

**NAME**

`fax2ps` – convert a TIFF facsimile to compressed POSTSCRIPT™

**SYNOPSIS**

`fax2ps` [ *options* ] [ *file...* ]

**DESCRIPTION**

`fax2ps` reads one or more TIFF facsimile image files and prints a compressed form of POSTSCRIPT on the standard output that is suitable for printing.

By default, each page is scaled to reflect the image dimensions and resolutions stored in the file. The `-x` and `-y` options can be used to specify the horizontal and vertical image resolutions (lines/inch), respectively. If the `-S` option is specified, each page is scaled to fill an output page. The default output page is 8.5 by 11 inches. Alternate page dimensions can be specified in inches with the `-W` and `-H` options.

By default `fax2ps` generates POSTSCRIPT for all pages in the file. The `-p` option can be used to select one or more pages from a multi-page document.

`fax2ps` generates a compressed form of POSTSCRIPT that is optimized for sending pages of text to a POSTSCRIPT printer attached to a host through a low-speed link (such as a serial line). Each output page is filled with white and then only the black areas are drawn. The POSTSCRIPT specification of the black drawing operations is optimized by using a special font that encodes the move-draw operations required to fill the black regions on the page. This compression scheme typically results in a substantially reduced POSTSCRIPT description, relative to the straightforward imaging of the page with a POSTSCRIPT *image* operator. This algorithm can, however, be ineffective for continuous-tone and white-on-black images. For these images, it sometimes is more efficient to send the raster bitmap image directly; see `tiff2ps(1)`.

**OPTIONS**

- `-p number` Print only the indicated page. Multiple pages may be printed by specifying this option more than once.
- `-x resolution`  
Use *resolution* as the horizontal resolution, in dots/inch, of the image data. By default this value is taken from the file.
- `-y resolution`  
Use *resolution* as the vertical resolution, in lines/inch, of the image data. By default this value is taken from the file.
- `-S` Scale each page of image data to fill the output page dimensions. By default images are presented according to the dimension information recorded in the TIFF file.
- `-W width` Use *width* as the width, in inches, of the output page. The default page width is 8.5 inches.
- `-H height` Use *height* as the height, in inches, of the output page. The default page height is 11 inches.

**DIAGNOSTICS**

Some messages about malformed TIFF images come from the TIFF library.

Various messages about badly formatted facsimile images may be generated due to transmission errors in received facsimile. `fax2ps` attempts to recover from such data errors by resynchronizing decoding at the end of the current scanline. This can result in long horizontal black lines in the resultant POSTSCRIPT image.

**NOTES**

If the destination printer supports POSTSCRIPT Level II then it is always faster to just send the encoded bitmap generated by the `tiff2ps(1)` program.

**BUGS**

`fax2ps` should probably figure out when it is doing a poor job of compressing the output and just generate POSTSCRIPT to image the bitmap raster instead.

**SEE ALSO**

`tiff2ps(1)`, `libtiff(3)`

**NAME**

fax2tiff – create a TIFF Class F fax file from raw fax data

**SYNOPSIS**

**fax2tiff** [ *options* ] [ **-o** *output.tif* ] *input.g3*

**DESCRIPTION**

*Fax2tiff* creates a TIFF file containing CCITT Group 3 or Group 4 encoded data from one or more files containing “raw” Group 3 encoded data (typically obtained directly from a fax modem). By default, each row of data in the resultant TIFF file is 2-dimensionally encoded and padded or truncated to 1728 pixels, as needed. The resultant image is a set of low resolution (98 lines/inch) or medium resolution (196 lines/inch) pages, each of which is a single strip of data. The generated file conforms to the TIFF Class F (FAX) specification for storing facsimile data. This means, in particular, that each page of the data does **not** include the trailing *return to control* (RTC) code; as required for transmission by the CCITT Group 3 specifications. The old, “classic”, format is created if the **-c** option is used. (The Class F format can also be requested with the **-f** option.)

The default name of the output image is *fax.tif*; this can be changed with the **-o** option. Each input file is assumed to be a separate page of facsimile data from the same document. The order in which input files are specified on the command line is the order in which the resultant pages appear in the output file.

**OPTIONS**

Options that affect the interpretation of input data are:

- 2** Assume input data is 2-d Huffman encoded.
- B** Assume input data was encoded with black as 0 and white as 1.
- L** Treat input data as having bits filled from least significant bit (LSB) to most significant bit (MSB). (This is the default.)
- M** Treat input data as having bits filled from most significant bit (MSB) to most least bit (LSB).
- R** Specify the vertical resolution, in lines/inch, of the input images. By default input are assumed to have a vertical resolution of 196 lines/inch. If images are low resolution facsimile, a value of 98 lines/inch should be specified.
- W** Assume input data was encoded with black as 1 and white as 0. (This is the default.)

Options that affect the output file format are:

- 1** Force output to be compressed with the 1-dimensional version of the CCITT Group 3 Huffman encoding algorithm.
- 4** Force output to be compressed with the CCITT Group 4 Huffman encoding.
- o** Specify the name of the output file.
- p** Force the last bit of each *End Of Line* (EOL) code to land on a byte boundary. This “zero padding” will be reflected in the contents of the *Group3Options* tag of the resultant TIFF file.
- s** Stretch the input image vertically by writing each input row of data twice to the output file.
- v** Force *fax2tiff* to print the number of rows of data it retrieved from the input file.

**DIAGNOSTICS**

The following warnings and errors come from the decoding routines in the library.

**Warning, %s: Premature EOL at scanline %d (x %d).\n.** The input data had a row that was shorter than the expected value of 1728. The row is padded with white.

**%s: Premature EOF at scanline %d (x %d).\n.** The decoder ran out of data in the middle of a scanline. The resultant row is padded with white.

**%s: Bad code word at row %d, x %d\n.** An invalid Group 3 *code* was encountered while decoding the input file. The row number and horizontal position is given. The remainder of the input row is discarded, while the corresponding output row is padded with white.

**%s: Bad 2D code word at scanline %d.\n.** An invalid Group 4 or 2D Group 3 *code* was encountered while decoding the input file. The row number and horizontal position is given. The remainder of the

input row is discarded, while the corresponding output row is padded with white.

**BUGS**

Should not have the constant width 1728 built into it. Input data are assumed to have a “top left” orientation; it should be possible to override this assumption from the command line.

**SEE ALSO**

*CCITT Recommendation T.4* (Standardization of Group 3 Facsimile Apparatus for Document Transmission).

*The Spirit of TIFF Class F*, an appendix to the TIFF 5.0 specification prepared by Cygnet Technologies.

*tiffinfo(1)*, *tiffdither(1)*, *tiffgt(1)*, *libtiff(3)*.

**NAME**

gif2tiff – create a TIFF file from a GIF87 format image file

**SYNOPSIS**

**gif2tiff** [ *options* ] *input.gif output.tif*

**DESCRIPTION**

*Gif2tiff* converts a file in the GIF87 format to TIFF. The TIFF image is created as a palette image, with samples compressed with the Lempel-Ziv & Welch algorithm (*Compression=5*). These characteristics can be overridden, or explicitly specified with the options described below.

**OPTIONS**

- c** Specify a compression scheme to use when writing image data: **-c none** for no compression, **-c packbits** for the PackBits compression algorithm, **-c zip** for the Deflate compression algorithm, and **-c lzw** for Lempel-Ziv & Welch (the default).
- r** Write data with a specified number of rows per strip; by default the number of rows/strip is selected so that each strip is approximately 8 kilobytes.

**NOTES**

The program is based on Paul Haeberli's *fromgif* program which, in turn, is based on Marcel J.E. Mol's GIF reader.

**BUGS**

Should have more options to control output format.

**SEE ALSO**

*pal2rgb(1)*, *tiffinfo(1)*, *tiffcp(1)*, *tiffmedian(1)*, *libtiff(3)*

**NAME**

pal2rgb – convert a palette color TIFF image to a full color image

**SYNOPSIS**

**pal2rgb** [ *options* ] *input.tif* *output.tif*

**DESCRIPTION**

*Pal2rgb* converts a palette color TIFF image to a full color image by applying the colormap of the palette image to each sample to generate a full color RGB image.

**OPTIONS**

Options that affect the interpretation of input data are:

- C** This option overrides the default behaviour of *pal2rgb* in determining whether or not colormap entries contain 16-bit or 8-bit values. By default the colormap is inspected and if no colormap entry greater than 255 is found, the colormap is assumed to have only 8-bit values; otherwise 16-bit values (as required by the TIFF specification) are assumed. The –**C** option can be used to explicitly specify the number of bits for colormap entries: –**C 8** for 8-bit values, –**C 16** for 16-bit values.

Options that affect the output file format are:

- p** Explicitly select the planar configuration used in organizing data samples in the output image: –**p contig** for samples packed contiguously, and –**p separate** for samples stored separately. By default samples are packed.
- c** Use the specific compression algorithm to encoded image data in the output file: –**c packbits** for Macintosh Packbits, –**c lzw** for Lempel-Ziv & Welch, –**c zip** for Deflate, –**c none** for no compression. If no compression-related option is specified, the input file's compression algorithm is used.
- r** Explicitly specify the number of rows in each strip of the output file. If the –**r** option is not specified, a number is selected such that each output strip has approximately 8 kilobytes of data in it.

**BUGS**

Only 8-bit images are handled.

**SEE ALSO**

*tiffinfo*(1), *tiffcp*(1), *tiffmedian*(1), *libtiff*(3)

**NAME**

ppm2tiff – create a TIFF file from a PPM image file

**SYNOPSIS**

**ppm2tiff** [ *options* ] [ *input.ppm* ] *output.tif*

**DESCRIPTION**

*ppm2tiff* converts a file in the PPM image format to TIFF. By default, the TIFF image is created with data samples packed (*PlanarConfiguration*=1), compressed with the Lempel-Ziv & Welch algorithm (*Compression*=5), and with each strip no more than 8 kilobytes. These characteristics can be overridden, or explicitly specified with the options described below

If the PPM file contains greyscale data, then the *PhotometricInterpretation* tag is set to 1 (min-is-black), otherwise it is set to 2 (RGB).

If no PPM file is specified on the command line, *ppm2tiff* will read from the standard input.

**OPTIONS**

- c** Specify a compression scheme to use when writing image data: **-c none** for no compression, **-c packbits** for the PackBits compression algorithm, **-c jpeg** for the baseline JPEG compression algorithm, **-c zip** for the Deflate compression algorithm, and **-c lzw** for Lempel-Ziv & Welch compression (the default).
- r** Write data with a specified number of rows per strip; by default the number of rows/strip is selected so that each strip is approximately 8 kilobytes.
- R** Mark the resultant image to have the specified X and Y resolution (in dots/inch).

**SEE ALSO**

*tiffinfo*(1), *tiffcp*(1), *tiffmedian*(1), *libtiff*(3)

**NAME**

ras2tiff – create a TIFF file from a Sun rasterfile

**SYNOPSIS**

**ras2tiff** [ *options* ] *input.ras output.tif*

**DESCRIPTION**

*ras2tiff* converts a file in the Sun rasterfile format to TIFF. By default, the TIFF image is created with data samples packed (*PlanarConfiguration*=1), compressed with the Lempel-Ziv & Welch algorithm (*Compression*=5), and with each strip no more than 8 kilobytes. These characteristics can be overridden, or explicitly specified with the options described below.

Any colormap information in the rasterfile is carried over to the TIFF file by including a *Colormap* tag in the output file. If the rasterfile has a colormap, the *PhotometricInterpretation* tag is set to 3 (palette); otherwise it is set to 2 (RGB) if the depth is 24 or 1 (min-is-black) if the depth is not 24.

**OPTIONS**

- c** Specify a compression scheme to use when writing image data: **-c none** for no compression, **-c packbits** for the PackBits compression algorithm, **-c jpeg** for the baseline JPEG compression algorithm, **-c zip** for the Deflate compression algorithm, and **-c lzw** for Lempel-Ziv & Welch (the default).
- r** Write data with a specified number of rows per strip; by default the number of rows/strip is selected so that each strip is approximately 8 kilobytes.

**BUGS**

Does not handle all possible rasterfiles. In particular, *ras2tiff* does not handle run-length encoded images.

**SEE ALSO**

*pal2rgb*(1), *tiffinfo*(1), *tiffcp*(1), *tiffmedian*(1), *libtiff*(3)

**NAME**

`rgb2ycbcr` – convert non-YCbCr TIFF images to a YCbCr TIFF image

**SYNOPSIS**

`rgb2ycbcr` [ *options* ] *src1.tif src2.tif ... dst.tif*

**DESCRIPTION**

`rgb2ycbcr` converts RGB color, greyscale, or bi-level TIFF images to YCbCr images by transforming and sampling pixel data. If multiple files are specified on the command line each source file is converted to a separate directory in the destination file.

By default, chrominance samples are created by sampling 2 by 2 blocks of luminance values; this can be changed with the `-h` and `-v` options. Output data are compressed with the LZW compression scheme, by default; an alternate scheme can be selected with the `-c` option. By default, output data are compressed in strips with the number of rows in each strip selected so that the size of a strip is never more than 8 kilobytes; the `-r` option can be used to explicitly set the number of rows per strip.

**OPTIONS**

- `-c` Specify a compression scheme to use when writing image data: `-c none` for no compression, `-c packbits` for the PackBits compression algorithm, `-c jpeg` for the JPEG compression algorithm, and `-c lzw` for Lempel-Ziv & Welch (the default).
- `-h` Set the horizontal sampling dimension to one of: 1, 2 (default), or 4.
- `-r` Write data with a specified number of rows per strip; by default the number of rows/strip is selected so that each strip is approximately 8 kilobytes.
- `-v` Set the vertical sampling dimension to one of: 1, 2 (default), or 4.

**SEE ALSO**

*tiffinfo(1)*, *tiffcp(1)*, *libtiff(3)*

**NAME**

`sgi2tiff` – create a TIFF file from an SGI image file

**SYNOPSIS**

`sgi2tiff` [ *options* ] *input.rgb* *output.tif*

**DESCRIPTION**

`sgi2tiff` converts a file in the SGI image format to TIFF. By default, the TIFF image is created with data samples packed (*PlanarConfiguration*=1), compressed with the Lempel-Ziv & Welch algorithm (*Compression*=5), and with each strip no more than 8 kilobytes. These characteristics can be overridden, or explicitly specified with the options described below.

**OPTIONS**

- c** Specify a compression scheme to use when writing image data: **-c none** for no compression, **-c packbits** for the PackBits compression algorithm), **-c jpeg** for the baseline JPEG compression algorithm, **-c zip** for the Deflate compression algorithm, and **-c lzw** for Lempel-Ziv & Welch (the default).
- p** Explicitly select the planar configuration used in organizing data samples in the output image: **-p contig** for samples packed contiguously, and **-p separate** for samples stored separately. By default samples are packed.
- r** Write data with a specified number of rows per strip; by default the number of rows/strip is selected so that each strip is approximately 8 kilobytes.

**BUGS**

Does not record colormap information.

**SEE ALSO**

*tiffinfo*(1), *tiffcp*(1), *tiffmedian*(1), *libtiff*(3)

**NAME**

thumbnail – create a TIFF file with thumbnail images

**SYNOPSIS**

**thumbnail** [ *options* ] *input.tif* *output.tif*

**DESCRIPTION**

*thumbnail* is a program written to show how one might use the SubIFD tag (#330) to store thumbnail images. *thumbnail* copies a TIFF Class F facsimile file to the output file and for each image an 8-bit greyscale *thumbnail sketch*. The output file contains the thumbnail image with the associated full-resolution page linked below with the SubIFD tag.

By default, thumbnail images are 216 pixels wide by 274 pixels high. Pixels are calculated by sampling and filtering the input image with each pixel value passed through a contrast curve.

**OPTIONS**

- w** Specify the width of thumbnail images in pixels.
- h** Specify the height of thumbnail images in pixels.
- c** Specify a contrast curve to apply in generating the thumbnail images. By default pixels values are passed through a linear contrast curve that simply maps the pixel value ranges. Alternative curves are: **exp50** for a 50% exponential curve, **exp60** for a 60% exponential curve, **exp70** for a 70% exponential curve, **exp80** for a 80% exponential curve, **exp90** for a 90% exponential curve, **exp** for a pure exponential curve, **linear** for a linear curve.

**BUGS**

There are no options to control the format of the saved thumbnail images.

**SEE ALSO**

*tiffdump*(1), *tiffgt*(1), *tiffinfo*(1), *libtiff*(3)

**NAME**

tiff2bw – convert a color TIFF image to greyscale

**SYNOPSIS**

**tiff2bw** [ options ] *input.tif output.tif*

**DESCRIPTION**

*Tiff2bw* converts an RGB or Palette color TIFF image to a greyscale image by combining percentages of the red, green, and blue channels. By default, output samples are created by taking 28% of the red channel, 59% of the green channel, and 11% of the blue channel. To alter these percentages, the **-R**, **-G**, and **-B** options may be used.

**OPTIONS**

- c** Specify a compression scheme to use when writing image data: **-c none** for no compression, **-c packbits** for the PackBits compression algorithm, **-c zip** for the Deflate compression algorithm, **-c g3** for the CCITT Group 3 compression algorithm, **-c g4** for the CCITT Group 4 compression algorithm, and **-c lzw** for Lempel-Ziv & Welch (the default).
- r** Write data with a specified number of rows per strip; by default the number of rows/strip is selected so that each strip is approximately 8 kilobytes.
- R** Specify the percentage of the red channel to use (default 28).
- G** Specify the percentage of the green channel to use (default 59).
- B** Specify the percentage of the blue channel to use (default 11).

**SEE ALSO**

*pal2rgb(1), tiffinfo(1), tiffcp(1), tiffmedian(1), libtiff(3)*

**NAME**

tiff2ps – convert a TIFF image to POSTSCRIPT™

**SYNOPSIS**

**tiff2ps** [ *options* ] *input.tif* ...

**DESCRIPTION**

*tiff2ps* reads TIFF images and writes POSTSCRIPT or Encapsulated POSTSCRIPT (EPS) on the standard output. By default, *tiff2ps* writes Encapsulated POSTSCRIPT for the first image in the specified TIFF image file.

