Apple Computer, Inc., is not responsible for the contents of this article.

TIFF (Tag Image File Format): Specifications (5 of 7)

TIFF (Tag Image File Format): Specifications

This article last reviewed: 12 February 1988

This series of seven articles (prepared jointly by Aldus and Microsoft) has eight parts:

Introduction

This memorandum has been prepared jointly by Aldus and Microsoft in conjunction with leading scanner and printer manufacturers. This document does not represent a commitment on the part of either Microsoft or Aldus to provide support for this file format in any application. When responding to specific issues raised in this memo, or when requesting additional tag or field assignments, please address your correspondence to either:

Tim Davenport
Aldus Corporation
411 First Ave. South
Suite 200
Seattle, WA 98104

Manny Vellon
Windows Marketing Group
Microsoft Corporation
16011 NE 36th Way
Box 97017
Redmond, WA 98073-9717

Revision Notes

This release of the TIFF specification has been given a Revision number. It is really the fourth major revision, so the Revision number was set to 4.0.

Abstract

This document describes TIFF, a tag based file format that is designed to promote the interchange of digital image data.

The fields were defined primarily with desktop publishing and related applications in mind, although it is conceivable that other sorts of imaging applications may find TIFF to be useful.

The general scenario for which TIFF was invented assumes that applications software for scanning or painting creates a TIFF file, which can then be read and incorporated into a "document" or "publication" by an application such as a desktop publishing package.

The intent of TIFF is to organize and codify existing practice with respect to the definition and usage of "desktop" digital data, not to blaze new paths or promote unproven techniques. Yet a very high priority has been given to structuring the data in such a way as to minimize the pain of future additions. TIFF was designed to be a very extensible interchange format.

TIFF is not a printer language or page description language, nor is it intended to be a general document interchange standard. It may be useful as is for some image editing applications, but is probably inappropriate for -- and would thus need to be translated into some intermediate data structures by -- other image editing applications. The primary design goal was to provide a rich environment within which the exchange of image data between application programs can be accomplished. This richness is required in order to take advantage of the varying capabilities of scanners and similar devices. TIFF is therefore designed to be a superset of existing image file formats for "desktop" scanners (and paint programs and anything else that produces images with pixels in them) and will be enhanced on a continuing basis as new capabilities arise.

Although TIFF is claimed to be in some sense a rich format, it can easily be used for simple scanners and applications as well, since the application developer need only be concerned with the capabilities that he requires. The mechanisms for accomplishing this goal are discussed in the next section.

TIFF is intended to be independent of specific operating systems, filing systems, compilers, and processors. The only significant assumption is that the storage medium supports something like a "file," defined as a sequence of 8-bit bytes, where the bytes are numbered from 0 to N. The largest possible TIFF file is 2**32 bytes. Since pointers (byte offsets) are used liberally, a TIFF file is most easily read from a random access device, although it is possible to read and write TIFF files on sequential media such as magnetic tape.

The recommended MS-DOS file extension for TIFF files is ".TIF". The recommended Macintosh filetype is "TIFF". Conventions in other computing environments have not yet been established.

1. Conformance

Many of the application programs that read the contents of TIFF image files will not support all of the features described in this document. In some cases, little more than the default options will be supported. It is up to each organization to determine the costs and benefits associated with different levels of conformity. Therefore, claims of conformity to this specification should be interpreted with a certain amount of caution.

It follows that the usage of this specification does not preclude the need for coordination between image file writers and image file readers. It is up to the application designer that initially writes a file in this format to verify that the desired file options are supported by the applications that will read the file.

2. Structure

In TIFF, individual fields are identified with a unique tag. This allows particular fields to be present or absent from the file as required by the application.

Some TIFF files will have only a few fields in them; others will have many. Software that creates TIFF files should write out as many fields as it believes will be meaningful and useful (and no more). Software that reads TIFF files should do the best it can with the fields that it finds there.

There are many ways in which a tag-oriented file format scheme can be implemented. TIFF uses the following approach:

There are three main parts to a TIFF file. First is a short image file header. Next is a directory of all the fields that are to be found in this file. Finally, we have the data for the fields.

