vImage Conversion Reference

Performance > Graphics & Imaging

2007-07-12

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

Figures

vImage Conversion Reference

Framework:	Accelerate/vImage
Companion guide	vImage Programming Guide
Declared in	Conversion.h

Overview

Conversion functions change an image from one image format into another. These functions work with the formats supported by vImage (Planar8, PlanarF, ARGB8888, ARGBFFFF, RGBA8888, and RGBAFFFF) but they can also change between a supported format to one that's not supported by vImage (such as RGB565). Conversion functions can also fill buffers with a color, overwrite channels, permute channels, flatten data, and clip data.

Conversion functions use a vImage buffer structure (`vImage_Buffer`—see *vImage Data Types and Constants Reference*) to receive and supply image data. This buffer contains a pointer to image data, the height and width (in pixels) of the image data, and the number of row bytes. You actually pass a pointer to a vImage buffer structure. For some functions, you can provide a pointer to the same vImage buffer structure for the source images and the destination image because the function "works in place". That is , the source and destination images can occupy the same memory if the they are strictly aligned pixel for pixel.

Functions by Task

Filling Buffers

`vImageBufferFill_ARGB8888` (page 11)
Fills an ARGB8888 buffer with a specified color.

`vImageBufferFill_ARGBFFFF` (page 12)
Fills an ARGBFFFF buffer with a specified color.

Permuting Channels

`vImagePermuteChannels_ARGB8888` (page 52)
Reorders the channels in an ARGB8888 image.

`vImagePermuteChannels_ARGBFFFF` (page 54)
Reorders the channels in an ARGBFFFF image.

Overwriting Channels

`vImageSelectChannels_ARGB8888` (page 54)
Overwrites the specified channels in an ARGB8888 image buffer with the provided channels from an ARGB8888 image buffer.

`vImageSelectChannels_ARGBFFFF` (page 55)
Overwrites the specified channels in an ARGBFFFF image buffer with the provided channels in an ARGBFFFF image buffer.

`vImageOverwriteChannels_ARGB8888` (page 50)
Overwrites one or more planes of an ARGB8888 image buffer with the provided planar buffer.

`vImageOverwriteChannels_ARGBFFFF` (page 51)
Overwrites one or more planes of an ARGBFFFF image buffer with the provided planar buffer.

`vImageOverwriteChannelsWithScalar_ARGB8888` (page 47)
Overwrites the pixels of one or more planes of an ARGB8888 image buffer with the provided scalar value.

`vImageOverwriteChannelsWithScalar_ARGBFFFF` (page 48)
Overwrites the pixels of one or more planes of an ARGBFFFF image buffer with the provided scalar value.

`vImageOverwriteChannelsWithScalar_Planar8` (page 49)
Overwrites a Planar8 image buffer with the provided value.

`vImageOverwriteChannelsWithScalar_PlanarF` (page 49)
Overwrites a PlanarF image buffer with the provided value.

`vImageOverwriteChannelsWithPixel_ARGB8888` (page 45)
Overwrites an ARGB8888 image buffer with the provided pixel value.

`vImageOverwriteChannelsWithPixel_ARGBFFFF` (page 46)
Overwrites an ARGBFFFF image buffer with the provided pixel value.

Converting From 16 Bit

`vImageConvert_16SToF` (page 13)
Converts an image in a special planar format—in which each pixel value is a 16-bit signed integer—to a PlanarF format.

`vImageConvert_16UToF` (page 14)
Converts an image in a special planar format—in which each pixel value is a 16-bit unsigned integer—to a PlanarF format.

`vImageConvert_16UToPlanar8` (page 15)
Converts an image in a special planar format—in which each pixel value is a 16-bit unsigned integer—to a Planar8 image.

Transforming Using Table Lookups

`vImageTableLookUp_ARGB8888` (page 56)
Transforms an ARGB8888 image by substituting pixel values with pixel values provided by four lookup tables.

`vImageTableLookUp_Planar8` (page 58)
Transforms an Planar8 image by substituting pixel values with pixel values provided by four lookup tables.

Flattening Data

`vImageFlatten_ARGB8888ToRGB888` (page 43)
Transforms an ARGB8888 image to an RGB888 image against an opaque background of the provided color.

`vImageFlatten_ARGBFFFFToRGBFFF` (page 44)
Transforms an ARGBFFFF image to an RGBFFF image against an opaque background of the provided color.

Clipping Data

`vImageClip_PlanarF` (page 12)
Clips the pixel values of an image in PlanarF format, using the provided minimum and maximum values.

Converting Between Chunky and Planar

These convenience functions allow you to convert between various interleaved (or *chunky*) formats that vImage does not explicitly support (and that may have less than or more than four channels) and the formats that vImage supports explicitly. You can represent some non-interleaved formats as well. The functions are not fast or vectorized.

`vImageConvert_PlanarToChunky8` (page 36)
Combines a collection of planar source images into a single interleaved destination image, with one 8-bit channel for each planar image.

`vImageConvert_PlanarToChunkyF` (page 37)
Combines a collection of planar source images into a single interleaved destination image, with one floating-point channel for each planar image.

`vImageConvert_ChunkyToPlanar8` (page 22)
Separates a source image into a collection of corresponding planar destination images, one for each 8-bit channel of the original image.

`vImageConvert_ChunkyToPlanarF` (page 23)
Separates a source image into a collection of corresponding planar destination images, one for each floating-point channel of the original image.

Converting From Planar Formats

`vImageConvert_Planar8To16U` (page 27)
Converts a Planar8 image to a 16U image .

`vImageConvert_Planar8toARGB1555` (page 28)
Combines four Planar8 images into one ARGB1555 image.

vImageConvert_Planar8toARGB8888 (page 29)
Combines four Planar8 images into one ARGB8888 image.

vImageConvert_Planar8toPlanarF (page 30)
Converts a Planar8 image to a PlanarF image.

vImageConvert_Planar8toRGB565 (page 31)
Combines three Planar8 images into one RGB565 image.

vImageConvert_Planar8toRGB888 (page 32)
Combines three Planar8 images into one RGB888 image.

vImageConvert_PlanarFtoRGBFFF (page 35)
Combines three PlanarF images into one RGBFFF image.

vImageConvert_PlanarFtoARGBFFFF (page 32)
Combines four PlanarF images into one ARGBFFFF image.

vImageConvert_PlanarFtoPlanar16F (page 33)
Converts a PlanarF image to a Planar16F image.

vImageConvert_PlanarFtoPlanar8 (page 34)
Converts a PlanarF image to a Planar8 image, clipping values to the provided minimum and maximum values.

vImageConvert_Planar16FtoPlanarF (page 26)
Converts a Planar16F image to a PlanarF image.

vImageConvert_FTo16S (page 25)
Converts a PlanarF image into a special format in which each pixel is a 16-bit signed integer.

vImageConvert_FTo16U (page 25)
Converts a PlanarF image into a special format in which each pixel is a 16-bit unsigned integer.

