DRAFT TIFF Technical Note #2 17-Mar-95  
============================  
  
This Technical Note describes serious problems that have been found in  
TIFF 6.0's design for embedding JPEG-compressed data in TIFF (Section 22  
of the TIFF 6.0 spec of 3 June 1992). A replacement TIFF/JPEG  
specification is given. Some corrections to Section 21 are also given.  
  
To permit TIFF implementations to continue to read existing files, the 6.0  
JPEG fields and tag values will remain reserved indefinitely. However,  
TIFF writers are strongly discouraged from using the 6.0 JPEG design. It  
is expected that the next full release of the TIFF specification will not  
describe the old design at all, except to note that certain tag numbers  
are reserved. The existing Section 22 will be replaced by the  
specification text given in the second part of this Tech Note.  
  
  
Problems in TIFF 6.0 JPEG  
=========================  
  
Abandoning a published spec is not a step to be taken lightly. This  
section summarizes the reasons that have forced this decision.  
TIFF 6.0's JPEG design suffers from design errors and limitations,  
ambiguities, and unnecessary complexity.  
  
  
Design errors and limitations  
-----------------------------  
  
The fundamental design error in the existing Section 22 is that JPEG's  
various tables and parameters are broken out as separate fields which the  
TIFF control logic must manage. This is bad software engineering: that  
information should be treated as private to the JPEG codec  
(compressor/decompressor). Worse, the fields themselves are specified  
without sufficient thought for future extension and without regard to  
well-established TIFF conventions. Here are some of the significant  
problems:  
  
\* The JPEGxxTable fields do not store the table data directly in the  
IFD/field structure; rather, the fields hold pointers to information  
elsewhere in the file. This requires special-purpose code to be added to  
\*every\* TIFF-manipulating application, whether it needs to decode JPEG  
image data or not. Even a trivial TIFF editor, for example a program to  
add an ImageDescription field to a TIFF file, must be explicitly aware of  
the internal structure of the JPEG-related tables, or else it will probably  
break the file. Every other auxiliary field in the TIFF spec contains  
data, not pointers, and can be copied or relocated by standard code that  
doesn't know anything about the particular field. This is a crucial  
property of the TIFF format that must not be given up.  
  
\* To manipulate these fields, the TIFF control logic is required to know a  
great deal about JPEG details, for example such arcana as how to compute  
the length of a Huffman code table --- the length is not supplied in the  
field structure and can only be found by inspecting the table contents.  
This is again a violation of good software practice. Moreover, it will  
prevent easy adoption of future JPEG extensions that might change these  
low-level details.  
  
\* The design neglects the fact that baseline JPEG codecs support only two  
sets of Huffman tables: it specifies a separate table for each color  
component. This implies that encoders must waste space (by storing  
duplicate Huffman tables) or else violate the well-founded TIFF convention  
that prohibits duplicate pointers. Furthermore, baseline decoders must  
test to find out which tables are identical, a waste of time and code  
space.  
  
\* The JPEGInterchangeFormat field also violates TIFF's proscription against  
duplicate pointers: the normal strip/tile pointers are expected to point  
into the larger data area pointed to by JPEGInterchangeFormat. All TIFF  
editing applications must be specifically aware of this relationship, since  
they must maintain it or else delete the JPEGInterchangeFormat field. The  
JPEGxxTables fields are also likely to point into the JPEGInterchangeFormat  
area, creating additional pointer relationships that must be maintained.  
  
\* The JPEGQTables field is fixed at a byte per table entry; there is no  
way to support 16-bit quantization values. This is a serious impediment  
to extending TIFF to use 12-bit JPEG.  
  
\* The 6.0 design cannot support using different quantization tables in  
different strips/tiles of an image (so as to encode some areas at higher  
quality than others). Furthermore, since quantization tables are tied  
one-for-one to color components, the design cannot support table switching  
options that are likely to be added in future JPEG revisions.  
  
  
Ambiguities  
-----------  
  
Several incompatible interpretations are possible for 6.0's treatment of  
JPEG restart markers:  
  
 \* It is unclear whether restart markers must be omitted at TIFF segment  
 (strip/tile) boundaries, or whether they are optional.  
  
