Function

vImageLookupTable_Planar8toPlanar48(_:_:_:_:)

Uses a lookup table to transform an 8-bit planar image to a 16-bit-per-channel, three-channel interleaved image.

iOS 7.0+ | iPadOS 7.0+ | Mac Catalyst 13.1+ | macOS 10.9+ | tvOS 7.0+ | visionOS 1.0+ | watchOS 1.0+

```
func vImageLookupTable_Planar8toPlanar48(
    _ src: UnsafePointer<vImage_Buffer>,
    _ dest: UnsafePointer<vImage_Buffer>,
    _ table: UnsafePointer<UInt64>,
    _ flags: vImage_Flags
) -> vImage_Error
```

Parameters

`src`
The source vImage buffer.

`dest`
A pointer to the destination vImage buffer structure. You're 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 this structure points to contains the destination image data. When you no longer need the data buffer, deallocate the memory to prevent memory leaks.

`table`
A lookup table that contains 256 `uint64_t` values. The upper 48 bits of each value stores the red, green, and blue values.

`flags`
: The options to use when performing the operation. If your code implements its own tiling or its own multithreading, pass `kvImageDoNotTile`; otherwise, pass `kvImageNoFlags`.

Return Value

`kvImageNoError`; otherwise, one of the error codes in Data Types and Constants.

Discussion

For each pixel, this function uses the 8-bit value from the source image as an index to the 48-bit value from the table. The per-pixel conversion calculation is equivalent to the following:

```
uint64_t table[256];       // 64 bits per pixel and 16 bits per channel.
uint16_t *t_ptr = (uint16_t*)(table + srcRow[i]);
uint16_t r = t_ptr[1];
uint16_t g = t_ptr[2];
uint16_t b = t_ptr[3];
dstRow[3*i+0] = r;
dstRow[3*i+1] = g;
dstRow[3*i+2] = b;
```

The following code shows how to create a lookup table that maps all source pixel values to a constant RGB value of `(10, 20, 30)`:

```
let red: Pixel_16F = 10
let green: Pixel_16F = 20
let blue: Pixel_16F = 30
let lookupValue = (UInt64(blue) << 48) | (UInt64(green) << 32) | (UInt64(red) << 16)

let lookupTable = [UInt64](repeating: lookupValue, count: 256)

let source = vImage.PixelBuffer<vImage.Planar8>(
    size: .init(width: 5, height: 1))
let destination = vImage.PixelBuffer<vImage.Planar16F>(
    size: .init(width: 15, height: 1))

source.withUnsafePointerToVImageBuffer { src in
    destination.withUnsafeVImageBuffer { dest in
```

```
        var intermediate = vImage_Buffer(
            data: dest.data,
            height: dest.height,
            width: 5,
            rowBytes: dest.rowBytes)

        _ = vImageLookupTable_Planar8toPlanar48(
            src,
            &intermediate,
            lookupTable,
            vImage_Flags(kvImageNoFlags))

    }
}

// Prints "[10, 20, 30,  10, 20, 30,  10, 20, 30,  10, 20, 30,  10, 20, 30]".
print(destination array)
```

See Also

Transforming planar-to-interleaved with a lookup table

`func vImageLookupTable_Planar8toPlanar24(UnsafePointer<vImage_Buffer>, UnsafePointer<vImage_Buffer>, UnsafePointer<UInt32>, vImage_Flags) -> vImage_Error`

Uses a lookup table to transform an 8-bit planar image to an 8-bit-per-channel, three-channel interleaved image.

`func vImageLookupTable_Planar8toPlanar96(UnsafePointer<vImage_Buffer>, UnsafePointer<vImage_Buffer>, UnsafePointer<Pixel_FFFF>, vImage_Flags) -> vImage_Error`

Uses a lookup table to transform an 8-bit planar image to a 32-bit-per-channel, three-channel interleaved image.

`func vImageLookupTable_Planar8toPlanar128(UnsafePointer<vImage_Buffer>, UnsafePointer<vImage_Buffer>, UnsafePointer<Pixel_FFFF>, vImage_Flags) -> vImage_Error`

Uses a lookup table to transform an 8-bit planar image to a 32-bit-per-channel, four-channel interleaved image.