Converting From ARGB Formats

vImageConvert_ARGB1555toARGB8888 (page 16)
Converts an ARGB1555 image to an ARGB8888 image.

vImageConvert_ARGB1555toPlanar8 (page 16)
Separates an ARGB1555 image into four Planar8 images.

vImageConvert_ARGB8888toARGB1555 (page 18)
Converts an ARGB8888 image into an ARGB1555 image.

vImageConvert_ARGB8888toPlanar8 (page 18)
Separates an ARGB8888 image into four Planar8 images.

vImageConvert_ARGB8888toRGB565 (page 20)
Converts an ARGB8888 image into an RGB565 image.

vImageConvert_ARGB8888toRGB888 (page 20)
Converts an ARGB8888 image into an RGB888 image.

vImageConvert_ARGBFFFFtoPlanarF (page 21)
Separates an ARGBFFFF image into four PlanarF images.

Converting From RGB Formats

`vImageConvert_RGB565toPlanar8` (page 39)
Separates an RGB565 image into three Planar8 images.

`vImageConvert_RGB565toARGB8888` (page 38)
Converts an RGB565 image into an ARGB8888 image, using the provided 8-bit alpha value.

`vImageConvert_RGB888toARGB8888` (page 40)
Converts an RGB888 image into an ARGB8888 image, using the provided alpha value (either as planar or pixel data).

`vImageConvert_RGB888toPlanar8` (page 41)
Separates an RGB888 image into three Planar8 images.

`vImageConvert_RGBFFFtoPlanarF` (page 42)
Separates an RGBFFF image into three PlanarF images.

Functions

vImageBufferFill_ARGB8888

Fills an ARGB8888 buffer with a specified color.

```
vImage_Error vImageBufferFill_ARGB8888 (
   const vImage_Buffer *dest,
   const Pixel_8888 color,
   vImage_Flags flags
);
```

Parameters

dest
A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data created with the fill color. When you no longer need the data buffer, you must deallocate the memory.

color
The color to fill the buffer with.

flags
Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability
Available in Mac OS X v10.4 and later.

See Also
`vImageOverwriteChannelsWithScalar_Planar8` (page 49)

Declared In
`Conversion.h`

vImageBufferFill_ARGBFFFF

Fills an ARGBFFFF buffer with a specified color.

```
vImage_Error vImageBufferFill_ARGBFFFF (
   const vImage_Buffer *dest,
   const Pixel_FFFF color,
   vImage_Flags flags
);
```

Parameters

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data created with the fill color. When you no longer need the data buffer, you must deallocate the memory.

color

The color to fill the buffer with.

flags

Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.4 and later.

See Also

`vImageOverwriteChannelsWithScalar_PlanarF` (page 49)

Declared In

`Conversion.h`

vImageClip_PlanarF

Clips the pixel values of an image in PlanarF format, using the provided minimum and maximum values.

```
vImage_Error vImageClip_PlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   Pixel_F maxFloat,
   Pixel_F minFloat,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to clip.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

maxFloat

A maximum pixel value. The function clips larger values to this value in the destination image.

minFloat

A maximum pixel value. The function clips smaller values to this value in the destination image.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_16SToF

Converts an image in a special planar format—in which each pixel value is a 16-bit signed integer— to a PlanarF format.

```
vImage_Error vImageConvert_16SToF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   float offset,
   float scale,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image (for which each pixel value is a 16-bit signed integer) whose data you want to overwrite.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data in PlanarF format. When you no longer need the data buffer, you must deallocate the memory.

offset

The offset value to add to each pixel.

scale

The value to multiply each pixel by.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The function changes each pixel value to a floating-point value, scales each value and then adds the offset value. The calculation is

```
resultPixel = (float) sourcePixel * scale + offset
```

The functions `vImageConvert_16SToF` and `vImageConvert_FTo16S` are inverse transformations when you use the same offset and scale values for each. (The inversion is not precise due to round-off error.) This requires the two functions to use these values differently (and in a different order).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Conversion.h`

vImageConvert_16UToF

Converts an image in a special planar format—in which each pixel value is a 16-bit unsigned integer— to a PlanarF format.

```
vImage_Error vImageConvert_16UToF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   float offset,
   float scale,
   vImage_Flags flags
);
```

Parameters

src
: A pointer to a vImage buffer structure that contains the source image (for which each pixel value is a 16-bit unsigned integer) whose data you want to overwrite.

dest
: A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data in PlanarF format. When you no longer need the data buffer, you must deallocate the memory.

offset
: The offset value to add to each pixel.

scale
: The value to multiply each pixel by.

flags
: The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
Each pixel value is changed to a floating-point value, then scaled and offset by the provided values. The calculation is

```
resultPixel =   SATURATED_CLIP_SHRT_MIN_to_SHRT_MAX( (sourcePixel
                - offset) / scale  + 0.5f)
```

The functions `vImageConvert_16SToF` and `vImageConvert_FTo16S` are inverse transformations when you use the same offset and scale values for each. (The inversion is not precise due to round-off error.) This requires the two functions to use these values differently (and in a different order).

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Conversion.h`

vImageConvert_16UToPlanar8

Converts an image in a special planar format—in which each pixel value is a 16-bit unsigned integer—to a Planar8 image.

```
vImage_Error vImageConvert_16UToPlanar8 (
const vImage_Buffer *src,
const vImage_Buffer *dest,
vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The conversion from 16-bit to 8-bit values is:

```
uint8_t result = (srcPixel * 255 + 32767) / 65535
```

You can also use this function to convert a 4-channel interleaved 16U image to an ARGB8888. image. Simply multiply the width of the destination buffer by four.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageConvert_ARGB1555toARGB8888

Converts an ARGB1555 image to an ARGB8888 image.

```
vImage_Error vImageConvert_ARGB1555toARGB8888 (
const vImage_Buffer *src,
const vImage_Buffer *dest,
vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the converted data. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The ARGB1555 format has 16-bit pixels with 1 bit for alpha and 5 bits each for red, green, and blue. The function calculates the 8-bit pixels in the destination image as follows:

```
Pixel8 alpha =  1bitAlphaChannel * 255
Pixel8 red   = (5bitRedChannel   * 255 + 15) / 31
Pixel8 green = (5bitGreenChannel * 255 + 15) / 31
Pixel8 blue  = (5bitBlueChannel  * 255 + 15) / 31
```

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageConvert_ARGB1555toPlanar8

Separates an ARGB1555 image into four Planar8 images.

```
vImage_Error vImageConvert_ARGB1555toPlanar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *destA,
   const vImage_Buffer *destR,
   const vImage_Buffer *destG,
   const vImage_Buffer *destB,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to separate.

destA

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the alpha channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destR

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the red channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destG

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the green channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destB

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the blue channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The ARGB1555 format has 16-bit pixels with 1 bit for alpha and 5 bits each for red, green, and blue. The function calculates the 8-bit pixels in the destination image as follows:

```
Pixel8 alpha =  1bitAlphaChannel * 255
Pixel8 red   = (5bitRedChannel   * 255 + 15) / 31
Pixel8 green = (5bitGreenChannel * 255 + 15) / 31
Pixel8 blue  = (5bitBlueChannel  * 255 + 15) / 31
```

This function works in place for one destination buffer; the others must be allocated separately.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageConvert_ARGB8888toARGB1555

Converts an ARGB8888 image into an ARGB1555 image.