 \* It is unclear whether the segment size is required to be chosen as  
 a multiple of the specified restart interval (if any); perhaps the  
 JPEG codec is supposed to be reset at each segment boundary as if  
 there were a restart marker there, even if the boundary does not fall  
 at a multiple of the nominal restart interval.  
  
 \* The spec fails to address the question of restart marker numbering:  
 do the numbers begin again within each segment, or not?  
  
That last point is particularly nasty. If we make numbering begin again  
within each segment, we give up the ability to impose a TIFF strip/tile  
structure on an existing JPEG datastream with restarts (which was clearly a  
goal of Section 22's authors). But the other choice interferes with random  
access to the image segments: a reader must compute the first restart  
number to be expected within a segment, and must have a way to reset its  
JPEG decoder to expect a nonzero restart number first. This may not even  
be possible with some JPEG chips.  
  
The tile height restriction found on page 104 contradicts Section 15's  
general description of tiles. For an image that is not vertically  
downsampled, page 104 specifies a tile height of one MCU or 8 pixels; but  
Section 15 requires tiles to be a multiple of 16 pixels high.  
  
This Tech Note does not attempt to resolve these ambiguities, so  
implementations that follow the 6.0 design should be aware that  
inter-application compatibility problems are likely to arise.  
  
  
Unnecessary complexity  
----------------------  
  
The 6.0 design creates problems for implementations that need to keep the  
JPEG codec separate from the TIFF control logic --- for example, consider  
using a JPEG chip that was not designed specifically for TIFF. JPEG codecs  
generally want to produce or consume a standard ISO JPEG datastream, not  
just raw compressed data. (If they were to handle raw data, a separate  
out-of-band mechanism would be needed to load tables into the codec.)  
With such a codec, the TIFF control logic must parse JPEG markers emitted  
by the codec to create the TIFF table fields (when writing) or synthesize  
JPEG markers from the TIFF fields to feed the codec (when reading). This  
means that the control logic must know a great deal more about JPEG details  
than we would like. The parsing and reconstruction of the markers also  
represents a fair amount of unnecessary work.  
  
Quite a few implementors have proposed writing "TIFF/JPEG" files in which  
a standard JPEG datastream is simply dumped into the file and pointed to  
by JPEGInterchangeFormat. To avoid parsing the JPEG datastream, they  
suggest not writing the JPEG auxiliary fields (JPEGxxTables etc) nor even  
the basic TIFF strip/tile data pointers. This approach is incompatible  
with implementations that handle the full TIFF 6.0 JPEG design, since they  
will expect to find strip/tile pointers and auxiliary fields. Indeed this  
is arguably not TIFF at all, since \*all\* TIFF-reading applications expect  
to find strip or tile pointers. A subset implementation that is not  
upward-compatible with the full spec is clearly unacceptable. However,  
the frequency with which this idea has come up makes it clear that  
implementors find the existing Section 22 too complex.  
  
  
Overview of the solution  
========================  
  
To solve these problems, we adopt a new design for embedding  
JPEG-compressed data in TIFF files. The new design uses only complete,  
uninterpreted ISO JPEG datastreams, so it should be much more forgiving of  
extensions to the ISO standard. It should also be far easier to implement  
using unmodified JPEG codecs.  
  
To reduce overhead in multi-segment TIFF files, we allow JPEG overhead  
tables to be stored just once in a JPEGTables auxiliary field. This  
feature does not violate the integrity of the JPEG datastreams, because it  
uses the notions of "tables-only datastreams" and "abbreviated image  
datastreams" as defined by the ISO standard.  
  
To prevent confusion with the old design, the new design is given a new  
Compression tag value, Compression=7. Readers that need to handle  
existing 6.0 JPEG files may read both old and new files, using whatever  
interpretation of the 6.0 spec they did before. Compression tag value 6  
and the field tag numbers defined by 6.0 section 22 will remain reserved  
indefinitely, even though detailed descriptions of them will be dropped  
from future editions of the TIFF specification.  
  
