TIFF (Tag Image File Format): Specifications (2 of 7)

Revised: 3/22/88
Security: Everyone

TIFF (Tag Image File Format): Specifications (2 of 7)

==========

This article last reviewed: 12 February 1988

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.

Tech Info Library Article Number:2665