3. Header and Directory

A TIFF file begins with a small amount of positionally defined data, containing the following information:

Bytes 0-1:

The first word of the file serves to specify the byte order used within the file. The currently defined values are:

"II" (hex 4949)
"MM" (hex 4D4D)

In the "II" format, byte order is always from least significant to most significant, for both 16-bit and 32-bit integers.

In the "MM" format, byte order is always from most significant to least significant, for both 16-bit and 32-bit integers.

In both formats, character strings are stored into sequential byte locations.

It is certainly not required that all applications software be able to handle both formats. It should be apparent which is the native format for a particular machine.

Bytes 2-3:

The second word of the file is the TIFF version number. This number shouldn't change. This document describes Version 42, so 42 (2A in hex) should be stored in this word.

Bytes 4-7:

This long word contains the offset (in bytes) of the first Image File Directory. The directory may be at any location in the file after the header but must begin on a word boundary.

(The term "byte offset" is always used in this document to refer to a location with respect to the beginning of the file. The first byte of the file has an offset of 0.)

An IFD consists of a 2-byte count of the number of entries (i.e., the number of fields), followed by a sequence of 12-byte field entries, followed by a 4-byte offset of the next Image File Directory (or 0 if none). Each 12-byte field entry has the following format:

Bytes 0-1 contain the Tag for the field. Bytes 2-3 contain the field Type. Bytes 4-7 contain the Length ("Count" might have been a better term) of the field. Bytes 8-11 contain the file offset (in bytes) of the Value for the field. The Value is expected to begin on a word boundary; the corresponding file offset will thus be an even number.

The entries in an IFD must be sorted in ascending order by Tag. Note that this is not the order in which the fields are described in this document. The Values to which directory entries point need not be in any particular order in the file.

If the Value fits within 4 bytes, the Offset is interpreted to contain the

Value instead of pointing to the Value, to save a little time and space. If the Value is less than 4 bytes, it is left-justified. Whether or not it fits within 4 bytes can be determined by looking at the Type and Length of the field.

The Length part is specified in terms of the data type. A single 16-bit word (SHORT) has a Length of 1, not 2, for example. The data types and their lengths are described below:

1 = BYTE. 8-bit unsigned integer.
2 = ASCII. 8-bit bytes that store ASCII codes; the last byte must be null.
3 = SHORT. A 16-bit (2-byte) unsigned integer.
4 = LONG. A 32-bit (4-byte) unsigned integer.
5 = RATIONAL. Two LONGs: the first represents the numerator of a fraction, the second the denominator.

The value of the Length part of an ASCII field entry includes the null. If padding is necessary, the Length does not include the pad byte.

The reader should check the type to ensure that it is what he expects. TIFF currently allows more than 1 valid type for a given field. For example, ImageWidth and ImageLength were specified as having type SHORT. Very large images with more than 64k rows or columns are possible with some devices even now. Rather than add parallel LONG tags for these fields, it is cleaner to allow both SHORT and LONG for ImageWidth and similar fields. Writers of TIFF files are, however, encouraged to use the default type values as indicated in this document to insure compatibility with existing TIFF reader applications.

Note that there may be more than one IFD. Each IFD is said to define a "subfile." One potential use of subsequent subfiles is to describe a "sub-image" that is somehow related to the main image, such as a reduced resolution or "screen resolution" image. Another use is to represent multiple "page" images -- for example, a facsimile document requiring more than one page. Subsequent IFDs will in general contain many of the same fields as the first IFD but will usually point to or contain different values for those fields.

4. Definitions

The TIFF structure itself is not specific to imaging applications in any way. It is only the definitions of the fields themselves that jointly describe an image. Before we begin describing the fields, a few image related definitions may be useful.

An image is defined to be a rectangular array of "pixels," each of which

consists of one or more "samples." With monochromatic data, we have one sample per pixel, and "sample" and "pixel" can be used interchangeably. Color data usually contains three samples per pixel, as in, for example, an RGB scheme.

5. The Fields

The following fields are defined in this version of TIFF. More will be added in future versions, if possible in such a way so as not to break old software that encounters a newer TIFF file. An attempt has been made to group related fields, although the grouping is necessarily somewhat arbitrary.