By default, *tiff2ps* will generate POSTSCRIPT that fills a printed area specified by the TIFF tags in the input file. If the file does not contain *XResolution* or *YResolution* tags, then the printed area is set according to the image dimensions. The **-w** and **-h** options (see below) can be used to set the dimensions of the printed area in inches; overriding any relevant TIFF tags.

The POSTSCRIPT generated for RGB, palette, and CMYK images uses the *colorimage* operator. The POSTSCRIPT generated for greyscale and bilevel images uses the *image* operator. When the *colorimage* operator is used, POSTSCRIPT code to emulate this operator on older POSTSCRIPT printers is also generated. Note that this emulation code can be very slow.

Color images with associated alpha data are composited over a white background.

**OPTIONS**

- 1** Generate POSTSCRIPT Level I (the default).
- 2** Generate POSTSCRIPT Level II.
- a** Generate output for all IFDs (pages) in the input file.
- d** Set the initial TIFF directory to the specified directory number. (NB: directories are numbered starting at zero.) This option is useful for selecting individual pages in a multi-page (e.g. facsimile) file.
- e** Force the generation of Encapsulated POSTSCRIPT.
- h** Specify the vertical size of the printed area (in inches).
- i** Enable/disable pixel interpolation. This option requires a single numeric value: zero to disable pixel interpolation and non-zero to enable. The default is enabled.
- m** Where possible render using the **imagemask** POSTSCRIPT operator instead of the image operator. When this option is specified *tiff2ps* will use **imagemask** for rendering 1 bit deep images. If this option is not specified or if the image depth is greater than 1 then the image operator is used.
- o** Set the initial TIFF directory to the IFD at the specified file offset. This option is useful for selecting thumbnail images and the like which are hidden using the SubIFD tag.
- p** Force the generation of (non-Encapsulated) POSTSCRIPT.
- s** Generate output for a single IFD (page) in the input file.
- w** Specify the horizontal size of the printed area (in inches).
- z** When generating POSTSCRIPT Level II, data is scaled so that it does not image into the *dead-zone* on a page (the outer margin that the printing device is unable to mark). This option suppresses this behaviour. When POSTSCRIPT Level I is generated, data is imaged to the entire printed page and this option has no affect.

**EXAMPLES**

The following generates POSTSCRIPT Level II for all pages of a facsimile:

```
tiff2ps -a2 fax.tif | lpr
```

Note also that if you have version 2.6.1 or newer of Ghostscript then you can efficiently preview facsimile generated with the above command.

To generate Encapsulated POSTSCRIPT for a the image at directory 2 of an image use:

```
tiff2ps -d 1 foo.tif
```

(notice that directories are numbered starting at zero.)

**BUGS**

Because POSTSCRIPT does not support the notion of a colormap, 8-bit palette images produce 24-bit POSTSCRIPT images. This conversion results in output that is six times bigger than the original image and which takes a long time to send to a printer over a serial line. Matters are even worse for 4-, 2-, and 1-bit palette images.

**BUGS**

Does not handle tiled images when generating PS Level I output.

**SEE ALSO**

*pal2rgb(1), tiffinfo(1), tiffcp(1), tiffgt(1), tiffmedian(1), tiff2bw(1), tiffsv(1), libtiff(3)*

**NAME**

tiff2rgba – convert a TIFF image to RGBA color space

**SYNOPSIS**

**tiff2rgba** [ options ] *input.tif output.tif*

**DESCRIPTION**

*Tiff2rgba* converts a wide variety of TIFF images into an RGBA TIFF image. This includes the ability to translate different color spaces and photometric interpretation into RGBA, support for alpha blending, and translation of many different bit depths into a 32bit RGBA image.

Internally this program is implemented using the *TIFFReadRGBAImage()* function, and it suffers any limitations of that image. This includes limited support for > 8 BitsPerSample images, and flaws with some esoteric combinations of BitsPerSample, photometric interpretation, block organization and planar configuration.

The generated images are stripped images with four samples per pixel (red, green, blue and alpha) or if the -n flag is used, three samples per pixel (red, green, and blue). The resulting images are always planar configuration contiguous. For this reason, this program is a useful utility for transform exotic TIFF files into a form ingestable by almost any TIFF supporting software.

**OPTIONS**

- c Specify a compression scheme to use when writing image data: **-c none** for no compression (the default), **-c packbits** for the PackBits compression algorithm, **-c zip** for the Deflate compression algorithm, **-c jpeg** for the JPEG compression algorithm, and **-c lzw** for Lempel-Ziv & Welch.
- r Write data with a specified number of rows per strip; by default the number of rows/strip is selected so that each strip is approximately 8 kilobytes.
- b Process the image one block (strip/tile) at a time instead of by reading the whole image into memory at once. This may be necessary for very large images on systems with limited RAM.
- n Drop the alpha component from the output file, producing a pure RGB file. Currently this does not work if the -b flag is also in effect.

**SEE ALSO**

*tiff2bw(1)*, *TIFFReadRGBAImage(3t)*, *libtiff(3)*

**NAME**

tiffcmp – compare two TIFF files

**SYNOPSIS**

**tiffcmp** [ *options* ] *file1.tif file2.tif*

**DESCRIPTION**

*Tiffcmp* compares the tags and data in two files created according to the Tagged Image File Format, Revision 6.0. The schemes used for compressing data in each file are immaterial when data are compared—data are compared on a scanline-by-scanline basis after decompression. Most directory tags are checked; notable exceptions are: *GrayResponseCurve*, *ColorResponseCurve*, and *ColorMap* tags. Data will not be compared if any of the *BitsPerSample*, *SamplesPerPixel*, or *ImageWidth* values are not equal. By default, *tiffcmp* will terminate if it encounters any difference.

**OPTIONS**

- l List each byte of image data that differs between the files.
- t Ignore any differences in directory tags.

**BUGS**

Tags that are not recognized by the library are not compared; they may also generate spurious diagnostics.

The image data of tiled files is not compared, since the `TIFFReadScanline()` function is used. A error will be reported for tiled files.

The pixel and/or sample number reported in differences may be off in some exotic cases.

**SEE ALSO**

*pal2rgb(1)*, *tiffinfo(1)*, *tiffcp(1)*, *tiffmedian(1)*, *libtiff(3)*

**NAME**

tiffcp – copy (and possibly convert) a TIFF file

**SYNOPSIS**

**tiffcp** [ *options* ] *src1.tif ... srcN.tif dst.tif*

**DESCRIPTION**

*tiffcp* combines one or more files created according to the Tag Image File Format, Revision 6.0 into a single TIFF file. Because the output file may be compressed using a different algorithm than the input files, *tiffcp* is most often used to convert between different compression schemes.

By default, *tiffcp* will copy all the understood tags in a TIFF directory of an input file to the associated directory in the output file.

*tiffcp* can be used to reorganize the storage characteristics of data in a file, but it is explicitly intended to not alter or convert the image data content in any way.

**OPTIONS****-b image**

subtract the following monochrome image from all others processed. This can be used to remove a noise bias from a set of images. This bias image is typically an image of noise the camera saw with its shutter closed.

**-B** Force output to be written with Big-Endian byte order. This option only has an effect when the output file is created or overwritten and not when it is appended to.

**-C** Suppress the use of “strip chopping” when reading images that have a single strip/tile of uncompressed data.

**-c** Specify the compression to use for data written to the output file: **none** for no compression, **packbits** for PackBits compression, **lzw** for Lempel-Ziv & Welch compression, **jpeg** for baseline JPEG compression, **zip** for Deflate compression, **g3** for CCITT Group 3 (T.4) compression, and **g4** for CCITT Group 4 (T.6) compression. By default *tiffcp* will compress data according to the value of the *Compression* tag found in the source file.

The CCITT Group 3 and Group 4 compression algorithms can only be used with bilevel data.

Group 3 compression can be specified together with several T.4-specific options: **1d** for 1-dimensional encoding, **2d** for 2-dimensional encoding, and **fill** to force each encoded scanline to be zero-filled so that the terminating EOL code lies on a byte boundary. Group 3-specific options are specified by appending a “:”-separated list to the “g3” option; e.g. **-c g3:2d:fill** to get 2D-encoded data with byte-aligned EOL codes.

LZW compression can be specified together with a *predictor* value. A predictor value of 2 causes each scanline of the output image to undergo horizontal differencing before it is encoded; a value of 1 forces each scanline to be encoded without differencing. LZW-specific options are specified by appending a “:”-separated list to the “lzw” option; e.g. **-c lzw:2** for LZW compression with horizontal differencing.

**-f** Specify the bit fill order to use in writing output data. By default, *tiffcp* will create a new file with the same fill order as the original. Specifying **-f lsb2msb** will force data to be written with the FillOrder tag set to LSB2MSB, while **-f msb2lsb** will force data to be written with the FillOrder tag set to MSB2LSB.

**-I** Specify the length of a tile (in pixels). *tiffcp* attempts to set the tile dimensions so that no more than 8 kilobytes of data appear in a tile.

**-L** Force output to be written with Little-Endian byte order. This option only has an effect when the output file is created or overwritten and not when it is appended to.

**-M** Suppress the use of memory-mapped files when reading images.

**-p** Specify the planar configuration to use in writing image data that has one 8-bit sample per pixel. By default, *tiffcp* will create a new file with the same planar configuration as the original. Specifying **-p contig** will force data to be written with multi-sample data packed together, while **-p separate** will force samples to be written in separate planes.

- r** Specify the number of rows (scanlines) in each strip of data written to the output file. By default, *tiffcp* attempts to set the rows/strip that no more than 8 kilobytes of data appear in a strip.
- s** Force the output file to be written with data organized in strips (rather than tiles).
- t** Force the output file to be written with data organized in tiles (rather than strips). options can be used to force the resultant image to be written as strips or tiles of data, respectively.
- w** Specify the width of a tile (in pixels). *tiffcp* attempts to set the tile dimensions so that no more than 8 kilobytes of data appear in a tile. *tiffcp* attempts to set the tile dimensions so that no more than 8 kilobytes of data appear in a tile.
- ,={character}**  
substitute {character} for ',' in parsing image directory indices in files. This is necessary if filenames contain commas. Note that ',=' with whitespace immediately following will disable the special meaning of the ',' entirely. See examples.

## EXAMPLES

The following concatenates two files and writes the result using LZW encoding:

```
tiffcp -c lzw a.tif b.tif result.tif
```

To convert a G3 1d-encoded TIFF to a single strip of G4-encoded data the following might be used:

```
tiffcp -c g4 -r 10000 g3.tif g4.tif
```

(1000 is just a number that is larger than the number of rows in the source file.)

To extract a selected set of images from a multi-image TIFF file, the file name may be immediately followed by a ',' separated list of image directory indices. The first image is always in directory 0. Thus, to copy the 1st and 3rd images of image file "album.tif" to "result.tif":

```
tiffcp album.tif,0,2 result.tif
```

Given file "CCD.tif" whose first image is a noise bias followed by images which include that bias, subtract the noise from all those images following it (while decompressing) with the command:

```
tiffcp -c none -b CCD.tif CCD.tif,1, result.tif
```

If the file above were named "CCD,X.tif", the "-,=" option would be required to correctly parse this filename with image numbers, as follows:

```
tiffcp -c none -,=% -b CCD,X.tif CCD,X%1%.tif result.tif
```

## SEE ALSO

*pal2rgb(1)*, *tiffinfo(1)*, *tiffcmp(1)*, *tiffmedian(1)*, *tiffsplit(1)*, *libtiff(3)*

**NAME**

tiffdither – convert a greyscale image to bilevel using dithering

**SYNOPSIS**

**tiffdither** [ *options* ] *input.tif* *output.tif*

**DESCRIPTION**

*tiffdither* converts a single channel 8-bit greyscale image to a bilevel image using Floyd-Steinberg error propagation with thresholding.

**OPTIONS**

**-c** Specify the compression to use for data written to the output file: **none** for no compression, **packbits** for PackBits compression, **lzw** for Lempel-Ziv & Welch compression, **zip** for Deflate compression, **g3** for CCITT Group 3 (T.4) compression, and **g4** for CCITT Group 4 (T.6) compression. By default *tiffdither* will compress data according to the value of the *Compression* tag found in the source file.

The CCITT Group 3 and Group 4 compression algorithms can only be used with bilevel data.

Group 3 compression can be specified together with several T.4-specific options: **1d** for 1-dimensional encoding, **2d** for 2-dimensional encoding, and **fill** to force each encoded scanline to be zero-filled so that the terminating EOL code lies on a byte boundary. Group 3-specific options are specified by appending a “:”-separated list to the “g3” option; e.g. **-c g3:2d:fill** to get 2D-encoded data with byte-aligned EOL codes.

LZW compression can be specified together with a *predictor* value. A predictor value of 2 causes each scanline of the output image to undergo horizontal differencing before it is encoded; a value of 1 forces each scanline to be encoded without differencing. LZW-specific options are specified by appending a “:”-separated list to the “lzw” option; e.g. **-c lzw:2** for LZW compression with horizontal differencing.

**-f** Specify the bit fill order to use in writing output data. By default, *tiffdither* will create a new file with the same fill order as the original. Specifying **-f lsb2msb** will force data to be written with the FillOrder tag set to LSB2MSB, while **-f msb2lsb** will force data to be written with the FillOrder tag set to MSB2LSB.

**-t** Set the threshold value for dithering. By default the threshold value is 128.

**NOTES**

The dither algorithm is taken from the *tiffmedian*(1) program (written by Paul Heckbert).

**SEE ALSO**

*pal2rgb*(1), *fax2tiff*(1), *tiffinfo*(1), *tiffcp*(1), *tiff2bw*(1), *libtiff*(3)

**NAME**

`tiffdump` – print verbatim information about TIFF files

**SYNOPSIS**

**tiffdump** [ *options* ] *name* ...

**DESCRIPTION**

*tiffdump* displays directory information from files created according to the Tag Image File Format, Revision 6.0. The header of each TIFF file (magic number, version, and first directory offset) is displayed, followed by the tag contents of each directory in the file. For each tag, the name, datatype, count, and value(s) is displayed. When the symbolic name for a tag or datatype is known, the symbolic name is displayed followed by its numeric (decimal) value. Tag values are displayed enclosed in “<>” characters immediately preceded by the value of the count field. For example, an *ImageWidth* tag might be displayed as “ImageWidth (256) SHORT (3) 1<800>”.

*tiffdump* is particularly useful for investigating the contents of TIFF files that *libtiff* does not understand.

**OPTIONS**

- h** Force numeric data to be printed in hexadecimal rather than the default decimal.
- o** Dump the contents of the IFD at the a particular file offset. The file offset may be specified using the usual C-style syntax; i.e. a leading “0x” for hexadecimal and a leading “0” for octal.

**SEE ALSO**

*tiffinfo*(1), *libtiff*(3)

**NAME**

tiffgt – display an image stored in a TIFF file (Silicon Graphics version)

**SYNOPSIS**

**tiffgt** [ *options* ] *input.tif* ...

**DESCRIPTION**

*tiffgt* displays one or more images stored using the Tag Image File Format, Revision 6.0. Each image is placed in a fixed size window that the user must position on the display (unless configured otherwise through X defaults). If the display has fewer than 24 bitplanes, or if the image does not warrant full color, then RGB color values are mapped to the closest values that exist in the colormap (this is done using the *rgbi* routine found in the graphics utility library **-lgutil**.)

*tiffgt* correctly handles files with any of the following characteristics:

BitsPerSample	1, 2, 4, 8, 16
SamplesPerPixel	1, 3, 4 (the 4th sample is ignored)
PhotometricInterpretation	0 (min-is-white), 1 (min-is-black), 2 (RGB), 3 (palette), 6 (YCbCr)
PlanarConfiguration	1 (contiguous), 2 (separate)
Orientation	1 (top-left), 4 (bottom-left)

Data may be organized as strips or tiles and may be compressed with any of the compression algorithms supported by the *libtiff*(3) library.

For palette images (*PhotometricInterpretation*=3), *tiffgt* inspects the colormap values and assumes either 16-bit or 8-bit values according to the maximum value. That is, if no colormap entry greater than 255 is found, *tiffgt* assumes the colormap has only 8-bit values; otherwise it assumes 16-bit values. This inspection is done to handle old images written by previous (incorrect) versions of *libtiff*.

*tiffgt* can be used to display multiple images one-at-a-time. The left mouse button switches the display to the first image in the *next* file in the list of files specified on the command line. The right mouse button switches to the first image in the *previous* file in the list. The middle mouse button causes the first image in the first file specified on the command line to be displayed. In addition the following keyboard commands are recognized:

- b** Use a *PhotometricInterpretation* of MinIsBlack in displaying the current image.
- l** Use a *FillOrder* of lsb-to-msb in decoding the current image.
- m** Use a *FillOrder* of msb-to-lsb in decoding the current image.
- c** Use a colormap visual to display the current image.
- r** Use a true color (24-bit RGB) visual to display the current image.
- w** Use a *PhotometricInterpretation* of MinIsWhite in displaying the current image.
- W** Toggle (enable/disable) display of warning messages from the TIFF library when decoding images.
- E** Toggle (enable/disable) display of error messages from the TIFF library when decoding images.
- z** Reset all parameters to their default settings (*FillOrder*, *PhotometricInterpretation*, handling of warnings and errors).

**PageUp**

Display the previous image in the current file or the last image in the previous file.

**PageDown**

Display the next image in the current file or the first image in the next file.

**Home** Display the first image in the current file.

**End** Display the last image in the current file (unimplemented).

**OPTIONS**

- c** Force image display in a colormap window.
- d** Specify an image to display by directory number. By default the first image in the file is displayed. Directories are numbered starting at zero.

- e Enable reporting of error messages from the TIFF library. By default *tiffgt* silently ignores images that cannot be read.
- f Force *tiffgt* to run as a foreground process. By default *tiffgt* will place itself in the background once it has opened the requested image file.
- l Force the presumed bit ordering to be LSB to MSB.
- m Force the presumed bit ordering to be MSB to LSB.
- o Specify an image to display by directory offset. By default the first image in the file is displayed. Directory offsets may be specified using C-style syntax; i.e. a leading “0x” for hexadecimal and a leading “0” for octal.
- p Override the value of the *PhotometricInterpretation* tag; the parameter may be one of: *miniswhite*, *minisblack*, *rgb*, *palette*, *mask*, *separated*, *ycbcr*, and *cielab*.
- r Force image display in a full color window.
- s Stop on the first read error. By default all errors in the input data are ignored and *tiffgt* does it’s best to display as much of an image as possible.
- w Enable reporting of warning messages from the TIFF library. By default *tiffgt* ignores warning messages generated when reading an image.
- v Place information in the title bar describing what type of window (full color or colormap) is being used, the name of the input file, and the directory index of the image (if non-zero). By default, the window type is not shown in the title bar.

