NAME

TIFFRGBAImageOK, TIFFRGBAImageBegin, TIFFRGBAImageGet, TIFFRGBAImageEnd - 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_t*** *raster***, uint32_t** *width* **, uint32_t** *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 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.)

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 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_t* raster,
   uint32_t x, uint32_t y, uint32_t w, uint32_t h,
   int32_t fromskew, int32_t 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*(3TIFF) 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.

SEE ALSO

TIFFOpen(3TIFF), **TIFFReadRGBAImage**(3TIFF), **TIFFReadRGBAImageOriented**(3TIFF), **TIFFReadRGBAStrip**(3TIFF), **TIFFReadRGBATile**(3TIFF), **libtiff**(3TIFF)

Libtiff library home page: **http://www.simplesystems.org/libtiff/**