  
Replacement TIFF/JPEG specification  
===================================  
  
[This section of the Tech Note is expected to replace Section 22 in the  
next release of the TIFF specification.]  
  
This section describes TIFF compression scheme 7, a high-performance  
compression method for continuous-tone images.  
  
Introduction  
------------  
  
This TIFF compression method uses the international standard for image  
compression ISO/IEC 10918-1, usually known as "JPEG" (after the original  
name of the standards committee, Joint Photographic Experts Group). JPEG  
is a joint ISO/CCITT standard for compression of continuous-tone images.  
  
The JPEG committee decided that because of the broad scope of the standard,  
no one algorithmic procedure was able to satisfy the requirements of all  
applications. Instead, the JPEG standard became a "toolkit" of multiple  
algorithms and optional capabilities. Individual applications may select  
a subset of the JPEG standard that meets their requirements.  
  
The most important distinction among the JPEG processes is between lossy  
and lossless compression. Lossy compression methods provide high  
compression but allow only approximate reconstruction of the original  
image. JPEG's lossy processes allow the encoder to trade off compressed  
file size against reconstruction fidelity over a wide range. Typically,  
10:1 or more compression of full-color data can be obtained while keeping  
the reconstructed image visually indistinguishable from the original. Much  
higher compression ratios are possible if a low-quality reconstructed image  
is acceptable. Lossless compression provides exact reconstruction of the  
source data, but the achievable compression ratio is much lower than for  
the lossy processes; JPEG's rather simple lossless process typically  
achieves around 2:1 compression of full-color data.  
  
The most widely implemented JPEG subset is the "baseline" JPEG process.  
This provides lossy compression of 8-bit-per-channel data. Optional  
extensions include 12-bit-per-channel data, arithmetic entropy coding for  
better compression, and progressive/hierarchical representations. The  
lossless process is an independent algorithm that has little in  
common with the lossy processes.  
  
It should be noted that the optional arithmetic-coding extension is subject  
to several US and Japanese patents. To avoid patent problems, use of  
arithmetic coding processes in TIFF files intended for inter-application  
interchange is discouraged.  
  
All of the JPEG processes are useful only for "continuous tone" data,  
in which the difference between adjacent pixel values is usually small.  
Low-bit-depth source data is not appropriate for JPEG compression, nor  
are palette-color images good candidates. The JPEG processes work well  
on grayscale and full-color data.  
  
Describing the JPEG compression algorithms in sufficient detail to permit  
implementation would require more space than we have here. Instead, we  
refer the reader to the References section.  
  
  
What data is being compressed?  
------------------------------  
  
In lossy JPEG compression, it is customary to convert color source data  
to YCbCr and then downsample it before JPEG compression. This gives  
2:1 data compression with hardly any visible image degradation, and it  
permits additional space savings within the JPEG compression step proper.  
However, these steps are not considered part of the ISO JPEG standard.  
The ISO standard is "color blind": it accepts data in any color space.  
  
For TIFF purposes, the JPEG compression tag is considered to represent the  
ISO JPEG compression standard only. The ISO standard is applied to the  
same data that would be stored in the TIFF file if no compression were  
used. Therefore, if color conversion or downsampling are used, they must  
be reflected in the regular TIFF fields; these steps are not considered to  
be implicit in the JPEG compression tag value. PhotometricInterpretation  
and related fields shall describe the color space actually stored in the  
file. With the TIFF 6.0 field definitions, downsampling is permissible  
only for YCbCr data, and it must correspond to the YCbCrSubSampling field.  
(Note that the default value for this field is not 1,1; so the default for  
YCbCr is to apply downsampling!) It is likely that future versions of TIFF  
will provide additional PhotometricInterpretation values and a more general  
way of defining subsampling, so as to allow more flexibility in  
JPEG-compressed files. But that issue is not addressed in this Tech Note.  
  
Implementors should note that many popular JPEG codecs  
(compressor/decompressors) provide automatic color conversion and  
downsampling, so that the application may supply full-size RGB data which  
is nonetheless converted to downsampled YCbCr. This is an implementation  
convenience which does not excuse the TIFF control layer from its  
responsibility to know what is really going on. The  
PhotometricInterpretation and subsampling fields written to the file must  
describe what is actually in the file.  
  
