TITLE: Additional Extension Segments for JBIG2

SOURCE: Dave Tompkins (davet@ece.ubc.ca) and Faouzi Kossentini (faouzi@ece.ubc.ca)  
Department of Electrical and Computer Engineering  
University of British Columbia  
Vancouver BC Canada V6T 1Z4

PROJECT: JBIG2

STATUS: Discussion

REQUESTED ACTION: For Inclusion in the JBIG2 Standard

DISTRIBUTION: July 99 Meeting - Vancouver

Contact:  
ISO/IEC JTC 1/SC 29/WG 1 Convener - Dr. Daniel T. Lee  
Hewlett-Packard Company, 11000 Wolfe Road, MS42U0, Cupertion, California 95014, USA  
Tel: +1 408 447 4160, Fax: +1 408 447 2842, E-mail: Daniel_Lee@hp.com
Summary

We are proposing three extension segments to be included in the JBIG2 standard. The first is a Version Extension segment, which will help facilitate any future revisions and enhancements to the JBIG2 standard. The other two segments are designed to enable JBIG2 to support multi-level images.

Version Extension Segment

We propose that a Version Extension segment be added to the JBIG2 standard. This segment would contain a major and a minor revision number. A decoder shall be able to decode a bitstream with the same major revision number and any minor revision number. However, a decoder may not be able to decode a bitstream with a higher major revision number.

Any bitstream with a major revision number greater than 1 shall include a Version Extension Segment.

Extension Type:
0xA0000020  Version segment.

A version segment shall contain 2 additional byte fields:

- **Major Version**: 1 Byte, Currently 1
- **Minor Version**: 1 Byte, Currently 0

An alternative to consider is including the version in the file format syntax.

Multi-Level Encoding with JBIG2

Although JBIG2 is designed for bi-level images, there are several types of multi-level images that can be compressed quite efficiently with JBIG2 techniques. Targeted images may include:

- Colour images that contain a lot of text
- Predominately black & white images that contain small areas of colour
- Images that contain only a few colours.

We propose a method of including multi-level colour information in JBIG2 bitstreams. Our method has the following features:

- A wide variety of colour varieties can be supported, from grayscale to single spot colours to full 24-bit colour. Variations within the colour format can be used to achieve the best compression.
- For predominately black & white documents, the black & white regions can be encoded as bi-level data.
- Encoders can provide an alternative bi-level image for decoders that do not support colour.
- With the use of extension segments, bi-level decoders will not fail, or decode “garbage”.
- Although any image can be supported, images with “white” backgrounds will be easily encoded.
Currently, a page is encoded with the following segments:

| Page Information Segment | (optional) Support Segments (dictionaries, tables, etc.) | Region Segment(s) | End of Page Segment |

We propose the following organization to support multi-level images:

| Page Information Segment | Multi-Level Page Segment | (optional) Support Segments (for bi-level and multi-level segments) | (optional) Region Segment(s) (bi-level component of the multi-level image) | Multi-Level Region Segment(s) | End of Page Segment |

Each of the new segments is explained below, followed by a decoding procedure.

### Multi-Level Page Segment

Extension Type:

**0xA0000010  Multi-Level Page Segment**

A Multi-Level Page Segment shall immediately follow a Page Information Segment.

A Multi-Level Page Segment contains a 1-byte flag field, followed by a 1-byte field corresponding to **MLLEVELS**, and ends with an optional colour palette section.

The flag field is as follows:

**Bit 0  MLPLANES**

If this bit is 0, each colour is coded as a separate level. If this bit is 1, the colour data is coded in bit-planes.

**Bit 1  MLREVERSED**

If MLPLANES is 0, this bit shall be 0. If this bit is 0 planes shall be ordered lowest to highest. If this bit is 1 the planes ordered highest to lowest.

**Bit 2  MLXOR**

If MLPLANES is 0, this bit shall be 0. If this bit is 0, then each plane is coded directly. If this bit is 1, then the first plane is coded directly, and each subsequent plane is coded as the XOR of the previous plane, in the same manner as gray-scale images in Annex C.

