Accelerate Reference Update

Performance > Vector Engines

2007-07-18

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

Simultaneously published in the United States and Canada.

Contents

Introduction to Accelerate Reference Update 5

Organization of This Document 5
See Also 5

10.5 Symbol Changes 7

C Symbols 7
  vecLib 7
  vImage 9

10.4 Symbol Changes 11

C Symbols 11
  vecLib 11
  vImage 35

10.3 Symbol Changes 45

C Symbols 45
  vecLib 45
  vImage 47

Document Revision History 59

Introduction to Accelerate Reference Update

This document summarizes the symbols that have been added to the Accelerate framework. The full reference documentation notes in what version a symbol was introduced, but sometimes it's useful to see only the new symbols for a given release.

If you are not familiar with this framework you should refer to the complete framework reference documentation.

Organization of This Document

Symbols are grouped by class or protocol for Objective-C and by header file for C. For each symbol there is a link to complete documentation, if available, and a brief description, if available.

See Also

For reference documentation on this framework, see *Accelerate framework reference*.

10.5 Symbol Changes

This article lists the symbols added to `Accelerate.framework` in Mac OS X v10.5.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

vecLib

vBasicOps.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vLL128Shift`	
`vLR128Shift`	
`vS128Neg`	
`vS64Neg`	
`vU128Neg`	
`vU64Neg`	

vDSP.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vDSP_HALF_WINDOW`	
`vDSP_HANN_DENORM`	
`vDSP_HANN_NORM`	

vDSP_Length	
vDSP_Stride	

vForce.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

vvcopysignf	
vvexpm1f	
vvfabf	
vvfmodf	
vvlog1pf	
vvlogbf	
vvnextafterf	
vvremainderf	

vfp.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

vceilf	
vfloorf	
vintf	
vlog10f	
vnintf	
vrecf	
vsincosf	

vImage

Conversion.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vImageOverwriteChannelsWithPixel_ARGB8888`	Overwrites an ARGB8888 image buffer with the provided pixel value.
`vImageOverwriteChannelsWithPixel_ARGBFFFF`	Overwrites an ARGBFFFF image buffer with the provided pixel value.
`vImageSelectChannels_ARGB8888`	Overwrites the specified channels in an ARGB8888 image buffer with the provided channels from an ARGB8888 image buffer.
`vImageSelectChannels_ARGBFFFF`	Overwrites the specified channels in an ARGBFFFF image buffer with the provided channels in an ARGBFFFF image buffer.

10.4 Symbol Changes

This article lists the symbols added to `Accelerate.framework` in Mac OS X v10.4.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

vecLib

cblas.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`ATLU_DestroyThreadMemory`	
`catlas_caxpby`	
`catlas_cset`	
`catlas_daxpby`	
`catlas_dset`	
`catlas_saxpby`	
`catlas_sset`	
`catlas_zaxpby`	
`catlas_zset`	

vDSP.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

vDSP_acorD	Autocorrelation with automatic selection of domain.
vDSP_acorfD	Frequency-domain autocorrelation.
vDSP_acortD	Time-domain autocorrelation.
vDSP_blkman_window	Creates a Blackman window.
vDSP_blkman_windowD	Creates a Blackman window.
vDSP_conv	Performs either correlation or convolution on two vectors.
vDSP_convD	Performs either correlation or convolution on two vectors.
vDSP_create_fftsetup	Builds a data structure that contains precalculated data for use by Fourier Transform functions.
vDSP_create_fftsetupD	Builds a data structure that contains precalculated data for use by Fourier Transform functions.
vDSP_ctoz	Copies the contents of an interleaved complex vector C to a split complex vector Z.
vDSP_ctozD	Copies the contents of an interleaved complex vector C to a split complex vector Z.
vDSP_deq22	Difference equation, 2 poles, 2 zeros.
vDSP_deq22D	Difference equation, 2 poles, 2 zeros.
vDSP_desamp	Convolution with decimation.
vDSP_desampD	Convolution with decimation.
vDSP_destroy_fftsetup	Frees an existing Fourier Transforms data structure.
vDSP_destroy_fftsetupD	Frees an existing Fourier Transforms data structure.
vDSP_dotpr	Computes the dot or scalar product of vectors A and B and leaves the result in scalar C.
vDSP_dotprD	Computes the dot or scalar product of vectors A and B and leaves the result in scalar C.
vDSP_f3x3	Filters an image by performing a two-dimensional convolution with a 3x3 kernel on the input matrix A. The resulting image is placed in the output matrix C.
vDSP_f3x3D	Filters an image by performing a two-dimensional convolution with a 3x3 kernel on the input matrix A. The resulting image is placed in the output matrix C.
vDSP_f5x5	Filters an image by performing a two-dimensional convolution with a 5x5 kernel on the input matrix signal. The resulting image is placed in the output matrix result.

vDSP_f5x5D	Filters an image by performing a two-dimensional convolution with a 5x5 kernel on the input matrix signal. The resulting image is placed in the output matrix result.
vDSP_fft2d_zip	Computes an in-place complex discrete Fourier transform of matrix represented by signal, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zipD	Computes an in-place complex discrete Fourier transform of matrix represented by signal, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zipt	Computes an in-place complex discrete Fourier transform of matrix represented by signal, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_ziptD	Computes an in-place complex discrete Fourier transform of matrix represented by signal, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zop	Computes an out-of-place complex discrete Fourier transform of the matrix represented by signal, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zopD	Computes an out-of-place complex discrete Fourier transform of the matrix represented by signal, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zopt	Computes an out-of-place complex discrete Fourier transform of the matrix represented by signal, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zoptD	Computes an out-of-place complex discrete Fourier transform of the matrix represented by signal, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zrip	Computes an in-place real discrete Fourier transform, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zripD	Computes an in-place real discrete Fourier transform, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).

vDSP_fft2d_zript	Computes an in-place real discrete Fourier transform, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zriptD	Computes an in-place real discrete Fourier transform, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zrop	Computes an out-of-place real discrete Fourier transform, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zropD	Computes an out-of-place real discrete Fourier transform, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zropt	Computes an out-of-place real discrete Fourier transform, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft2d_zroptD	Computes an out-of-place real discrete Fourier transform, either from the spatial domain to the frequency domain (forward) or from the frequency domain to the spatial domain (inverse).
vDSP_fft3_zop	Computes an out-of-place radix-3 complex Fourier transform, either forward or inverses. The number of input and output values processed equals 3 times the power of 2 specified by parameter log2n.
vDSP_fft3_zopD	Computes an out-of-place radix-3 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 3 times the power of 2 specified by parameter log2n.
vDSP_fft5_zop	Computes an out-of-place radix-5 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 5 times the power of 2 specified by parameter log2n.
vDSP_fft5_zopD	Computes an out-of-place radix-5 complex Fourier transform, either forward or inverse. The number of input and output values processed equals 5 times the power of 2 specified by parameter log2n.
vDSP_fftm_zip	Performs multiple Fourier transforms with a single call.
vDSP_fftm_zipD	Performs multiple Fourier transforms with a single call.
vDSP_fftm_zipt	Performs multiple Fourier transforms with a single call.
vDSP_fftm_ziptD	Performs multiple Fourier transforms with a single call.
vDSP_fftm_zop	Performs multiple Fourier transforms with a single call.
vDSP_fftm_zopD	Performs multiple Fourier transforms with a single call.
vDSP_fftm_zopt	Performs multiple Fourier transforms with a single call.
vDSP_fftm_zoptD	Performs multiple Fourier transforms with a single call.

vDSP_fftm_zrip	Performs multiple Fourier transform with a single call.
vDSP_fftm_zripD	Performs multiple Fourier transform with a single call.
vDSP_fftm_zript	Performs multiple Fourier transform with a single call.
vDSP_fftm_zriptD	Performs multiple Fourier transform with a single call.
vDSP_fftm_zrop	Performs multiple Fourier transforms with a single call.
vDSP_fftm_zropD	Performs multiple Fourier transforms with a single call.
vDSP_fftm_zropt	Performs multiple Fourier transforms with a single call.
vDSP_fftm_zroptD	Performs multiple Fourier transforms with a single call.
vDSP_fft_zip	Computes an in-place complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zipD	Computes an in-place complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zipt	Computes an in-place complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_ziptD	Computes an in-place complex discrete Fourier transform of the input/output vector signal, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zop	Computes an out-of-place complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zopD	Computes an out-of-place complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zopt	Computes an out-of-place complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zoptD	Computes an out-of-place complex discrete Fourier transform of the input vector, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zrip	Computes an in-place real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).