A JPEG-compressed TIFF file will typically have PhotometricInterpretation =  
YCbCr and YCbCrSubSampling = [2,1] or [2,2], unless the source data was  
grayscale or CMYK.  
  
  
Basic representation of JPEG-compressed images  
----------------------------------------------  
  
JPEG compression works in either strip-based or tile-based TIFF files.  
Rather than repeating "strip or tile" constantly, we will use the term  
"segment" to mean either a strip or a tile.  
  
When the Compression field has the value 7, each image segment contains  
a complete JPEG datastream which is valid according to the ISO JPEG  
standard (ISO/IEC 10918-1). Any sequential JPEG process can be used,  
including lossless JPEG, but progressive and hierarchical processes are not  
supported. Since JPEG is useful only for continuous-tone images, the  
PhotometricInterpretation of the image shall not be 3 (palette color) nor  
4 (transparency mask). The bit depth of the data is also restricted as  
specified below.  
  
Each image segment in a JPEG-compressed TIFF file shall contain a valid  
JPEG datastream according to the ISO JPEG standard's rules for  
interchange-format or abbreviated-image-format data. The datastream shall  
contain a single JPEG frame storing that segment of the image. The  
required JPEG markers within a segment are:  
 SOI (must appear at very beginning of segment)  
 SOFn  
 SOS (one for each scan, if there is more than one scan)  
 EOI (must appear at very end of segment)  
The actual compressed data follows SOS; it may contain RSTn markers if DRI  
is used.  
  
Additional JPEG "tables and miscellaneous" markers may appear between SOI  
and SOFn, between SOFn and SOS, and before each subsequent SOS if there is  
more than one scan. These markers include:  
 DQT  
 DHT  
 DAC (not to appear unless arithmetic coding is used)  
 DRI  
 APPn (shall be ignored by TIFF readers)  
 COM (shall be ignored by TIFF readers)  
DNL markers shall not be used in TIFF files. Readers should abort if any  
other marker type is found, especially the JPEG reserved markers;  
occurrence of such a marker is likely to indicate a JPEG extension.  
  
The tables/miscellaneous markers may appear in any order. Readers are  
cautioned that although the SOFn marker refers to DQT tables, JPEG does not  
require those tables to precede the SOFn, only the SOS. Missing-table  
checks should be made when SOS is reached.  
  
If no JPEGTables field is used, then each image segment shall be a complete  
JPEG interchange datastream. Each segment must define all the tables it  
references. To allow readers to decode segments in any order, no segment  
may rely on tables being carried over from a previous segment.  
  
When a JPEGTables field is used, image segments may omit tables that have  
been specified in the JPEGTables field. Further details appear below.  
  
The SOFn marker shall be of type SOF0 for strict baseline JPEG data, of  
type SOF1 for non-baseline lossy JPEG data, or of type SOF3 for lossless  
JPEG data. (SOF9 or SOF11 would be used for arithmetic coding.) All  
segments of a JPEG-compressed TIFF image shall use the same JPEG  
compression process, in particular the same SOFn type.  
  
The data precision field of the SOFn marker shall agree with the TIFF  
BitsPerSample field. (Note that when PlanarConfiguration=1, this implies  
that all components must have the same BitsPerSample value; when  
PlanarConfiguration=2, different components could have different bit  
depths.) For SOF0 only precision 8 is permitted; for SOF1, precision 8 or  
12 is permitted; for SOF3, precisions 2 to 16 are permitted.  
  
The image dimensions given in the SOFn marker shall agree with the logical  
dimensions of that particular strip or tile. For strip images, the SOFn  
image width shall equal ImageWidth and the height shall equal RowsPerStrip,  
except in the last strip; its SOFn height shall equal the number of rows  
remaining in the ImageLength. (In other words, no padding data is counted  
in the SOFn dimensions.) For tile images, each SOFn shall have width  
TileWidth and height TileHeight; adding and removing any padding needed in  
the edge tiles is the concern of some higher level of the TIFF software.  
(The dimensional rules are slightly different when PlanarConfiguration=2,  
as described below.)  
  