## BUGS

Images wider and taller than the display are silently truncated to avoid crashing old versions of the window manager.

## SEE ALSO

*tiffdump*(1), *tiffinfo*(1), *tiffcp*(1), *libtiff*(3)

**NAME**

tiffinfo – print information about TIFF files

**SYNOPSIS**

**tiffinfo** [ *options* ] *input.tif* ...

**DESCRIPTION**

*Tiffinfo* displays information about files created according to the Tag Image File Format, Revision 6.0. By default, the contents of each TIFF directory in each file is displayed, with the value of each tag shown symbolically (where sensible).

**OPTIONS**

- c     Display the colormap and color/gray response curves, if present.
- D     In addition to displaying the directory tags, read and decompress all the data in each image (but not display it).
- d     In addition to displaying the directory tags, print each byte of decompressed data in hexadecimal.
- j     Display any JPEG-related tags that are present.
- o     Set the initial TIFF directory according to the specified file offset. The file offset may be specified using the usual C-style syntax; i.e. a leading “0x” for hexadecimal and a leading “0” for octal.
- s     Display the offsets and byte counts for each data strip in a directory.
- z     Enable strip chopping when reading image data.
- #     Set the initial TIFF directory to #.

**SEE ALSO**

*pal2rgb(1)*, *tiffcp(1)*, *tiffcmp(1)*, *tiffmedian(1)*, *libtiff(3)*

**NAME**

`tiffmedian` – apply the median cut algorithm to data in a TIFF file

**SYNOPSIS**

`tiffmedian` [ *options* ] *input.tif* *output.tif*

**DESCRIPTION**

*tiffmedian* applies the median cut algorithm to an RGB image in *input.tif* to generate a palette image that is written to *output.tif*. The generated colormap has, by default, 256 entries. The image data is quantized by mapping each pixel to the closest color values in the colormap.

**OPTIONS**

- c Specify the compression to use for data written to the output file: **none** for no compression, **packbits** for PackBits compression, **lzw** for Lempel-Ziv & Welch compression, and **zip** for Deflate compression. By default *tiffmedian* will compress data according to the value of the *Compression* tag found in the source file.  
  
LZW compression can be specified together with a *predictor* value. A predictor value of 2 causes each scanline of the output image to undergo horizontal differencing before it is encoded; a value of 1 forces each scanline to be encoded without differencing. LZW-specific options are specified by appending a “:”-separated list to the “lzw” option; e.g. –c **lzw:2** for LZW compression with horizontal differencing.
- C Specify the number of entries to use in the generated colormap. By default all 256 entries/colors are used.
- f Apply Floyd-Steinberg dithering before selecting a colormap entry.
- r Specify the number of rows (scanlines) in each strip of data written to the output file. By default, *tiffmedian* attempts to set the rows/strip that no more than 8 kilobytes of data appear in a strip.

**NOTES**

This program is derived from Paul Heckbert’s *median* program.

**SEE ALSO**

*pal2rgb*(1), *tiffinfo*(1), *tiffcp*(1), *tiffcmp*(1), *libtiff*(3)

"Color Image Quantization for Frame Buffer Display", Paul Heckbert, SIGGRAPH proceedings, 1982, pp. 297-307.

**NAME**

`tiffsplit` – split a multi-image TIFF into single-image TIFF files

**SYNOPSIS**

`tiffsplit` *src.tif* [ *prefix* ]

**DESCRIPTION**

*tiffsplit* takes a multi-directory (page) TIFF file and creates one or more single-directory (page) TIFF files from it. The output files are given names created by concatenating a prefix, a lexically ordered suffix in the range [aa-zz], the suffix *.tif* (e.g. *xaa.tif*, *xab.tif*, *xzz.tif*). If a prefix is not specified on the command line, the default prefix of *x* is used.

**OPTIONS**

None.

**BUGS**

Only a select set of “known tags” is copied when splitting.

**SEE ALSO**

*tiffcp*(1), *tiffinfo*(1), *libtiff*(3)

**NAME**

tiffsv – save an image from the framebuffer in a TIFF file (Silicon Graphics version)

**SYNOPSIS**

**tiffsv** [ *options* ] *output.tif* [ *x1 x2 y1 y2* ]

**DESCRIPTION**

*tiffsv* saves all or part of the framebuffer in a file using the Tag Image File Format, Revision 6.0. By default, the image is saved with data samples packed (*PlanarConfiguration=1*), compressed with the Lempel-Ziv & Welch algorithm (*Compression=5*), and with each strip no more than 8 kilobytes. These characteristics can be overridden, or explicitly specified with the options described below.

**OPTIONS**

- b** Save the image as a greyscale image as if it were processed by *tiff2bw*(1). This option is included for compatibility with the standard *scrsave*(6D) program.
- c** Specify the compression to use for data written to the output file: **none** for no compression, **packbits** for PackBits compression, **jpeg** for baseline JPEG compression, **zip** for Deflate compression, and **lzw** for Lempel-Ziv & Welch compression (default).  
LZW compression can be specified together with a *predictor* value. A predictor value of 2 causes each scanline of the output image to undergo horizontal differencing before it is encoded; a value of 1 forces each scanline to be encoded without differencing. LZW-specific options are specified by appending a “:”-separated list to the “lzw” option; e.g. **-c lzw:2** for LZW compression with horizontal differencing.
- p** Specify the planar configuration to use in writing image data. By default, *tiffsv* will create a new file with the data samples packed contiguously. Specifying **-p contig** will force data to be written with multi-sample data packed together, while **-p separate** will force samples to be written in separate planes.
- r** Specify the number of rows (scanlines) in each strip of data written to the output file. By default, *tiffsv* attempts to set the rows/strip that no more than 8 kilobytes of data appear in a strip.

**NOTE**

Except for the use of TIFF, this program is equivalent to the standard *scrsave* program. This means, for example, that you can use it in conjunction with the standard *icut* program simply by creating a link called *scrsave*, or by creating a shell script called *scrsave* that invokes *tiffgt* with the appropriate options.

**BUGS**

If data are saved compressed and in separate planes, then the rows in each strip is silently set to one to avoid limitations in the *libtiff*(3) library.

**SEE ALSO**

*scrsave*(6D) *pal2rgb*(1), *tiffdump*(1), *tiffgt*(1), *tiffinfo*(1), *tiffcp*(1), *tiffmedian*(1), *libtiff*(3)

**NAME**

libtiff – introduction to *libtiff*, a library for reading and writing TIFF files

**SYNOPSIS**

```
#include <tiffio.h>
cc file.c -ltiff
```

**DESCRIPTION**

*libtiff* is a library for reading and writing data files encoded with the *Tag Image File* format, Revision 6.0 (or revision 5.0 or revision 4.0). This file format is suitable for archiving multi-color and monochromatic image data.

The library supports several compression algorithms, as indicated by the *Compression* field, including: no compression (1), CCITT 1D Huffman compression (2), CCITT Group 3 Facsimile compression (3), CCITT Group 4 Facsimile compression (4), Lempel-Ziv & Welch compression (5), baseline JPEG compression (7), word-aligned 1D Huffman compression (32771), and PackBits compression (32773). In addition, several nonstandard compression algorithms are supported: the 4-bit compression algorithm used by the *ThunderScan* program (32809) (decompression only), NeXT's 2-bit compression algorithm (32766) (decompression only), an experimental LZ-style algorithm known as Deflate (32946), and an experimental CIE LogLuv compression scheme designed for images with high dynamic range (32845 for LogL and 32845 for LogLuv). Directory information may be in either little- or big-endian byte order—byte swapping is automatically done by the library. Data bit ordering may be either Most Significant Bit (MSB) to Least Significant Bit (LSB) or LSB to MSB. Finally, the library does not support files in which the *BitsPerSample*, *Compression*, *MinSampleValue*, or *MaxSampleValue* fields are defined differently on a per-sample basis (in Rev. 6.0 the *Compression* tag is not defined on a per-sample basis, so this is immaterial).

**DATA TYPES**

The library makes extensive use of C typedefs to promote portability. Two sets of typedefs are used, one for communication with clients of the library and one for internal data structures and parsing of the TIFF format. The following typedefs are exposed to users either through function definitions or through parameters passed through the varargs interfaces.

```
typedef unsigned short uint16;    16-bit unsigned integer
typedef unsigned <thing> uint32; 32-bit unsigned integer

typedef unsigned int ttag_t;      directory tag
typedef uint16 tdir_t;           directory index
typedef uint16 tsample_t;        sample number
typedef uint32 tstrip_t;         strip number
typedef uint32 ttile_t;          tile number
typedef int32 tsize_t;           i/o size in bytes
typedef void* tdata_t;           image data ref
typedef void* thandle_t;         client data handle
typedef int32 toff_t;            file offset
```

Note that *tstrip\_t*, *ttile\_t*, and *tsize\_t* are constrained to be no more than 32-bit quantities by 32-bit fields they are stored in in the TIFF image. Likewise *tsample\_t* is limited by the 16-bit field used to store the *SamplesPerPixel* tag. *tdir\_t* constrains the maximum number of IFDs that may appear in an image and may be an arbitrary size (w/o penalty). *ttag\_t* must be either int, unsigned int, pointer, or double because the library uses a varargs interface and ANSI C restricts the type of the parameter before an ellipsis to be a promoted type. *toff\_t* is defined as int32 because TIFF file offsets are (unsigned) 32-bit quantities. A signed value is used because some interfaces return -1 on error. Finally, note that user-specified data references are passed as opaque handles and only cast at the lowest layers where their type is presumed.

**LIST OF ROUTINES**

The following routines are part of the library. Consult specific manual pages for details on their operation. The manual page names listed below are for systems where the full function names can not be encoded in the filesystem; on most systems doing “man function-name” will work.

<i>Name</i>	<i>Appears on Page</i>	<i>Description</i>
TIFFCheckTile	tile.3t	very x,y,z,sample is within image

TIFFClientOpen	open.3t	open a file for reading or writing
TIFFClose	close.3t	close an open file
TIFFComputeStrip	strip.3t	return strip containing y,sample
TIFFComputeTile	tile.3t	return tile containing x,y,z,sample
TIFFCurrentDirectory	query.3t	return index of current directory
TIFFCurrentRow	query.3t	return index of current scanline
TIFFCurrentStrip	query.3t	return index of current strip
TIFFCurrentTile	query.3t	return index of current tile
TIFFError	error.3t	library error handler
TIFFFdOpen	open.3t	open a file for reading or writing
TIFFFileName	query.3t	return name of open file
TIFFFileno	query.3t	return open file descriptor
TIFFFlush	flush.3t	flush all pending writes
TIFFFlushData	flush.3t	flush pending data writes
TIFFGetBitRevTable	swab.3t	return bit reversal table
TIFFGetField	getfield.3t	return tag value in current directory
TIFFGetFieldDefaulted	getfield.3t	return tag value in current directory
TIFFGetMode	query.3t	return open file mode
TIFFGetVersion	query.3t	return library version string
TIFFIsTiled	query.3t	return true if image data is tiled
TIFFIsByteSwapped	query.3t	return true if image data is byte-swapped
TIFFNumberOfStrips	strip.3t	return number of strips in an image
TIFFNumberOfTiles	tile.3t	return number of tiles in an image
TIFFOpen	open.3t	open a file for reading or writing
TIFFPrintDirectory	print.3t	print description of the current directory
TIFFReadBufferSetup	rdbuf.3t	specify i/o buffer for reading
TIFFReadDirectory	readdir.3t	read the next directory
TIFFReadEncodedStrip	rdstrip.3t	read and decode a strip of data
TIFFReadEncodedTile	rdtile.3t	read and decode a tile of data
TIFFReadRawStrip	rdrstrip.3t	read a raw strip of data
TIFFReadRawTile	rdrtile.3t	read a raw tile of data
TIFFReadRGBAImage	rdimage.3t	read an image into a fixed format raster
TIFFReadScanline	readline.3t	read and decode a row of data
TIFFReadTile	readtile.3t	read and decode a tile of data
TIFFReverseBits	swab.3t	reverse bits in an array of bytes
TIFFRGBAImageBegin	rgbaimage.3t	setup decoder state for TIFFRGBAImageGet
TIFFRGBAImageEnd	rgbaimage.3t	release TIFFRGBAImage decoder state
TIFFRGBAImageGet	rgbaimage.3t	read and decode an image
TIFFRGBAImageOK	rgbaimage.3t	is image readable by TIFFRGBAImageGet
TIFFScanlineSize	size.3t	return size of a scanline
TIFFSetDirectory	setdir.3t	set the current directory
TIFFSetSubDirectory	setdir.3t	set the current directory
TIFFSetErrorHandler	error.3t	set error handler function
TIFFSetField	setfield.3t	set a tag's value in the current directory
TIFFSetWarningHandler	error.3t	set warning handler function
TIFFStripSize	size.3t	return size of a strip
TIFFSwabShort	swab.3t	swap bytes of short
TIFFSwabLong	swab.3t	swap bytes of long
TIFFSwabArrayOfShort	swab.3t	swap bytes of an array of shorts
TIFFSwabArrayOfLong	swab.3t	swap bytes of an array of longs
TIFFTileRowSize	size.3t	return size of a row in a tile
TIFFTileSize	size.3t	return size of a tile
TIFFVGetField	getfield.3t	return tag value in current directory
TIFFVGetFieldDefaulted	getfield.3t	return tag value in current directory
TIFFVSetField	setfield.3t	set a tag's value in the current directory
TIFFWarning	warning.3t	library warning handler
TIFFWriteDirectory	writedir.3t	write the current directory
TIFFWriteEncodedStrip	wrestrip.3t	compress and write a strip of data
TIFFWriteEncodedTile	wretile.3t	compress and write a tile of data

TIFFWriteRawStrip	wrrstrip.3t	write a raw strip of data
TIFFWriteRawTile	wrrtile.3t	write a raw tile of data
TIFFWriteScanline	writeline.3t	write a scanline of data
TIFFWriteTile	wrrtile.3t	compress and write a tile of data

## TAG USAGE

The table below lists the TIFF tags that are recognized and handled by the library. If no use is indicated in the table, then the library reads and writes the tag, but does not use it internally. Note that some tags are meaningful only when a particular compression scheme is being used; e.g. *Group3Options* is only useful if *Compression* is set to CCITT Group 3 encoding. Tags of this sort are considered *codec-specific* tags and the library does not recognize them except when the *Compression* tag has been previously set to the relevant compression scheme.

<i>Tag Name</i>	<i>Value</i>	<i>R/W</i>	<i>Library Use/Notes</i>
Artist	315	R/W	
BadFaxLines	326	R/W	
BitsPerSample	258	R/W	lots
CellLength	265		parsed but ignored
CellWidth	264		parsed but ignored
CleanFaxData	327	R/W	
ColorMap	320	R/W	
ColorResponseUnit	300		parsed but ignored
Compression	259	R/W	choosing codec
ConsecutiveBadFaxLines	328	R/W	
DataType	32996	R	obsoleted by SampleFormat tag
DateTime	306	R/W	
DocumentName	269	R/W	
DotRange	336	R/W	
ExtraSamples	338	R/W	lots
FaxRecvParams	34908	R/W	
FaxSubAddress	34909	R/W	
FaxRecvTime	34910	R/W	
FillOrder	266	R/W	control bit order
FreeByteCounts	289		parsed but ignored
FreeOffsets	288		parsed but ignored
GrayResponseCurve	291		parsed but ignored
GrayResponseUnit	290		parsed but ignored
Group3Options	292	R/W	used by Group 3 codec
Group4Options	293	R/W	
HostComputer	316	R/W	
ImageDepth	32997	R/W	tile/strip calculations
ImageDescription	270	R/W	
ImageLength	257	R/W	lots
ImageWidth	256	R/W	lots
InkNames	333	R/W	
InkSet	332	R/W	
JPEGTables	347	R/W	used by JPEG codec
Make	271	R/W	
Matteing	32995	R	obsoleted by ExtraSamples tag
MaxSampleValue	281	R/W	
MinSampleValue	280	R/W	
Model	272	R/W	
NewSubFileType	254	R/W	called SubFileType in spec
NumberOfInks	334	R/W	
Orientation	274	R/W	
PageName	285	R/W	
PageNumber	297	R/W	
PhotometricInterpretation	262	R/W	used by Group 3 and JPEG codecs
PlanarConfiguration	284	R/W	data i/o

Predictor	317	R/W	used by LZW and Deflate codecs
PrimaryChromaticities	319	R/W	
ReferenceBlackWhite	532	R/W	
ResolutionUnit	296	R/W	used by Group 3 codec
RowsPerStrip	278	R/W	data i/o
SampleFormat	339	R/W	
SamplesPerPixel	277	R/W	lots
SMinSampleValue	340	R/W	
SMaxSampleValue	341	R/W	
Software	305	R/W	
StoNits	37439	R/W	
StripByteCounts	279	R/W	data i/o
StripOffsets	273	R/W	data i/o
SubFileType	255	R/W	called OSubFileType in spec
TargetPrinter	337	R/W	
Thresholding	263	R/W	
TileByteCounts	324	R/W	data i/o
TileDepth	32998	R/W	tile/strip calculations
TileLength	323	R/W	data i/o
TileOffsets	324	R/W	data i/o
TileWidth	322	R/W	data i/o
TransferFunction	301	R/W	
WhitePoint	318	R/W	
XPosition	286	R/W	
XResolution	282	R/W	
YCbCrCoefficients	529	R/W	used by TIFFRGBAImage support
YCbCrPositioning	531	R/W	tile/strip size calculations
YCbCrSubsampling	530	R/W	
YPosition	286	R/W	
YResolution	283	R/W	used by Group 3 codec

## PSEUDO TAGS

In addition to the normal TIFF tags the library supports a collection of tags whose values lie in a range outside the valid range of TIFF tags. These tags are termed *pseud-tags* and are used to control various codec-specific functions within the library. The table below summarizes the defined pseudo-tags.