vDSP_fft_zripD	Computes an in-place real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zript	Computes an in-place real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zriptD	Computes an in-place real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zrop	Computes an out-of-place real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zropD	Computes an out-of-place real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zropt	Computes an out-of-place real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_fft_zroptD	Computes an out-of-place real discrete Fourier transform, either from the time domain to the frequency domain (forward) or from the frequency domain to the time domain (inverse).
vDSP_hamm_window	Creates a Hamming window.
vDSP_hamm_windowD	Creates a Hamming window.
vDSP_hann_window	Creates a Hanning window.
vDSP_hann_windowD	Creates a Hanning window.
vDSP_imgfir	Filters an image by performing a two-dimensional convolution with a kernel.
vDSP_imgfirD	Filters an image by performing a two-dimensional convolution with a kernel.
vDSP_maxmgv	Vector maximum magnitude.
vDSP_maxmgvD	Vector maximum magnitude.
vDSP_maxmgvi	Vector maximum magnitude with index.
vDSP_maxmgviD	Vector maximum magnitude with index.
vDSP_maxv	Vector maximum value.
vDSP_maxvD	Vector maximum value.
vDSP_maxvi	Vector maximum value with index.

vDSP_maxviD	Vector maximum value with index.
vDSP_meamgv	Vector mean magnitude.
vDSP_meamgvD	Vector mean magnitude.
vDSP_meanv	Vector mean value.
vDSP_meanvD	Vector mean value.
vDSP_measqv	Vector mean square value.
vDSP_measqvD	Vector mean square value.
vDSP_minmgv	Vector minimum magnitude.
vDSP_minmgvD	Vector minimum magnitude.
vDSP_minmgvi	Vector minimum magnitude with index.
vDSP_minmgviD	Vector minimum magnitude with index.
vDSP_minv	Vector minimum value.
vDSP_minvD	Vector minimum value.
vDSP_minvi	Vector minimum value with index.
vDSP_minviD	Vector minimum value with index.
vDSP_mmov	The contents of a submatrix are copied to another submatrix.
vDSP_mmovD	The contents of a submatrix are copied to another submatrix.
vDSP_mmul	Multiplies an M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C. This function can only be performed out-of-place.
vDSP_mmulD	Multiplies an M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C. This function can only be performed out-of-place.
vDSP_mtrans	Creates a transposed matrix C from a source matrix A.
vDSP_mtransD	Creates a transposed matrix C from a source matrix A.
vDSP_mvessq	Vector mean of signed squares.
vDSP_mvessqD	Vector mean of signed squares.
vDSP_nzcros	Find zero crossings.
vDSP_nzcrosD	Find zero crossings.
vDSP_polar	Rectangular to polar conversion.
vDSP_polarD	Rectangular to polar conversion.

vDSP_rect	Polar to rectangular conversion.
vDSP_rectD	Polar to rectangular conversion.
vDSP_rmsqv	Vector root-mean-square.
vDSP_rmsqvD	Vector root-mean-square.
vDSP_svdiv	Divide scalar by vector.
vDSP_svdivD	Divide scalar by vector.
vDSP_sve	Vector sum.
vDSP_sveD	Vector sum.
vDSP_svemg	Vector sum of magnitudes.
vDSP_svemgD	Vector sum of magnitudes.
vDSP_svesq	Vector sum of squares.
vDSP_svesqD	Vector sum of squares.
vDSP_svs	Vector sum of signed squares.
vDSP_svsD	Vector sum of signed squares.
vDSP_vaam	Vector add, add, and multiply.
vDSP_vaamD	Vector add, add, and multiply.
vDSP_vabs	Vector absolute values.
vDSP_vabsD	Vector absolute values.
vDSP_vabsi	Integer vector absolute values.
vDSP_vadd	Adds vector A to vector B and leaves the result in vector C.
vDSP_vaddD	Adds vector A to vector B and leaves the result in vector C.
vDSP_vam	Adds vectors A and B, multiplies the sum by vector C, and leaves the result in vector D.
vDSP_vamD	Adds vectors A and B, multiplies the sum by vector C, and leaves the result in vector D.
vDSP_vasbm	Vector add, subtract, and multiply.
vDSP_vasbmD	Vector add, subtract, and multiply.
vDSP_vasm	Vector add and scalar multiply.
vDSP_vasmD	Vector add and scalar multiply.

vDSP_vavlin	Vector linear average.
vDSP_vavlinD	Vector linear average.
vDSP_vclip	Vector clip.
vDSP_vclipc	Vector clip and count.
vDSP_vclipcD	Vector clip and count.
vDSP_vclipD	Vector clip.
vDSP_vclr	Vector clear.
vDSP_vclrD	Vector clear.
vDSP_vcmprs	Vector compress.
vDSP_vcmprsD	Vector compress.
vDSP_vdbcon	Vector convert power or amplitude to decibels.
vDSP_vdbconD	Vector convert power or amplitude to decibels.
vDSP_vdist	Vector distance.
vDSP_vdistD	Vector distance.
vDSP_vdiv	Vector divide.
vDSP_vdivD	Vector divide.
vDSP_vdivi	Vector divide.
vDSP_vdpsp	Vector convert double-precision to single-precision.
vDSP_venvlp	Vector envelope.
vDSP_venvlpD	Vector envelope.
vDSP_veqvi	Vector equivalence, 32-bit logical.
vDSP_vfill	Vector fill.
vDSP_vfillD	Vector fill.
vDSP_vfilli	Integer vector fill.
vDSP_vfix16	
vDSP_vfix16D	
vDSP_vfix32	
vDSP_vfix32D	

vDSP_vfix8	
vDSP_vfix8D	
vDSP_vfixr16	
vDSP_vfixr16D	
vDSP_vfixr32	
vDSP_vfixr32D	
vDSP_vfixr8	
vDSP_vfixr8D	
vDSP_vfixru16	
vDSP_vfixru16D	
vDSP_vfixru32	
vDSP_vfixru32D	
vDSP_vfixru8	
vDSP_vfixru8D	
vDSP_vfixu16	
vDSP_vfixu16D	
vDSP_vfixu32	
vDSP_vfixu32D	
vDSP_vfixu8	
vDSP_vfixu8D	
vDSP_vflt16	
vDSP_vflt16D	
vDSP_vflt32	
vDSP_vflt32D	
vDSP_vflt8	
vDSP_vflt8D	
vDSP_vfltu16	
vDSP_vfltu16D	

vDSP_vfltu32	
vDSP_vfltu32D	
vDSP_vfltu8	
vDSP_vfltu8D	
vDSP_vfrac	Vector truncate to fraction.
vDSP_vfracD	Vector truncate to fraction.
vDSP_vgathr	Vector gather.
vDSP_vgathra	Vector gather, absolute pointers.
vDSP_vgathraD	Vector gather, absolute pointers.
vDSP_vgathrD	Vector gather.
vDSP_vgen	Vector tapered ramp.
vDSP_vgenD	Vector tapered ramp.
vDSP_vgenp	Vector generate by extrapolation and interpolation.
vDSP_vgenpD	Vector generate by extrapolation and interpolation.
vDSP_viclip	Vector inverted clip.
vDSP_viclipD	Vector inverted clip.
vDSP_vindex	Vector index.
vDSP_vindexD	Vector index.
vDSP_vintb	Vector linear interpolation between vectors.
vDSP_vintbD	Vector linear interpolation between vectors.
vDSP_vlim	Vector test limit.
vDSP_vlimD	Vector test limit.
vDSP_vlint	Vector linear interpolation between neighboring values.
vDSP_vlintD	Vector linear interpolation between neighboring values.
vDSP_vma	Vector multiply and add.
vDSP_vmaD	Vector multiply and add.
vDSP_vmax	Vector maxima.
vDSP_vmaxD	Vector maxima.

vDSP_vmaxmg	Vector maximum magnitudes.
vDSP_vmaxmgD	Vector maximum magnitudes.
vDSP_vmin	Vector minima.
vDSP_vminD	Vector minima.
vDSP_vminmg	Vector minimum magnitudes.
vDSP_vminmgD	Vector minimum magnitudes.
vDSP_vmma	Vector multiply, multiply, and add.
vDSP_vmmaD	Vector multiply, multiply, and add.
vDSP_vmmsb	Vector multiply, multiply, and subtract.
vDSP_vmmsbD	Vector multiply, multiply, and subtract.
vDSP_vmsa	Vector multiply and scalar add.
vDSP_vmsaD	Vector multiply and scalar add.
vDSP_vmsb	Vector multiply and subtract.
vDSP_vmsbD	Vector multiply and subtract.
vDSP_vmul	Multiplies vector signal1 by vector signal2 and leaves the result in vector result.
vDSP_vmulD	Multiplies vector signal1 by vector signal2 and leaves the result in vector result.
vDSP_vnabs	Vector negative absolute value.
vDSP_vnabsD	Vector negative absolute value.
vDSP_vneg	Vector negative value.
vDSP_vnegD	Vector negative value.
vDSP_vpoly	Vector polynomial.
vDSP_vpolyD	Vector polynomial.
vDSP_vpythg	Vector pythagoras.
vDSP_vpythgD	Vector pythagoras.
vDSP_vqint	Vector quadratic interpolation.
vDSP_vqintD	Vector quadratic interpolation.
vDSP_vramp	Build ramped vector.