The documentation for each field contains the name of the field (quite arbitrary, but convenient), the Tag value, the field Type, the Number of Values (N) expected (per IFD, in the case of multiple subfiles), comments describing the field, and the default, if any. The default value is used if the field does not exist.

A fairly large number of fields has already been defined, and the number will grow. Please keep in mind that many common images can be described using only a handful of these fields (see the Examples section).

General Description

SubfileType
Tag = 255 (FF)
Type = SHORT
N = 1

A general indication of the kind of data that is contained in this subfile. Currently defined values are:

1 = full resolution image data - ImageWidth, ImageLength, and StripOffsets are required fields.

2 = reduced resolution image data - ImageWidth, ImageLength, and StripOffsets are required fields. It is further assumed that a reduced resolution image is a reduced version of the entire extent of the corresponding full resolution data.

3 = Single page of a multi-page image (see the PageNumber tag description).

If your kind of image data doesn't fit nicely into either description, contact either Aldus or Microsoft to define an additional value. Note that both image types can be found in a single TIFF file, with each subfile described by its own IFD.
No default.

Data Architecture

ImageWidth
Tag = 256 (100)
Type = SHORT
N = 1

The image's width, in pixels (X: horizontal). The number of columns in the image.
No default.

ImageLength
Tag = 257 (101)
Type = SHORT
N = 1

The image's length (height) in pixels (Y: vertical). The number of rows (sometimes described as "scan lines") in the image. ImageLength and ImageWidth refer only to how the pixels are stored in the file and do not imply anything about where the visual "top" or "left side" of the image may be. See Orientation for this information.
No default.

RowsPerStrip
Tag = 278 (116)
Type = SHORT or LONG
N = 1

The number of rows per strip. The image data is organized into strips for fast access to individual rows when the data is compressed (though this field is valid even if the data is not compressed).

Note that either SHORT or LONG values can be used to specify RowsPerStrip. SHORT values may be used for small TIFF files. It should be noted, however, that earlier TIFF specifications required LONG values and that some software may not expect SHORT values.

Default is 2**32 - 1, which is effectively infinity. That is, the entire image is one strip.

[StripsPerImage]
N = 1

The number of strips per image. This value is not a field, since it can be computed from two other fields, but it is convenient to give it a name in order to clarify the use of other fields. The equation to use is StripsPerImage = (ImageLength + RowsPerStrip - 1) / RowsPerStrip, assuming

integer arithmetic.

StripOffsets
Tag = 273 (111)
Type = SHORT or LONG
N = StripsPerImage for PlanarConfiguration equal to 1.
= SamplesPerPixel * StripsPerImage for PlanarConfiguration equal to 2

For each strip, the byte offset of that strip. The offset is specified with respect to the beginning of the TIFF file. Note that this implies that each strip has a location independent of the locations of other strips. This feature may be useful for certain editing applications. This field is the only way for a reader to find the image data, and hence must exist.

Note that either SHORT or LONG values can be used to specify the strip offsets. SHORT values may be used for small TIFF files. It should be noted, however, that earlier TIFF specifications required LONG strip offsets and that some software may not expect SHORT values.
No default.

5. The Fields, continued

StripByteCounts
Tag = 279 (117)
Type = LONG
N = StripsPerImage for PlanarConfiguration equal to 1.
= SamplesPerPixel * StripsPerImage for PlanarConfiguration equal to 2

For each strip, the number of bytes in that strip.
No default.

SamplesPerPixel
Tag = 277 (115)
Type = SHORT
N = 1

The number of samples per pixel. Usually 1 for monochromatic data and 3 for color data (i.e. one sample for each of the color planes.)
Default = 1.

BitsPerSample
Tag = 258 (102)
Type = SHORT
N = SamplesPerPixel

Number of bits per sample. Note that this tag allows a different number of bits per sample for each sample corresponding to a pixel. For example, RGB color data could use a different number of bits per sample for each of the three color planes.
Default = 1.