<i>Tag Name</i>	<i>Codec</i>	<i>R/W</i>	<i>Library Use/Notes</i>
TIFFTAG_FAXMODE	G3	R/W	general codec operation
TIFFTAG_FAXFILLFUNC	G3/G4	R/W	bitmap fill function
TIFFTAG_JPEGQUALITY	JPEG	R/W	compression quality control
TIFFTAG_JPEGCOLORMODE	JPEG	R/W	control colorspace conversions
TIFFTAG_JPEGTABLESMODE	JPEG	R/W	control contents of <i>JPEGTables</i> tag
TIFFTAG_ZIPQUALITY	Deflate	R/W	compression quality level
TIFFTAG_PIXARLOGDATAFMT	PixarLog		R/Wuser data format
TIFFTAG_PIXARLOGQUALITY	PixarLog		R/Wcompression quality level
TIFFTAG_SGILOGDATAFMT	SGILog	R/W	user data format

### TIFFTAG\_FAXMODE

Control the operation of the Group 3 codec. Possible values (independent bits that can be combined by or'ing them together) are: FAXMODE\_CLASSIC (enable old-style format in which the RTC is written at the end of the last strip), FAXMODE\_NORTC (opposite of FAXMODE\_CLASSIC; also called FAXMODE\_CLASSF), FAXMODE\_NOEOL (do not write EOL codes at the start of each row of data), FAXMODE\_BYTEALIGN (align each encoded row to an 8-bit boundary), FAXMODE\_WORDALIGN (align each encoded row to an 16-bit boundary), The default value is dependent on the compression scheme; this pseudo-tag is used by the various G3 and G4 codecs to share code.

### TIFFTAG\_FAXFILLFUNC

Control the function used to convert arrays of black and white runs to packed bit arrays. This hook can be used to image decoded scanlines in multi-bit depth rasters (e.g. for display in colormap mode) or for other purposes. The default value is a pointer to a builtin function that

images packed bilevel data.

#### **TIFFTAG\_IPTCNEWSPHOTO**

Tag contains image metadata per the IPTC newsphoto spec: Headline, captioning, credit, etc... Used by most wire services.

#### **TIFFTAG\_PHOTOSHOP**

Tag contains Photoshop captioning information and metadata. Photoshop uses in parallel and redundantly alongside IPTCNEWSPHOTO information.

#### **TIFFTAG\_JPEGQUALITY**

Control the compression quality level used in the baseline algorithm. Note that quality levels are in the range 0-100 with a default value of 75.

#### **TIFFTAG\_JPEGCOLORMODE**

Control whether or not conversion is done between RGB and YCbCr colorspace. Possible values are: JPEGCOLORMODE\_RAW (do not convert), and JPEGCOLORMODE\_RGB (convert to/from RGB) The default value is JPEGCOLORMODE\_RAW.

#### **TIFFTAG\_JPEGTABLESMODE**

Control the information written in the *JPEGTables* tag. Possible values (independent bits that can be combined by or'ing them together) are: JPEGTABLESMODE\_QUANT (include quantization tables), and JPEGTABLESMODE\_HUFF (include Huffman encoding tables). The default value is JPEGTABLESMODE\_QUANT|JPEGTABLESMODE\_HUFF.

#### **TIFFTAG\_ZIPQUALITY**

Control the compression technique used by the Deflate codec. Quality levels are in the range 1-9 with larger numbers yielding better compression at the cost of more computation. The default quality level is 6 which yields a good time-space tradeoff.

#### **TIFFTAG\_PIXARLOGDATAFMT**

Control the format of user data passed *in* to the PixarLog codec when encoding and passed *out* from when decoding. Possible values are: PIXARLOGDATAFMT\_8BIT for 8-bit unsigned pixels, PIXARLOGDATAFMT\_8BITABGR for 8-bit unsigned ABGR-ordered pixels, PIXARLOGDATAFMT\_11BITLOG for 11-bit log-encoded raw data, PIXARLOGDATAFMT\_12BITPICIO for 12-bit PICIO-compatible data, PIXARLOGDATAFMT\_16BIT for 16-bit signed samples, and PIXARLOGDATAFMT\_FLOAT for 32-bit IEEE floating point samples.

#### **TIFFTAG\_PIXARLOGQUALITY**

Control the compression technique used by the PixarLog codec. This value is treated identically to TIFFTAG\_ZIPQUALITY; see the above description.

#### **TIFFTAG\_SGILOGDATAFMT**

Control the format of client data passed *in* to the SGILOG codec when encoding and passed *out* from when decoding. Possible values are: SGILOGDATAFMT\_FLTXYZ for converting between LogLuv and 32-bit IEEE floating valued XYZ pixels, SGILOGDATAFMT\_16BITLUV for 16-bit encoded Luv pixels, SGILOGDATAFMT\_32BITRAW and SGILOGDATAFMT\_24BITRAW for no conversion of data, SGILOGDATAFMT\_8BITRGB for returning 8-bit RGB data (valid only when decoding LogLuv-encoded data), SGILOGDATAFMT\_FLTY for converting between LogL and 32-bit IEEE floating valued Y pixels, SGILOGDATAFMT\_16BITL for 16-bit encoded L pixels, and SGILOGDATAFMT\_8BITGRY for returning 8-bit greyscale data (valid only when decoding LogL-encoded data).

### **DIAGNOSTICS**

All error messages are directed through the *TIFFError* routine. By default messages are directed to **stderr** in the form: *module: message\n*. Warning messages are likewise directed through the *TIFFWarning* routine.

### **SEE ALSO**

*fax2tiff(1)*, *gif2tiff(1)*, *pal2rgb(1)*, *ppm2tiff(1)*, *rgb2ycbcr(1)*, *ras2tiff(1)*, *sgi2tiff(1)*, *tiff2bw(1)*, *tiffdither(1)*, *tiffdump(1)*, *tiffcp(1)*, *tiffcmp(1)*, *tiffgt(1)*, *tiffinfo(1)*, *tiffmedian(1)*, *tiffsplit(1)*, *tiffsv(1)*,

*Tag Image File Format Specification — Revision 6.0*, an Aldus Technical Memorandum.

*The Spirit of TIFF Class F*, an appendix to the TIFF 5.0 specification prepared by Cygnet

Technologies.

**BUGS**

The library does not support multi-sample images where some samples have different bits/sample.

The library does not support random access to compressed data that is organized with more than one row per tile or strip. The library discards unknown tags. The library should do more validity checking of a directory's contents.

**NAME**

TIFFReadBufferSetup, TIFFWriteBufferSetup – I/O buffering control routines

**SYNOPSIS**

```
#include <tiffio.h>
int TIFFReadBufferSetup(TIFF*, tdata_t buffer, tsize_t size);
int TIFFWriteBufferSetup(TIFF*, tdata_t buffer, tsize_t size);
```

**DESCRIPTION**

The following routines are provided for client-control of the I/O buffers used by the library. Applications need never use these routines; they are provided only for “intelligent clients” that wish to optimize memory usage and/or eliminate potential copy operations that can occur when working with images that have data stored without compression.

*TIFFReadBufferSetup* sets up the data buffer used to read raw (encoded) data from a file. If the specified pointer is NULL (zero), then a buffer of the appropriate size is allocated. Otherwise the caller must guarantee that the buffer is large enough to hold any individual strip of raw data. *TIFFReadBufferSetup* returns a non-zero value if the setup was successful and zero otherwise.

*TIFFWriteBufferSetup* sets up the data buffer used to write raw (encoded) data to a file. If the specified *size* is -1 then the buffer size is selected to hold a complete tile or strip, or at least 8 kilobytes, whichever is greater. If the specified *buffer* is NULL (zero), then a buffer of the appropriate size is dynamically allocated. *TIFFWriteBufferSetup* returns a non-zero value if the setup was successful and zero otherwise.

**DIAGNOSTICS**

**%s: No space for data buffer at scanline %ld.** *TIFFReadBufferSetup* was unable to dynamically allocate space for a data buffer.

**%s: No space for output buffer.** *TIFFWriteBufferSetup* was unable to dynamically allocate space for a data buffer.

**SEE ALSO**

*libtiff*(3T)

**NAME**

TIFFClose – close a previously opened TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
void TIFFClose(TIFF* tif)
```

**DESCRIPTION**

*TIFFClose* closes a file that was previously opened with *TIFFOpen(3T)*. Any buffered data are flushed to the file, including the contents of the current directory (if modified); and all resources are reclaimed.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine. Likewise, warning messages are directed to the *TIFFWarning(3T)* routine.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*

**NAME**

TIFFFindCODEC, TIFFRegisterCODEC, TIFFUnRegisterCODEC – codec-related utility routines

**SYNOPSIS**

```
#include <tiffio.h>
const TIFFCodec* TIFFFindCODEC(uint16 scheme);
TIFFCodec* TIFFRegisterCODEC(uint16 scheme, const char* method, TIFFInitMethod init);
void TIFFUnRegisterCODEC(TIFFCodec* codec);
```

**DESCRIPTION**

*libtiff* supports a variety of compression schemes implemented by software *codecs*. Each codec adheres to a modular interface that provides for the decoding and encoding of image data; as well as some other methods for initialization, setup, cleanup, and the control of default strip and tile sizes. Codecs are identified by the associated value of the TIFF *Compression* tag; e.g. 5 for LZW compression.

The *TIFFRegisterCODEC* routine can be used to augment or override the set of codecs available to an application. If the specified *scheme* already has a registered codec then it is *overridden* and any images with data encoded with this compression scheme will be decoded using the supplied coded.

**DIAGNOSTICS**

**No space to register compression scheme %s.** *TIFFRegisterCODEC* was unable to allocate memory for the data structures needed to register a codec.

**Cannot remove compression scheme %s; not registered.** *TIFFUnRegisterCODEC* did not locate the specified codec in the table of registered compression schemes.

**SEE ALSO**

*libtiff*(3T)

**NAME**

TIFFError, TIFFSetErrorHandler – library error handling interface

**SYNOPSIS**

```
#include <tiffio.h>
void TIFFError(const char* module, const char* fmt, ...)

#include <stdarg.h>
typedef void (*TIFFErrorHandler)(const char* module, const char* fmt, va_list ap);
TIFFErrorHandler TIFFSetErrorHandler(TIFFErrorHandler handler);
```

**DESCRIPTION**

*TIFFError* invokes the library-wide error handling function to (normally) write an error message to the **stderr**. The *fmt* parameter is a *printf*(3S) format string, and any number arguments can be supplied. The *module* parameter, if non-zero, is printed before the message; it typically is used to identify the software module in which an error is detected.

Applications that desire to capture control in the event of an error should use *TIFFSetErrorHandler* to override the default error handler. A NULL (0) error handling function may be installed to suppress error messages.

**RETURN VALUES**

*TIFFSetErrorHandler* returns a reference to the previous error handling function.

**SEE ALSO**

*libtiff*(3T), *TIFFWarning*(3T), *printf*(3S)

**NAME**

TIFFFlush, TIFFFlushData – flush pending writes to an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
int TIFFFlush(TIFF* tif)
int TIFFFlushData(TIFF* tif)
```

**DESCRIPTION**

*TIFFFlush* causes any pending writes for the specified file (including writes for the current directory) to be done. In normal operation this call is never needed– the library automatically does any flushing required.

*TIFFFlushData* flushes any pending image data for the specified file to be written out; directory-related data are not flushed. In normal operation this call is never needed– the library automatically does any flushing required.

**RETURN VALUES**

0 is returned if an error is encountered, otherwise 1 is returned.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError*(3T) routine.

**SEE ALSO**

*libtiff*(3T), *TIFFOpen*(3T), *TIFFWriteEncodedStrip*(3T), *TIFFWriteEncodedTile*(3T), *TIFFWriteRawStrip*(3T), *TIFFWriteRawTile*(3T), *TIFFWriteScanline*(3T), *TIFFWriteTile*(3T)

**NAME**

TIFFGetField, TIFFVGetField – get the value(s) of a tag in an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
int TIFFGetField(TIFF* tif, ttag_t tag, ...)

#include <stdarg.h>
int TIFFVGetField(TIFF* tif, ttag_t tag, va_list ap)

int TIFFGetFieldDefaulted(TIFF* tif, ttag_t tag, ...)
int TIFFVGetFieldDefaulted(TIFF* tif, ttag_t tag, va_list ap)
```

**DESCRIPTION**

*TIFFGetField* returns the value of a tag or pseudo-tag associated with the the current directory of the open TIFF file *tif*. (A *pseudo-tag* is a parameter that is used to control the operation of the TIFF library but whose value is not read or written to the underlying file.) The file must have been previously opened with *TIFFOpen(3T)*. The tag is identified by *tag*, one of the values defined in the include file **tiff.h** (see also the table below). The type and number of values returned is dependent on the tag being requested. The programming interface uses a variable argument list as prescribed by the *stdarg(3)* interface. The returned values should only be interpreted if *TIFFGetField* returns 1.

*TIFFVGetField* is functionally equivalent to *TIFFGetField* except that it takes a pointer to a variable argument list. *TIFFVGetField* is useful for layering interfaces on top of the functionality provided by *TIFFGetField*.

*TIFFGetFieldDefaulted* and *TIFFVGetFieldDefaulted* are identical to *TIFFGetField* and *TIFFVGetField*, except that if a tag is not defined in the current directory and it has a default value, then the default value is returned.

The tags understood by *libtiff*, the number of parameter values, and the types for the returned values are shown below. The data types are specified as in C and correspond to the types used to specify tag values to *TIFFSetField(3T)*. Remember that *TIFFGetField* returns parameter values, so all the listed data types are pointers to storage where values should be returned. Consult the TIFF specification for information on the meaning of each tag and their possible values.

Tag Name	Count	Types	Notes
TIFFTAG_ARTIST	1	char**	
TIFFTAG_BADFAXLINES	1	uint32*	
TIFFTAG_BITSPERSAMPLE	1	uint16*	
TIFFTAG_CLEANFAXDATA	1	uint16*	
TIFFTAG_COLORMAP	3	uint16**	1<<BitsPerSample arrays
TIFFTAG_COMPRESSION	1	uint16*	
TIFFTAG_CONSECUTIVEBADFAXLINES	1	uint32*	
TIFFTAG_COPYRIGHT	1	char*	
TIFFTAG_DATATYPE	1	uint16*	
TIFFTAG_DATETIME	1	char**	
TIFFTAG_DOCUMENTNAME	1	char**	
TIFFTAG_DOTRANGE	2	uint16*	
TIFFTAG_EXTRASAMPLES	2	uint16*,uint16**	count & types array
TIFFTAG_FAXMODE	1	int*	G3/G4 compression pseudo-tag
TIFFTAG_FAXFILLFUNC	1	TIFFFaxFillFunc*	G3/G4 compression pseudo-tag
TIFFTAG_FILLORDER	1	uint16*	
TIFFTAG_GROUP3OPTIONS	1	uint32*	
TIFFTAG_GROUP4OPTIONS	1	uint32*	
TIFFTAG_HALFTONEHINTS	2	uint16*	
TIFFTAG_HOSTCOMPUTER	1	char**	
TIFFTAG_IMAGEDEPTH	1	uint32*	
TIFFTAG_IMAGEDESCRIPTION	1	char**	
TIFFTAG_IMAGELENGTH	1	uint32*	
TIFFTAG_IMAGEWIDTH	1	uint32*	
TIFFTAG_INKNAMES	1	char**	
TIFFTAG_INKSET	1	uint16*	

TIFFTAG_JPEGTABLES	2	u_short*,void**	count & tables
TIFFTAG_JPEGQUALITY	1	int*	JPEG pseudo-tag
TIFFTAG_JPEGCOLORMODE	1	int*	JPEG pseudo-tag
TIFFTAG_JPEGTABLESMODE	1	int*	JPEG pseudo-tag
TIFFTAG_MAKE	1	char**	
TIFFTAG_MATTEING	1	uint16*	
TIFFTAG_MAXSAMPLEVALUE	1	uint16*	
TIFFTAG_MINSAMPLEVALUE	1	uint16*	
TIFFTAG_MODEL	1	char**	
TIFFTAG_ORIENTATION	1	uint16*	
TIFFTAG_PAGENAME	1	char**	
TIFFTAG_PAGENUMBER	2	uint16*	
TIFFTAG_PHOTOMETRIC	1	uint16*	
TIFFTAG_PLANARCONFIG	1	uint16*	
TIFFTAG_PREDICTOR	1	uint16*	
TIFFTAG_PRIMARYCHROMATICITIES	1	float**	6-entry array
TIFFTAG_REFERENCEBLACKWHITE	1	float**	2*SamplesPerPixel array
TIFFTAG_RESOLUTIONUNIT	1	uint16*	
TIFFTAG_ROWSPERSTRIP	1	uint32*	
TIFFTAG_SAMPLEFORMAT	1	uint16*	
TIFFTAG_SAMPLESPELPIXEL	1	uint16*	
TIFFTAG_SMAXSAMPLEVALUE	1	double*	
TIFFTAG_SMINSAMPLEVALUE	1	double*	
TIFFTAG_SOFTWARE	1	char**	
TIFFTAG_STONITS	1	double**	
TIFFTAG_STRIPBYTECOUNTS	1	uint32**	
TIFFTAG_STRIPOFFSETS	1	uint32**	
TIFFTAG_SUBFILETYPE	1	uint32*	
TIFFTAG_SUBIFD	2	uint16*,uint32**	count & offsets array
TIFFTAG_TARGETPRINTER	1	char**	
TIFFTAG_THRESHHOLDING	1	uint16*	
TIFFTAG_TILEBYTECOUNTS	1	uint32**	
TIFFTAG_TILEDEPTH	1	uint32*	
TIFFTAG_TILELENGTH	1	uint32*	
TIFFTAG_TILEOFFSETS	1	uint32**	
TIFFTAG_TILEWIDTH	1	uint32*	
TIFFTAG_TRANSFERFUNCTION	1 or 3†	uint16**	1<<BitsPerSample entry arrays
TIFFTAG_WHITEPOINT	1	float**	2-entry array
TIFFTAG_XPOSITION	1	float*	
TIFFTAG_XRESOLUTION	1	float*	
TIFFTAG_YCBCRCOEFFICIENTS	1	float**	3-entry array
TIFFTAG_YCBCRPOSITIONING	1	uint16*	
TIFFTAG_YCBCRSUBSAMPLING	2	uint16*	
TIFFTAG_YPOSITION	1	float*	
TIFFTAG_YRESOLUTION	1	float*	
TIFFTAG_ICCPROFILE	2	uint32*,void**	count, profile data‡

† If *SamplesPerPixel* is one, then a single array is returned; otherwise three arrays are returned.

‡ The contents of this field are quite complex. See *The ICC Profile Format Specification*, Annex B.3 "Embedding ICC Profiles in TIFF Files" (available at <http://www.color.org>) for an explanation.

## RETURN VALUES

1 is returned if the tag is defined in the current directory; otherwise a 0 is returned.

## DIAGNOSTICS

All error messages are directed to the *TIFFError(3T)* routine.

**Unknown field, tag 0x%x.** An unknown tag was supplied.

## SEE ALSO

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFSetField(3T)*, *TIFFSetDirectory(3T)*, *TIFFReadDirectory(3T)*, *TIFFWriteDirectory(3T)*

**NAME**

`_TIFFmalloc`, `_TIFFrealloc`, `_TIFFfree`, `_TIFFmemset`, `_TIFFmemcpy`, `_TIFFmemcmp`, – memory management-related functions for use with TIFF files

**SYNOPSIS**