The ISO JPEG standard only permits images up to 65535 pixels in width or  
height, due to 2-byte fields in the SOFn markers. In TIFF, this limits  
the size of an individual JPEG-compressed strip or tile, but the total  
image size can be greater.  
  
The number of components in the JPEG datastream shall equal SamplesPerPixel  
for PlanarConfiguration=1, and shall be 1 for PlanarConfiguration=2. The  
components shall be stored in the same order as they are described at the  
TIFF field level. (This applies both to their order in the SOFn marker,  
and to the order in which they are scanned if multiple JPEG scans are  
used.) The component ID bytes are arbitrary so long as each component  
within an image segment is given a distinct ID. To avoid any possible  
confusion, we require that all segments of a TIFF image use the same ID  
code for a given component.  
  
In PlanarConfiguration 1, the sampling factors given in SOFn markers shall  
agree with the sampling factors defined by the related TIFF fields (or with  
the default values that are specified in the absence of those fields).  
  
When DCT-based JPEG is used in a strip TIFF file, RowsPerStrip is required  
to be a multiple of 8 times the largest vertical sampling factor, i.e., a  
multiple of the height of an interleaved MCU. (For simplicity of  
specification, we require this even if the data is not actually  
interleaved.) For example, if YCbCrSubSampling = [2,2] then RowsPerStrip  
must be a multiple of 16. An exception to this rule is made for  
single-strip images (RowsPerStrip >= ImageLength): the exact value of  
RowsPerStrip is unimportant in that case. This rule ensures that no data  
padding is needed at the bottom of a strip, except perhaps the last strip.  
Any padding required at the right edge of the image, or at the bottom of  
the last strip, is expected to occur internally to the JPEG codec.  
  
When DCT-based JPEG is used in a tiled TIFF file, TileLength is required  
to be a multiple of 8 times the largest vertical sampling factor, i.e.,  
a multiple of the height of an interleaved MCU; and TileWidth is required  
to be a multiple of 8 times the largest horizontal sampling factor, i.e.,  
a multiple of the width of an interleaved MCU. (For simplicity of  
specification, we require this even if the data is not actually  
interleaved.) All edge padding required will therefore occur in the course  
of normal TIFF tile padding; it is not special to JPEG.  
  
Lossless JPEG does not impose these constraints on strip and tile sizes,  
since it is not DCT-based.  
  
Note that within JPEG datastreams, multibyte values appear in the MSB-first  
order specified by the JPEG standard, regardless of the byte ordering of  
the surrounding TIFF file.  
  
  
JPEGTables field  
----------------  
  
The only auxiliary TIFF field added for Compression=7 is the optional  
JPEGTables field. The purpose of JPEGTables is to predefine JPEG  
quantization and/or Huffman tables for subsequent use by JPEG image  
segments. When this is done, these rather bulky tables need not be  
duplicated in each segment, thus saving space and processing time.  
JPEGTables may be used even in a single-segment file, although there is no  
space savings in that case.  
  
JPEGTables:  
 Tag = 347 (15B.H)  
 Type = UNDEFINED  
 N = number of bytes in tables datastream, typically a few hundred  
JPEGTables provides default JPEG quantization and/or Huffman tables which  
are used whenever a segment datastream does not contain its own tables, as  
specified below.  
  
Notice that the JPEGTables field is required to have type code UNDEFINED,  
not type code BYTE. This is to cue readers that expanding individual bytes  
to short or long integers is not appropriate. A TIFF reader will generally  
need to store the field value as an uninterpreted byte sequence until it is  
fed to the JPEG decoder.  
  
Multibyte quantities within the tables follow the ISO JPEG convention of  
MSB-first storage, regardless of the byte ordering of the surrounding TIFF  
file.  
  