PlanarConfiguration
Tag = 284 (11C)
Type = SHORT
N = 1

1 = the sample values for each pixel are stored contiguously, so that there is a single image plane. See PhotometricInterpretation to determine the order of the samples within the pixel data.
2 = the samples are stored in separate "sample planes." The values in StripOffsets and StripByteCounts are then arranged as a 2-dimensional array, with SamplesPerPixel rows and StripsPerImage columns. (All of the columns for row 0 are stored first, followed by the columns of row 1, and so on.) PhotometricInterpretation describes the type of data that is stored in each sample plane.

If SamplesPerPixel is 1, a PlanarConfiguration value of 1 is equivalent to a value of 2.
No default.

Compression
Tag = 259 (103)
Type= SHORT
N= SamplesPerPixel for PlanarConfiguration equal to 1 or 2.

Note that a value is provided for each sample, allowing different compression schemes to be applied to different planes of data.

1 = No compression, but pack data into bytes as tightly as possible, with no unused bits except at the end of a row. See also FillOrder. The bytes are stored as an array of type BYTE, for BitsPerSample <= 8, SHORT if BitsPerSample > 8 and <= 16, and LONG if BitsPerSample > 16 and <= 32. The byte ordering of data >8 bits must be consistent with that specified in the TIFF file header (bytes 0 and 1). "Intel" format files will have the least significant bytes preceeding the most significant bytes while "Motorola" format files will have the opposite.

If the number of bits per sample is not a power of 2, and you are willing to give up some space for better performance, you may wish to use the next higher power of 2. For example, if your data can be represented in 6 bits, you may wish to specify that it is 8 bits deep. If you take this approach,

you should be sure that MinSampleValue and MaxSampleValue are given correct values (probably 0 and 63 for intrinsically 6-bit data.) TIFF file readers should use MinSampleValue and MaxSampleValue to determine the range of values in the data rather than BitsPerSample.

Rows are required to begin on byte boundaries.

2 = CCITT Group 3 1-Dimensional Modified Huffman run length encoding. BitsPerSample must be 1, since this type of compression is defined only for "binary" images.

3 = Facsimile-compatible CCITT Group 3, exactly as specified in "Standardization of Group 3 facsimile apparatus for document transmission," Recommendation T.4, Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, 1985, pages 16 through 31. Each strip must begin on a byte boundary. (But recall that an image can be a single strip.) Rows that are not the first row of a strip are not required to begin on a byte boundary. The data is stored as bytes, not words -- byte-reversal is not allowed. Note that the FillOrder field still applies. See the Group3Options field for Group 3 options such as 1D vs 2D coding.

4 = Facsimile-compatible CCITT Group 4, exactly as specified in "Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus," Recommendation T.6, Volume VII, Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services, The International Telegraph and Telephone Consultative Committee (CCITT), Geneva, 1985, pages 40 through 48. Each strip must begin on a byte boundary. Rows that are not the first row of a strip are not required to begin on a byte boundary. The data is stored as bytes, not words. Note that the FillOrder field still applies. See the Group4Options field for Group 4 options.

32771 = the same thing as Compression type 1 (no compression), except that each row begins on the next available word boundary, instead of byte boundary.

32773 = PackBits compression, a relatively simple byte-oriented run-length scheme.

Data compression only applies to pixel data, as pointed to by StripOffsets. All other TIFF information is unaffected.

To be determined are additional compression schemes for gray and colored images. We encourage your suggestions, especially if accompanied by full specifications and performance information. It is of course desirable to minimize the number of compression schemes that are being used, but this is clearly an area in which extremely significant time and space tradeoffs

exist.
Default = 1.

Group3Options
Tag = 292 (124)
Type = LONG
N = 1

This field is made up of a set of 32 flag bits. Unused bits are expected to be 0. Bit 0 is the low-order bit. It is probably not safe to try to read the file if any bit of this field is set that you don't know the meaning of.

Bit 0 is 1 for 2-dimensional coding (else 1-dimensional is assumed). For 2-D coding, if more than one strip is specified, each strip must begin with a 1-dimensionally coded line. That is, RowsPerStrip should be a multiple of "Parameter K" as documented in the CCITT specification.

Bit 1 is 1 if uncompressed mode is used.

Bit 2 is 1 if fill bits have been added as necessary before EOL codes such that EOL always ends on a byte boundary, thus ensuring an eol-sequence of a 1 byte preceded by a zero nibble: xxxx-0000 0000-0001.