vDSP_vrampD	Build ramped vector.
vDSP_vrsum	Vector running sum integration.
vDSP_vrsumD	Vector running sum integration.
vDSP_vrvrs	Vector reverse order, in place.
vDSP_vrvrsD	Vector reverse order, in place.
vDSP_vsadd	Vector scalar add.
vDSP_vsaddD	Vector scalar add.
vDSP_vsaddi	Integer vector scalar add.
vDSP_vsbm	Vector subtract and multiply.
vDSP_vsbmD	Vector subtract and multiply.
vDSP_vsbsbm	Vector subtract, subtract, and multiply.
vDSP_vsbsbmD	Vector subtract, subtract, and multiply.
vDSP_vsbsm	Vector subtract and scalar multiply.
vDSP_vsbsmD	Vector subtract and scalar multiply.
vDSP_vsdiv	Vector scalar divide.
vDSP_vsdivD	Vector scalar divide.
vDSP_vsdivi	Integer vector scalar divide.
vDSP_vsimps	Simpson integration.
vDSP_vsimpsD	Simpson integration.
vDSP_vsma	Vector scalar multiply and vector add.
vDSP_vsmaD	Vector scalar multiply and vector add.
vDSP_vsmsa	Vector scalar multiply and scalar add.
vDSP_vsmsaD	Vector scalar multiply and scalar add.
vDSP_vsmsb	Vector scalar multiply and vector subtract.
vDSP_vsmsbD	Vector scalar multiply and vector subtract.
vDSP_vsmul	Multiplies vector signal1 by scalar signal2 and leaves the result in vector result.
vDSP_vsmulD	Multiplies vector signal1 by scalar signal2 and leaves the result in vector result.

vDSP_vsort	Vector in-place sort.
vDSP_vsortD	Vector in-place sort.
vDSP_vsorti	Vector integer in-place sort.
vDSP_vsortiD	Vector integer in-place sort.
vDSP_vspdp	Vector convert single-precision to double-precision.
vDSP_vsq	Computes the squared values of vector signal1 and leaves the result in vector result.
vDSP_vsqD	Computes the squared values of vector signal1 and leaves the result in vector result.
vDSP_vssq	Computes the signed squares of vector signal1 and leaves the result in vector result.
vDSP_vssqD	Computes the signed squares of vector signal1 and leaves the result in vector result.
vDSP_vsub	Subtracts vector signal2 from vector signal1 and leaves the result in vector result.
vDSP_vsubD	Subtracts vector signal2 from vector signal1 and leaves the result in vector result.
vDSP_vswap	Vector swap.
vDSP_vswapD	Vector swap.
vDSP_vswsum	Vector sliding window sum.
vDSP_vswsumD	Vector sliding window sum.
vDSP_vtabi	Vector interpolation, table lookup.
vDSP_vtabiD	Vector interpolation, table lookup.
vDSP_vthr	Vector threshold.
vDSP_vthrD	Vector threshold.
vDSP_vthres	Vector threshold with zero fill.
vDSP_vthresD	Vector threshold with zero fill.
vDSP_vthrsc	Vector threshold with signed constant.
vDSP_vthrscD	Vector threshold with signed constant.
vDSP_vtmerg	Vector tapered merge of two vectors.
vDSP_vtmergD	Vector tapered merge of two vectors.

vDSP_vtrapz	Vector trapezoidal integration.
vDSP_vtrapzD	Vector trapezoidal integration.
vDSP_wiener	Wiener-Levinson general convolution.
vDSP_wienerD	Wiener-Levinson general convolution.
vDSP_zaspec	Computes an accumulating autospectrum.
vDSP_zaspecD	Computes an accumulating autospectrum.
vDSP_zcoher	Coherence function of two signals.
vDSP_zcoherD	Coherence function of two signals.
vDSP_zconv	Performs either correlation or convolution on two complex vectors.
vDSP_zconvD	Performs either correlation or convolution on two complex vectors.
vDSP_zcspec	Accumulating cross-spectrum on two complex vectors.
vDSP_zcspecD	Accumulating cross-spectrum on two complex vectors.
vDSP_zdotpr	Calculates the complex dot product of complex vectors signal1 and signal2 and leaves the result in complex vector result.
vDSP_zdotprD	Calculates the complex dot product of complex vectors signal1 and signal2 and leaves the result in complex vector result.
vDSP_zidotpr	Calculates the conjugate dot product (or inner dot product) of complex vectors signal1 and signal2 and leave the result in complex vector result.
vDSP_zidotprD	Calculates the conjugate dot product (or inner dot product) of complex vectors signal1 and signal2 and leave the result in complex vector result.
vDSP_zmma	Multiplies a matrix A by matrix B , adds the product to matrix C, and stores the result in matrix D. A is an M-by-P matrix, B is a P-by-N matrix, C and D are by M-by-N matrixes. This function can only be performed out-of-place.
vDSP_zmmaD	Multiplies a matrix A by matrix B , adds the product to matrix C, and stores the result in matrix D. A is an M-by-P matrix, B is a P-by-N matrix, C and D are by M-by-N matrixes. This function can only be performed out-of-place.
vDSP_zmms	Multiplies an matrix a by matrix b , subtracts matrix c from the product, and stores the result in matrix d. a is an M-by-P matrix, b is a P-by-N matrix, c and d are by M-by-N matrixes. The function can only be performed out of place.

vDSP_zmmsD	Multiplies an matrix a by matrix b , subtracts matrix c from the product, and stores the result in matrix d. a is an M-by-P matrix, b is a P-by-N matrix, c and d are by M-by-N matrixes. The function can only be performed out of place.
vDSP_zmmul	Multiplies an M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C. The function can only be performed out of place
vDSP_zmmulD	Multiplies an M-by-P matrix A by a P-by-N matrix B and stores the results in an M-by-N matrix C. The function can only be performed out of place.
vDSP_zmsm	Multiplies an matrix a by matrix b , subtracts the product from matrix c, and stores the result in matrix d. a is an M-by-P matrix, b is a P-by-N matrix, c and d are by M-by-N matrixes. The function can only be performed out of place.
vDSP_zmsmD	Multiplies an matrix a by matrix b , subtracts the product from matrix c, and stores the result in matrix d. a is an M-by-P matrix, b is a P-by-N matrix, c and d are by M-by-N matrixes. The function can only be performed out of place.
vDSP_zrdesamp	Complex/real downsample with anti-aliasing.
vDSP_zrdesampD	Complex/real downsample with anti-aliasing.
vDSP_zrdotpr	Calculates the complex dot product of complex vector A and real vector B and leaves the result in complex vector C.
vDSP_zrdotprD	Calculates the complex dot product of complex vector A and real vector B and leaves the result in complex vector C.
vDSP_zrvadd	Adds real vector B to complex vector A and leaves the result in complex vector C.
vDSP_zrvaddD	Adds real vector B to complex vector A and leaves the result in complex vector C.
vDSP_zrvdiv	Divides complex vector A by real vector B and leaves the result in vector C.
vDSP_zrvdivD	Divides complex vector A by real vector B and leaves the result in vector C.
vDSP_zrvmul	Multiplies complex vector A by real vector B and leaves the result in vector C.
vDSP_zrvmulD	Multiplies complex vector A by real vector B and leaves the result in vector C.
vDSP_zrvsub	Subtracts real vector B from complex vector A and leaves the result in complex vector C.
vDSP_zrvsubD	Subtracts real vector B from complex vector A and leaves the result in complex vector C.