```
vImage_Error vImageConvert_ARGB8888toARGB1555 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the converted image data. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The ARGB1555 format has 16-bit pixels with 1 bit for alpha and 5 bits each for red, green, and blue. The function calculates the 16-bit pixels in the destination image as follows:

```
uint32_t alpha = (8bitAlphaChannel      + 127) / 255
uint32_t red   = (8bitRedChannel   * 31 + 127) / 255
uint32_t green = (8bitGreenChannel * 31 + 127) / 255
uint32_t blue  = (8bitBlueChannel  * 31 + 127) / 255
uint16_t ARGB1555pixel = (alpha << 15) | (red <<  10) | (green << 5) | blue
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageConvert_ARGB8888toPlanar8

Separates an ARGB8888 image into four Planar8 images.

```
vImage_Error vImageConvert_ARGB8888toPlanar8 (
   const vImage_Buffer *srcARGB,
   const vImage_Buffer *destA,
   const vImage_Buffer *destR,
   const vImage_Buffer *destG,
   const vImage_Buffer *destB,
   vImage_Flags flags
);
```

Parameters

srcARGB

A pointer to a vImage buffer structure that contains the source image whose data you want to separate.

destA

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the alpha channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destR

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the red channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destG

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the green channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destB

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the blue channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The source image, and the *destA*, *destR*, *destG*, and *destB* destination buffers, must have the same height and the same width. This function works in place for one destination buffer. The others must be allocated separately.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_ARGB8888toRGB565

Converts an ARGB8888 image into an RGB565 image.

```
vImage_Error vImageConvert_ARGB8888toRGB565 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the data in RGB565 format. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The alpha channel in the ARGB8888 image is ignored. (The RGB565 format has 16-bit pixels with 5 bits for red, 6 for green, and 5 for blue.) The function calculates the pixels in the destination image as follows:

```
    uint32_t red   = (8bitRedChannel   * (31*2) + 255) / (255*2)
    uint32_t green = (8bitGreenChannel * 63 + 127) / 255
    uint32_t blue  = (8bitBlueChannel  * 31 + 127) / 255
    uint16_t RGB565pixel =  (red << 11) | (green <<  5) | blue
```

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageConvert_ARGB8888toRGB888

Converts an ARGB8888 image into an RGB888 image.

```
vImage_Error vImageConvert_ARGB8888toRGB888 (
   const vImage_Buffer *argbSrc,
   const vImage_Buffer *rgbDest,
   vImage_Flags flags
);
```

Parameters

argbSrc

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

rgbDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `argbSrc` data buffer. On return, the data buffer pointed to by this structure contains the data in RGB888 format. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The red, green, and blue channels are simply copied.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageConvert_ARGBFFFFtoPlanarF

Separates an ARGBFFFF image into four PlanarF images.

```
vImage_Error vImageConvert_ARGBFFFFtoPlanarF (
   const vImage_Buffer *srcARGB,
   const vImage_Buffer *destA,
   const vImage_Buffer *destR,
   const vImage_Buffer *destG,
   const vImage_Buffer *destB,
   vImage_Flags flags
);
```

Parameters

srcARGB

A pointer to a vImage buffer structure that contains the source image whose data you want to separate.

destA

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a PlanarF image equivalent to the alpha channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destR

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a PlanarF image equivalent to the red channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destG

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a PlanarF image equivalent to the green channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destB

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a PlanarF image equivalent to the blue channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The source image, and the *destA*, *destR*, *destG*, and *destB* destination buffers, must have the same height and the same width. This function works in place for one destination buffer. The others must be allocated separately.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_ChunkyToPlanar8

Separates a source image into a collection of corresponding planar destination images, one for each 8-bit channel of the original image.

```
vImage_Error vImageConvert_ChunkyToPlanar8 (
   const void *srcChannels[],
   const vImage_Buffer *destPlanarBuffers[],
   unsigned int channelCount,
   size_t srcStrideBytes,
   vImagePixelCount srcWidth,
   vImagePixelCount srcHeight,
   size_t srcRowBytes,
   vImage_Flags flags
);
```

Parameters

srcChannels

An array of pointers to channels of the source image. Each pointer points to the start of the data for one source channel.

destPlanarBuffers

An array of vImage buffer structures, each of which contains image data in Planar8 format. Each structure must have the same width and height values, but may have different row byte values. On return, the data buffer in each vImage buffer structure contains a planar image equivalent to the corresponding channel of the source image.

channelCount

The number of channels in the source image.

srcStrideBytes

The number of bytes from one pixel value in a given channel to the next pixel of that channel (within a row). This value must be the same for all channels.

srcWidth

The number of pixels in a row. This value must be the same for all channels in the source image, and for all the destination buffers.

srcHeight

The number of rows. This value must be the same for all channels in the source image, and for all the destination buffers.

srcRowBytes

The number of bytes from the beginning of a channel row to the beginning of the next row of the channel. This value must be the same for all channels of the source image. (It does not have the be the same as the *rowBytes* values of the destination buffers. Each destination buffer can have its own *rowBytes* value.)

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_ChunkyToPlanarF

Separates a source image into a collection of corresponding planar destination images, one for each floating-point channel of the original image.

```
vImage_Error vImageConvert_ChunkyToPlanarF (
   const void *srcChannels[],
   const vImage_Buffer *destPlanarBuffers[],
   unsigned int channelCount,
   size_t srcStrideBytes,
   vImagePixelCount srcWidth,
   vImagePixelCount srcHeight,
   size_t srcRowBytes,
   vImage_Flags flags
);
```

Parameters

srcChannels

An array of pointers to channels of the source image. Each pointer points to the start of the data for one source channel.

destPlanarBuffers

An array of vImage buffer structures, each of which contains image data in PlanarF format. Each structure must have the same width and height values, but may have different row byte values. On return, the data buffer in each each vImage buffer structure contains a planar image equivalent to the corresponding channel of the source image.

channelCount

The number of channels in the source image.

srcStrideBytes

The number of bytes from one pixel value in a given channel to the next pixel of that channel (within a row). This value must be the same for all channels.

srcWidth

The number of pixels in a row. This value must be the same for all channels in the source image, and for all the destination buffers.

srcHeight

The number of rows. This value must be the same for all channels in the source image, and for all the destination buffers.

srcRowBytes

The number of bytes from the beginning of a channel row to the beginning of the next row of the channel. This value must be the same for all channels of the source image. (It does not have the be the same as the *rowBytes* values of the destination buffers. Each destination buffer can have its own *rowBytes* value.)