Default is 0, for basic 1-dimensional coding.

Group4Options
Tag = 293 (125)
Type = LONG
N = 1

This field is made up of a set of 32 flag bits. Unused bits are expected to be 0. Bit 0 is the low-order bit. It is probably not safe to try to read the file if any bit of this field is set that you don't know the meaning of. Gray scale and color coding schemes are under study, and will be added when finalized.

For 2-D coding, each strip is encoded as if it were a separate image. In particular, each strip begins on a byte boundary; and the coding for the first row of a strip is encoded independently of the previous row, using horizontal codes, as if the previous row is entirely white. Each strip ends with the 24-bit end-of-facsimile block (EOFB).

Bit 0 is unused.

Bit 1 is 1 if uncompressed mode is used.

Default is 0, for basic 2-dimensional binary compression.

FillOrder
Tag = 266 (10A)
Type = SHORT
N = 1

The order of data values within a byte.
1 = most significant bits of the byte are filled first. That is, data values (or code words) are ordered from high order bit to low order bit within a byte.
2 = least significant bits are filled first.
Default is FillOrder = 1.

5. The Fields, continued

Threshholding
Tag = 263 (107)
Type = SHORT
N = 1

1 = a bilevel "line art" scan. BitsPerSample must be 1.
2 = a "halftone" or "dithered" scan, usually of continuous tone data such as photographs. BitsPerSample must be 1.
3 = Error Diffused.
Default is Threshholding = 1.

CellWidth
Tag = 264 (108)
Type = SHORT
N = 1

The width, in 1-bit samples, of the dithering/halftoning matrix. Assumes that Threshholding = 2. That is, this field is only relevant if Threshholding = 2.
No default.

CellLength
Tag = 265 (109)
Type = SHORT
N = 1

The length, in 1-bit samples, of the dithering/halftoning matrix. Assumes that Threshholding = 2. This field and the previous field may be useful for converting from halftoned to true gray level data.
No default.

Photometrics

These fields are useful in determining the visual meaning of the sample data.

MinSampleValue
Tag = 280 (118)
Type = SHORT
N = SamplesPerPixel

The minimum valid sample value.
Default is 0.

MaxSampleValue
Tag = 281 (119)
Type = SHORT
N = SamplesPerPixel

The maximum valid sample value.
Default is 2**(BitsPerSample) - 1.

PhotometricInterpretation
Tag = 262 (106)
Type = SHORT
N = 1

0 = MinSampleValue should be imaged as white. MaxSampleValue should be imaged as black. If the bit-map represents gray scale, then the values between the minimum and maximum sample values should be interpreted according to either the gray scale response curve information (if included) or according to the result of some more arbitrary rule. See GrayResponseCurve.

1 = MinSampleValue should be imaged as black. MaxSampleValue should be imaged as white. If the bit-map represents gray scale, then the values between the minimum and maximum sample values should be interpreted according to either the gray scale response curve information (if included) or according to the result of some more arbitrary rule.

2 = RGB. In the RGB model, a color is described as a combination of the three primary colors of light (red, green, and blue) in particular concentrations. For each of the three samples, MinSampleValue represents minimum intensity, and MaxSampleValue represents maximum intensity. For

PlanarConfiguration = 1, the samples are stored in the indicated order within a pixel: first Red, then Green, then Blue. For PlanarConfiguration = 2, the sample planes are stored in the indicated order: first the Red sample plane, then the Green plane, then the Blue plane.

The Red, Green and Blue intensity values are defined according to the NTSC specifications for primary color chromaticity. These specifications assume the illumination to be CIE D6500. See the Red, Green and Blue color response curve tags.

Note: some compression schemes, such as the CCITT schemes, imply a particular PhotometricInterpretation. Therefore, when reading such data, TIFF readers should ignore PhotometricInterpretation. And, ideally, TIFF writers should not write out the field when using one of these schemes.

No default.

GrayResponseUnit
Tag = 290 (122)
Type = SHORT
N = 1

1 = number represents tenths of a unit.
2 = number represents hundredths of a unit.
3 = number represents thousandths of a unit.
4 = number represents ten-thousandths of a unit.
5 = number represents hundred-thousandths of a unit.
Default is 2.