vDSP_ztoc	Copies the contents of a split complex vector A to an interleaved complex vector C.
vDSP_ztocD	Copies the contents of a split complex vector A to an interleaved complex vector C.
vDSP_ztrans	Transfer function.
vDSP_ztransD	Transfer function.
vDSP_zvabs	Complex vector absolute value.
vDSP_zvabsD	Complex vector absolute value.
vDSP_zvadd	Adds complex vectors A and B and leaves the result in complex vector C
vDSP_zvaddD	Adds complex vectors A and B and leaves the result in complex vector C
vDSP_zvcma	Multiplies complex vector B by the complex conjugates of complex vector A, adds the products to complex vector C, and stores the results in complex vector D.
vDSP_zvcmaD	Multiplies complex vector B by the complex conjugates of complex vector A, adds the products to complex vector C, and stores the results in complex vector D.
vDSP_zvcmul	Complex vector conjugate and multiply.
vDSP_zvcmulD	Complex vector conjugate and multiply.
vDSP_zvconj	Complex vector conjugate.
vDSP_zvconjD	Complex vector conjugate.
vDSP_zvdiv	Complex vector divide.
vDSP_zvdivD	Complex vector divide.
vDSP_zvfill	Complex vector fill.
vDSP_zvfillD	Complex vector fill.
vDSP_zvmags	Complex vector magnitudes squared.
vDSP_zvmagsD	Complex vector magnitudes squared.
vDSP_zvmgsa	Complex vector magnitudes square and add.
vDSP_zvmgsaD	Complex vector magnitudes square and add.
vDSP_zvmov	Complex vector move.
vDSP_zvmovD	Complex vector move.

vDSP_zvmul	Multiplies complex vectors A and B and leaves the result in complex vector C.
vDSP_zvmulD	Multiplies complex vectors A and B and leaves the result in complex vector C.
vDSP_zvneg	Complex vector negate.
vDSP_zvnegD	Complex vector negate.
vDSP_zvphas	Complex vector phase.
vDSP_zvphasD	Complex vector phase.
vDSP_zvsma	Complex vector scalar multiply and add.
vDSP_zvsmaD	Complex vector scalar multiply and add.
vDSP_zvsub	Subtracts complex vector B from complex vector A and leaves the result in complex vector C
vDSP_zvsubD	Subtracts complex vector B from complex vector A and leaves the result in complex vector C
vDSP_zvzsml	Complex vector multiply by complex scalar.
vDSP_zvzsmlD	Complex vector multiply by complex scalar.

vDSP_translate.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

vDSP_conv	
vDSP_convD	
vDSP_create_fftsetup	
vDSP_create_fftsetupD	
vDSP_ctoz	
vDSP_ctozD	
vDSP_destroy_fftsetup	
vDSP_destroy_fftsetupD	
vDSP_dotpr	
vDSP_dotprD	

vDSP_f3x3	
vDSP_f3x3D	
vDSP_f5x5	
vDSP_f5x5D	
vDSP_fft2d_zip	
vDSP_fft2d_zipD	
vDSP_fft2d_zipt	
vDSP_fft2d_ziptD	
vDSP_fft2d_zop	
vDSP_fft2d_zopD	
vDSP_fft2d_zopt	
vDSP_fft2d_zoptD	
vDSP_fft2d_zrip	
vDSP_fft2d_zripD	
vDSP_fft2d_zript	
vDSP_fft2d_zriptD	
vDSP_fft2d_zrop	
vDSP_fft2d_zropD	
vDSP_fft2d_zropt	
vDSP_fft2d_zroptD	
vDSP_fft3_zop	
vDSP_fft3_zopD	
vDSP_fft5_zop	
vDSP_fft5_zopD	
vDSP_fftm_zip	
vDSP_fftm_zipD	
vDSP_fftm_zipt	
vDSP_fftm_ziptD	

vDSP_fftm_zop	
vDSP_fftm_zopD	
vDSP_fftm_zopt	
vDSP_fftm_zoptD	
vDSP_fftm_zrip	
vDSP_fftm_zripD	
vDSP_fftm_zript	
vDSP_fftm_zriptD	
vDSP_fftm_zrop	
vDSP_fftm_zropD	
vDSP_fftm_zropt	
vDSP_fftm_zroptD	
vDSP_fft_cip	
vDSP_fft_cipt	
vDSP_fft_cop	
vDSP_fft_copt	
vDSP_fft_zip	
vDSP_fft_zipD	
vDSP_fft_zipt	
vDSP_fft_ziptD	
vDSP_fft_zop	
vDSP_fft_zopD	
vDSP_fft_zopt	
vDSP_fft_zoptD	
vDSP_fft_zrip	
vDSP_fft_zripD	
vDSP_fft_zript	
vDSP_fft_zriptD	

vDSP_fft_zrop	
vDSP_fft_zropD	
vDSP_fft_zropt	
vDSP_fft_zroptD	
vDSP_imgfir	
vDSP_imgfirD	
vDSP_mmul	
vDSP_mmulD	
vDSP_mtrans	
vDSP_mtransD	
vDSP_vadd	
vDSP_vaddD	
vDSP_vam	
vDSP_vamD	
vDSP_vmul	
vDSP_vmulD	
vDSP_vsmul	
vDSP_vsmulD	
vDSP_vsq	
vDSP_vsqD	
vDSP_vssq	
vDSP_vssqD	
vDSP_vsub	
vDSP_vsubD	
vDSP_zconv	
vDSP_zconvD	
vDSP_zdotpr	
vDSP_zdotprD	

vDSP_zidotpr	
vDSP_zidotprD	
vDSP_zmma	
vDSP_zmmaD	
vDSP_zmms	
vDSP_zmmsD	
vDSP_zmmul	
vDSP_zmmulD	
vDSP_zmsm	
vDSP_zmsmD	
vDSP_zrdotpr	
vDSP_zrdotprD	
vDSP_zrvadd	
vDSP_zrvaddD	
vDSP_zrvmul	
vDSP_zrvmulD	
vDSP_zrvsub	
vDSP_zrvsubD	
vDSP_ztoc	
vDSP_ztocD	
vDSP_zvadd	
vDSP_zvaddD	
vDSP_zvcma	
vDSP_zvcmaD	
vDSP_zvmul	
vDSP_zvmulD	
vDSP_zvsub	
vDSP_zvsubD	

vForce.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vvacos`	For each double-precision array element, sets y to the arccosine of x.
`vvacosf`	For each single-precision array element, sets y to the arccosine of x.
`vvacosh`	For each double-precision array element, sets y to the inverse hyperbolic cosine of x.
`vvacoshf`	For each single-precision array element, sets y to the inverse hyperbolic cosine of x.
`vvasin`	For each double-precision array element, sets y to the arcsine of x.
`vvasinf`	For each single-precision array element, sets y to the arcsine of x.
`vvasinh`	For each double-precision array element, sets y to the inverse hyperbolic sine of x.
`vvasinhf`	For each single-precision array element, sets y to the inverse hyperbolic sine of x.
`vvatan`	For each double-precision array element, sets y to the arctangent of x.
`vvatan2`	For each double-precision array element, sets z to the arctangent of y/x.
`vvatan2f`	For each single-precision array element, sets z to the arctangent of y/x.
`vvatanf`	For each single-precision array element, sets y to the arctangent of x.
`vvatanh`	For each double-precision array element, sets y to the inverse hyperbolic tangent of x.
`vvatanhf`	For each single-precision array element, sets y to the inverse hyperbolic tangent of x.
`vvceil`	For each double-precision array element, sets y to the ceiling of x.
`vvceilf`	For each single-precision array element, sets y to the ceiling of x.
`vvcos`	For each double-precision array element, sets y to the cosine of x.
`vvcosf`	For each single-precision array element, sets y to the cosine of x.
`vvcosh`	For each double-precision array element, sets y to the hyperbolic cosine of x.
`vvcoshf`	For each single-precision array element, sets y to the hyperbolic cosine of x.
`vvcosisin`	For each double-precision array element, sets the real part of C to the sine of x and the imaginary part of C to the cosine of x.

`vvcosisinf`	For each single-precision array element, sets the real part of C to the sine of x and the imaginary part of C to the cosine of x.
`vvdiv`	For each double-precision array element, sets z to y/x.
`vvdivf`	For each single-precision array element, sets z to y/x.
`vvexp`	For each double-precision array element, sets y to the exponential of x.
`vvexpf`	For each single-precision array element, sets y to the exponential of x.
`vvfloor`	For each double-precision array element, sets y to the floor of x.
`vvfloorf`	For each single-precision array element, sets y to the floor of x.
`vvint`	For each double-precision array element, sets y to the integer truncation of x.
`vvintf`	For each single-precision array element, sets y to the integer truncation of x.
`vvlog`	For each double-precision array element, sets y to the natural logarithm of x.
`vvlog10`	For each double-precision array element, sets y to the base 10 logarithm of x.
`vvlog10f`	For each single-precision array element, sets y to the base 10 logarithm of x.
`vvlogf`	For each single-precision array element, sets y to the natural logarithm of x.
`vvnint`	For each double-precision array element, sets y to the nearest integer to x.
`vvnintf`	For each single-precision array element, sets y to the nearest integer to x.
`vvpow`	For each double-precision array element, sets z to x raised to the power of y.
`vvpowf`	For each single-precision array element, sets z to x raised to the power of y.
`vvrec`	For each double-precision array element, sets y to the reciprocal of y.
`vvrecf`	For each single-precision array element, sets y to the reciprocal of y.
`vvrsqrt`	For each double-precision array element, sets y to the reciprocal of the square root of x.
`vvrsqrtf`	For each single-precision array element, sets y to the reciprocal of the square root of x.
`vvsin`	For each double-precision array element, sets y to the sine of x.
`vvsincos`	For each double-precision array element, sets z to the sine of x and y to the cosine of x.
`vvsincosf`	For each single-precision array element, sets z to the sine of x and y to the cosine of x.
`vvsinf`	For each single-precision array element, sets y to the sine of x.
`vvsinh`	For each double-precision array element, sets y to the hyperbolic sine of x.

