 317 362 | 317 364 | 317 364 | 317 384 |
| 0 | 0 | 1 | 1951 | 315 887 | 317 110 | 317 112 | 317 112 | 317 132 |
| 1 | 8 | 0 | 128 | 252 557 | 253 593 | 253 625 | 253 633 | 253 653 |

Further data for these three tests is provided by Table 30. The entry in the **SCD** column duplicates the final column of Table 27. The protected stripe coded data (**PSCD**) is obtained from the stripe coded data (**SCD**) by replacing each byte aligned 0xff with 0xff (**ESC**) and 0x00 (**STUFF**). The stripe data entity (**SDE**) is obtained from the **PSCD** by appending an 0xff (**ESC**) and 0x02 (**SDNORM**) byte at the end of each **PSCD**. The binary image data has the same number of bytes in it as the **SDE** except for the final test where it is eight bytes larger because of the AT movement marker segment. Finally, the **BIE** is 20 bytes larger (the header size) than the **BID**.

**Table 30 – Trace parameters for the enconding of artificial image**

| Layer | $X_d$ | $Y_d$ | $L_d$ | TP lines | TP exceptions | TP pixels | DP pixels | Encoded pixels | Coded bytes |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 1960 | 1951 | 128 | 137 | 7033 | 375 520 | 589 344 | 2 859 096 | 188 817 |
| 5 | 980 | 976 | 64 | 186 | 1442 | 93 120 | 128 642 | 734 718 | 65 584 |
| 4 | 490 | 488 | 32 | 181 | 135 | 22 792 | 30 230 | 186 098 | 16 565 |
| 3 | 245 | 244 | 16 | 117 | 8 | 5406 | 7246 | 47 128 | 4994 |
| 2 | 123 | 122 | 8 | 61 | 0 | 1238 | 1769 | 11 999 | 1430 |
| 1 | 62 | 61 | 4 | 31 | 0 | 248 | 452 | 3082 | 370 |
| 0 | 31 | 31 | 2 | 3 | – | 93 | – | 868 | 113 |

### 7.2.3 Progressive and progressive-compatible sequential test

For the test of this subclause, the input image is again the artificial image, but this time the parameters are set as follows (see Table 9): $D_L = 0, D = 6, P = 1, X_6 = 1960, Y_6 = 1951, L_0 = 2, M_X = 8, M_Y = 0$, **HITOLO** = 0 (immaterial), **SEQ** = 0 (immaterial), **ILEAVE** = 0 (immaterial), **SMID** = 0 (immaterial), **LRLTWO** = 0, **VLENGTH** = 0 (immaterial), **TPDON** = 1, **TPBON** = 1, **DPPRIV** = 0, and **DPLAST** = 0 (immaterial). Although the above parameterization is a progressive-coding parameterization, the value of **SEQ** is immaterial and identical byte counts are generated under progressive-compatible sequential coding. Hence, the test data here pertain to both modes. AT is again implemented as suggested in Annex C with any switches found to be desirable deferred until the beginning of the next stripe.

Table 31 provides trace data for coding the artificial image with the above parameters. The coded byte count shown in the last column is the sum of the number of bytes from all 16 **SCD** of that layer. As before all possible 0x00 bytes are removed.

There are two adaptive template switches, one in layer 6 and one in layer 5. Data pertinent to these two switches is provided in Table 31.

**Table 31 – AT change information**

| Layer | Stripe | Line | $\tau_X$ | $C_{all}$ | $C_0$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ | $C_8$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 6 | 9 | 0 | 8 | 3243 | 1984 | 2014 | 2055 | 2031 | 2001 | 2212 | 2924 |
| 5 | 10 | 0 | 4 | 2580 | 1323 | 1401 | 2259 | 1440 | 1447 | 1426 | 1966 |

Per-layer byte counts for all 16 **PSCD** and **SDE** are shown in Table 32, as well as the total number of bytes in the **BID** and **BIE** (see 6.2). The protected stripe coded data (**PSCD**) is obtained from the stripe coded data (**SCD**) by replacing each byte aligned 0xff with 0xff (**ESC**) and 0x00 (**STUFF**). The stripe data entity (**SDE**) is obtained from the **PSCD** by appending an 0xff (**ESC**) byte and an 0x02 (**SDNORM**) byte at the end of each **PSCD**. Finally, the binary image data field **BID** is obtained by concatenating all 112 **SDE**s (7 layers with 16 stripes for each layer) and the two **ATMOVE** marker segments, and the **BIE** is obtained by appending the **BID** to the twenty byte **BIH**.

**Table 32 – Byte counts for the artificial image**

| Layer | SCD | PSCD | SDE | BID | BIE |
|---|---|---|---|---|---|
| 6 | 188 817 | 189 584 | 189 616 | | |
| 5 | 65 584 | 65 905 | 65 937 | | |
| 4 | 16 565 | 16 634 | 16 666 | | |
| 3 | 4994 | 5010 | 5042 | | |
| 2 | 1430 | 1434 | 1466 | | |
| 1 | 370 | 373 | 405 | | |
| 0 | 113 | 114 | 146 | | |
| Total | 277 873 | 279 054 | 279 278 | 279 294 | 279 314 |

## 7.3    Datastream examples

A sample **BIE** for single-progression sequential coding of a binary image 1728 pixels wide and 2376 pixels tall with one stripe for the entire image and all binary parameters set to zero is (in hexadecimal):

```
|0x00|0x01|0x01|0x00|0x00 0x00 0x06 0xc0|0x00 0x00 0x09 0x48|

|0x00 0x00 0x09 0x48|0x00|0x00|0x00|0x00|

|      PSCD for entire image |0xff|0x02|
```

(The vertical bars show logical groupings, but otherwise are no different from a simple space.)