GrayResponseCurve
Tag = 291 (123)
Type = SHORT
N = 2**BitsPerSample

The purpose of the gray response curve and the gray units is to further provide photometric interpretation information for gray scale image data. The gray response curve specifies for given levels of gray between the minimum and maximum sample values the actual photometric gray level of the value. It represents this gray level in terms of optical density.

The GrayScaleResponseUnits specifies the accuracy of the information contained in the curve. Since optical density is specified in terms of fractional numbers, this tag is necessary to know how to interpret the stored integer information. For example, if GrayScaleResponseUnits is set to 4 (ten-thousandths of a unit), and a GrayScaleResponseCurve number for gray level 4 is 3455, then the resulting actual value is 0.3455.

If the gray scale response curve is known for the data in the TIFF file, and if the gray scale response of the output device is known, then an intelligent conversion can be made between the input data and the output device. For example, the output can be made to look just like the input. In addition, if the input image lacks contrast (as can be seen from the response curve), then appropriate contrast enhancements can be made.

The purpose of the grey scale response curve is to act as a "lookup" table mapping values from 0 to 2**BitsPerSample-1 into specific intensity values. Refer to the PhotometricInterpretation tag to determine how the mapping should be done.

5. The fields, concluded

ColorResponseUnit
Tag = 300 (12C)
Type = SHORT
N = 1

1 = number represents tenths of a unit.
2 = number represents hundredths of a unit.
3 = number represents thousandths of a unit.
4 = number represents ten-thousandths of a unit.
5 = number represents hundred-thousandths of a unit.
Default is 2.

ColorResponseCurves
Tag = 301 (12D)
Type = SHORT
N = 2**BitsPerSample (for Red samples) +
2**BitsPerSample (for Green samples) +
2**BitsPerSample (for Blue samples)

This tag defines three color response curves (one each for Red, Green, and Blue color information). The curves are stored sequentially (in red-green-blue order). The size of each table is 2**BitsPerSample, using the BitsPerSample value corresponding to the respective color. The ColorResponseUnit further specifies how each entry in the table is to be interpreted.

The purpose of the color response curves is to act as a "lookup" table mapping values from 0 to 2**BitsPerSample-1 into specific intensity values. The intensity values are as specified by the NTSC color strandard assuming illumination to be CIE D6500.

Correspondence to the Physical World

XResolution
Tag = 282 (11A)
Type = RATIONAL
N = 1

The number of pixels per ResolutionUnit (see below) in the X direction, i.e., in the ImageWidth direction. It is, of course, not mandatory that the image be actually printed at the size implied by this parameter. It is up to the application to use this information as it wishes.

As is the case for many of these fields, XResolution may be invalid and irrelevant for some images (e.g., images made with a hand-held digitizing camera, which has a three-dimensional nature) and should therefore be absent from the image file.
No default.

YResolution
Tag = 283 (11B)
Type = RATIONAL
N = 1

The number of pixels per ResolutionUnit in the Y direction, i.e., in the ImageLength direction.
No default.

ResolutionUnit
Tag = 296 (128)
Type = SHORT
N = 1

To be used with XResolution and YResolution.
1 = no absolute unit of measurement. Used for images that may have a non-square aspect ratio, but no meaningful absolute dimensions.
2 = inch
3 = centimeter
Default is 2

Orientation
Tag = 274 (112)
Type = SHORT
N = 1

1 = The 0th row represents the visual top of the image, and the 0th column represents the visual left hand side.
2 = The 0th row represents the visual top of the image, and the 0th column

represents the visual right hand side.
3 = The 0th row represents the visual bottom of the image, and the 0th column represents the visual right hand side.
4 = The 0th row represents the visual bottom of the image, and the 0th column represents the visual left hand side.
5 = The 0th row represents the visual left hand side of the image, and the 0th column represents the visual top.
6 = The 0th row represents the visual right hand side of the image, and the 0th column represents the visual top.
7 = The 0th row represents the visual right hand side of the image, and the 0th column represents the visual bottom.
8 = The 0th row represents the visual left hand side of the image, and the 0th column represents the visual bottom.
Default is 1.