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_FTo16S

Converts a PlanarF image into a special format in which each pixel is a 16-bit signed integer.

```
vImage_Error vImageConvert_FTo16S (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   float offset,
   float scale,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the destination image converted to 16-bit signed integer format. When you no longer need the data buffer, you must deallocate the memory.

offset

The offset value to subtract from every pixel.

scale

The scale value to divide each pixel by.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

Each pixel value is first offset and scaled by the provided values, and then changed to a 16-bit signed integer (rounded and clipped as necessary). The calculation is as follows:

```
resultPixel =   SATURATED_CLIP_0_to_USHRT_MAX( (srcPixel - offset)
                / scale  + 0.5f)
```

The functions `vImageConvert_16SToF` and `vImageConvert_FTo16S` are inverse transformations when you use the same offset and scale values for each. (The inversion is not precise due to round-off error.) This requires the two functions to use these values differently (and in a different order).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_FTo16U

Converts a PlanarF image into a special format in which each pixel is a 16-bit unsigned integer.

```
vImage_Error vImageConvert_FTo16U (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   float offset,
   float scale,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the destination image converted to 16-bit unsigned integer format. When you no longer need the data buffer, you must deallocate the memory.

offset

The offset value to subtract from every pixel.

scale

The scale value to divide each pixel by.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

Each pixel value is first offset and scaled by user-supplied values, and then changed to a 16-bit unsigned integer (rounded and clipped as necessary). The calculation is as follows:

```
resultPixel =   SATURATED_CLIP_0_to_USHRT_MAX( (sourcePixel  - offset)
                / scale  + 0.5f)
```

The functions `vImageConvert_16UToF` and `vImageConvert_FTo16U` are inverse transformations when you use the same offset and scale values for each. (The inversion is not precise due to round-off error.) This requires the two functions to use these values differently (and in a different order).

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_Planar16FtoPlanarF

Converts a Planar16F image to a PlanarF image.

```
vImage_Error vImageConvert_Planar16FtoPlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image converted to PlanarF format. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The Planar16F format is identical to the OpenEXR format; it uses 16-bit floating-point numbers. In conformance with IEEE-754, the function quiets signaling NaNs during the conversion. (OpenEXR-1.2.1 does not do this.)

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageConvert_Planar8To16U

Converts a Planar8 image to a 16U image .

```
vImage_Error vImageConvert_Planar8To16U (
const vImage_Buffer *src,
const vImage_Buffer *dest,
vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The function converts from 8-bit to 16-bit values as follows:

```
uint16_t result = (srcPixel * 65535 + 127 ) / 255
```

You can also use this function to convert an ARGB8888 image to a 4-channel interleaved 16U image. Simply multiply the width of the destination buffer by four.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageConvert_Planar8toARGB1555

Combines four Planar8 images into one ARGB1555 image.

```
vImage_Error vImageConvert_Planar8toARGB1555 (
   const vImage_Buffer *srcA,
   const vImage_Buffer *srcR,
   const vImage_Buffer *srcG,
   const vImage_Buffer *srcB,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

srcA

A pointer to vImage buffer structure that contains the Planar8 image to use as the alpha channel of the destination image.

srcR

A pointer to vImage buffer structure that contains the Planar8 image to use as the red channel of the destination image.

srcG

A pointer to vImage buffer structure that contains the Planar8 image to use as the green channel of the destination image.

srcB

A pointer to vImage buffer structure that contains the Planar8 image to use as the blue channel of the destination image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data in ARGB1555 format (16-bit pixels with 1 bit for alpha and 5 bits each for red, green, and blue). When you no longer need the data buffer, you must deallocate the memory. The destination buffer can be the same as the source buffer.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
This function calculates the 8-bit pixels in the destination image as follows:

```
Pixel8 alpha =  1bitAlphaChannel * 255
Pixel8 red   = (5bitRedChannel   * 255 + 15) / 31
Pixel8 green = (5bitGreenChannel * 255 + 15) / 31
Pixel8 blue  = (5bitBlueChannel  * 255 + 15) / 31
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageConvert_Planar8toARGB8888

Combines four Planar8 images into one ARGB8888 image.

```
vImage_Error vImageConvert_Planar8toARGB8888 (
   const vImage_Buffer *srcA,
   const vImage_Buffer *srcR,
   const vImage_Buffer *srcG,
   const vImage_Buffer *srcB,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

srcA

A pointer to vImage buffer structure that contains the Planar8 image to use as the alpha channel of the destination image.

srcR

A pointer to vImage buffer structure that contains the Planar8 image to use as the red channel of the destination image.

srcG

A pointer to vImage buffer structure that contains the Planar8 image to use as the green channel of the destination image.

srcB

A pointer to vImage buffer structure that contains the Planar8 image to use as the blue channel of the destination image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image in ARGB8888 format. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The source and destination buffers must have the same height and width.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Conversion.h`

vImageConvert_Planar8toPlanarF

Converts a Planar8 image to a PlanarF image.

```
vImage_Error vImageConvert_Planar8toPlanarF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   Pixel_F maxFloat,
   Pixel_F minFloat,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image converted to PlanarF format. When you no longer need the data buffer, you must deallocate the memory.

maxFloat

The maximum pixel value for the destination image.

minFloat

The minimum pixel value for the destination image.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
This function transforms a Planar8 image to a PlanarF image, using a *minFloat* value and a *maxFloat* value to specify the range of values for the PlanarF image. The function maps each source pixel value(which can be in the range of 0 to 255 inclusive) linearly into the range *minFloat* to *maxFloat*, using the following mapping (where `i` is the old pixel value):

```
new pixel value = i* (maxFloat - minFloat)/255.0f + minFloat
```

The two buffers must have the same number of rows and the same number of columns.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Conversion.h`

vImageConvert_Planar8toRGB565

Combines three Planar8 images into one RGB565 image.

```
vImage_Error vImageConvert_Planar8toRGB565 (
   const vImage_Buffer *srcR,
   const vImage_Buffer *srcG,
   const vImage_Buffer *srcB,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

srcR

A pointer to vImage buffer structure that contains the Planar8 image to use as the red channel of the destination image.

srcG

A pointer to vImage buffer structure that contains the Planar8 image to use as the green channel of the destination image.

srcB

A pointer to vImage buffer structure that contains the Planar8 image to use as the blue channel of the destination image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image in RGB565 format (16-bit pixels with 5 bits for red, 6 for green, and 5 for blue). When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The function calculates the pixels in the destination image as follows:

```
    uint32_t red   = (8bitRedChannel   * (31*2) + 255) / (255*2)
    uint32_t green = (8bitGreenChannel * 63 + 127) / 255
    uint32_t blue  = (8bitBlueChannel  * 31 + 127) / 255
    uint16_t RGB565pixel =  (red << 11) | (green <<  5) | blue
```

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageConvert_Planar8toRGB888

Combines three Planar8 images into one RGB888 image.

```
vImage_Error vImageConvert_Planar8toRGB888 (
   const vImage_Buffer *planarRed,
   const vImage_Buffer *planarGreen,
   const vImage_Buffer *planarBlue,
   const vImage_Buffer *rgbDest,
   vImage_Flags flags
);
```

Parameters

planarRed

A pointer to vImage buffer structure that contains the Planar8 image to use as the red channel of the destination image.

planarGreen

A pointer to vImage buffer structure that contains the Planar8 image to use as the green channel of the destination image.

planarBlue

A pointer to vImage buffer structure that contains the Planar8 image to use as the blue channel of the destination image.

rgbDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image in RGB888 format. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The source and destination buffers must have the same height and width.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageConvert_PlanarFtoARGBFFFF

Combines four PlanarF images into one ARGBFFFF image.