When the JPEGTables field is present, it shall contain a valid JPEG  
"abbreviated table specification" datastream. This datastream shall begin  
with SOI and end with EOI. It may contain zero or more JPEG "tables and  
miscellaneous" markers, namely:  
 DQT  
 DHT  
 DAC (not to appear unless arithmetic coding is used)  
 DRI  
 APPn (shall be ignored by TIFF readers)  
 COM (shall be ignored by TIFF readers)  
Since JPEG defines the SOI marker to reset the DAC and DRI state, these two  
markers' values cannot be carried over into any image datastream, and thus  
they are effectively no-ops in the JPEGTables field. To avoid confusion,  
it is recommended that writers not place DAC or DRI markers in JPEGTables.  
However readers must properly skip over them if they appear.  
  
When JPEGTables is present, readers shall load the table specifications  
contained in JPEGTables before processing image segment datastreams.  
Image segments may simply refer to these preloaded tables without defining  
them. An image segment can still define and use its own tables, subject to  
the restrictions below.  
  
An image segment may not redefine any table defined in JPEGTables. (This  
restriction is imposed to allow readers to process image segments in random  
order without having to reload JPEGTables between segments.) Therefore, use  
of JPEGTables divides the available table slots into two groups: "global"  
slots are defined in JPEGTables and may be used but not redefined by  
segments; "local" slots are available for local definition and use in each  
segment. To permit random access, a segment may not reference any local  
tables that it does not itself define.  
  
  
Special considerations for PlanarConfiguration 2  
------------------------------------------------  
  
In PlanarConfiguration 2, each image segment contains data for only one  
color component. To avoid confusing the JPEG codec, we wish the segments  
to look like valid single-channel (i.e., grayscale) JPEG datastreams. This  
means that different rules must be used for the SOFn parameters.  
  
In PlanarConfiguration 2, the dimensions given in the SOFn of a subsampled  
component shall be scaled down by the sampling factors compared to the SOFn  
dimensions that would be used in PlanarConfiguration 1. This is necessary  
to match the actual number of samples stored in that segment, so that the  
JPEG codec doesn't complain about too much or too little data. In strip  
TIFF files the computed dimensions may need to be rounded up to the next  
integer; in tiled files, the restrictions on tile size make this case  
impossible.  
  
Furthermore, all SOFn sampling factors shall be given as 1. (This is  
merely to avoid confusion, since the sampling factors in a single-channel  
JPEG datastream have no real effect.)  
  
Any downsampling will need to happen externally to the JPEG codec, since  
JPEG sampling factors are defined with reference to the full-precision  
component. In PlanarConfiguration 2, the JPEG codec will be working on  
only one component at a time and thus will have no reference component to  
downsample against.  
  
  
Minimum requirements for TIFF/JPEG  
----------------------------------  
  
ISO JPEG is a large and complex standard; most implementations support only  
a subset of it. Here we define a "core" subset of TIFF/JPEG which readers  
must support to claim TIFF/JPEG compatibility. For maximum  
cross-application compatibility, we recommend that writers confine  
themselves to this subset unless there is very good reason to do otherwise.  
  
Use the ISO baseline JPEG process: 8-bit data precision, Huffman coding,  
with no more than 2 DC and 2 AC Huffman tables. Note that this implies  
BitsPerSample = 8 for each component. We recommend deviating from baseline  
JPEG only if 12-bit data precision or lossless coding is required.  
  
Use no subsampling (all JPEG sampling factors = 1) for color spaces other  
than YCbCr. (This is, in fact, required with the TIFF 6.0 field  
definitions, but may not be so in future revisions.) For YCbCr, use one of  
the following choices:  
 YCbCrSubSampling field JPEG sampling factors  
 1,1 1h1v, 1h1v, 1h1v  
 2,1 2h1v, 1h1v, 1h1v  
 2,2 (default value) 2h2v, 1h1v, 1h1v  
We recommend that RGB source data be converted to YCbCr for best compression  
results. Other source data colorspaces should probably be left alone.  
Minimal readers need not support JPEG images with colorspaces other than  
YCbCr and grayscale (PhotometricInterpretation = 6 or 1).  
  
A minimal reader also need not support JPEG YCbCr images with nondefault  
values of YCbCrCoefficients or YCbCrPositioning, nor with values of  
ReferenceBlackWhite other than [0,255,128,255,128,255]. (These values  
correspond to the RGB<=>YCbCr conversion specified by JFIF, which is widely  
implemented in JPEG codecs.)  
  