Document Context

DocumentName
Tag = 269 (10D)
Type = ASCII

The name of the document from which this image was scanned.
No default.

PageName
Tag = 285 (11D)
Type = ASCII

The name of the page from which this image was scanned.
No default.

XPosition
Tag = 286 (11E)
Type = RATIONAL

The X offset of the left side of the image, with respect to the left side of the page, in inches.
No default.

YPosition
Tag = 287 (11F)
Type = RATIONAL

The Y offset of the top of the image, with respect to the top of the page, in inches. In the TIFF coordinate scheme, the positive Y direction is down,

so that YPosition is always positive.
No default.

PageNumber
Tag = 297 (129)
Type = SHORT
N = 2

This tag is used to specify page numbers of a multiple page (e.g. facsimile) document. Two SHORT values are specified. The first value is the page number; the second value is the total number of pages in the document.

Note that pages need not appear in numerical order.

Miscellaneous Strings

ImageDescription
Tag = 270 (10E)
Type = ASCII

Useful or interesting information about the image.
No default.

Make
Tag = 271 (10F)
Type = ASCII

The name of the scanner manufacturer.
No default.

Model
Tag = 272 (110)
Type = ASCII

The model name/number of the scanner.
No default.

Storage Management

These fields may be useful in certain dynamic editing situations. Software that merely reads TIFF files will probably not need to care about these fields. And, of course, software that creates TIFF files is by no means required to write these fields.

FreeOffsets
Tag = 288 (120)
Type = LONG

For each "free block" in the file, its byte offset.
No default.

FreeByteCounts
Tag = 289 (121)
Type = LONG

For each "free block" in the file, the number of bytes in the block.

This article last reviewed: 12 February 1988

6. Examples

A binary image from a paint program might contain only SubfileType, ImageWidth, ImageLength, StripOffsets, and PhotometricInterpretation fields.

A typical line art scan might require that XResolution and YResolution be added to the above list.

7. Private Fields

An organization may wish to store with the image file information that is meaningful only to that organization. Tags numbered 32768 or higher are reserved for that purpose. Upon request, the administrator will allocate and register a block of private tags for an organization, to avoid possible conflicts with other organizations.

Private enumerated values can be accommodated in a similar fashion. Enumeration constants numbered 32768 or higher are reserved for private usage. Upon request, the administrator will allocate and register a block of enumerated values for a particular field, to avoid possible conflicts.

Tags and values which are allocated in the private number range are not prohibited from being included in a future revision of this specification. Several such instances can be found in this revision.

8. A List of Possible Future Enhancements

In the future TIFF will very likely be expanded to support more compression schemes, more photometric schemes, color lookup tables, and non-rectangular images. Please refer all questions regarding enhancements to TIFF to the

contacts listed at the beginning of the document. Written submissions should be in Microsoft Windows Write format, to ensure timely and error-free incorporation into the specification.

TIFF: Caution - Not All TIFF Files are the Same

This article last reviewed: 22 June 1989

TOPIC ----------

A user tried to copy a file created with MacView/AGFA Scanner for porting to Microsoft Windows, and it didn't work. Are there differences between the TIFF used on the Macintosh and the one used by Microsoft?

DISCUSSION ----------

Unfortunately, Apple Tech Comm as neither a MacView/AGFA Scanner nor a copy of Microsoft Windows to test this problem. However, the ability of applications to read TIFF files, created by another application, is based on the conformance of both applications to the TIFF specification, not the CPU type or operating system. Below is a copy of the TIFF Conformance Specification.

Conformance

"Many of the application programs that read the contents of TIFF image files will not support all of the features described in this document. In some cases, little more than the default options will be supported. It is up to each organization to determine the costs and benefits associated with different levels of conformity. Therefore, claims of conformity to this specification should be interpreted with a certain amount of caution.

"It follows that the usage of this specification does not preclude the need for coordination between image file writers and image file readers. It is up to the application designer that initially writes a file in this format to verify that the desired file options are supported by the applications that will read the file."

To resolve this problem, contact the vendors of both products to confirm the conformance of the applications to the TIFF specification