```
#include <tiffio.h>
tdata_t _TIFFmalloc(tsize_t);
tdata_t _TIFFrealloc(tdata_t, tsize_t);
void _TIFFfree(tdata_t);
void _TIFFmemset(tdata_t, int, tsize_t);
void _TIFFmemcpy(tdata_t, const tdata_t, tsize_t);
int _TIFFmemcmp(const tdata_t, const tdata_t, tsize_t);
```

**DESCRIPTION**

These routines are provided for writing portable software that uses *libtiff*; they hide any memory-management related issues, such as dealing with segmented architectures found on 16-bit machines.

`_TIFFmalloc` and `_TIFFrealloc` are used to dynamically allocate and reallocate memory used by *libtiff*; such as memory passed into the I/O routines. Memory allocated through these interfaces is released back to the system using the `_TIFFfree` routine.

Memory allocated through one of the above interfaces can be set to a known value using `_TIFFmemset`, copied to another memory location using `_TIFFmemcpy`, or compared for equality using `_TIFFmemcmp`. These routines conform to the equivalent ANSI C routines: `memset`, `memcpy`, and `memcmp`, respectively.

**DIAGNOSTICS**

None.

**SEE ALSO**

*libtiff*(3T), *malloc*(3C), *memory*(3C)

**NAME**

TIFFOpen, TIFFFdOpen, TIFFClientOpen – open a TIFF file for reading or writing

**SYNOPSIS**

```
#include <tiffio.h>
TIFF* TIFFOpen(const char* filename, const char* mode)
TIFF* TIFFFdOpen(const int fd, const char* filename, const char* mode)

typedef tsize_t (*TIFFReadWriteProc)(thandle_t, tdata_t, tsize_t);
typedef toff_t (*TIFFSeekProc)(thandle_t, toff_t, int);
typedef int (*TIFFCloseProc)(thandle_t);
typedef toff_t (*TIFFSizeProc)(thandle_t);
typedef int (*TIFFMapFileProc)(thandle_t, tdata_t*, toff_t*);
typedef void (*TIFFUnmapFileProc)(thandle_t, tdata_t, toff_t);

TIFF* TIFFClientOpen(const char* filename, const char* mode, thandle_t clientdata,
    TIFFReadWriteProc readproc, TIFFReadWriteProc writeproc, TIFFSeekProc seekproc,
    TIFFCloseProc closeproc, TIFFSizeProc sizeproc, TIFFMapFileProc mapproc,
    TIFFUnmapFileProc unmapproc)
```

**DESCRIPTION**

*TIFFOpen* opens a TIFF file whose name is *filename* and returns a handle to be used in subsequent calls to routines in *libtiff*. If the open operation fails, then zero is returned. The *mode* parameter specifies if the file is to be opened for reading (“r”), writing (“w”), or appending (“a”) and, optionally, whether to override certain default aspects of library operation (see below). When a file is opened for appending, existing data will not be touched; instead new data will be written as additional subfiles. If an existing file is opened for writing, all previous data is overwritten.

If a file is opened for reading, the first TIFF directory in the file is automatically read (also see *TIFFSetDirectory*(3T) for reading directories other than the first). If a file is opened for writing or appending, a default directory is automatically created for writing subsequent data. This directory has all the default values specified in TIFF Revision 6.0: *BitsPerSample*=1, *ThreshHolding*=bilevel art scan, *FillOrder*=1 (most significant bit of each data byte is filled first), *Orientation*=1 (the 0th row represents the visual top of the image, and the 0th column represents the visual left hand side), *SamplesPerPixel*=1, *RowsPerStrip*=infinity, *ResolutionUnit*=2 (inches), and *Compression*=1 (no compression). To alter these values, or to define values for additional fields, *TIFFSetField*(3T) must be used.

*TIFFFdOpen* is like *TIFFOpen* except that it opens a TIFF file given an open file descriptor *fd*. The file’s name and mode must reflect that of the open descriptor. The object associated with the file descriptor **must support random access**.

*TIFFClientOpen* is like *TIFFOpen* except that the caller supplies a collection of functions that the library will use to do UNIX-like I/O operations. The *readproc* and *writeproc* are called to read and write data at the current file position. *seekproc* is called to change the current file position a la *lseek*(2). *closeproc* is invoked to release any resources associated with an open file. *sizeproc* is invoked to obtain the size in bytes of a file. *mapproc* and *unmapproc* are called to map and unmap a file’s contents in memory; c.f. *mmap*(2) and *munmap*(2). The *clientdata* parameter is an opaque “handle” passed to the client-specified routines passed as parameters to *TIFFClientOpen*.

**OPTIONS**

The open mode parameter can include the following flags in addition to the “r”, “w”, and “a” flags. Note however that option flags must follow the read-write-append specification.

- I** When creating a new file force information be written with Little-Endian byte order (but see below). By default the library will create new files using the native CPU byte order.
- b** When creating a new file force information be written with Big-Endian byte order (but see below). By default the library will create new files using the native CPU byte order.
- L** Force image data that is read or written to be treated with bits filled from Least Significant Bit (LSB) to Most Significant Bit (MSB). Note that this is the opposite to the way the library has worked from its inception.
- B** Force image data that is read or written to be treated with bits filled from Most Significant Bit (MSB) to Least Significant Bit (LSB); this is the default.

- H** Force image data that is read or written to be treated with bits filled in the same order as the native CPU.
- M** Enable the use of memory-mapped files for images opened read-only. If the underlying system does not support memory-mapped files or if the specific image being opened cannot be memory-mapped then the library will fallback to using the normal system interface for reading information. By default the library will attempt to use memory-mapped files.
- m** Disable the use of memory-mapped files.
- C** Enable the use of “strip chopping” when reading images that are comprised of a single strip or tile of uncompressed data. Strip chopping is a mechanism by which the library will automatically convert the single-strip image to multiple strips, each of which has about 8 Kilobytes of data. This facility can be useful in reducing the amount of memory used to read an image because the library normally reads each strip in its entirety. Strip chopping does however alter the apparent contents of the image because when an image is divided into multiple strips it looks as though the underlying file contains multiple separate strips. Finally, note that default handling of strip chopping is a compile-time configuration parameter. The default behaviour, for backwards compatibility, is to enable strip chopping.
- c** Disable the use of strip chopping when reading images.

## BYTE ORDER

The TIFF specification (**all versions**) states that compliant readers *must be capable of reading images written in either byte order*. Nonetheless some software that claims to support the reading of TIFF images is incapable of reading images in anything but the native CPU byte order on which the software was written. (Especially notorious are applications written to run on Intel-based machines.) By default the library will create new files with the native byte-order of the CPU on which the application is run. This ensures optimal performance and is portable to any application that conforms to the TIFF specification. To force the library to use a specific byte-order when creating a new file the “b” and “l” option flags may be included in the call to open a file; for example, “wb” or “wl”.

## RETURN VALUES

Upon successful completion *TIFFOpen*, *TIFFFdOpen*, and *TIFFClientOpen* return a TIFF pointer. Otherwise, NULL is returned.

## DIAGNOSTICS

All error messages are directed to the *TIFFError(3T)* routine. Likewise, warning messages are directed to the *TIFFWarning(3T)* routine.

**“%s”: Bad mode.** The specified *mode* parameter was not one of “r” (read), “w” (write), or “a” (append).

**%s: Cannot open.** *TIFFOpen()* was unable to open the specified filename for read/writing.

**Cannot read TIFF header.** An error occurred while attempting to read the header information.

**Error writing TIFF header.** An error occurred while writing the default header information for a new file.

**Not a TIFF file, bad magic number %d (0x%x).** The magic number in the header was not (hex) 0x4d4d or (hex) 0x4949.

**Not a TIFF file, bad version number %d (0x%x).** The version field in the header was not 42 (decimal).

**Cannot append to file that has opposite byte ordering.** A file with a byte ordering opposite to the native byte ordering of the current machine was opened for appending (“a”). This is a limitation of the library.

## SEE ALSO

*libtiff(3T)*, *TIFFClose(3T)*

**NAME**

TIFFPrintDirectory – print a description of a TIFF directory

**SYNOPSIS**

```
#include <tiffio.h>
void TIFFPrintDirectory(TIFF* tif, FILE* fd, long flags)
```

**DESCRIPTION**

*TIFFPrintDirectory* prints a description of the current directory in the specified TIFF file to the standard I/O output stream *fd*. The *flags* parameter is used to control the *level of detail* of the printed information; it is a bit-or of the flags defined in **tiffio.h**:

```
#define TIFFPRINT_NONE          0x0    /* no extra info */
#define TIFFPRINT_STRIPS       0x1    /* strips/tiles info */
#define TIFFPRINT_CURVES      0x2    /* color/gray response curves */
#define TIFFPRINT_COLORMAP    0x4    /* colormap */
#define TIFFPRINT_JPEGQTABLES 0x100  /* JPEG Q matrices */
#define TIFFPRINT_JPEGACTABLES 0x200 /* JPEG AC tables */
#define TIFFPRINT_JPEGDCTABLES 0x200 /* JPEG DC tables */
```

**NOTES**

In C++ the *flags* parameter defaults to 0.

**RETURN VALUES**

None.

**DIAGNOSTICS**

None.

**SEE ALSO**

*libtiff*(3T), *TIFFOpen*(3T), *TIFFReadDirectory*(3T), *TIFFSetDirectory*(3T)

**NAME**

TIFFCurrentRow, TIFFCurrentStrip, TIFFCurrentTile, TIFFCurrentDirectory, TIFFLastDirectory, TIFFFileno, TIFFFileName, TIFFGetMode, TIFFIsTiled, TIFFIsByteSwapped, TIFFIsUpSampled, TIFFIsMSB2LSB – query routines

**SYNOPSIS**

```
#include <tiffio.h>
uint32 TIFFCurrentRow(TIFF* tif)
tstrip_t TIFFCurrentStrip(TIFF* tif)
ttile_t TIFFCurrentTile(TIFF* tif)
tdir_t TIFFCurrentDirectory(TIFF* tif)
int TIFFLastDirectory(TIFF* tif)
int TIFFFileno(TIFF* tif)
char* TIFFFileName(TIFF* tif)
int TIFFGetMode(TIFF* tif)
int TIFFIsTiled(TIFF* tif)
int TIFFIsByteSwapped(TIFF* tif)
int TIFFIsUpSampled(TIFF* tif)
int TIFFIsMSB2LSB(TIFF* tif)
const char* TIFFGetVersion(void)
```

**DESCRIPTION**

The following routines return status information about an open TIFF file.

*TIFFCurrentDirectory* returns the index of the current directory (directories are numbered starting at 0). This number is suitable for use with the *TIFFSetDirectory* routine.

*TIFFLastDirectory* returns a non-zero value if the current directory is the last directory in the file; otherwise zero is returned.

*TIFFCurrentRow*, *TIFFCurrentStrip*, and *TIFFCurrentTile*, return the current row, strip, and tile, respectively, that is being read or written. These values are updated each time a read or write is done.

*TIFFFileno* returns the underlying file descriptor used to access the TIFF image in the filesystem.

*TIFFFileName* returns the pathname argument passed to *TIFFOpen* or *TIFFFdOpen*.

*TIFFGetMode* returns the mode with which the underlying file was opened. On UNIX systems, this is the value passed to the *open(2)* system call.

*TIFFIsTiled* returns a non-zero value if the image data has a tiled organization. Zero is returned if the image data is organized in strips.

*TIFFIsByteSwapped* returns a non-zero value if the image data was in a different byte-order than the host machine. Zero is returned if the TIFF file and local host byte-orders are the same. Note that *TIFFReadTile()*, *TIFFReadStrip()* and *TIFFReadScanline()* functions already normally perform byte swapping to local host order if needed.

*TIFFIsUpSampled* returns a non-zero value if image data returned through the read interface routines is being up-sampled. This can be useful to applications that want to calculate I/O buffer sizes to reflect this usage (though the usual strip and tile size routines already do this).

*TIFFIsMSB2LSB* returns a non-zero value if the image data is being returned with bit 0 as the most significant bit.

*TIFFGetVersion* returns an ASCII string that has a version stamp for the TIFF library software.

**DIAGNOSTICS**

None.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFFdOpen(3T)*

**NAME**

TIFFReadDirectory – get the contents of the next directory in an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
int TIFFReadDirectory(TIFF* tif)
```

**DESCRIPTION**

Read the next directory in the specified file and make it the current directory. Applications only need to call *TIFFReadDirectory* to read multiple subfiles in a single TIFF file—the first directory in a file is automatically read when *TIFFOpen* is called.

**NOTES**

If the library is compiled with STRIPCHOP\_SUPPORT enabled, then images that have a single uncompressed strip or tile of data are automatically treated as if they were made up of multiple strips or tiles of approximately 8 kilobytes each. This operation is done only in-memory; it does not alter the contents of the file. However, the construction of the “chopped strips” is visible to the application through the number of strips [tiles] returned by *TIFFNumberOfStrips* [*TIFFNumberOfTiles*].

**RETURN VALUES**

If the next directory was successfully read, 1 is returned. Otherwise, 0 is returned if an error was encountered, or if there are no more directories to be read.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError*(3T) routine. All warning messages are directed to the *TIFFWarning*(3T) routine.

**Seek error accessing TIFF directory.** An error occurred while positioning to the location of the directory.

**Wrong data type %d for field "%s".** The tag entry in the directory had an incorrect data type. For example, an *ImageDescription* tag with a SHORT data type.

**TIFF directory is missing required "%s" field.** The specified tag is required to be present by the TIFF 5.0 specification, but is missing. The directory is (usually) unusable.

**%s: Rational with zero denominator.** A directory tag has a RATIONAL value whose denominator is zero.

**Incorrect count %d for field "%s" (%lu, expecting %lu); tag ignored.** The specified tag’s count field is bad. For example, a count other than 1 for a *SubFileType* tag.

**Cannot handle different per-sample values for field "%s".** The tag has *SamplesPerPixel* values and they are not all the same; e.g. *BitsPerSample*. The library is unable to handle images of this sort.

**Count mismatch for field "%s"; expecting %d, got %d.** The count field in a tag does not agree with the number expected by the library. This should never happen, so if it does, the library refuses to read the directory.

**Invalid TIFF directory; tags are not sorted in ascending order.** The directory tags are not properly sorted as specified in the TIFF 5.0 specification. This error is not fatal.

**Ignoring unknown field with tag %d (0x%x).** An unknown tag was encountered in the directory; the library ignores all such tags.

**TIFF directory is missing required "ImageLength" field.** The image violates the specification by not having a necessary field. There is no way for the library to recover from this error.

**TIFF directory is missing required "PlanarConfig" field.** The image violates the specification by not having a necessary field. There is no way for the library to recover from this error.

**TIFF directory is missing required "StripOffsets" field.** The image has multiple strips, but is missing the tag that specifies the file offset to each strip of data. There is no way for the library to recover from this error.

**TIFF directory is missing required "TileOffsets" field.** The image has multiple tiles, but is missing the tag that specifies the file offset to each tile of data. There is no way for the library to recover from this error.

**TIFF directory is missing required "StripByteCounts" field.** The image has multiple strips, but is

missing the tag that specifies the size of each strip of data. There is no way for the library to recover from this error.

**TIFF directory is missing required "StripByteCounts" field, calculating from imagelength.** The image violates the specification by not having a necessary field. However, when the image is comprised of only one strip or tile, the library will estimate the missing value based on the file size.

**Bogus "StripByteCounts" field, ignoring and calculating from imagelength.** Certain vendors violate the specification by writing zero for the StripByteCounts tag when they want to leave the value unspecified. If the image has a single strip, the library will estimate the missing value based on the file size.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFWriteDirectory(3T)*, *TIFFSetDirectory(3T)*, *TIFFSetSubDirectory(3T)*

**NAME**

TIFFReadEncodedStrip – read and decode a strip of data from an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
tsize_t TIFFReadEncodedStrip(TIFF* tif, tstrip_t strip, tdata_t buf, tsize_t size)
```

**DESCRIPTION**

Read the specified strip of data and place up to *size* bytes of decompressed information in the (user supplied) data buffer.

**NOTES**

The value of *strip* is a “raw strip number.” That is, the caller must take into account whether or not the data are organized in separate planes (*PlanarConfiguration=2*). To read a full strip of data the data buffer should typically be at least as large as the number returned by *TIFFStripSize(3T)*.

The library attempts to hide bit- and byte-ordering differences between the image and the native machine by converting data to the native machine order. Bit reversal is done if the *FillOrder* tag is opposite to the native machine bit order. 16- and 32-bit samples are automatically byte-swapped if the file was written with a byte order opposite to the native machine byte order,

**RETURN VALUES**

The actual number of bytes of data that were placed in *buf* is returned; *TIFFReadEncodedStrip* returns -1 if an error was encountered.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFReadRawStrip(3T)*, *TIFFReadScanline(3T)*

**NAME**

TIFFReadEncodedTile – read and decode a tile of data from an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
```

```
int TIFFReadEncodedTile(TIFF* tif, u_long tile, u_char* buf, u_long size)
```

**DESCRIPTION**

Read the specified tile of data and place up to *size* bytes of decompressed information in the (user supplied) data buffer.