```
vImage_Error vImageConvert_PlanarFtoARGBFFFF (
   const vImage_Buffer *srcA,
   const vImage_Buffer *srcR,
   const vImage_Buffer *srcG,
   const vImage_Buffer *srcB,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

srcA

A pointer to vImage buffer structure that contains the PlanarF image to use as the alpha channel of the destination image.

srcR

A pointer to vImage buffer structure that contains the PlanarF image to use as the red channel of the destination image.

srcG

A pointer to vImage buffer structure that contains the PlanarF image to use as the green channel of the destination image.

srcB

A pointer to vImage buffer structure that contains the PlanarF image to use as the blue channel of the destination image.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data in ARGBFFFF format. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The source and destination buffers must have the same height and width.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_PlanarFtoPlanar16F

Converts a PlanarF image to a Planar16F image.

```
vImage_Error vImageConvert_PlanarFtoPlanar16F (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the destination image converted to Planar16F format. The destination buffer can be the same as the source buffer. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

Planar16F pixels are 16-bit floating-point numbers, conforming to the OpenEXR standard. Denormals, NaNs. and +/– Infinity are supported. In conformance with IEEE-754, all signaling NaNs are quieted during the conversion (OpenEXR-1.2.1 does not do this.)

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageConvert_PlanarFtoPlanar8

Converts a PlanarF image to a Planar8 image, clipping values to the provided minimum and maximum values.

```
vImage_Error vImageConvert_PlanarFtoPlanar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   Pixel_F maxFloat,
   Pixel_F minFloat,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the destination image converted to Planar8 format. When you no longer need the data buffer, you must deallocate the memory.

maxFloat

A maximum pixel value. The function clips larger values to this value in the destination image.

maxFloat

A minimum pixel value. The function clips smaller values to this value in the destination image.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The minimum and maximum value determine the mapping of intensity values to the destination image. The mapping is:

```
if oldPixel < minFloat
newPixel = 0

if minfloat <= oldPixel <= maxFloat
newPixel = (oldPixel - minFloat) * 255.0f / (maxFloat - minFloat)

if oldPixel > maxFloat
newPixel = 255
```

The source and destination buffers must have the same height and width.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Conversion.h`

vImageConvert_PlanarFtoRGBFFF

Combines three PlanarF images into one RGBFFF image.

```
vImage_Error vImageConvert_PlanarFtoRGBFFF (
   const vImage_Buffer *planarRed,
   const vImage_Buffer *planarGreen,
   const vImage_Buffer *planarBlue,
   const vImage_Buffer *rgbDest,
   vImage_Flags flags
);
```

Parameters

planarRed

A pointer to vImage buffer structure that contains the PlanarF image to use as the red channel of the destination image.

planarGreen

A pointer to vImage buffer structure that contains the PlanarF image to use as the green channel of the destination image.

planarBlue

A pointer to vImage buffer structure that contains the PlanarF image to use as the blue channel of the destination image.

rgbDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image in RGBFFF format. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The source and destination buffers must have the same height and width.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageConvert_PlanarToChunky8

Combines a collection of planar source images into a single interleaved destination image, with one 8-bit channel for each planar image.

```
vImage_Error vImageConvert_PlanarToChunky8 (
   const vImage_Buffer *srcPlanarBuffers[],
   void *destChannels[],
   unsigned int channelCount,
   size_t destStrideBytes,
   vImagePixelCount destWidth,
   vImagePixelCount destHeight,
   size_t destRowBytes,
   vImage_Flags flags
);
```

Parameters

srcPlanarBuffers

An array of vImage buffer structures, each of which contains image data in Planar8 format. Each structure must have the same width and height values, but may have different row byte values.

destChannels

An array of pointers to channels of the destination image. Each pointer points to the start of the data for one destination channel. The function fills the pixel values of each channel, using the corresponding source image for each channel.

channelCount

The number of vImage buffer structures in the *srcPlanarBuffers* array and the number of channels in the destination image.

destStrideBytes

The number of bytes from one pixel value in a given channel to the next pixel of that channel (within a row). This value is used for all channels.

destWidth

The number of pixels in a row. This value is used for all channels. It must be the same as the width of each of the planar source images.

destHeight

The number of rows. This value will be used for all channels. It must be the same as the height of each of the planar source images.

destRowBytes

The number of bytes from the beginning of a channel row to the beginning of the next row in that channel. This value is used for all channels. (It does not have to be the same as the *rowBytes* values of the source buffers. Each source buffer can have its own *rowBytes* value.)

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_PlanarToChunkyF

Combines a collection of planar source images into a single interleaved destination image, with one floating-point channel for each planar image.

```
vImage_Error vImageConvert_PlanarToChunkyF (
   const vImage_Buffer *srcPlanarBuffers[],
   void *destChannels[],
   unsigned int channelCount,
   size_t destStrideBytes,
   vImagePixelCount destWidth,
   vImagePixelCount destHeight,
   size_t destRowBytes,
   vImage_Flags flags
);
```

Parameters

srcPlanarBuffers

An array of vImage buffer structures, each of which contains image data in PlanarF format. Each structure must have the same width and height values, but may have different row byte values.

destChannels

An array of pointers to channels of the destination image. Each pointer points to the start of the data for one destination channel. The function fills the pixel values of each channel, using the corresponding source image for each channel.

channelCount

The number of vImage buffer structures in the *srcPlanarBuffers* array and the number of channels in the destination image.

destStrideBytes

The number of bytes from one pixel value in a given channel to the next pixel of that channel (within a row). This value is used for all channels.

destWidth

The number of pixels in a row. This value is used for all channels. It must be the same as the width of each of the planar source images.

destHeight

The number of rows. This value is used for all channels. It must be the same as the height of each of the planar source images.

destRowBytes

The number of bytes from the beginning of a channel row to the beginning of the next row in that channel. This value is used for all channels. (It does not have to be the same as the *rowBytes* values of the source buffers. Each source buffer can have its own *rowBytes* value.)

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.3 and later.

Declared In

`Conversion.h`

vImageConvert_RGB565toARGB8888

Converts an RGB565 image into an ARGB8888 image, using the provided 8-bit alpha value.

```
vImage_Error vImageConvert_RGB565toARGB8888 (
   Pixel_8 alpha,
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

alpha

A value of type `Pixel_8` to be used as the alpha value for all pixels in the destination image.

src

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the destination data converted to ARGB8888 format.. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The RGB565 format has 16-bit pixels with 5 bits for red, 6 for green, and 5 for blue. The function calculates the pixels in the destination image as follows:

```
Pixel8 alpha = alpha
Pixel8 red   = (5bitRedChannel   * 255 + 15) / 31
Pixel8 green = (6bitGreenChannel * 255 + 31) / 63
Pixel8 blue  = (5bitBlueChannel  * 255 + 15) / 31
```

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageConvert_RGB565toPlanar8

Separates an RGB565 image into three Planar8 images.