`vvsinhf`	For each single-precision array element, sets y to the hyperbolic sine of x.
`vvsqrt`	For each double-precision array element, sets y to the square root of x.
`vvsqrtf`	For each single-precision array element, sets y to the square root of x.
`vvtan`	For each double-precision array element, sets y to the tangent of x.
`vvtanf`	For each single-precision array element, sets y to the tangent of x.
`vvtanh`	For each double-precision array element, sets y to the hyperbolic tangent of x.
`vvtanhf`	For each single-precision array element, sets y to the hyperbolic tangent of x.

vImage

Alpha.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vImageAlphaBlend_NonpremultipliedToPremultiplied_ARGB8888`	Performs mixed alpha compositing of a nonpremultiplied ARGB8888 image over a premultiplied ARGB8888 image, placing the premultiplied result in a destination buffer.
`vImageAlphaBlend_NonpremultipliedToPremultiplied_ARGBFFFF`	Performs mixed alpha compositing of a nonpremultiplied ARGBFFFF image over a premultiplied ARGBFFFF image, placing the premultiplied result in a destination buffer.
`vImageAlphaBlend_NonpremultipliedToPremultiplied_Planar8`	Performs mixed alpha compositing of a nonpremultiplied Planar8 image over a premultiplied Planar8 image, placing the premultiplied result in a destination buffer.
`vImageAlphaBlend_NonpremultipliedToPremultiplied_PlanarF`	Performs mixed alpha compositing of a nonpremultiplied PlanarF image over a premultiplied PlanarF image, placing the premultiplied result in a destination buffer.

`vImageClipToAlpha_ARGB8888`	Sets the color channel of each pixel in an ARGB8888 image to the smaller of two values—either the color channel or the alpha value for that pixel.
`vImageClipToAlpha_ARGBFFFF`	Sets the color channel of each pixel in an ARGBFFFF image to the smaller of two values—either the color channel or the alpha value for that pixel.
`vImageClipToAlpha_Planar8`	Sets the color channel of each pixel in a Planar8 image to the smaller of two values—either the color channel or the alpha value for that pixel.
`vImageClipToAlpha_PlanarF`	Sets the color channel of each pixel in a PlanarF image to the smaller of two values—either the color channel or the alpha value for that pixel.
`vImagePremultipliedConstAlphaBlend_ARGB8888`	Performs premultiplied alpha compositing of two ARGB8888 images, using a single alpha value for the whole image and placing the result in a destination buffer.
`vImagePremultipliedConstAlphaBlend_ARGBFFFF`	Performs premultiplied alpha compositing of two ARGBFFFF images, using a single alpha value for the whole image and placing the result in a destination buffer.
`vImagePremultipliedConstAlphaBlend_Planar8`	Performs premultiplied alpha compositing of two Planar8 images, using a single alpha value for the entire image and placing the result in a destination buffer.
`vImagePremultipliedConstAlphaBlend_PlanarF`	Performs premultiplied alpha compositing of a two PlanarF images, using a single alpha value for the whole image and placing the result in a destination buffer.
`vImagePremultiplyData_RGBA8888`	Takes an RGBA8888 image in nonpremultiplied alpha format and transforms it into an image in premultiplied alpha format.
`vImagePremultiplyData_RGBAFFFF`	Takes an RGBAFFFF image in nonpremultiplied alpha format and transforms it into an image in premultiplied alpha format.

`vImageUnpremultiplyData_RGBA8888`	Takes an RGBA8888 image in premultiplied alpha format and transforms it into an image in nonpremultiplied alpha format.
`vImageUnpremultiplyData_RGBAFFFF`	Takes an RGBAFFFF image in premultiplied alpha format and transforms it into an image in nonpremultiplied alpha format.

BasicImageTypes.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vImagePNGDecompressionFilter`	Performs PNG decompression filtering.

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`kvImage_PNG_FILTER_VALUE_AVG`	A filter that predicts a pixel value from the average of the pixels to the left and above the predicted pixel location.
`kvImage_PNG_FILTER_VALUE_NONE`	No filtering.
`kvImage_PNG_FILTER_VALUE_PAETH`	A filter that predicts a pixel value by applying a linear function to the pixels located to the left, above, and to the upper left of the predicted pixel location.
`kvImage_PNG_FILTER_VALUE_SUB`	A filter that computes the difference between each byte of a pixel and the value of the corresponding byte of the pixel located to the left.
`kvImage_PNG_FILTER_VALUE_UP`	A filter that computes the difference between each byte of a pixel and the value of the corresponding byte of the pixel located above.

Conversion.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vImageBufferFill_ARGB8888`	Fills an ARGB8888 buffer with a specified color.

vImageBufferFill_ARGBFFFF	Fills an ARGBFFFF buffer with a specified color.
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.
vImageConvert_ARGB1555toARGB8888	Converts an ARGB1555 image to an ARGB8888 image.
vImageConvert_ARGB1555toPlanar8	Separates an ARGB1555 image into four Planar8 images.
vImageConvert_ARGB8888toARGB1555	Converts an ARGB8888 image into an ARGB1555 image.
vImageConvert_ARGB8888toRGB565	Converts an ARGB8888 image into an RGB565 image.
vImageConvert_ARGB8888toRGB888	Converts an ARGB8888 image into an RGB888 image.
vImageConvert_Planar16FtoPlanarF	Converts a Planar16F image to a PlanarF image.
vImageConvert_Planar8To16U	Converts a Planar8 image to a 16U image .
vImageConvert_Planar8toARGB1555	Combines four Planar8 images into one ARGB1555 image.
vImageConvert_Planar8toRGB565	Combines three Planar8 images into one RGB565 image.
vImageConvert_Planar8toRGB888	Combines three Planar8 images into one RGB888 image.
vImageConvert_PlanarFtoPlanar16F	Converts a PlanarF image to a Planar16F image.
vImageConvert_PlanarFtoRGBFFF	Combines three PlanarF images into one RGBFFF image.
vImageConvert_RGB565toARGB8888	Converts an RGB565 image into an ARGB8888 image, using the provided 8-bit alpha value.
vImageConvert_RGB565toPlanar8	Separates an RGB565 image into three Planar8 images.
vImageConvert_RGB888toARGB8888	Converts an RGB888 image into an ARGB8888 image, using the provided alpha value (either as planar or pixel data).
vImageConvert_RGB888toPlanar8	Separates an RGB888 image into three Planar8 images.
vImageConvert_RGBFFFtoPlanarF	Separates an RGBFFF image into three PlanarF images.

vImageFlatten_ARGB8888ToRGB888	Transforms an ARGB8888 image to an RGB888 image against an opaque background of the provided color.
vImageFlatten_ARGBFFFFToRGBFFF	Transforms an ARGBFFFF image to an RGBFFF image against an opaque background of the provided color.
vImageOverwriteChannelsWithScalar_ARGB8888	Overwrites the pixels of one or more planes of an ARGB8888 image buffer with the provided scalar value.
vImageOverwriteChannelsWithScalar_ARGBFFFF	Overwrites the pixels of one or more planes of an ARGBFFFF image buffer with the provided scalar value.
vImageOverwriteChannelsWithScalar_Planar8	Overwrites a Planar8 image buffer with the provided value.
vImageOverwriteChannelsWithScalar_PlanarF	Overwrites a PlanarF image buffer with the provided value.
vImageOverwriteChannels_ARGB8888	Overwrites one or more planes of an ARGB8888 image buffer with the provided planar buffer.
vImageOverwriteChannels_ARGBFFFF	Overwrites one or more planes of an ARGBFFFF image buffer with the provided planar buffer.
vImagePermuteChannels_ARGB8888	Reorders the channels in an ARGB8888 image.
vImagePermuteChannels_ARGBFFFF	Reorders the channels in an ARGBFFFF image.