Writers are reminded that a ReferenceBlackWhite field \*must\* be included  
when PhotometricInterpretation is YCbCr, because the default  
ReferenceBlackWhite values are inappropriate for YCbCr.  
  
If any subsampling is used, PlanarConfiguration=1 is preferred to avoid the  
possibly-confusing requirements of PlanarConfiguration=2. In any case,  
readers are not required to support PlanarConfiguration=2.  
  
If possible, use a single interleaved scan in each image segment. This is  
not legal JPEG if there are more than 4 SamplesPerPixel or if the sampling  
factors are such that more than 10 blocks would be needed per MCU; in that  
case, use a separate scan for each component. (The recommended color  
spaces and sampling factors will not run into that restriction, so a  
minimal reader need not support more than one scan per segment.)  
  
To claim TIFF/JPEG compatibility, readers shall support multiple-strip TIFF  
files and the optional JPEGTables field; it is not acceptable to read only  
single-datastream files. Support for tiled TIFF files is strongly  
recommended but not required.  
  
  
Other recommendations for implementors  
--------------------------------------  
  
The TIFF tag Compression=7 guarantees only that the compressed data is  
represented as ISO JPEG datastreams. Since JPEG is a large and evolving  
standard, readers should apply careful error checking to the JPEG markers  
to ensure that the compression process is within their capabilities. In  
particular, to avoid being confused by future extensions to the JPEG  
standard, it is important to abort if unknown marker codes are seen.  
  
The point of requiring that all image segments use the same JPEG process is  
to ensure that a reader need check only one segment to determine whether it  
can handle the image. For example, consider a TIFF reader that has access  
to fast but restricted JPEG hardware, as well as a slower, more general  
software implementation. It is desirable to check only one image segment  
to find out whether the fast hardware can be used. Thus, writers should  
try to ensure that all segments of an image look as much "alike" as  
possible: there should be no variation in scan layout, use of options such  
as DRI, etc. Ideally, segments will be processed identically except  
perhaps for using different local quantization or entropy-coding tables.  
  
Writers should avoid including "noise" JPEG markers (COM and APPn markers).  
Standard TIFF fields provide a better way to transport any non-image data.  
Some JPEG codecs may change behavior if they see an APPn marker they  
think they understand; since the TIFF spec requires these markers to be  
ignored, this behavior is undesirable.  
  
It is possible to convert an interchange-JPEG file (e.g., a JFIF file) to  
TIFF simply by dropping the interchange datastream into a single strip.  
(However, designers are reminded that the TIFF spec discourages huge  
strips; splitting the image is somewhat more work but may give better  
results.) Conversion from TIFF to interchange JPEG is more complex. A  
strip-based TIFF/JPEG file can be converted fairly easily if all strips use  
identical JPEG tables and no RSTn markers: just delete the overhead markers  
and insert RSTn markers between strips. Converting tiled images is harder,  
since the data will usually not be in the right order (unless the tiles are  
only one MCU high). This can still be done losslessly, but it will require  
undoing and redoing the entropy coding so that the DC coefficient  
differences can be updated.  
  
There is no default value for JPEGTables: standard TIFF files must define all  
tables that they reference. For some closed systems in which many files will  
have identical tables, it might make sense to define a default JPEGTables  
value to avoid actually storing the tables. Or even better, invent a  
private field selecting one of N default JPEGTables settings, so as to allow  
for future expansion. Either of these must be regarded as a private  
extension that will render the files unreadable by other applications.  
  
  
References  
----------  
  
[1] Wallace, Gregory K. "The JPEG Still Picture Compression Standard",  
Communications of the ACM, April 1991 (vol. 34 no. 4), pp. 30-44.  
  
This is the best short technical introduction to the JPEG algorithms.  
It is a good overview but does not provide sufficiently detailed  
information to write an implementation.  
  
[2] Pennebaker, William B. and Mitchell, Joan L. "JPEG Still Image Data  
Compression Standard", Van Nostrand Reinhold, 1993, ISBN 0-442-01272-1.  
638pp.  
  