```
vImage_Error vImageConvert_RGB565toPlanar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *destR,
   const vImage_Buffer *destG,
   const vImage_Buffer *destB,
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to separate.

destR

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the red channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destG

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the green channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

destB

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the blue channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The RGB565 format has 16-bit pixels with 5 bits for red, 6 for green, and 5 for blue. The function calculates the pixels in the destination image as follows:

```
Pixel8 red   = (5bitRedChannel   * 255 + 15) / 31
Pixel8 green = (6bitGreenChannel * 255 + 31) / 63
Pixel8 blue  = (5bitBlueChannel  * 255 + 15) / 31
```

This function works in place for one destination buffer. You must allocate the others separately.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageConvert_RGB888toARGB8888

Converts an RGB888 image into an ARGB8888 image, using the provided alpha value (either as planar or pixel data).

```
vImage_Error vImageConvert_RGB888toARGB8888 (
   const vImage_Buffer *rgbSrc,
   const vImage_Buffer *aSrc,
   Pixel_8 alpha,
   const vImage_Buffer *argbDest,
   bool premultiply,
   vImage_Flags flags
);
```

Parameters

rgbSrc

A pointer to a vImage buffer structure that contains the source image whose data you want to convert.

aSrc

A pointer to a vImage buffer structure that contains a Planar8 alpha plane to use as the alpha values for in the destination image. If you pass `NULL`, the function assigns the value of the `alpha` parameter for all pixels in the destination image.

alpha

An alpha value for all pixels in the destination image. The function ignores this value if the `aSrc` parameter is not `NULL`.

argbDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the converted image data. When you no longer need the data buffer, you must deallocate the memory.

premultiply

Pass `YES` if the data is premultiplied by the alpha value; `NO` otherwise.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

If you specify premultiplied data, the function calculates each channel in the destination image as follows:

```
(alpha * sourceValue + 127) / 255
```

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageConvert_RGB888toPlanar8

Separates an RGB888 image into three Planar8 images.

```
vImage_Error vImageConvert_RGB888toPlanar8 (
   const vImage_Buffer *rgbSrc,
   const vImage_Buffer *redDest,
   const vImage_Buffer *greenDest,
   const vImage_Buffer *blueDest,
   vImage_Flags flags
);
```

Parameters

rgbSrc

A pointer to a vImage buffer structure that contains the source image whose data you want to separate.

redDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the red channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

greenDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the green channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

blueDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a Planar8 image equivalent to the blue channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The source image and the destination buffers, must all have the same height and the same width. This function works in place for one destination buffer. You must allocate the others separately.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageConvert_RGBFFFtoPlanarF

Separates an RGBFFF image into three PlanarF images.

```
vImage_Error vImageConvert_RGBFFFtoPlanarF (
   const vImage_Buffer *rgbSrc,
   const vImage_Buffer *redDest,
   const vImage_Buffer *greenDest,
   const vImage_Buffer *blueDest,
   vImage_Flags flags
);
```

Parameters

rgbSrc

A pointer to a vImage buffer structure that contains the source image whose data you want to separate.

redDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a PlanarF image equivalent to the red channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

greenDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a PlanarF image equivalent to the green channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

blueDest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains data for a PlanarF image equivalent to the blue channel of the source image. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The source image and the destination buffers, must all have the same height and the same width. This function works in place for one destination buffer. You must allocated the others separately.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageFlatten_ARGB8888ToRGB888

Transforms an ARGB8888 image to an RGB888 image against an opaque background of the provided color.

```
vImage_Error vImageFlatten_ARGB8888ToRGB888 (
const vImage_Buffer *argb8888Src,
const vImage_Buffer *rgb888dest,
Pixel_8888 backgroundColor,
bool isImagePremultiplied,
vImage_Flags flags
);
```

Parameters

argb8888Src

A pointer to a vImage buffer structure that contains the source image whose data you want to flatten.

rgb888dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `argb8888Src` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory. The destination buffer can be the same as the source buffer.

backgroundColor

An 8-bit interleaved pixel value.

isImagePremultiplied

`TRUE` if the source image uses premultiplied data , `FALSE` otherwise.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

If the source image uses premultiplied data, the function calculates each channel value for a pixel in the destination image as follows (where i is the source value for the channel):

```
new value = (i * 255 + (255 - alpha) * backgroundColor + 127)  / 255
```

If the source image does not use premultiplied data, the function calculates each channel value for a pixel in the destination image as follows (where i is the source value for the channel):

```
new value = (i * alpha + (255 - alpha) * backgroundColor + 127)  / 255
```

The source and destinations buffer must have the same height and the same width.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageFlatten_ARGBFFFFToRGBFFF

Transforms an ARGBFFFF image to an RGBFFF image against an opaque background of the provided color.

```
vImage_Error vImageFlatten_ARGBFFFFToRGBFFF (
const vImage_Buffer *argbFFFFSrc,
const vImage_Buffer *rgbFFFdest,
Pixel_FFFF backgroundColor,
bool isImagePremultiplied,
vImage_Flags flags
);
```

Parameters

argbFFFFSrc

A pointer to a vImage buffer structure that contains the source image whose data you want to flatten.

rgbFFFdest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `argbFFFFSrc` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory. The destination buffer can be the same as the source buffer.

backgroundColor

A floating-point interleaved pixel value.

isImagePremultiplied

True if the source image is premultiplied, false otherwise.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

If the source image uses premultiplied data, the function calculates each channel value for a pixel in the destination image as follows (where i is the source value for the channel):

```
newcolor = i + (1.0f - alpha) * backgroundColor
```

If the source image does not use premultiplied data, the function calculates each channel value for a pixel in the destination image as follows (where i is the source value for the channel):

```
newcolor = i * alpha + (1.0f - alpha) * backgroundColor
```

The source and destinations buffer must have the same height and the same width.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageOverwriteChannelsWithPixel_ARGB8888

Overwrites an ARGB8888 image buffer with the provided pixel value.

```
vImage_Error vImageOverwriteChannelsWithPixel_ARGB8888 (
   const Pixel_8888 the_pixel,
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   uint8_t copyMask,
   vImage_Flags flags
);
```

Parameters

the_pixel

An ARGB pixel value.

src

A pointer to a vImage buffer structure that contains the source image whose data you want to overwrite.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `origSrc` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

copyMask

An output value that selects the plane (or planes) from the ARGB8888 source buffer that you want replaced with the pixel value. The value `0x8` selects the alpha channel, `0x4` the red channel, `0x2` the green channel, and `0x1` the blue channel. You can add these values together to select multiple channels.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`Conversion.h`

vImageOverwriteChannelsWithPixel_ARGBFFFF

Overwrites an ARGBFFFF image buffer with the provided pixel value.

```
vImage_Error vImageOverwriteChannelsWithPixel_ARGBFFFF (
   const Pixel_FFFF the_pixel,
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   uint8_t copyMask,
   vImage_Flags flags
);
```

Parameters

the_pixel

An ARGB pixel value.