Convolution.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

vImageBoxConvolve_ARGB8888	Convolves a region of interest within an ARGB8888 source image by an implicit M x N kernel that has the effect of a box filter.
vImageBoxConvolve_Planar8	Convolves a region of interest within a Planar8 source image by an implicit M x N kernel that has the effect of a box filter.
vImageConvolveMultiKernel_ARGB8888	Convolves each channel of a region of interest within an ARGB8888 source image by one of the four M x N kernels, then divides the pixel values by one of the four divisors.

vImageConvolveMultiKernel_ARGBFFFF	Convolves each channel of a region of interest within an ARGBFFFF source image by one of the four M x N kernels.
vImageConvolveWithBias_ARGB8888	Convolves a region of interest within an ARGB8888 source image by an M x N kernel, then normalizes the pixel values.
vImageConvolveWithBias_ARGBFFFF	Convolves a region of interest within an ARGBFFFF source image by an M x N kernel.
vImageConvolveWithBias_Planar8	Convolves a region of interest within a Planar8 source image by an M x N kernel, then normalizes the pixel values.
vImageConvolveWithBias_PlanarF	Convolves a region of interest within a PlanarF source image by an M x N kernel.
vImageRichardsonLucyDeConvolve_ARGB8888	Sharpens an ARGB8888 image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.
vImageRichardsonLucyDeConvolve_ARGBFFFF	Sharpens an ARGBFFFF image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.
vImageRichardsonLucyDeConvolve_Planar8	Sharpens a Planar8 image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.
vImageRichardsonLucyDeConvolve_PlanarF	Sharpens a PlanarF image by undoing a previous convolution that blurred the image, such as diffraction effects in a camera lens.
vImageTentConvolve_ARGB8888	Convolves a region of interest within an ARGB8888 source image by an implicit M x N kernel that has the effect of a tent filter.
vImageTentConvolve_Planar8	Convolves a region of interest within a Planar8 source image by an implicit M x N kernel that has the effect of a tent filter.

Transform.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

vImageCreateGammaFunction	Returns a gamma function object.
vImageDestroyGammaFunction	Destroys a gamma function object created.

`vImageGamma_Planar8toPlanarF`	Applies a gamma function to a Planar8 image to produce a PlanarF image.
`vImageGamma_PlanarF`	Applies a gamma function to a PlanarF image.
`vImageGamma_PlanarFtoPlanar8`	Applies a gamma function to an image in PlanarF format to an image in Planar8 format.
`vImageInterpolatedLookupTable_PlanarF`	Uses a lookup table to transform an image in PlanarF format to an image in PlanarF format.
`vImageLookupTable_Planar8toPlanarF`	Uses a lookup table to transform an image in Planar8 format to an image in PlanarF format.
`vImageLookupTable_PlanarFtoPlanar8`	Uses a lookup table to transform an image in PlanarF format to an image in Planar8 format.
`vImageMatrixMultiply_ARGB8888`	Operates upon an interleaved 8-bit source image, multiplying each pixel by the provided matrix to produce an interleaved 8-bit destination image.
`vImageMatrixMultiply_ARGBFFFF`	Operates upon an interleaved floating-point source image, multiplying each pixel by the provided matrix to produce an interleaved floating-point destination image.
`vImageMatrixMultiply_Planar8`	Operates on a set of 8-bit source image planes, multiplying each pixel by the provided matrix to produce a set of 8-bit destination image planes.
`vImageMatrixMultiply_PlanarF`	Operates upon a set of floating-point source image planes, multiplying each pixel by the provided matrix to produce a set of floating-point destination image planes.
`vImagePiecewisePolynomial_Planar8toPlanarF`	Applies a set of piecewise polynomials to transform an image in Planar8 format to an image in PlanarF format.
`vImagePiecewisePolynomial_PlanarF`	Applies a set of piecewise polynomials to an image in PlanarF format.
`vImagePiecewisePolynomial_PlanarFtoPlanar8`	Applies a set of piecewise polynomials to transform an image in PlanarF format to an image in Planar8 format.
`vImagePiecewiseRational_PlanarF`	Applies a piecewise rational expression to an image in PlanarF format.

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`kvImageGamma_11_over_5_half_precision`	Half-precision calculation using a gamma value of 11/5 or 2.2. On exit, gamma is 5/11.
`kvImageGamma_11_over_9_half_precision`	Half-precision calculation using a gamma value of 11/9 or (11/5)/(9/5).
`kvImageGamma_5_over_11_half_precision`	Half-precision calculation using a gamma value of 5/11 or 1/2.2.
`kvImageGamma_5_over_9_half_precision`	Half-precision calculation using a gamma value of 5/9 or 1/1.8.
`kvImageGamma_9_over_11_half_precision`	Half-precision calculation using a gamma value of 9/11 or (9/5)/(11/5).
`kvImageGamma_9_over_5_half_precision`	Half-precision calculation using a gamma value of 9/5 or 1.8.
`kvImageGamma_BT709_forward_half_precision`	ITU-R BT.709 standard. This is like kvImageGamma_sRGB_forward_half_precision above but without the 1.125 viewing gamma for computer graphics: x<0.081? x/4.5: pow((x+0.099)/1.099, 1/0.45).
`kvImageGamma_BT709_reverse_half_precision`	ITU-R BT.709 standard reverse. This is like kvImageGamma_sRGB_reverse_half_precision above but without the 1.125 viewing gamma for computer graphics: x<0.018? 4.5*x: 1.099*pow(x,0.45) - 0.099.
`kvImageGamma_sRGB_forward_half_precision`	Half-precision calculation using the sRGB standard gamma value of 2.2.
`kvImageGamma_sRGB_reverse_half_precision`	Half-precision calculation using the sRGB standard gamma value of 1/2.2.
`kvImageGamma_UseGammaValue`	Full-precision calculation using the gamma value set in vImageCreateGammaFunction.
`kvImageGamma_UseGammaValue_half_precision`	Half-precision calculation using the gamma value set in vImageCreateGammaFunction.

vImage_Types.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`GammaFunction`	A type for a gamma function.

`kvImageGetTempBufferSize`	Get the minimum temporary buffer size for the operation, given the parameters provided. When you set this flag, the function returns the number of bytes required for the temporary buffer. A negative value specifies an error.
`kvImageTruncateKernel`	Use the part of the kernel that overlaps the image. This flag is valid only for convolution operations. When you set this flag, vImage restricts calculations to the portion of the kernel overlapping the image. It corrects the calculated pixel by first multiplying by the sum of all the kernel elements, then dividing by the sum of the kernel elements that are actually used. This preserves image brightness at the edges.
`vImagePixelCount`	A type for the number of pixels.

10.3 Symbol Changes

This article lists the symbols added to `Accelerate.framework` in Mac OS X v10.3.

C Symbols

All of the header files with new symbols are listed alphabetically, with their new symbols described.

vecLib

cblas.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`dMultMatMat_16x16`	
`dMultMatMat_32x32`	
`dMultMatMat_4x4`	
`dMultMatMat_8x8`	
`dMultMatVec_16x16`	
`dMultMatVec_32x32`	
`dMultMatVec_4x4`	
`dMultMatVec_8x8`	
`dMultVecMat_16x16`	
`dMultVecMat_32x32`	
`dMultVecMat_4x4`	
`dMultVecMat_8x8`	

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`VectorFloat`	

vBigNum.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`v3`	
`v4`	
`v5`	
`v6`	
`v7`	
`v8`	

vecLibTypes.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vBool32`	A 128-bit vector packed with bool int values.
`vFloat`	A 128-bit vector packed with float values.
`vSInt16`	A 128-bit vector packed with signed short values.
`vSInt32`	A 128-bit vector packed with signed int values.
`vSInt8`	A 128-bit vector packed with signed char values.
`vUInt16`	A 128-bit vector packed with unsigned short values.
`vUInt32`	A 128-bit vector packed with unsigned int values.
`vUInt8`	A 128-bit vector packed with unsigned char values.

vImage

Alpha.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vImageAlphaBlend_ARGB8888`	Performs nonpremultiplied alpha compositing of two ARGB8888 images, placing the result in a destination buffer.
`vImageAlphaBlend_ARGBFFFF`	Performs nonpremultiplied alpha compositing of two ARGBFFFF images, placing the result in a destination buffer.
`vImageAlphaBlend_Planar8`	Performs nonpremultiplied alpha compositing of two Planar8 images, placing the result in a destination buffer.
`vImageAlphaBlend_PlanarF`	Performs nonpremultiplied alpha compositing of two PlanarF images, placing the result in a destination buffer.
`vImagePremultipliedAlphaBlend_ARGB8888`	Performs premultiplied alpha compositing of two ARGB8888 images, placing the result in a destination buffer.
`vImagePremultipliedAlphaBlend_ARGBFFFF`	Performs premultiplied alpha compositing of two ARGBFFFF images, placing the result in a destination buffer.
`vImagePremultipliedAlphaBlend_Planar8`	Performs premultiplied alpha compositing of two Planar8 images, placing the result in a destination buffer.
`vImagePremultipliedAlphaBlend_PlanarF`	Performs premultiplied alpha compositing of two PlanarF images, placing the result in a destination buffer.
`vImagePremultiplyData_ARGB8888`	Takes an ARGB8888 image in nonpremultiplied alpha format and transforms it into an image in premultiplied alpha format.
`vImagePremultiplyData_ARGBFFFF`	Takes an ARGBFFFF image in nonpremultiplied alpha format and transforms it into an image in premultiplied alpha format.
`vImagePremultiplyData_Planar8`	Takes a Planar8 image in nonpremultiplied alpha format, along with alpha information, and transforms it into an image in premultiplied alpha format.

