Function

vImageMatrixMultiply_ARGB8888ToPlanar8(_:_:_:_:_:_:_:)

Multiplies each pixel in an interleaved four-channel, 8-bit source image by a matrix to produce a planar 8-bit destination image.

iOS 9.0+ | iPadOS 9.0+ | Mac Catalyst 13.1+ | macOS 10.11+ | tvOS 9.0+ | visionOS 1.0+ | watchOS 2.0+

```
func vImageMatrixMultiply_ARGB8888ToPlanar8(
    _ src: UnsafePointer<vImage_Buffer>,
    _ dest: UnsafePointer<vImage_Buffer>,
    _ matrix: UnsafePointer<Int16>,
    _ divisor: Int32,
    _ pre_bias: UnsafePointer<Int16>!,
    _ post_bias: Int32,
    _ 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.

`matrix`
An array of values that contains the four elements in the column-matrix.

`divisor`
A value that the function divides the matrix multiplication result by after adding postbias.

`pre_bias`
An array that contains four bias values. The function adds the corresponding bias value to each source value before the matrix multiplication step. Pass `nil` to specify zero prebias.

`post_bias`
The postbias value. The function adds the bias value to the destination value after the matrix multiplication step.

`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

This function treats the four-channel source pixels as a four-element vector, and multiplies each source pixel by the matrix. The operation is equivalent to computing the dot product of the source pixel and the matrix.

The following code shows a four-channel interleaved buffer multiplied by a four-element column matrix to produce a planar destination image. The values in the matrix represent the Rec. 709 luminance coefficients that convert the color source pixels to a grayscale destination image. The source buffer contains a red pixel, a green pixel, and a blue pixel.

```
let size = vImage.Size(width: 3, height: 1)

let source = vImage.PixelBuffer<vImage.Interleaved8x4>(
    pixelValues: [0, 255, 0, 0,  // red
                  0, 0, 255, 0,  // green
                  0, 0, 0, 255], // blue
    size: size)
```

```
let destination = vImage.PixelBuffer<vImage.Planar8>(
    size: size)

let divisor: Int32 = 0x1000
let fDivisor = Float(divisor)

let redCoefficient: Float = 0.2126
let greenCoefficient: Float = 0.7152
let blueCoefficient: Float = 0.0722

let matrix = [
    0,
    Int16(redCoefficient * fDivisor),
    Int16(greenCoefficient * fDivisor),
    Int16(blueCoefficient * fDivisor)
]

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

        _ = vImageMatrixMultiply_ARGB8888ToPlanar8(
            src,
            dest,
            matrix,
            divisor,
            nil,
            0,
            vImage_Flags(kvImageNoFlags))
    }
}

// Prints "[0.21176471, 0.7137255, 0.07058824]".
print(destination.array.map { Float($0) / 255 })
```

See Also

Multiplying interleaved pixels by a matrix

```
func vImageMatrixMultiply_ARGB8888(UnsafePointer<vImage_Buffer>, Unsafe
Pointer<vImage_Buffer>, UnsafePointer<Int16>, Int32, UnsafePointer<
Int16>!, UnsafePointer<Int32>!, vImage_Flags) -> vImage_Error
```

Multiplies each pixel in an interleaved four-channel, 8-bit source image by a matrix to produce an interleaved four-channel, 8-bit destination image.

```
func vImageMatrixMultiply_ARGBFFFF(UnsafePointer<vImage_Buffer>, Unsafe
Pointer<vImage_Buffer>, UnsafePointer<Float>, UnsafePointer<Float>!,
UnsafePointer<Float>!, vImage_Flags) -> vImage_Error
```

Multiplies each pixel in an interleaved four-channel, 32-bit source image by a matrix to produce an interleaved four-channel, 32-bit destination image.

```
func vImageMatrixMultiply_ARGBFFFFToPlanarF(UnsafePointer<vImage_Buffer
>, UnsafePointer<vImage_Buffer>, UnsafePointer<Float>, UnsafePointer<
Float>!, Float, vImage_Flags) -> vImage_Error
```

Multiplies each pixel in an interleaved four-channel, 32-bit source image by a matrix to produce a planar 32-bit destination image.