A sample **BIE** for a progressive-compatible sequential encoding of a binary image 1728 pixels wide and 2376 pixels tall with 1 differential layer, 64 lines per stripe in the reduced image, and all binary parameters except **SEQ** set to zero is (in hexadecimal):

```
|0x00|0x01|0x01|0x00|0x00 0x00 0x06 0xc0|0x00 0x00 0x09 0x48|

|0x00 0x00 0x00 0x40|0x00|0x00|0x04|0x00|

| PSCD for stripe  0 and layer 0 (base  image lines    0 to   63)|0xff|0x02|
| PSCD for stripe  0 and layer 1 (diff. image lines    0 to  127)|0xff|0x02|
| PSCD for stripe  1 and layer 0 (base  image lines   64 to  127)|0xff|0x02|
| PSCD for stripe  1 and layer 1 (diff. image lines  128 to  255)|0xff|0x02|

                                   .
                                   .
                                   .

| PSCD for stripe 17 and layer 0 (base  image lines 1088 to 1151)|0xff|0x02|
| PSCD for stripe 17 and layer 1 (diff. image lines 2176 to 2303)|0xff|0x02|
| PSCD for stripe 18 and layer 0 (base  image lines 1152 to 1187)|0xff|0x02|
| PSCD for stripe 18 and layer 1 (diff. image lines 2304 to 2375)|0xff|0x02|
```

The parenthetical comments indicate which lines of the base and differential images are encoded in which stripe.

A sample **BIE** for a progressive encoding of a binary image 1728 pixels wide and 2376 pixels tall with 1 differential layer, 64 lines per stripe in the reduced image, and all binary parameters set to zero is (in hexadecimal):

```
|0x00|0x01|0x01|0x00|0x00 0x00 0x06 0xc0|0x00 0x00 0x09 0x48|

|0x00 0x00 0x00 0x40|0x00|0x00|0x00|0x00|

| PSCD for stripe  0 and layer 0 (base  image lines    0 to   63)|0xff|0x02|
| PSCD for stripe  1 and layer 0 (base  image lines   64 to  127)|0xff|0x02|

                                   .
                                   .
                                   .

| PSCD for stripe 17 and layer 0 (base  image lines 1088 to 1151)|0xff|0x02|
| PSCD for stripe 18 and layer 0 (base  image lines 1152 to 1187)|0xff|0x02|
| PSCD for stripe  0 and layer 1 (diff. image lines    0 to  127)|0xff|0x02|
| PSCD for stripe  1 and layer 1 (diff. image lines  128 to  255)|0xff|0x02|

                                   .
                                   .
                                   .

| PSCD for stripe 17 and layer 1 (diff. image lines 2176 to 2303)|0xff|0x02|
| PSCD for stripe 18 and layer 1 (diff. image lines 2304 to 2375)|0xff|0x02|
```

# Annex A
# Suggested minimum support for free parameters

(This annex does not form an integral part of this Recommendation | International Standard)

Applications may set any of the 19 free parameters to any values within the ranges specified in Table 9. The suggestions on minimum support contained in this annex are being given so that it might be possible for a broad range of applications to share hardware and exchange decodable image data. Unless absolutely necessary, applications are encouraged to not choose parameter values outside the suggested support ranges.

Table A.1 lists suggested minimum ranges of decoder support. Hardware and software intended for general use with a variety of different output devices should be such that, if provisioned with sufficient external memory, support is available for parameter choices within the indicated ranges. A complete decoder that includes a specific output device should provide support over these same ranges, but with the obvious exceptions that there need be no support for

- image dimensions $X_D$ and $Y_D$ beyond the capability of the particular output device that is available;

- values of $P$ beyond the capability of the particular output device that is available; and

- more than one stripe order as defined by **SEQ**, **ILEAVE**, and **SMID**.

NOTE – In single-progression sequential applications the maximum support for $D$ is zero, the maximum support for $L_0$ is $Y_D$, and no support is required for the parameters **HITOLO**, **SEQ**, **TPDON**, **DPON**, **DPPRIV**, and **DPLAST**.

### Table A.1 – Suggested minimum support for free parameters

| Parameter | Minimum | Maximum |
|---|---|---|
| $D_L$ | 0 | $D$ |
| $D$ | $D_L$ | 6 |
| $P$ | 1 | 4 |
| $X_D$ | 1 | 5184 |
| $Y_D$ | 1 | 8192 |
| $L_0$ | 1 | $128/2^D$, for $D > 0$ $Y_D$, for $D = 0$ |
| $M_X$ | 0 | 8, for encoders 16, for decoders |
| $M_Y$ | 0 | 0 |
| **HITOLO** | 0 | 0 |
| **SEQ** | 0 | 1 |
| **ILEAVE** | 0 | 1 |
| **SMID** | 0 | 1 |
| **LRLTWO** | 0 | 1 |
| **VLENGTH** | 0 | 1 |
| **TPDON** | 0 | 1 |
| **TPBON** | 0 | 1 |
| **DPON** | 0 | 1 |
| **DPPRIV** | 0 | 1 |
| **DPLAST** | 0 | 1 |