src

A pointer to a vImage buffer structure that contains the source image whose data you want to overwrite.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `origSrc` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

copyMask

An output value that selects the plane (or planes) from the ARGBFFFF source buffer that you want replaced with the pixel value. The value `0x8` selects the alpha channel, `0x4` the red channel, `0x2` the green channel, and `0x1` the blue channel. You can add these values together to select multiple channels.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`Conversion.h`

vImageOverwriteChannelsWithScalar_ARGB8888

Overwrites the pixels of one or more planes of an ARGB8888 image buffer with the provided scalar value.

```
vImage_Error vImageOverwriteChannelsWithScalar_ARGB8888 (
   Pixel_8 scalar,
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   uint8_t copyMask,
   vImage_Flags flags
);
```

Parameters

scalar

An 8-bit pixel value.

src

A pointer to a vImage buffer structure that contains the source image whose data you want to overwrite.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `origSrc` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

copyMask

An output value that selects the plane (or planes) from the ARGB8888 source buffer that you want replaced with the scalar value. The value `0x8` selects the alpha channel, `0x4` the red channel, `0x2` the green channel, and `0x1` the blue channel. You can add these values together to select multiple channels.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageOverwriteChannelsWithScalar_ARGBFFFF

Overwrites the pixels of one or more planes of an ARGBFFFF image buffer with the provided scalar value.

```
vImage_Error vImageOverwriteChannelsWithScalar_ARGBFFFF (
   Pixel_F scalar,
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   uint8_t copyMask,
   vImage_Flags flags
);
```

Parameters

scalar

A floating-point pixel value.

src

A pointer to a vImage buffer structure that contains the source image whose data you want to overwrite.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `origSrc` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

copyMask

An output value that selects the plane (or planes) from the ARGBFFFF source buffer that you want replaced with the scalar value. The value `0x8` selects the alpha channel, `0x4` the red channel, `0x2` the green channel, and `0x1` the blue channel. You can add these values together to select multiple channels.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageOverwriteChannelsWithScalar_Planar8

Overwrites a Planar8 image buffer with the provided value.

```
vImage_Error vImageOverwriteChannelsWithScalar_Planar8 (
   Pixel_8 scalar,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

scalar

An 8-bit pixel value.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination data. When you no longer need the data buffer, you must deallocate the memory.

flags

Set the `kvImageDoNotTile` field in the flags parameter to prevent vImage from using tiling internally. (This is appropriate if you are doing tiling yourself.)

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`Conversion.h`

vImageOverwriteChannelsWithScalar_PlanarF

Overwrites a PlanarF image buffer with the provided value.

```
vImage_Error vImageOverwriteChannelsWithScalar_PlanarF (
   Pixel_F scalar,
   const vImage_Buffer *dest,
   vImage_Flags flags
);
```

Parameters

scalar

A floating-point pixel value.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageOverwriteChannels_ARGB8888

Overwrites one or more planes of an ARGB8888 image buffer with the provided planar buffer.

```
vImage_Error vImageOverwriteChannels_ARGB8888 (
   const vImage_Buffer *newSrc,
   const vImage_Buffer *origSrc,
   const vImage_Buffer *dest,
   uint8_t copyMask,
   vImage_Flags flags
);
```

Parameters

`newSrc`

A pointer to a vImage buffer structure that contains the data, in Planar8 format, for overwriting the `origSrc` image data.

`origSrc`

A pointer to a vImage buffer structure that contains the source image whose data you want to overwrite.

`dest`

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `origSrc` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

`copyMask`

An output value that selects the plane (or planes) from the ARGB8888 source buffer that you want replaced with data from the Planar8 source buffer. The value `0x8` selects the alpha channel, `0x4` the red channel, `0x2` the green channel, and `0x1` the blue channel. You can add these values together to select multiple channels.

`flags`

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The function overwrites pixel values in the `origSrc` image buffer using the corresponding pixel value from the `newSrc` image buffer.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageOverwriteChannels_ARGBFFFF

Overwrites one or more planes of an ARGBFFFF image buffer with the provided planar buffer.

```
vImage_Error vImageOverwriteChannels_ARGBFFFF (
   const vImage_Buffer *newSrc,
   const vImage_Buffer *origSrc,
   const vImage_Buffer *dest,
   uint8_t copyMask,
   vImage_Flags flags
);
```

Parameters

newSrc

A pointer to a vImage buffer structure that contains the data, in PlanarF format, for overwriting the `origSrc` image data.

origSrc

A pointer to a vImage buffer structure that contains the source image whose data you want to overwrite.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `origSrc` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

copyMask

An output value that selects the plane (or planes) from the ARGBFFFF source buffer that you want replaced with data from the Planar8 source buffer. The value `0x8` selects the alpha channel, `0x4` the red channel, `0x2` the green channel, and `0x1` the blue channel. You can add these values together to select multiple channels.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
The function overwrites pixel values in the `origSrc` image buffer using the corresponding pixel value from the `newSrc` image buffer.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImagePermuteChannels_ARGB8888

Reorders the channels in an ARGB8888 image.

```
vImage_Error vImagePermuteChannels_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   const uint8_t permuteMap[4],
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to permute.

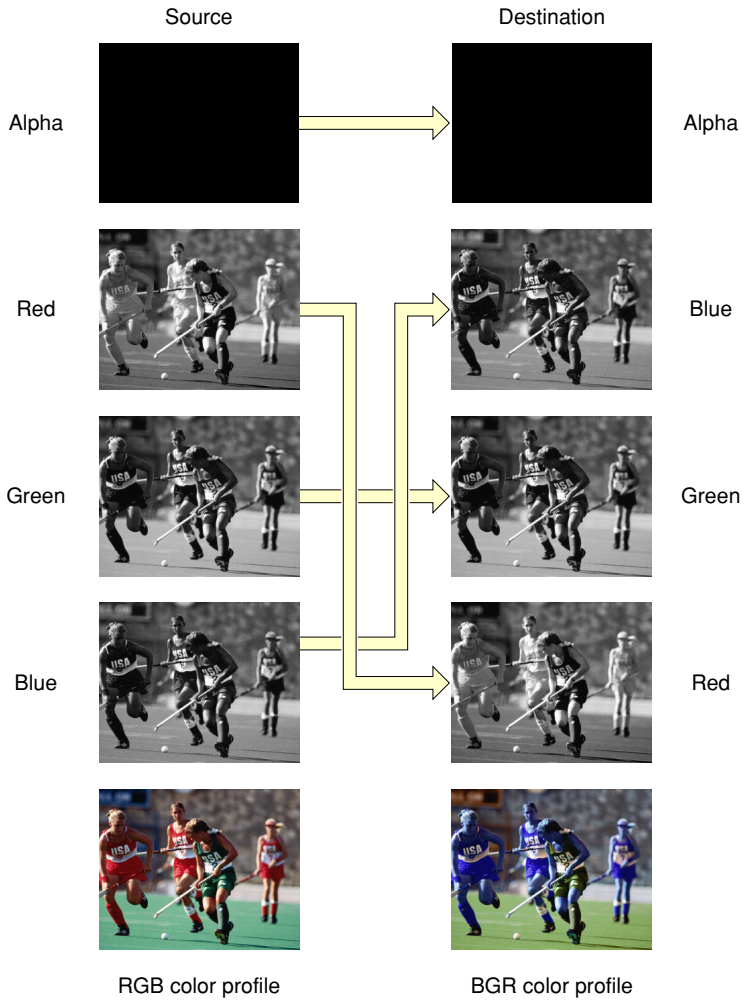
dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

permuteMap

An array of four 8-bit integers with the values 0, 1, 2, and 3, in some order. The ith value specifies the plane from the source image that you want copied to the ith plane of the destination image. 0 denotes the alpha channel, 1 the red channel, 2 the green channel, and 3 the blue channel. The following figure shows the result of using a permute map shows values are (0, 3, 2, 1). The data in the alpha and green channels remain the same, but the data in the source red channel maps to the destination blue channel while the data in the source blue channel maps to the destination red channel.