**NOTES**

The value of *tile* is a “raw tile number.” That is, the caller must take into account whether or not the data are organized in separate planes (*PlanarConfiguration=2*). *TIFFComputeTile* automatically does this when converting an (x,y,z,sample) coordinate quadruple to a tile number. To read a full tile of data the data buffer should be at least as large as the value returned by *TIFFTileSize*.

The library attempts to hide bit- and byte-ordering differences between the image and the native machine by converting data to the native machine order. Bit reversal is done if the *FillOrder* tag is opposite to the native machine bit order. 16- and 32-bit samples are automatically byte-swapped if the file was written with a byte order opposite to the native machine byte order,

**RETURN VALUES**

The actual number of bytes of data that were placed in *buf* is returned; *TIFFReadEncodedTile* returns -1 if an error was encountered.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFReadRawTile(3T)*, *TIFFReadTile(3T)*

**NAME**

TIFFReadRawStrip – return the undecoded contents of a strip of data from an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
```

```
tsize_t TIFFReadRawStrip(TIFF* tif, tstrip_t strip, tdata_t buf, tsize_t size)
```

**DESCRIPTION**

Read the contents of the specified strip into the (user supplied) data buffer. Note that the value of *strip* is a “raw strip number.” That is, the caller must take into account whether or not the data is organized in separate planes (*PlanarConfiguration=2*). To read a full strip of data the data buffer should typically be at least as large as the number returned by *TIFFStripSize*.

**RETURN VALUES**

The actual number of bytes of data that were placed in *buf* is returned; *TIFFReadEncodedStrip* returns -1 if an error was encountered.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFReadEncodedStrip(3T)*, *TIFFReadScanline(3T)*, *TIFFStripSize(3T)*

**NAME**

TIFFReadRawTile – return an undecoded tile of data from an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
```

```
tsize_t TIFFReadRawTile(TIFF* tif, ttile_t tile, tdata_t buf, tsize_t size)
```

**DESCRIPTION**

Read the contents of the specified tile into the (user supplied) data buffer. Note that the value of *tile* is a “raw tile number.” That is, the caller must take into account whether or not the data is organized in separate planes (*PlanarConfiguration=2*). *TIFFComputeTile* automatically does this when converting an (x,y,z,sample) coordinate quadruple to a tile number. To read a full tile of data the data buffer should typically be at least as large as the value returned by *TIFFTileSize*.

**RETURN VALUES**

The actual number of bytes of data that were placed in *buf* is returned; *TIFFReadEncodedTile* returns -1 if an error was encountered.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFReadEncodedTile(3T)*, *TIFFReadTile(3T)*, *TIFFTileSize(3T)*

**NAME**

TIFFReadRGBAImage – read and decode an image into a fixed-format raster

**SYNOPSIS**

```
#include <tiffio.h>
#define TIFFGetR(abgr) ((abgr) & 0xff)
#define TIFFGetG(abgr) (((abgr) >> 8) & 0xff)
#define TIFFGetB(abgr) (((abgr) >> 16) & 0xff)
#define TIFFGetA(abgr) (((abgr) >> 24) & 0xff)

int TIFFReadRGBAImage(TIFF* tif, u_long width, u_long height, u_long* raster, int stopOnError)
```

**DESCRIPTION**

*TIFFReadRGBAImage* reads a strip- or tile-based image into memory, storing the result in the user supplied *raster*. The raster is assumed to be an array of *width* times *height* 32-bit entries, where *width* must be less than or equal to the width of the image (*height* may be any non-zero size). If the raster dimensions are smaller than the image, the image data is cropped to the raster bounds. If the raster height is greater than that of the image, then the image data are placed in the lower part of the raster. (Note that the raster is assumed to be organized such that the pixel at location  $(x,y)$  is  $raster[y*width+x]$ ; with the raster origin in the lower-left hand corner.)

Raster pixels are 8-bit packed red, green, blue, alpha samples. The macros *TIFFGetR*, *TIFFGetG*, *TIFFGetB*, and *TIFFGetA* should be used to access individual samples. Images without Associated Alpha matting information have a constant Alpha of 1.0 (255).

*TIFFReadRGBAImage* converts non-8-bit images by scaling sample values. Palette, grayscale, bilevel, CMYK, and YCbCr images are converted to RGB transparently. Raster pixels are returned uncorrected by any colorimetry information present in the directory.

The parameter *stopOnError* specifies how to act if an error is encountered while reading the image. If *stopOnError* is non-zero, then an error will terminate the operation; otherwise *TIFFReadRGBAImage* will continue processing data until all the possible data in the image have been requested.

**NOTES**

In C++ the *stopOnError* parameter defaults to 0.

Samples must be either 1, 2, 4, 8, or 16 bits. Colorimetric samples/pixel must be either 1, 3, or 4 (i.e. *SamplesPerPixel* minus *ExtraSamples*).

Palette image colormaps that appear to be incorrectly written as 8-bit values are automatically scaled to 16-bits.

*TIFFReadRGBAImage* is just a wrapper around the more general *TIFFRGBAImage(3T)* facilities.

**RETURN VALUES**

1 is returned if the image was successfully read and converted. Otherwise, 0 is returned if an error was encountered and *stopOnError* is zero.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**Sorry, can not handle %d-bit pictures.** The image had *BitsPerSample* other than 1, 2, 4, 8, or 16.

**Sorry, can not handle %d-channel images.** The image had *SamplesPerPixel* other than 1, 3, or 4.

**Missing needed "PhotometricInterpretation" tag.** The image did not have a tag that describes how to display the data.

**No "PhotometricInterpretation" tag, assuming RGB.** The image was missing a tag that describes how to display it, but because it has 3 or 4 samples/pixel, it is assumed to be RGB.

**No "PhotometricInterpretation" tag, assuming min-is-black.** The image was missing a tag that describes how to display it, but because it has 1 sample/pixel, it is assumed to be a grayscale or bilevel image.

**No space for photometric conversion table.** There was insufficient memory for a table used to convert image samples to 8-bit RGB.

**Missing required "Colormap" tag.** A Palette image did not have a required *Colormap* tag.

**No space for tile buffer.** There was insufficient memory to allocate an i/o buffer.

**No space for strip buffer.** There was insufficient memory to allocate an i/o buffer.

**Can not handle format.** The image has a format (combination of *BitsPerSample*, *SamplesPerPixel*, and *PhotometricInterpretation*) that *TIFFReadRGBAImage* can not handle.

**No space for B&W mapping table.** There was insufficient memory to allocate a table used to map grayscale data to RGB.

**No space for Palette mapping table.** There was insufficient memory to allocate a table used to map data to 8-bit RGB.

## BUGS

Orientations other than bottom-left, or top-left are not handled correctly.

## SEE ALSO

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFRGBAImage(3T)*, *TIFFReadRGBAStrip(3T)*, *TIFFReadRGBATile(3T)*

**NAME**

TIFFReadRGBAStrip – read and decode an image strip into a fixed-format raster

**SYNOPSIS**

```
#include <tiffio.h>
#define TIFFGetR(abgr) ((abgr) & 0xff)
#define TIFFGetG(abgr) (((abgr) >> 8) & 0xff)
#define TIFFGetB(abgr) (((abgr) >> 16) & 0xff)
#define TIFFGetA(abgr) (((abgr) >> 24) & 0xff)

int TIFFReadRGBAStrip(TIFF* tif, uint32 row, uint32 * raster )
```

**DESCRIPTION**

*TIFFReadRGBAStrip* reads a single strip of a strip-based image into memory, storing the result in the user supplied RGBA *raster*. The raster is assumed to be an array of width times rowsperstrip 32-bit entries, where width is the width of the image (TIFFTAG\_IMAGEWIDTH) and rowsperstrip is the maximum lines in a strip (TIFFTAG\_ROWSPERSTRIP).

The *row* value should be the row of the first row in the strip (strip \* rowsperstrip, zero based).

Note that the raster is assume to be organized such that the pixel at location (x,y) is *raster*[y\*width+x]; with the raster origin in the *lower-left hand corner* of the strip. That is bottom to top organization. When reading a partial last strip in the file the last line of the image will begin at the beginning of the buffer.

Raster pixels are 8-bit packed red, green, blue, alpha samples. The macros *TIFFGetR*, *TIFFGetG*, *TIFFGetB*, and *TIFFGetA* should be used to access individual samples. Images without Associated Alpha matting information have a constant Alpha of 1.0 (255).

See the *TIFFRGBAImage(3T)* page for more details on how various image types are converted to RGBA values.

**NOTES**

Samples must be either 1, 2, 4, 8, or 16 bits. Colorimetric samples/pixel must be either 1, 3, or 4 (i.e. *SamplesPerPixel* minus *ExtraSamples*).

Palette image colormaps that appear to be incorrectly written as 8-bit values are automatically scaled to 16-bits.

*TIFFReadRGBAStrip* is just a wrapper around the more general *TIFFRGBAImage(3T)* facilities. It's main advantage over the similar *TIFFReadRGBAImage()* function is that for large images a single buffer capable of holding the whole image doesn't need to be allocated, only enough for one strip. The *TIFFReadRGBATile()* function does a similar operation for tiled images.

**RETURN VALUES**

1 is returned if the image was successfully read and converted. Otherwise, 0 is returned if an error was encountered.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**Sorry, can not handle %d-bit pictures.** The image had *BitsPerSample* other than 1, 2, 4, 8, or 16.

**Sorry, can not handle %d-channel images.** The image had *SamplesPerPixel* other than 1, 3, or 4.

**Missing needed "PhotometricInterpretation" tag.** The image did not have a tag that describes how to display the data.

**No "PhotometricInterpretation" tag, assuming RGB.** The image was missing a tag that describes how to display it, but because it has 3 or 4 samples/pixel, it is assumed to be RGB.

**No "PhotometricInterpretation" tag, assuming min-is-black.** The image was missing a tag that describes how to display it, but because it has 1 sample/pixel, it is assumed to be a grayscale or bilevel image.

**No space for photometric conversion table.** There was insufficient memory for a table used to

convert image samples to 8-bit RGB.

**Missing required "Colormap" tag.** A Palette image did not have a required *Colormap* tag.

**No space for tile buffer.** There was insufficient memory to allocate an i/o buffer.

**No space for strip buffer.** There was insufficient memory to allocate an i/o buffer.

**Can not handle format.** The image has a format (combination of *BitsPerSample*, *SamplesPerPixel*, and *PhotometricInterpretation*) that *TIFFReadRGBAImage* can not handle.

**No space for B&W mapping table.** There was insufficient memory to allocate a table used to map grayscale data to RGB.

**No space for Palette mapping table.** There was insufficient memory to allocate a table used to map data to 8-bit RGB.

## BUGS

Orientations other than bottom-left, or top-left are not handled correctly.

## SEE ALSO

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFRGBAImage(3T)*, *TIFFReadRGBAImage(3T)*, *TIFFReadRGBATile(3T)*

**NAME**

TIFFReadRGBATile – read and decode an image tile into a fixed-format raster

**SYNOPSIS**

```
#include <tiffio.h>
#define TIFFGetR(abgr) ((abgr) & 0xff)
#define TIFFGetG(abgr) (((abgr) >> 8) & 0xff)
#define TIFFGetB(abgr) (((abgr) >> 16) & 0xff)
#define TIFFGetA(abgr) (((abgr) >> 24) & 0xff)

int TIFFReadRGBATile(TIFF* tif, uint32 x, uint32 y, uint32 * raster )
```

**DESCRIPTION**

*TIFFReadRGBATile* reads a single tile of a tile-based image into memory, storing the result in the user supplied RGBA *raster*. The raster is assumed to be an array of width times length 32-bit entries, where width is the width of a tile (TIFFTAG\_TILEWIDTH) and length is the height of a tile (TIFFTAG\_TILELENGTH).

The *x* and *y* values are the offsets from the top left corner to the top left corner of the tile to be read. They must be an exact multiple of the tile width and length.

Note that the raster is assume to be organized such that the pixel at location (*x,y*) is *raster[y\*width+x]*; with the raster origin in the *lower-left hand corner* of the tile. That is bottom to top organization. Edge tiles which partly fall off the image will be filled out with appropriate zeroed areas.

Raster pixels are 8-bit packed red, green, blue, alpha samples. The macros *TIFFGetR*, *TIFFGetG*, *TIFFGetB*, and *TIFFGetA* should be used to access individual samples. Images without Associated Alpha matting information have a constant Alpha of 1.0 (255).

See the *TIFFRGBAImage(3T)* page for more details on how various image types are converted to RGBA values.

**NOTES**

Samples must be either 1, 2, 4, 8, or 16 bits. Colorimetric samples/pixel must be either 1, 3, or 4 (i.e. *SamplesPerPixel* minus *ExtraSamples*).

Palette image colormaps that appear to be incorrectly written as 8-bit values are automatically scaled to 16-bits.

*TIFFReadRGBATile* is just a wrapper around the more general *TIFFRGBAImage(3T)* facilities. It's main advantage over the similar *TIFFReadRGBAImage()* function is that for large images a single buffer capable of holding the whole image doesn't need to be allocated, only enough for one tile. The *TIFFReadRGBAStrip()* function does a similar operation for striped images.

**RETURN VALUES**

1 is returned if the image was successfully read and converted. Otherwise, 0 is returned if an error was encountered.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**Sorry, can not handle %d-bit pictures.** The image had *BitsPerSample* other than 1, 2, 4, 8, or 16.

**Sorry, can not handle %d-channel images.** The image had *SamplesPerPixel* other than 1, 3, or 4.

**Missing needed "PhotometricInterpretation" tag.** The image did not have a tag that describes how to display the data.

**No "PhotometricInterpretation" tag, assuming RGB.** The image was missing a tag that describes how to display it, but because it has 3 or 4 samples/pixel, it is assumed to be RGB.

**No "PhotometricInterpretation" tag, assuming min-is-black.** The image was missing a tag that describes how to display it, but because it has 1 sample/pixel, it is assumed to be a grayscale or bilevel image.

**No space for photometric conversion table.** There was insufficient memory for a table used to

convert image samples to 8-bit RGB.

**Missing required "Colormap" tag.** A Palette image did not have a required *Colormap* tag.

**No space for tile buffer.** There was insufficient memory to allocate an i/o buffer.

**No space for strip buffer.** There was insufficient memory to allocate an i/o buffer.

**Can not handle format.** The image has a format (combination of *BitsPerSample*, *SamplesPerPixel*, and *PhotometricInterpretation*) that *TIFFReadRGBAImage* can not handle.

**No space for B&W mapping table.** There was insufficient memory to allocate a table used to map grayscale data to RGB.

**No space for Palette mapping table.** There was insufficient memory to allocate a table used to map data to 8-bit RGB.

## BUGS

Orientations other than bottom-left, or top-left are not handled correctly.

## SEE ALSO

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFRGBAImage(3T)*, *TIFFReadRGBAImage(3T)*, *TIFFReadRGBAStrip(3T)*

**NAME**

TIFFReadScanline – read and decode a scanline of data from an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
```

```
int TIFFReadScanline(TIFF* tif, tdata_t buf, uint32 row, tsample_t sample)
```

**DESCRIPTION**

Read the data for the specified row into the (user supplied) data buffer *buf*. The data are returned decompressed and, in the native byte- and bit-ordering, but are otherwise packed (see further below). The buffer must be large enough to hold an entire scanline of data. Applications should call the routine *TIFFScanlineSize* to find out the size (in bytes) of a scanline buffer. The *row* parameter is always used by *TIFFReadScanline*; the *sample* parameter is used only if data are organized in separate planes (*PlanarConfiguration=2*).

**NOTES**

The library attempts to hide bit- and byte-ordering differences between the image and the native machine by converting data to the native machine order. Bit reversal is done if the *FillOrder* tag is opposite to the native machine bit order. 16- and 32-bit samples are automatically byte-swapped if the file was written with a byte order opposite to the native machine byte order,

In C++ the *sample* parameter defaults to 0.

**RETURN VALUES**

*TIFFReadScanline* returns -1 if it detects an error; otherwise 1 is returned.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**Compression algorithm does not support random access.** Data was requested in a non-sequential order from a file that uses a compression algorithm and that has *RowsPerStrip* greater than one. That is, data in the image is stored in a compressed form, and with multiple rows packed into a strip. In this case, the library does not support random access to the data. The data should either be accessed sequentially, or the file should be converted so that each strip is made up of one row of data.

**BUGS**

Reading subsampled YCbCR data does not work correctly because, for *PlanarConfiguration=2* the size of a scanline is not calculated on a per-sample basis, and for *PlanarConfiguration=1* the library does not unpack the block-interleaved samples; use the strip- and tile-based interfaces to read these formats.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFReadEncodedStrip(3T)*, *TIFFReadRawStrip(3T)*

**NAME**

TIFFReadTile – read and decode a tile of data from an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
```

```
tsize_t TIFFReadTile(TIFF* tif, tdata_t buf, uint32 x, uint32 y, uint32 z, tsample_t sample)
```

**DESCRIPTION**

Return the data for the tile *containing* the specified coordinates. The data placed in *buf* are returned decompressed and, typically, in the native byte- and bit-ordering, but are otherwise packed (see further below). The buffer must be large enough to hold an entire tile of data. Applications should call the routine *TIFFTileSize* to find out the size (in bytes) of a tile buffer. The *x* and *y* parameters are always used by *TIFFReadTile*. The *z* parameter is used if the image is deeper than 1 slice (*ImageDepth*>1). The *sample* parameter is used only if data are organized in separate planes (*PlanarConfiguration*=2).

**NOTES**

The library attempts to hide bit- and byte-ordering differences between the image and the native machine by converting data to the native machine order. Bit reversal is done if the *FillOrder* tag is opposite to the native machine bit order. 16- and 32-bit samples are automatically byte-swapped if the file was written with a byte order opposite to the native machine byte order.

**RETURN VALUES**

*TIFFReadTile* returns -1 if it detects an error; otherwise the number of bytes in the decoded tile is returned.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError*(3T) routine.