`vImagePremultiplyData_PlanarF`	Takes a PlanarF image in nonpremultiplied alpha format, along with alpha information, and transforms it into an image in premultiplied alpha format.
`vImageUnpremultiplyData_ARGB8888`	Takes an ARGB8888 image in premultiplied alpha format and transforms it into an image in nonpremultiplied alpha format.
`vImageUnpremultiplyData_ARGBFFFF`	Takes an ARGBFFFF image in premultiplied alpha format and transforms it into an image in nonpremultiplied alpha format.
`vImageUnpremultiplyData_Planar8`	Takes a Planar8 image in premultiplied alpha format, along with alpha information, and transforms it into an image in nonpremultiplied alpha format.
`vImageUnpremultiplyData_PlanarF`	Takes a PlanarF image in premultiplied alpha format, along with alpha information, and transforms it into an image in nonpremultiplied alpha format.

Conversion.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vImageClip_PlanarF`	Clips the pixel values of an image in PlanarF format, using the provided minimum and maximum values.
`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.
`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.
`vImageConvert_ARGB8888toPlanar8`	Separates an ARGB8888 image into four Planar8 images.
`vImageConvert_ARGBFFFFtoPlanarF`	Separates an ARGBFFFF image into four PlanarF images.
`vImageConvert_ChunkyToPlanar8`	Separates a source image into a collection of corresponding planar destination images, one for each 8-bit channel of the original image.
`vImageConvert_ChunkyToPlanarF`	Separates a source image into a collection of corresponding planar destination images, one for each floating-point channel of the original image.
`vImageConvert_FTo16S`	Converts a PlanarF image into a special format in which each pixel is a 16-bit signed integer.

`vImageConvert_FTo16U`	Converts a PlanarF image into a special format in which each pixel is a 16-bit unsigned integer.
`vImageConvert_Planar8toARGB8888`	Combines four Planar8 images into one ARGB8888 image.
`vImageConvert_Planar8toPlanarF`	Converts a Planar8 image to a PlanarF image.
`vImageConvert_PlanarFtoARGBFFFF`	Combines four PlanarF images into one ARGBFFFF image.
`vImageConvert_PlanarFtoPlanar8`	Converts a PlanarF image to a Planar8 image, clipping values to the provided minimum and maximum values.
`vImageConvert_PlanarToChunky8`	Combines a collection of planar source images into a single interleaved destination image, with one 8-bit channel for each planar image.
`vImageConvert_PlanarToChunkyF`	Combines a collection of planar source images into a single interleaved destination image, with one floating-point channel for each planar image.
`vImageTableLookUp_ARGB8888`	Transforms an ARGB8888 image by substituting pixel values with pixel values provided by four lookup tables.
`vImageTableLookUp_Planar8`	Transforms an Planar8 image by substituting pixel values with pixel values provided by four lookup tables.

Convolution.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vImageConvolve_ARGB8888`	Convolves a region of interest within a source image by an M x N kernel, then divides the pixel values by a divisor.
`vImageConvolve_ARGBFFFF`	Convolves a region of interest within an ARGBFFFF source image by an M x N kernel.
`vImageConvolve_Planar8`	Convolves a region of interest within a source image by an M x N kernel, then divides the pixel values by a divisor.
`vImageConvolve_PlanarF`	Convolves a region of interest within a source image by an M x N kernel.
`vImageGetMinimumTempBufferSizeForConvolution`	Returns the minimum size, in bytes, for the temporary buffer that the caller supplies to any of the convolution functions.

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`dataIs1Channel`	
`dataIs8Bits`	
`printImageData`	

Geometry.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

`vImageAffineWarp_ARGB8888`	Applies an affine transform to an ARGB8888 source image.
`vImageAffineWarp_ARGBFFFF`	Applies an affine transform to an ARGBFFFF source image.
`vImageAffineWarp_Planar8`	Applies an affine transform to a Planar8 source image.
`vImageAffineWarp_PlanarF`	Applies an affine transform to a PlanarF source image.
`vImageDestroyResamplingFilter`	Disposes of a resampling filter object.
`vImageGetMinimumGeometryTempBufferSize`	Returns the minimum size, in bytes, for the temporary buffer needed by a high-level geometry function.
`vImageGetResamplingFilterSize`	Returns the minimum size, in bytes, for the buffer needed by the function vImageNewResamplingFilter-ForFunctionUsingBuffer.
`vImageHorizontalReflect_ARGB8888`	Reflects an ARGB8888 source image left to right across the center vertical line of the image.
`vImageHorizontalReflect_ARGBFFFF`	Reflects an ARGBFFFF source image left to right across the center vertical line of the image.
`vImageHorizontalReflect_Planar8`	Reflects a Planar9 source image left to right across the center vertical line of the image.

`vImageHorizontalReflect_PlanarF`	Reflects a PlanarF source image left to right across the center vertical line of the image, placing the result in a destination buffer.
`vImageHorizontalShear_ARGB8888`	Performs a horizontal shear operation on a region of interest of an ARGB8888 source image.
`vImageHorizontalShear_ARGBFFFF`	Performs a horizontal shear operation on a region of interest of an ARGBFFFF source image.
`vImageHorizontalShear_Planar8`	Performs a horizontal shear operation on a region of interest of a Planar8 source image.
`vImageHorizontalShear_PlanarF`	Performs a horizontal shear operation on a region of interest of a PlanarF source image.
`vImageNewResamplingFilter`	Creates a resampling filter object that corresponds to the default kernel supplied by the vImage framework.
`vImageNewResamplingFilterForFunctionUsingBuffer`	Creates a resampling filter object that encapsulates a resampling kernel function that you provide.
`vImageRotate90_ARGB8888`	Rotates an ARGB8888 source image by the provided factor of 90.
`vImageRotate90_ARGBFFFF`	Rotates an ARGBFFFF source image by the provided factor of 90.
`vImageRotate90_Planar8`	Rotates a Planar8 source image by the provided factor of 90.
`vImageRotate90_PlanarF`	Rotates a PlanarF source image by the provided factor of 90.
`vImageRotate_ARGB8888`	Rotates an ARGB8888 source image by the provided angle.
`vImageRotate_ARGBFFFF`	Rotates an ARGBFFFF source image by the provided angle.
`vImageRotate_Planar8`	Rotates a Planar8 source image by the provided angle.
`vImageRotate_PlanarF`	Rotates a PlanarF source image by the provided angle.
`vImageScale_ARGB8888`	Scales an ARGB8888 source image to fit a destination buffer.

vImageScale_ARGBFFFF	Scales an ARGBFFFF source image to fit a destination buffer.
vImageScale_Planar8	Scales a Planar8 source image to fit a destination buffer.
vImageScale_PlanarF	Scales a PlanarF source image to fit a destination buffer.
vImageVerticalReflect_ARGB8888	Reflects an ARGBFFFF source image top to bottom across the center vertical line of the image.
vImageVerticalReflect_ARGBFFFF	Reflects an ARGBFFFF source image top to bottom across the center vertical line of the image.
vImageVerticalReflect_Planar8	Reflects a Planar 8 source image top to bottom across the center vertical line of the image.
vImageVerticalReflect_PlanarF	Reflects a PlanarF source image top to bottom across the center vertical line of the image.
vImageVerticalShear_ARGB8888	Performs a vertical shear operation on a region of interest of an ARGB8888 source image.
vImageVerticalShear_ARGBFFFF	Performs a vertical shear operation on a region of interest of an ARGBFFFF source image.
vImageVerticalShear_Planar8	Performs a vertical shear operation on a region of interest of a Planar8 source image.
vImageVerticalShear_PlanarF	Performs a vertical shear operation on a region of interest of a PlanarF source image.