**Bit 3  MLTRANSPARENT**

If MLPLANES is 0, this bit shall be 0. If this bit is 0, then white ([255,255,255] in RGB space) shall be the transparent colour for superimposing colour data. If this bit is 1, then black ( [0,0,0] in RGB space) shall be the transparent colour.

**Bit 4  MLCOLOUR**

If this bit is 1, then the multi-level image has colour values. If this bit is 0, then the multi-level image is a gray-scale image.
Bit 5 MLPALETTE
If this bit is 1, then the colour values are determined from the palette section. If this bit is 0 then the colour values of the Multi-Level image are determined according to the following rules:

If MLCOLOUR is 0 then the colours shall be gray-scale intensities equally spaced between 0 (black) and MLNUMCOLOURS - 1 (white). If MLCOLOUR is 1, then MLNUMCOLOURS shall be divisible by three. The bits shall correspond to a triplet of RGB colour intensities equally spaced between 0 (no colour intensity) and \(2^{\text{MLNUMCOLOURS}/3} - 1\) (full colour intensity).

Bits 6-7 Reserved

MLLEVELS
MLLEVELS is coded as a 1-byte field. The value of MLLEVELS will determine the number of colours, MLNUMCOLOURS.

If MLPLANES is 0, then MLNUMCOLOURS = MLLEVELS.
If MLPLANES is 1, then MLNUMCOLOURS = 2^MLLEVELS.

PALETTE DATA
This section is only present if MLPALETTE is 1.

If MLCOLOUR is 0, then this section contains MLNUMCOLOURS bytes, with each byte corresponding to a gray-scale intensity ranging from 0 (black) to 255 (white).

If MLCOLOUR is 1, then this section contains 3*MLNUMCOLOURS bytes, with each 3-byte sequence corresponding to an RGB triplet of intensities ranging from 0 (no colour intensity) through to 255 (full colour intensity).

NOTE: When overlapping regions are to be used: If MLTRANSPARENT is 0 then the last colour should be white. If MLTRANSPARENT is 1 the first colour should be black.

Multi-Level Region Segment
Extension Type:
0xA0000011 Multi-Level Region Segment

Multi-Level Region shall appear after all regular region segments and after all support region segments for that page (or stripe).

A Multi-Level Region Segment is quite simple. It contains:

- A 1-byte field which contains the level of the region
- A 1-byte field which contains the region segment type (Same as the Segment Types in section 7.3)
- A regular region segment (without the header)

The Segment Header for the Multi-Level Region shall refer to and contain all of the data required by the regular region segment. The Data Length of the Multi-Level Region shall be the length of the regular region segment + 2.
Note: COMBOPREPLACE will be used for all Multi-Level Region Segments.

**Decoding a Multi-Level Image**

Note: For bi-level images, black shall be 1 and white shall be 0. For multi-level images, black shall be [0,0,0] and white shall be [255,255,255].

1. When the Page Information Segment is encountered, a bi-level image is constructed with the DEFPIXEL.
2. A Multi-Level Page Segment immediately following the Page Information Segment will indicate that the page contains multi-level data.
3. The colour levels are either calculated, or determined from the palette section.
4. For each level, (MLLEVELS) a bi-level image is constructed the same size as the page and filled with the value 0.
5. If MLPLANES is 1 and MLTRANSPARENT is 0 then the first image level is filled with the value 1. If MLXOR is 0 then the remaining image levels are also filled with the value 1.
6. All Support segments (dictionaries, tables, etc.) shall be decoded normally.
7. All regular region segments shall be decoded normally and placed on the bi-level image normally.
8. For each Multi-Level Region encountered, the following region is decoded normally, and then placed on the proper image level, with the combination operator COMBOPREPLACE.
9. When the end of Page Segment is encountered, the final multi-level image is constructed:
10. The bi-level image is converted to a multi-level image using the colour conventions described above.
11. If MLPLANES is 0 then for each level, any pixels that are 1 in the bi-level image have their corresponding multi-level pixel set to the appropriate colour.
12. If MLPLANES is 1 then another multi-level image is constructed by combining the planes into one image, setting the colour information accordingly. The multi-level image is then superimposed on the existing multi-level image, excluding any pixels corresponding to the transparent colour.