**SEE ALSO**

*libtiff*(3T), *TIFFCheckTile*(3T), *TIFFComputeTile*(3T), *TIFFOpen*(3T), *TIFFReadEncodedTile*(3T), *TIFFReadRawTile*(3T)

**NAME**

TIFFRGBAImage – read and decode an image into a raster

**SYNOPSIS**

```
#include <tiffio.h>
typedef unsigned char TIFFRGBValue;
typedef struct _TIFFRGBAImage TIFFRGBAImage;
int TIFFRGBAImageOK(TIFF* tif, char emsg[1024]);
int TIFFRGBAImageBegin(TIFFRGBAImage* img, TIFF* tif, int stopOnError, char emsg[1024]);
int TIFFRGBAImageGet(TIFFRGBAImage* img, uint32* raster, uint32 width, uint32 height);
void TIFFRGBAImageEnd(TIFFRGBAImage* img);
```

**DESCRIPTION**

The routines described here provide a high-level interface through which TIFF images may be read into memory. Images may be strip- or tile-based and have a variety of different characteristics: bits/sample, samples/pixel, photometric, etc. Decoding state is encapsulated in a *TIFFRGBAImage* structure making it possible to capture state for multiple images and quickly switch between them. The target raster format can be customized to a particular application's needs by installing custom routines that manipulate image data according to application requirements.

The default usage for these routines is: check if an image can be processed using *TIFFRGBAImageOK*, construct a decoder state block using *TIFFRGBAImageBegin*, read and decode an image into a target raster using *TIFFRGBAImageGet*, and then release resources using *TIFFRGBAImageEnd*. *TIFFRGBAImageGet* can be called multiple times to decode an image using different state parameters. If multiple images are to be displayed and there is not enough space for each of the decoded rasters, multiple state blocks can be managed and then calls can be made to *TIFFRGBAImageGet* as needed to display an image.

The generated raster is assumed to be an array of *width* times *height* 32-bit entries, where *width* must be less than or equal to the width of the image (*height* may be any non-zero size). If the raster dimensions are smaller than the image, the image data is cropped to the raster bounds. If the raster height is greater than that of the image, then the image data are placed in the lower part of the raster. (Note that the raster is assumed to be organized such that the pixel at location (*x,y*) is *raster[y\*width+x]*; with the raster origin in the lower-left hand corner.)

Raster pixels are 8-bit packed red, green, blue, alpha samples. The macros *TIFFGetR*, *TIFFGetG*, *TIFFGetB*, and *TIFFGetA* should be used to access individual samples. Images without Associated Alpha matting information have a constant Alpha of 1.0 (255).

*TIFFRGBAImageGet* converts non-8-bit images by scaling sample values. Palette, grayscale, bilevel, CMYK, and YCbCr images are converted to RGB transparently. Raster pixels are returned uncorrected by any colorimetry information present in the directory.

The parameter *stopOnError* specifies how to act if an error is encountered while reading the image. If *stopOnError* is non-zero, then an error will terminate the operation; otherwise *TIFFRGBAImageGet* will continue processing data until all the possible data in the image have been requested.

**ALTERNATE RASTER FORMATS**

To use the core support for reading and processing TIFF images, but write the resulting raster data in a different format one need only override the “*put methods*” used to store raster data. These methods are defined in the *TIFFRGBAImage* structure and initially setup by *TIFFRGBAImageBegin* to point to routines that pack raster data in the default ABGR pixel format. Two different routines are used according to the physical organization of the image data in the file: *PlanarConfiguration=1* (packed samples), and *PlanarConfiguration=2* (separated samples). Note that this mechanism can be used to transform the data before storing it in the raster. For example one can convert data to colormap indices for display on a colormap display.

**SIMULTANEOUS RASTER STORE AND DISPLAY**

It is simple to display an image as it is being read into memory by overriding the put methods as described above for supporting alternate raster formats. Simply keep a reference to the default put methods setup by *TIFFRGBAImageBegin* and then invoke them before or after each display operation. For example, the *tiffgt(1)* utility uses the following put method to update the display as the raster is being filled:

```

static void
putContigAndDraw(TIFFRGBAImage* img, uint32* raster,
                uint32 x, uint32 y, uint32 w, uint32 h,
                int32 fromskew, int32 toskew,
                unsigned char* cp)
{
    (*putContig)(img, raster, x, y, w, h, fromskew, toskew, cp);
    if (x+w == width) {
        w = width;
        if (img->orientation == ORIENTATION_TOPLEFT)
            lrectwrite(0, y-(h-1), w-1, y, raster-x-(h-1)*w);
        else
            lrectwrite(0, y, w-1, y+h-1, raster);
    }
}

```

(the original routine provided by the library is saved in the variable *putContig*.)

## SUPPORTING ADDITIONAL TIFF FORMATS

The *TIFFRGBAImage* routines support the most commonly encountered flavors of TIFF. It is possible to extend this support by overriding the “*get method*” invoked by *TIFFRGBAImageGet* to read TIFF image data. Details of doing this are a bit involved, it is best to make a copy of an existing get method and modify it to suit the needs of an application.

## NOTES

Samples must be either 1, 2, 4, 8, or 16 bits. Colorimetric samples/pixel must be either 1, 3, or 4 (i.e. *SamplesPerPixel* minus *ExtraSamples*).

Palette image colormaps that appear to be incorrectly written as 8-bit values are automatically scaled to 16-bits.

## RETURN VALUES

All routines return 1 if the operation was successful. Otherwise, 0 is returned if an error was encountered and *stopOnError* is zero.

## DIAGNOSTICS

All error messages are directed to the *TIFFError(3T)* routine.

**Sorry, can not handle %d-bit pictures.** The image had *BitsPerSample* other than 1, 2, 4, 8, or 16.

**Sorry, can not handle %d-channel images.** The image had *SamplesPerPixel* other than 1, 3, or 4.

**Missing needed "PhotometricInterpretation" tag.** The image did not have a tag that describes how to display the data.

**No "PhotometricInterpretation" tag, assuming RGB.** The image was missing a tag that describes how to display it, but because it has 3 or 4 samples/pixel, it is assumed to be RGB.

**No "PhotometricInterpretation" tag, assuming min-is-black.** The image was missing a tag that describes how to display it, but because it has 1 sample/pixel, it is assumed to be a grayscale or bilevel image.

**No space for photometric conversion table.** There was insufficient memory for a table used to convert image samples to 8-bit RGB.

**Missing required "Colormap" tag.** A Palette image did not have a required *Colormap* tag.

**No space for tile buffer.** There was insufficient memory to allocate an i/o buffer.

**No space for strip buffer.** There was insufficient memory to allocate an i/o buffer.

**Can not handle format.** The image has a format (combination of *BitsPerSample*, *SamplesPerPixel*, and *PhotometricInterpretation*) that can not be handled.

**No space for B&W mapping table.** There was insufficient memory to allocate a table used to map grayscale data to RGB.

**No space for Palette mapping table.** There was insufficient memory to allocate a table used to map data to 8-bit RGB.

**BUGS**

Orientations other than bottom-left, or top-left are not handled correctly.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFReadRGBAImage(3T)*, *TIFFReadRGBAStrip(3T)*, *TIFFReadRGBATile(3T)*

**NAME**

TIFFSetDirectory, TIFFSetSubDirectory – set the current directory for an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
int TIFFSetDirectory(TIFF* tif, tdir_t dirnum)
int TIFFSetSubDirectory(TIFF* tif, uint32 diroff)
```

**DESCRIPTION**

*TIFFSetDirectory* changes the current directory and reads its contents with *TIFFReadDirectory*. The parameter *dirnum* specifies the subfile/directory as an integer number, with the first directory numbered zero.

*TIFFSetSubDirectory* acts like *TIFFSetDirectory*, except the directory is specified as a file offset instead of an index; this is required for accessing subdirectories linked through a *SubIFD* tag.

**RETURN VALUES**

On successful return 1 is returned. Otherwise, 0 is returned if *dirnum* or *diroff* specifies a non-existent directory, or if an error was encountered while reading the directory's contents.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError*(3T) routine.

**%s: Error fetching directory count.** An error was encountered while reading the “directory count” field.

**%s: Error fetching directory link.** An error was encountered while reading the “link value” that points to the next directory in a file.

**SEE ALSO**

*libtiff*(3T), *TIFFCurrentDirectory*(3T), *TIFFOpen*(3T), *TIFFReadDirectory*(3T), *TIFFWriteDirectory*(3T)

**NAME**

TIFFSetField – set the value(s) of a tag in a TIFF file open for writing

**SYNOPSIS**

```
#include <tiffio.h>
int TIFFSetField(TIFF* tif, ttag_t tag, ...)
#include <stdarg.h>
int TIFFVSetField(TIFF* tif, ttag_t tag, va_list ap)
```

**DESCRIPTION**

*TIFFSetField* sets the value of a field or pseudo-tag in the current directory associated with the open TIFF file *tif*. (A *pseudo-tag* is a parameter that is used to control the operation of the TIFF library but whose value is not read or written to the underlying file.) To set the value of a field the file must have been previously opened for writing with *TIFFOpen*(3T); pseudo-tags can be set whether the file was opened for reading or writing. The field is identified by *tag*, one of the values defined in the include file *tiff.h* (see also the table below). The actual value is specified using a variable argument list, as prescribed by the *stdarg*(3) interface (or, on some machines, the *varargs*(3) interface.)

*TIFFVSetField* is functionally equivalent to *TIFFSetField* except that it takes a pointer to a variable argument list. *TIFFVSetField* is useful for writing routines that are layered on top of the functionality provided by *TIFFSetField*.

The tags understood by *libtiff*, the number of parameter values, and the expected types for the parameter values are shown below. The data types are: *char\** is null-terminated string and corresponds to the ASCII data type; *uint16* is an unsigned 16-bit value; *uint32* is an unsigned 32-bit value; *uint16\** is an array of unsigned 16-bit values. *void\** is an array of data values of unspecified type.

Consult the TIFF specification for information on the meaning of each tag.

Tag Name	Count	Types	Notes
TIFFTAG_ARTIST	1	char*	
TIFFTAG_BADFAXLINES	1	uint32	
TIFFTAG_BITSPERSAMPLE	1	uint16	†
TIFFTAG_CLEANFAXDATA	1	uint16	
TIFFTAG_COLORMAP	3	uint16*	1<<BitsPerSample arrays
TIFFTAG_COMPRESSION	1	uint16	†
TIFFTAG_CONSECUTIVEBADFAXLINES	1	uint32	
TIFFTAG_COPYRIGHT	1	char*	
TIFFTAG_DATETIME	1	char*	
TIFFTAG_DOCUMENTNAME	1	char*	
TIFFTAG_DOTRANGE	2	uint16	
TIFFTAG_EXTRASAMPLES	2	uint16,uint16*	† count & types array
TIFFTAG_FAXMODE	1	int	† G3/G4 compression pseudo-tag
TIFFTAG_FAXFILLFUNC	1	TIFFFaxFillFunc	G3/G4 compression pseudo-tag
TIFFTAG_FILLORDER	1	uint16	†
TIFFTAG_GROUP3OPTIONS	1	uint32	†
TIFFTAG_GROUP4OPTIONS	1	uint32	†
TIFFTAG_HALFTONEHINTS	2	uint16	
TIFFTAG_HOSTCOMPUTER	1	char*	
TIFFTAG_IMAGEDESCRIPTION	1	char*	
TIFFTAG_IMAGEDEPTH	1	uint32	†
TIFFTAG_IMAGELENGTH	1	uint32	
TIFFTAG_IMAGEWIDTH	1	uint32	†
TIFFTAG_INKNAMES	1	char*	
TIFFTAG_INKSET	1	uint16	†
TIFFTAG_JPEGTABLES	2	uint32*,void*	† count & tables
TIFFTAG_JPEGQUALITY	1	int	JPEG pseudo-tag
TIFFTAG_JPEGCOLORMODE	1	int	† JPEG pseudo-tag
TIFFTAG_JPEGTABLESMODE	1	int	† JPEG pseudo-tag
TIFFTAG_MAKE	1	char*	
TIFFTAG_MATTEING	1	uint16	†

TIFFTAG_MAXSAMPLEVALUE	1	uint16	
TIFFTAG_MINSAMPLEVALUE	1	uint16	
TIFFTAG_MODEL	1	char*	
TIFFTAG_ORIENTATION	1	uint16	
TIFFTAG_PAGENAME	1	char*	
TIFFTAG_PAGENUMBER	2	uint16	
TIFFTAG_PHOTOMETRIC	1	uint16	
TIFFTAG_PLANARCONFIG	1	uint16	†
TIFFTAG_PREDICTOR	1	uint16	†
TIFFTAG_PRIMARYCHROMATICITIES	1	float*	6-entry array
TIFFTAG_REFERENCEBLACKWHITE	1	float*	† 2*SamplesPerPixel array
TIFFTAG_RESOLUTIONUNIT	1	uint16	
TIFFTAG_ROWSPERSTRIP	1	uint32	† must be > 0
TIFFTAG_SAMPLEFORMAT	1	uint16	†
TIFFTAG_SAMPLESPPERPIXEL	1	uint16	† value must be <= 4
TIFFTAG_SMAXSAMPLEVALUE	1	double	
TIFFTAG_SMINSAMPLEVALUE	1	double	
TIFFTAG_SOFTWARE	1	char*	
TIFFTAG_STONITS	1	double	†
TIFFTAG_SUBFILETYPE	1	uint32	
TIFFTAG_SUBIFD	2	uint16,uint32*	count & offsets array
TIFFTAG_TARGETPRINTER	1	char*	
TIFFTAG_THRESHHOLDING	1	uint16	
TIFFTAG_TILEDEPTH	1	uint32	†
TIFFTAG_TILELENGTH	1	uint32	† must be a multiple of 8
TIFFTAG_TILEWIDTH	1	uint32	† must be a multiple of 8
TIFFTAG_TRANSFERFUNCTION	1 or 3 ‡	uint16*	1<<BitsPerSample entry arrays
TIFFTAG_XPOSITION	1	float	
TIFFTAG_XRESOLUTION	1	float	
TIFFTAG_WHITEPOINT	1	float*	2-entry array
TIFFTAG_YCBCRCOEFFICIENTS	1	float*	† 3-entry array
TIFFTAG_YCBCRPOSITIONING	1	uint16	†
TIFFTAG_YCBCRSAMPLING	2	uint16	†
TIFFTAG_YPOSITION	1	float	
TIFFTAG_YRESOLUTION	1	float	
TIFFTAG_ICCPROFILE	2	uint32,void*	count, profile data*

† Tag may not have its values changed once data is written.

‡ If *SamplesPerPixel* is one, then a single array is passed; otherwise three arrays should be passed.

\* The contents of this field are quite complex. See *The ICC Profile Format Specification*, Annex B.3 "Embedding ICC Profiles in TIFF Files" (available at <http://www.color.org>) for an explanation.

## RETURN VALUES

1 is returned if the tag is defined in the current directory; otherwise a 0 is returned.

## RETURN VALUES

1 is returned if the operation was successful. Otherwise, 0 is returned if an error was detected.

## DIAGNOSTICS

All error messages are directed to the *TIFFError(3T)* routine.

**%s: Cannot modify tag "%s" while writing.** Data has already been written to the file, so the specified tag's value can not be changed. This restriction is applied to all tags that affect the format of written data.

**%d: Bad value for "%s".** An invalid value was supplied for the named tag.

## SEE ALSO

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFGetField(3T)*, *TIFFSetDirectory(3T)*, *TIFFWriteDirectory(3T)*, *TIFFReadDirectory(3T)*

**NAME**

`TIFFScanlineSize`, `TIFFRasterScanlineSize`, – return the size of various items associated with an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
tsize_t TIFFRasterScanlineSize(TIFF* tif)
tsize_t TIFFScanlineSize(TIFF* tif)
```

**DESCRIPTION**

*TIFFScanlineSize* returns the size in bytes of a row of data as it would be returned in a call to *TIFFReadScanline*, or as it would be expected in a call to *TIFFWriteScanline*.

*TIFFRasterScanlineSize* returns the size in bytes of a complete decoded and packed raster scanline. Note that this value may be different from the value returned by *TIFFScanlineSize* if data is stored as separate planes.

**DIAGNOSTICS**

None.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFReadScanline(3T)*

**NAME**

TIFFDefaultStripSize, TIFFStripSize, TIFFVStripSize, TIFFComputeStrip, TIFFNumberOfStrips – strip-related utility routines

**SYNOPSIS**

```
#include <tiffio.h>
uint32 TIFFDefaultStripSize(TIFF* tif, uint32 estimate)
tsize_t TIFFStripSize(TIFF* tif)
tsize_t TIFFVStripSize(TIFF* tif, uint32 nrows)
tstrip_t TIFFComputeStrip(TIFF* tif, uint32 row, tsample_t sample)
tstrip_t TIFFNumberOfStrips(TIFF* tif)
```

**DESCRIPTION**

*TIFFDefaultStripSize* returns the number of rows for a reasonable-sized strip according to the current settings of the *ImageWidth*, *BitsPerSample*, *SamplesPerPixel*, tags and any compression-specific requirements. If the *estimate* parameter, if non-zero, then it is taken as an estimate of the desired strip size and adjusted according to any compression-specific requirements. The value returned by this function is typically used to define the *RowsPerStrip* tag. In lieu of any unusual requirements *TIFFDefaultStripSize* tries to create strips that have approximately 8 kilobytes of uncompressed data.

*TIFFStripSize* returns the equivalent size for a strip of data as it would be returned in a call to *TIFFReadEncodedStrip* or as it would be expected in a call to *TIFFWriteEncodedStrip*.

*TIFFVStripSize* returns the number of bytes in a strip with *nrows* rows of data.

*TIFFComputeStrip* returns the strip that contains the specified coordinates. A valid strip is always returned; out-of-range coordinate values are clamped to the bounds of the image. The *row* parameter is always used in calculating a strip. The *sample* parameter is used only if data are organized in separate planes (*PlanarConfiguration=2*).

*TIFFNumberOfStrips* returns the number of strips in the image.

**DIAGNOSTICS**

None.