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kRotate0DegreesClockwise	Rotate 0 degrees (that is, copy without rotating).
kRotate0DegreesCounterClockwise	Rotate 0 degrees (that is, copy without rotating).
kRotate180DegreesClockwise	Rotate 180 degrees clockwise.
kRotate180DegreesCounterClockwise	Rotate 180 degrees counter-clockwise.

kRotate270DegreesClockwise	Rotate 270 degrees clockwise.
kRotate270DegreesCounterClockwise	Rotate 270 degrees counter-clockwise.
kRotate90DegreesClockwise	Rotate 90 degrees clockwise.
kRotate90DegreesCounterClockwise	Rotate 90 degrees counter-clockwise.

Histogram.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

vImageContrastStretch_ARGB8888	Stretches the contrast of an ARGB8888 source image.
vImageContrastStretch_ARGBFFFF	Stretches the contrast of an ARGBFFFF source image.
vImageContrastStretch_Planar8	Stretches the contrast of a Planar8 source image.
vImageContrastStretch_PlanarF	Stretches the contrast of a PlanarF source image.
vImageEndsInContrastStretch_ARGB8888	Performs an ends-in contrast stretch operation on an ARGB8888 source image.
vImageEndsInContrastStretch_ARGBFFFF	Performs an ends-in contrast stretch operation on an ARGBFFFF source image.
vImageEndsInContrastStretch_Planar8	Performs an ends-in contrast stretch operation on a Planar8 source image.
vImageEndsInContrastStretch_PlanarF	Performs an ends-in contrast stretch operation on a PlanarF source image.
vImageEqualization_ARGB8888	Equalizes the histogram of an ARGB8888 source image.
vImageEqualization_ARGBFFFF	Equalizes the histogram of an ARGBFFFF source image.
vImageEqualization_Planar8	Equalizes the histogram of an ARGB8888 source image.
vImageEqualization_PlanarF	Equalizes the histogram of a PlanarF source image.
vImageGetMinimumTempBufferSizeForHistogram	Returns the minimum size, in bytes, for the temporary buffer needed by a histogram function.

vImageHistogramCalculation_ARGB8888	Calculates histograms for each channel of an ARGB8888 image.
vImageHistogramCalculation_ARGBFFFF	Calculates histograms for each channel of an ARGBFFFF image.
vImageHistogramCalculation_Planar8	Calculates a histogram for a Planar8 image.
vImageHistogramCalculation_PlanarF	Calculates the histogram a PlanarF image.
vImageHistogramSpecification_ARGB8888	Performs a histogram specification operation on an ARGB8888 source image.
vImageHistogramSpecification_ARGBFFFF	Performs a histogram specification operation on an ARGBFFFF source image.
vImageHistogramSpecification_Planar8	Performs a histogram specification operation on a Planar8 source image.
vImageHistogramSpecification_PlanarF	Performs a histogram specification operation on a PlanarF source image.

Morphology.h

Functions

All of the new functions in this header file are listed alphabetically, with links to documentation and abstracts, if available.

vImageDilate_ARGB8888	Dilates a region of interest within an ARGB8888 source image using an M x N kernel.
vImageDilate_ARGBFFFF	Dilates a region of interest within an ARGBFFFF source image using an M x N kernel.
vImageDilate_Planar8	Dilates a region of interest within a Planar8 source image using an M x N kernel.
vImageDilate_PlanarF	Dilates a region of interest within a PlanarF source image using an M x N kernel.
vImageErode_ARGB8888	Erodes a region of interest within an ARGB8888 source image using an M x N kernel.
vImageErode_ARGBFFFF	Erodes a region of interest within an ARGBFFFF source image using an M x N kernel.
vImageErode_Planar8	Erodes a region of interest within a Planar8 source image using an M x N kernel.
vImageErode_PlanarF	Erodes a region of interest within a PlanarF source image using an M x N kernel.

vImageGetMinimumTempBufferSizeForMinMax	Returns the minimum size, in bytes, for the temporary buffer that the caller supplies to any of the Min or Max morphological functions.
vImageMax_ARGB8888	Maximizes a region of interest within an ARGB8888 source image using an M x N kernel.
vImageMax_ARGBFFFF	Maximizes a region of interest within an ARGBFFFF source image using an M x N kernel.
vImageMax_Planar8	Maximizes a region of interest within a Planar8 source image using an M x N kernel.
vImageMax_PlanarF	Maximizes with a region of interest within a PlanarF source image using an M x N kernel.
vImageMin_ARGB8888	Minimizes a region of interest within an ARGB8888 source image using an M x N kernel.
vImageMin_ARGBFFFF	Minimizes a region of interest within an ARGBFFFF source image using an M x N kernel.
vImageMin_Planar8	Minimizes a region of interest within a Planar8 source image using an M x N kernel.
vImageMin_PlanarF	Minimizes a region of interest within a PlanarF source image using an M x N kernel.

vImage_Types.h

Data Types & Constants

All of the new data types and constants in this header file are listed alphabetically, with links to documentation and abstracts, if available.

kvImageBackgroundColorFill	A background color fill. The associated value is a background color (that is, a pixel value). When you set this flag, vImage assigns the pixel value to all pixels outside the image. You can set this flag for convolution and geometry functions. The morphology functions do not use this flag because they do not use pixels outside the image in any of their calculations.
kvImageBufferSizeMismatch	The function requires the source and destination buffers to have the same height and the same width, but they do not.

`kvImageCopyInPlace`	Copy the value of the edge pixel in the source to the destination. When you set this flag, and a convolution function is processing an image pixel for which some of the kernel extends beyond the image boundaries, vImage does not computer the convolution. Instead, the value of the particular pixel unchanged; it's simply copied to the destination image. This flag is valid only for convolution operations. The morphology functions do not use this flag because they do not use pixels outside the image in any of their calculations.
`kvImageDoNotTile`	Do not use vImage internal tiling routines. When you set this flag, vImage turns off internal tiling. Set this flag if you want to perform your own tiling or your own multithreading, or to use the minimum or maximum filters in place.
`kvImageEdgeExtend`	Extend the edges of the image infinitely. When you set this flag, vImage replicates the edges of the image outward. It repeats the top row of the image infinitely above the image, the bottom row infinitely below the image, the first column infinitely to the left of the image, and the last column infinitely to the right. For spaces that are diagonal to the image, vImage uses the value of the corresponding corner pixel. For example, for all pixels that are both above and to the left of the image, the upper-leftmost pixel of the image (the pixel at row 0, column 0) supplies the value. In this way, vImage assigns every pixel location outside the image some value. You can set this flag for convolution and geometry functions. The morphology functions do not use this flag because they do not use pixels outside the image in any of their calculations.
`kvImageHighQualityResampling`	Use a higher quality, slower resampling filter for for geometry operations—shear, scale, rotate, affine transform, and so forth.
`kvImageInvalidEdgeStyle`	
`kvImageInvalidKernelSize`	Either the kernel height, the kernel width, or both, are even.
`kvImageInvalidOffset_X`	The srcOffsetToROI_X parameter that specifies the left edge of the region of interest is greater than the width of the source image.
`kvImageInvalidOffset_Y`	The srcOffsetToROI_Y parameter that specifies the top edge of the region of interest is greater than the height of the source image.
`kvImageInvalidParameter`	Invalid parameter.
`kvImageLeaveAlphaUnchanged`	Operate on red, green, and blue channels only. When you set this flag, the alpha value is copied from source to destination. You can set this flag only for interleaved image formats.
`kvImageMemoryAllocationError`	An attempt to allocate memory failed.
`kvImageNoError`	The vImage function completed without error.

`kvImageNoFlags`	Do not set any flags.
`kvImageNullPointerArgument`	A pointer parameter is NULL and it must not be.
`kvImageRoiLargerThanInputBuffer`	The region of interest, as specified by the srcOffsetToROI_X and srcOffsetToROI_Y parameters and the height and width of the destination buffer, extends beyond the bottom edge or right edge of the source buffer.
`kvImageUnknownFlagsBit`	The flag is not recognized.
`Pixel_8`	A type for an 8-bit planar pixel value
`Pixel_8888`	A type for an interleaved, 8 bits per channel pixel value.
`Pixel_F`	A type for a floating-point planar pixel value
`Pixel_FFFF`	A type for an interleaved, floating-point pixel value.
`ResamplingFilter`	A pointer to a resampling filter callback function.
`vImage_Error`	A type for image errors.
`vImage_Flags`	A type for processing options.

Document Revision History

The table below describes the revisions to *Accelerate Reference Update*.

This table describes the changes to *Accelerate Reference Update*.

Date	Notes
2007-07-18	Updated with the symbols added to the Accelerate framework in Mac OS X v10.5.
2005-04-29	New document that summarizes the symbols added to the Accelerate framework in Mac OS X v10.4.