Figure 1 Permuting the red and blue channels



flags

The options to use when performing the permutation operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImagePermuteChannels_ARGBFFFF

Reorders the channels in an ARGBFFFF image.

```
vImage_Error vImagePermuteChannels_ARGBFFFF (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   const uint8_t permuteMap[4],
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to permute.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

permuteMap

An array of four 8-bit integers with the values 0, 1, 2, and 3, in some order. The ith value specifies the plane from the source image that will be copied to the ith plane of the destination image. 0 denotes the alpha channel, 1 the red channel, 2 the green channel, and 3 the blue channel. Figure 1 (page 53) shows the result of using a permute map shows values are (0, 3, 2, 1). The data in the alpha and green channels remain the same, but the data in the source red channel maps to the destination blue channel while the data in the source blue channel maps to the destination red channel.

flags

The options to use when performing the permutation operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability
Available in Mac OS X v10.4 and later.

Declared In
`Conversion.h`

vImageSelectChannels_ARGB8888

Overwrites the specified channels in an ARGB8888 image buffer with the provided channels from an ARGB8888 image buffer.

```
vImage_Error vImageSelectChannels_ARGB8888 (
   const vImage_Buffer *newSrc,
   const vImage_Buffer *origSrc,
   const vImage_Buffer *dest,
   uint8_t copyMask,
   vImage_Flags flags
);
```

Parameters

newSrc

A pointer to a vImage buffer structure that contains the data, in ARGB8888 format, for overwriting the `origSrc` image data.

origSrc

A pointer to a vImage buffer structure that contains the source image whose data you want to overwrite.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `origSrc` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

copyMask

An output value that selects the plane (or planes) from the ARGB8888 source buffer that you want replaced with the corresponding plane from the `newSrc` image buffer. The value `0x8` selects the alpha channel, `0x4` the red channel, `0x2` the green channel, and `0x1` the blue channel. You can add these values together to select multiple channels.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.5 and later.

See Also

`vImageOverwriteChannels_ARGB8888` (page 50)

Declared In

`Conversion.h`

vImageSelectChannels_ARGBFFFF

Overwrites the specified channels in an ARGBFFFF image buffer with the provided channels in an ARGBFFFF image buffer.

```
vImage_Error vImageSelectChannels_ARGBFFFF (
   const vImage_Buffer *newSrc,
   const vImage_Buffer *origSrc,
   const vImage_Buffer *dest,
   uint8_t copyMask,
   vImage_Flags flags
);
```

Parameters

newSrc

A pointer to a vImage buffer structure that contains the data, in ARGBFFFF format, for overwriting the `origSrc` image data.

origSrc

A pointer to a vImage buffer structure that contains the source image whose data you want to overwrite.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `origSrc` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

copyMask

An output value that selects the plane (or planes) from the ARGBFFFF source buffer that you want replaced with the corresponding plane from the `newSrc` image buffer. The value `0x8` selects the alpha channel, `0x4` the red channel, `0x2` the green channel, and `0x1` the blue channel. You can add these values together to select multiple channels.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`Conversion.h`

vImageTableLookUp_ARGB8888

Transforms an ARGB8888 image by substituting pixel values with pixel values provided by four lookup tables.

```
vImage_Error vImageTableLookUp_ARGB8888 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   const Pixel_8 alphaTable[256],
   const Pixel_8 redTable[256],
   const Pixel_8 greenTable[256],
   const Pixel_8 blueTable[256],
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to transform.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

alphaTable

The lookup table to use for the alpha channel of the source image. If you pass `NULL` for this table, the function copies the alpha channel unchanged to the destination buffer.

redTable

The lookup table to use for the red channel of the source image. If you pass `NULL` for this table, the function copies the red channel unchanged to the destination buffer.

greenTable

The lookup table to use for the green channel of the source image. If you pass `NULL` for this table, the function copies the green channel unchanged to the destination buffer.

blueTable

The lookup table to use for the blue channel of the source image. If you pass `NULL` for this table, the function copies the blue channel unchanged to the destination buffer.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value

`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion

The function separates each pixel into four channels—alpha, red, green, and blue. It substitutes each channel separately, using the appropriate table. Then the function recombines each pixel into a single ARGB8888 value, placing the transformed image into the destination buffer.

The source and destinations buffer must have the same height and the same width.

You cannot use this function to perform an arbitrary color mapping. For example, if two different source colors have the same green component, you must map them to two destination colors whose green components are equal. You can't map them to two arbitrary colors.

Availability

Available in Mac OS X v10.3 and later.

Declared In
`Conversion.h`

vImageTableLookUp_Planar8

Transforms an Planar8 image by substituting pixel values with pixel values provided by four lookup tables.

```
vImage_Error vImageTableLookUp_Planar8 (
   const vImage_Buffer *src,
   const vImage_Buffer *dest,
   const Pixel_8 table[256],
   vImage_Flags flags
);
```

Parameters

src

A pointer to a vImage buffer structure that contains the source image whose data you want to transform.

dest

A pointer to a vImage buffer data structure. You are responsible for filling out the `height`, `width`, and `rowBytes` fields of this structure, and for allocating a data buffer of the appropriate size. The destination data buffer can be the same as the `src` data buffer. On return, the data buffer pointed to by this structure contains the destination image data. When you no longer need the data buffer, you must deallocate the memory.

table

The lookup table to use.

flags

The options to use when performing this operation. Set the `kvImageDoNotTile` flag if you plan to perform your own tiling or use multithreading.

Return Value
`kvImageNoError`, otherwise one of the error codes described in *vImage Data Types and Constants Reference*.

Discussion
This function transforms a Planar8 image by replacing all pixels of a given intensity value with pixels of a new intensity value. It maps old values to new values using the provided 256-element lookup table (LUT).

The source and destinations buffer must have the same height and the same width.

Availability
Available in Mac OS X v10.3 and later.

Declared In
`Conversion.h`

Document Revision History

This table describes the changes to *vImage Conversion Reference*.

Date	Notes
2007-07-12	Updated for Mac OS X v10.5.
	The content in this document was formerly part of *Optimizing Image Processing With vImage*.
	Added `vImageSelectChannels_ARGB8888` (page 54), `vImageSelectChannels_ARGBFFFF` (page 55), `vImageOverwriteChannelsWithPixel_ARGB8888` (page 45), and `vImageOverwriteChannelsWithPixel_ARGBFFFF` (page 46).

Document Revision History

Index

V