**SEE ALSO**

*libtiff*(3T), *TIFFReadEncodedStrip*(3T), *TIFFReadRawStrip*(3T), *TIFFWriteEncodedStrip*(3T), *TIFFWriteRawStrip*(3T)

**NAME**

TIFFReverseBits, TIFFSwabShort, TIFFSwabLong, TIFFSwabArrayOfShort, TIFFSwabArrayOfLong  
– byte- and bit-swapping routines

**SYNOPSIS**

```
#include <tiffio.h>
const unsigned char* TIFFGetBitRevTable(int reversed);
void TIFFReverseBits(u_char* data, unsigned long nbytes)
void TIFFSwabShort(uint16* data)
void TIFFSwabLong(uint32* data)
void TIFFSwabArrayOfShort(uint16* data, unsigned long nshorts)
void TIFFSwabArrayOfLong(uint32* data, unsigned long nlongs)
```

**DESCRIPTION**

The following routines are used by the library to swap 16- and 32-bit data and to reverse the order of bits in bytes.

*TIFFSwabShort* and *TIFFSwabLong* swap the bytes in a single 16-bit and 32-bit item, respectively. *TIFFSwabArrayOfShort* and *TIFFSwabArrayOfLong* swap the bytes in an array of 16-bit and 32-bit items, respectively.

*TIFFReverseBits* replaces each byte in *data* with the equivalent bit-reversed value. This operation is done with a lookup table, *TIFFBitRevTable* which is declared public. A second table, *TIFFNoBitRevTable* is also declared public; it is a lookup table that can be used as an *identity function*; i.e. *TIFFNoBitRevTable[n] == n*.

**DIAGNOSTICS**

None.

**SEE ALSO**

*libtiff*(3T),

**NAME**

TIFFTileSize, TIFFTileRowSize, TIFFVTileSize, TIFFDefaultTileSize, TIFFComputeTile, TIFFCheckTile, TIFFNumberOfTiles – tile-related utility routines

**SYNOPSIS**

```
#include <tiffio.h>
void TIFFDefaultTileSize(TIFF* tif, uint32* tw, uint32* th)
tsize_t TIFFTileSize(TIFF* tif)
tsize_t TIFFTileRowSize(TIFF* tif)
tsize_t TIFFVTileSize(TIFF* tif, uint32 nrows)
ttile_t TIFFComputeTile(TIFF* tif, uint32 x, uint32 y, uint32 z, tsample_t sample)
int TIFFCheckTile(TIFF* tif, uint32 x, uint32 y, uint32 z, tsample_t sample)
ttile_t TIFFNumberOfTiles(TIFF* tif)
```

**DESCRIPTION**

*TIFFDefaultTileSize* returns the pixel width and height of a reasonable-sized tile; suitable for setting up the *TileWidth* and *TileLength* tags. If the *tw* and *th* values passed in are non-zero, then they are adjusted to reflect any compression-specific requirements. The returned width and height are constrained to be a multiple of 16 pixels to conform with the TIFF specification.

*TIFFTileSize* returns the equivalent size for a tile of data as it would be returned in a call to *TIFFReadTile* or as it would be expected in a call to *TIFFWriteTile*.

*TIFFVTileSize* returns the number of bytes in a row-aligned tile with *nrows* of data.

*TIFFTileRowSize* returns the number of bytes of a row of data in a tile.

*TIFFComputeTile* returns the tile that contains the specified coordinates. A valid tile is always returned; out-of-range coordinate values are clamped to the bounds of the image. The *x* and *y* parameters are always used in calculating a tile. The *z* parameter is used if the image is deeper than 1 slice (*ImageDepth*>1). The *sample* parameter is used only if data are organized in separate planes (*PlanarConfiguration*=2).

*TIFFCheckTile* returns a non-zero value if the supplied coordinates are within the bounds of the image and zero otherwise. The *x* parameter is checked against the value of the *ImageWidth* tag. The *y* parameter is checked against the value of the *ImageLength* tag. The *z* parameter is checked against the value of the *ImageDepth* tag (if defined). The *sample* parameter is checked against the value of the *SamplesPerPixel* parameter if the data are organized in separate planes.

*TIFFNumberOfTiles* returns the number of tiles in the image.

**DIAGNOSTICS**

None.

**SEE ALSO**

*libtiff*(3T), *TIFFReadEncodedTile*(3T), *TIFFReadRawTile*(3T), *TIFFReadTile*(3T), *TIFFWriteEncodedTile*(3T), *TIFFWriteRawTile*(3T), *TIFFWriteTile*(3T)

**NAME**

TIFFWarning, TIFFSetWarningHandler – library warning interface

**SYNOPSIS**

```
#include <tiffio.h>
void TIFFWarning(const char* module, const char* fmt, ...)

#include <stdarg.h>
typedef void (*TIFFWarningHandler)(const char* module, const char* fmt, va_list ap);
TIFFWarningHandler TIFFSetWarningHandler(TIFFWarningHandler handler);
```

**DESCRIPTION**

*TIFFWarning* invokes the library-wide warning handler function to (normally) write a warning message to the **stderr**. The *fmt* parameter is a *printf*(3S) format string, and any number arguments can be supplied. The *module* parameter is interpreted as a string that, if non-zero, should be printed before the message; it typically is used to identify the software module in which a warning is detected.

Applications that desire to capture control in the event of a warning should use *TIFFSetWarningHandler* to override the default warning handler. A NULL (0) warning handler function may be installed to suppress error messages.

**RETURN VALUES**

*TIFFSetWarningHandler* returns a reference to the previous error handling function.

**SEE ALSO**

*libtiff*(3T), *TIFFError*(3T), *printf*(3S)

**NAME**

TIFFWriteDirectory, TIFFRewriteDirectory – write the current directory in an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
int TIFFWriteDirectory(TIFF* tif)
int TIFFRewriteDirectory(TIFF* tif)
```

**DESCRIPTION**

*TIFFWriteDirectory* will write the contents of the current directory to the file and setup to create a new subfile in the same file. Applications only need to call *TIFFWriteDirectory* when writing multiple subfiles to a single TIFF file. *TIFFWriteDirectory* is automatically called by *TIFFClose* and *TIFFFlush* to write a modified directory if the file is open for writing.

The *TIFFRewriteDirectory* function operates similarly to *TIFFWriteDirectory* but can be called with directories previously read or written that already have an established location in the file. It will rewrite the directory, but instead of place it at it's old location (as *TIFFWriteDirectory* would) it will place them at the end of the file, correcting the pointer from the preceeding directory or file header to point to it's new location. This is particularly important in cases where the size of the directory and pointed to data has grown, so it won't fit in the space available at the old location.

**RETURN VALUES**

1 is returned when the contents are successfully written to the file. Otherwise, 0 is returned if an error was encountered when writing the directory contents.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**Error post-encoding before directory write.** Before writing the contents of the current directory, any pending data are flushed. This message indicates that an error occurred while doing this.

**Error flushing data before directory write.** Before writing the contents of the current directory, any pending data are flushed. This message indicates that an error occurred while doing this.

**Cannot write directory, out of space.** There was not enough space to allocate a temporary area for the directory that was to be written.

**Error writing directory count.** A write error occurred when writing the count of fields in the directory.

**Error writing directory contents.** A write error occurred when writing the directory fields.

**Error writing directory link.** A write error occurred when writing the link to the next directory.

**Error writing data for field "%s".** A write error occurred when writing indirect data for the specified field.

**Error writing TIFF header.** A write error occurred when re-writing header at the front of the file.

**Error fetching directory count.** A read error occurred when fetching the directory count field for a previous directory. This can occur when setting up a link to the directory that is being written.

**Error fetching directory link.** A read error occurred when fetching the directory link field for a previous directory. This can occur when setting up a link to the directory that is being written.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFError(3T)*, *TIFFReadDirectory(3T)*, *TIFFSetDirectory(3T)*

**NAME**

TIFFWriteEncodedStrip – compress and write a strip of data to an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
tsize_t TIFFWriteEncodedStrip(TIFF* tif, tstrip_t strip, tdata_t buf, tsize_t size)
```

**DESCRIPTION**

Compress *size* bytes of raw data from *buf* and write the result to the specified strip; replacing any previously written data. Note that the value of *strip* is a “raw strip number.” That is, the caller must take into account whether or not the data are organized in separate places (*PlanarConfiguration=2*).

**NOTES**

The library writes encoded data using the native machine byte order. Correctly implemented TIFF readers are expected to do any necessary byte-swapping to correctly process image data with BitsPerSample greater than 8.

The strip number must be valid according to the current settings of the *ImageLength* and *RowsPerStrip* tags. An image may be dynamically grown by increasing the value of *ImageLength* prior to each call to *TIFFWriteEncodedStrip*.

**RETURN VALUES**

–1 is returned if an error was encountered. Otherwise, the value of *size* is returned.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**%s: File not open for writing.** The file was opened for reading, not writing.

**Can not write scanlines to a tiled image.** The image is assumed to be organized in tiles because the *TileWidth* and *TileLength* tags have been set with *TIFFSetField(3T)*.

**%s: Must set "ImageWidth" before writing data.** The image’s width has not be set before the first write. See *TIFFSetField(3T)* for information on how to do this.

**%s: Must set "PlanarConfiguration" before writing data.** The organization of data has not be defined before the first write. See *TIFFSetField(3T)* for information on how to do this.

**%s: No space for strip arrays".** There was not enough space for the arrays that hold strip offsets and byte counts.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFWriteScanline(3T)*, *TIFFWriteRawStrip(3T)*

**NAME**

TIFFWriteEncodedTile – compress and write a tile of data to an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
tsize_t TIFFWriteEncodedTile(TIFF* tif, ttile_t tile, tdata_t buf, tsize_t size)
```

**DESCRIPTION**

Compress *size* bytes of raw data from *buf* and **append** the result to the end of the specified tile. Note that the value of *tile* is a “raw tile number.” That is, the caller must take into account whether or not the data are organized in separate places (*PlanarConfiguration=2*). *TIFFComputeTile* automatically does this when converting an (x,y,z,sample) coordinate quadruple to a tile number.

**NOTES**

The library writes encoded data using the native machine byte order. Correctly implemented TIFF readers are expected to do any necessary byte-swapping to correctly process image data with BitsPerSample greater than 8.

**RETURN VALUES**

–1 is returned if an error was encountered. Otherwise, the value of *size* is returned.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError*(3T) routine.

**%s: File not open for writing.** The file was opened for reading, not writing.

**Can not write tiles to a stripped image.** The image is assumed to be organized in strips because neither of the *TileWidth* or *TileLength* tags have been set with *TIFFSetField*(3T).

**%s: Must set "ImageWidth" before writing data.** The image’s width has not be set before the first write. See *TIFFSetField*(3T) for information on how to do this.

**%s: Must set "PlanarConfiguration" before writing data.** The organization of data has not be defined before the first write. See *TIFFSetField*(3T) for information on how to do this.

**%s: No space for tile arrays".** There was not enough space for the arrays that hold tile offsets and byte counts.

**SEE ALSO**

*libtiff*(3T), *TIFFOpen*(3T), *TIFFWriteTile*(3T), *TIFFWriteRawTile*(3T)

**NAME**

TIFFWriteRawStrip – write a strip of raw data to an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
tsize_t TIFFWriteRawStrip(TIFF* tif, tstrip_t strip, tdata_t buf, tsize_t size)
```

**DESCRIPTION**

Append *size* bytes of raw data to the specified strip.

**NOTES**

The strip number must be valid according to the current settings of the *ImageLength* and *RowsPerStrip* tags. An image may be dynamically grown by increasing the value of *ImageLength* prior to each call to *TIFFWriteRawStrip*.

**RETURN VALUES**

-1 is returned if an error occurred. Otherwise, the value of *size* is returned.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**%s: File not open for writing.** The file was opened for reading, not writing.

**Can not write scanlines to a tiled image.** The image is assumed to be organized in tiles because the *TileWidth* and *TileLength* tags have been set with *TIFFSetField(3T)*.

**%s: Must set "ImageWidth" before writing data.** The image's width has not be set before the first write. See *TIFFSetField(3T)* for information on how to do this.

**%s: Must set "PlanarConfiguration" before writing data.** The organization of data has not be defined before the first write. See *TIFFSetField(3T)* for information on how to do this.

**%s: No space for strip arrays".** There was not enough space for the arrays that hold strip offsets and byte counts.

**%s: Strip %d out of range, max %d.** The specified strip is not a valid strip according to the currently specified image dimensions.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFWriteEncodedStrip(3T)*, *TIFFWriteScanline(3T)*

**NAME**

TIFFWriteRawTile – write a tile of raw data to an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
tsize_t TIFFWriteRawTile(TIFF* tif, ttile_t tile, tdata_t buf, tsize_t size)
```

**DESCRIPTION**

Append *size* bytes of raw data to the specified tile.

**RETURN VALUES**

–1 is returned if an error occurred. Otherwise, the value of *size* is returned.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**%s: File not open for writing.** The file was opened for reading, not writing.

**Can not write tiles to a stripped image.** The image is assumed to be organized in strips because neither of the *TileWidth* or *TileLength* tags have been set with *TIFFSetField(3T)*.

**%s: Must set "ImageWidth" before writing data.** The image's width has not be set before the first write. See *TIFFSetField(3T)* for information on how to do this.

**%s: Must set "PlanarConfiguration" before writing data.** The organization of data has not be defined before the first write. See *TIFFSetField(3T)* for information on how to do this.

**%s: No space for tile arrays".** There was not enough space for the arrays that hold tile offsets and byte counts.

**%s: Specified tile %d out of range, max %d.** The specified tile is not valid according to the currently specified image dimensions.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFWriteEncodedTile(3T)*, *TIFFWriteScanline(3T)*

**NAME**

TIFFWriteScanline – write a scanline to an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
int TIFFWriteScanline(TIFF* tif, tdata_t buf, uint32 row, tsample_t sample)
```

**DESCRIPTION**

Write data to a file at the specified row. The *sample* parameter is used only if data are organized in separate planes (*PlanarConfiguration=2*). The data are assumed to be uncompressed and in the native bit- and byte-order of the host machine. The data written to the file is compressed according to the compression scheme of the current TIFF directory (see further below). If the current scanline is past the end of the current subfile, the *ImageLength* field is automatically increased to include the scanline (except for *PlanarConfiguration=2*, where the *ImageLength* cannot be changed once the first data are written). If the *ImageLength* is increased, the *StripOffsets* and *StripByteCounts* fields are similarly enlarged to reflect data written past the previous end of image.

**NOTES**

The library writes encoded data using the native machine byte order. Correctly implemented TIFF readers are expected to do any necessary byte-swapping to correctly process image data with *BitsPerSample* greater than 8. The library attempts to hide bit-ordering differences between the image and the native machine by converting data from the native machine order.

In C++ the *sample* parameter defaults to 0.

Once data are written to a file for the current directory, the values of certain tags may not be altered; see *TIFFSetField(3T)* for more information.

It is not possible to write scanlines to a file that uses a tiled organization. The routine *TIFFIsTiled* can be used to determine if the file is organized as tiles or strips.

**RETURN VALUES**

*TIFFWriteScanline* returns -1 if it immediately detects an error and 1 for a successful write.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError(3T)* routine.

**%s: File not open for writing .** The file was opened for reading, not writing.

**Can not write scanlines to a tiled image.** An attempt was made to write a scanline to a tiled image. The image is assumed to be organized in tiles because the *TileWidth* and *TileLength* tags have been set with *TIFFSetField(3T)*.

**Compression algorithm does not support random access.** Data was written in a non-sequential order to a file that uses a compression algorithm and that has *RowsPerStrip* greater than one. That is, data in the image is to be stored in a compressed form, and with multiple rows packed into a strip. In this case, the library does not support random access to the data. The data should either be written as entire strips, sequentially by rows, or the value of *RowsPerStrip* should be set to one.

**%s: Must set "ImageWidth" before writing data.** The image's width has not be set before the first write. See *TIFFSetField(3T)* for information on how to do this.

**%s: Must set "PlanarConfiguration" before writing data.** The organization of data has not be defined before the first write. See *TIFFSetField(3T)* for information on how to do this.

**Can not change "ImageLength" when using separate planes.** Separate image planes are being used (*PlanarConfiguration=2*), but the number of rows has not been specified before the first write. The library supports the dynamic growth of an image only when data are organized in a contiguous manner (*PlanarConfiguration=1*).

**%d: Sample out of range, max %d.** The *sample* parameter was greater than the value of the *SamplesPerPixel* tag.

**%s: No space for strip arrays .** There was not enough space for the arrays that hold strip offsets and byte counts.

**BUGS**

Writing subsampled YCbCR data does not work correctly because, for *PlanarConfiguration=2* the size of a scanline is not calculated on a per-sample basis, and for *PlanarConfiguration=1* the library does not pack the block-interleaved samples.

**SEE ALSO**

*libtiff(3T)*, *TIFFOpen(3T)*, *TIFFWriteEncodedStrip(3T)*, *TIFFWriteRawStrip(3T)*

**NAME**

TIFFWriteTile – encode and write a tile of data to an open TIFF file

**SYNOPSIS**

```
#include <tiffio.h>
```

```
tsize_t TIFFWriteTile(TIFF* tif, tdata_t buf, uint32 x, uint32 y, uint32 z, tsample_t sample)
```

**DESCRIPTION**

Write the data for the tile *containing* the specified coordinates. The data in *buf* are is (potentially) compressed, and written to the indicated file, normally being appended to the end of the file. The buffer must contain an entire tile of data. Applications should call the routine *TIFFTileSize* to find out the size (in bytes) of a tile buffer. The *x* and *y* parameters are always used by *TIFFWriteTile*. The *z* parameter is used if the image is deeper than 1 slice (*ImageDepth*>1). The *sample* parameter is used only if data are organized in separate planes (*PlanarConfiguration*=2).

**RETURN VALUES**

*TIFFWriteTile* returns -1 if it detects an error; otherwise the number of bytes in the tile is returned.

**DIAGNOSTICS**

All error messages are directed to the *TIFFError*(3T) routine.

**SEE ALSO**

*libtiff*(3T), *TIFFCheckTile*(3T), *TIFFComputeTile*(3T), *TIFFOpen*(3T), *TIFFReadTile*(3T), *TIFFWriteScanline*(3T), *TIFFWriteEncodedTile*(3T), *TIFFWriteRawTile*(3T)