This textbook is by far the most complete exposition of JPEG in existence.  
It includes the full text of the ISO JPEG standards (DIS 10918-1 and draft  
DIS 10918-2). No would-be JPEG implementor should be without it.  
  
[3] ISO/IEC IS 10918-1, "Digital Compression and Coding of Continuous-tone  
Still Images, Part 1: Requirements and guidelines", February 1994.  
ISO/IEC DIS 10918-2, "Digital Compression and Coding of Continuous-tone  
Still Images, Part 2: Compliance testing", final approval expected 1994.  
  
These are the official standards documents. Note that the Pennebaker and  
Mitchell textbook is likely to be cheaper and more useful than the official  
standards.  
  
  
Changes to Section 21: YCbCr Images  
===================================  
  
[This section of the Tech Note clarifies section 21 to make clear the  
interpretation of image dimensions in a subsampled image. Furthermore,  
the section is changed to allow the original image dimensions not to be  
multiples of the sampling factors. This change is necessary to support use  
of JPEG compression on odd-size images.]  
  
Add the following paragraphs to the Section 21 introduction (p. 89),  
just after the paragraph beginning "When a Class Y image is subsampled":  
  
 In a subsampled image, it is understood that all TIFF image  
 dimensions are measured in terms of the highest-resolution  
 (luminance) component. In particular, ImageWidth, ImageLength,  
 RowsPerStrip, TileWidth, TileLength, XResolution, and YResolution  
 are measured in luminance samples.  
  
 RowsPerStrip, TileWidth, and TileLength are constrained so that  
 there are an integral number of samples of each component in a  
 complete strip or tile. However, ImageWidth/ImageLength are not  
 constrained. If an odd-size image is to be converted to subsampled  
 format, the writer should pad the source data to a multiple of the  
 sampling factors by replication of the last column and/or row, then  
 downsample. The number of luminance samples actually stored in the  
 file will be a multiple of the sampling factors. Conversely,  
 readers must ignore any extra data (outside the specified image  
 dimensions) after upsampling.  
  
 When PlanarConfiguration=2, each strip or tile covers the same  
 image area despite subsampling; that is, the total number of strips  
 or tiles in the image is the same for each component. Therefore  
 strips or tiles of the subsampled components contain fewer samples  
 than strips or tiles of the luminance component.  
  
 If there are extra samples per pixel (see field ExtraSamples),  
 these data channels have the same number of samples as the  
 luminance component.  
  
Rewrite the YCbCrSubSampling field description (pp 91-92) as follows  
(largely to eliminate possibly-misleading references to  
ImageWidth/ImageLength of the subsampled components):  
  
 (first paragraph unchanged)  
  
 The two elements of this field are defined as follows:  
  
 Short 0: ChromaSubsampleHoriz:  
  
 1 = there are equal numbers of luma and chroma samples horizontally.  
  
 2 = there are twice as many luma samples as chroma samples  
 horizontally.  
  
 4 = there are four times as many luma samples as chroma samples  
 horizontally.  
  
 Short 1: ChromaSubsampleVert:  
  
 1 = there are equal numbers of luma and chroma samples vertically.  
  
 2 = there are twice as many luma samples as chroma samples  
 vertically.  
  
 4 = there are four times as many luma samples as chroma samples  
 vertically.  
  
 ChromaSubsampleVert shall always be less than or equal to  
 ChromaSubsampleHoriz. Note that Cb and Cr have the same sampling  
 ratios.  
  
 In a strip TIFF file, RowsPerStrip is required to be an integer  
 multiple of ChromaSubSampleVert (unless RowsPerStrip >=  
 ImageLength, in which case its exact value is unimportant).  
 If ImageWidth and ImageLength are not multiples of  
 ChromaSubsampleHoriz and ChromaSubsampleVert respectively, then the  
 source data shall be padded to the next integer multiple of these  
 values before downsampling.  
  
 In a tiled TIFF file, TileWidth must be an integer multiple of  
 ChromaSubsampleHoriz and TileLength must be an integer multiple of  
 ChromaSubsampleVert. Padding will occur to tile boundaries.  
  
 The default values of this field are [ 2,2 ]. Thus, YCbCr data is  
 downsampled by default!