Datamanip Toolbox Manual

Chapter 1

Introduction

Chapter 1 - Introduction

A. The Datamanip Toolbox

The datamanip toolbox contains general data manipulation operators, and the kdatamanip and ksegops libraries. Data manipulation operators include pointwise arithmetic, statistics calculations, data conversion, histogram, data organization, and size operators.

All datamanip programs are written to support the relationships defined by the polymorphic data model. This data model provides for general 5-dimensional data (width, height, depth, time, elements), with any combination of value, map, mask, location, and time data components. the datamanip operators are implemented using polymorphic data services, which is a part of the kappserv library in the Dataserv toolbox.

The datamanip toolbox requires the Bootstrap and Dataserv toolboxes to be installed.

B. General Operator Behavior

The following sections describe general behavior of operators with respect to the polymorphic data model. In the printed manual, these sections are referenced in the object's man page when they apply, and then exceptions to these rules are explicitly called out. When using online help and kman, these sections will be included in the text. The part of the system that helps generate this type of referencing is still under development, so sections are not called out in the manual pages - rather, the path to the general information file is given. This will be fixed by the next release of the system.

B.1. Data Type - Single Input (value_type_1input)

include file `$DATAMANIP/repos/shared/man/sections/value_type_1input`

The data type of the output object is the same as the input object's data type. Internally, the data is processed using one of the following: unsigned byte, long, unsigned long, double, or double complex. Data will not be cast to a lower type for processing.

B.2. Data Type - Two Input Objects (value_type_2input)

include file `$DATAMANIP/repos/shared/man/sections/value_type_2input`

The data type of the output object's data is cast to the highest order data type of the two input objects' data. Internally, the data is processed using one of the following: unsigned byte, long, unsigned long, double, or double complex. Data is not cast to a lower type for processing.

B.3. Map Data - Single Input (map_1input)

include file `$DATAMANIP/repos/shared/man/sections/map_1input`

If the input object contains a map, then the operation is performed on the map data. If the input object does not have a map, and has value data, then the operation is performed on the value data.

B.4. Map Data - Two Input Objects (map_2input)

include file `$DATAMANIP/repos/shared/man/sections/map_2input`

If either input object contains map data as well as value data, then the value data is mapped through the map before the operation is performed. The output object will contain value data with no associated map, and its data type is the highest of the input (map) data types.

If the data has a validity mask as well as a map, the mapped data is also masked. When mapping a value datum that has a mask value of zero (invalid), the mapping process will first look for an entry in the map that corresponds to the "invalid" value, if one does not exist (for example, the value is NaN), the mapped data is set to the pad value. In either case, the data remains masked. The mask is resized to correspond to the size of the mapped value data to properly mask it. For example, the value data may have the dimensions of

width = 100,
height = 50,
elements = depth = time = 1.

If the map dimensions are

map width = 3,
map height = whatever,
map element = map time = map spatial = 1,

the resulting value (and mask) size will be

width = 100,
height = 50,
elements = 3,
depth = time = 1.

B.5. Validity Mask - Single Input (mask_1input)

include file `$DATAMANIP/repos/shared/man/sections/mask_1input`

If there is a validity mask associated with the input object, the mask is transferred to the output object. To decrease processing time, the operation will still be performed on those data that have corresponding mask values equal to zero. If the data under the mask is genuinely *not* processable, such as NaN or infinity, the user can specify a that a value be substituted in place of the original value when it is accessed. The masked data substitution attributes for a data object can be explicitly set and stored using the *Set Attribute* operator, DATAMANIP::ksetdattr.

B.6. Validity Mask - Multiple Input Objects (mask_2input)

include file `$DATAMANIP/repos/shared/man/sections/mask_2input`

If there is a validity mask associated with any of the input objects, the resulting output object will have a mask. The output object mask is the logical AND of the input object masks. If only one input object has a mask, then the other input object's mask is assumed to be "all valid" (mask value = 1). If a mask is resized for the operation, the mask padding value will be 1.

To decrease processing time, the operation will still be performed on data that have corresponding mask values equal to zero (invalid data). If the data under the mask is genuinely *not* processable, such as NaN or infinity, the user can specify a that a value be substituted in place of the original value when it is accessed. The masked data substitution attributes for a data object can be explicitly set and stored using the *Set Attribute* operator, DATAMANIP::ksetdattr.

B.7. Input Objects of Different Sizes (resize_2input)

include file `$DATAMANIP/repos/shared/man/sections/resize_2input`

The input objects can be of different sizes. If the sizes are different, the size of the output object's data will be the maximum of the data sizes of the input objects. When resizing, the input data are padded with a constant value that is defined internally by the operation being performed. If the input objects have map data as well as value data, the maximum data size is determined after mapping.

The subobject position attribute of Input 2 is applied as an offset before the final output object size is determined and before the operation is performed. The subobject position of the first input object is not applied.

The subobject position attribute of a data object defines the coordinates at which that subobject was located in its parent data object. This attribute may be automatically set by programs such as the Extract operator (DATAMANIP::kextract), or it can be explicitly set by using the Set Data Attributes operator (DATAMANIP::ksetdattr). The subobject position coordinates can be printed using the Print File Information operator (DATAMANIP::kfileinfo).

B.8. Explicit Location and Time Data - Single Input (loc_and_time_1input)

include file `$DATAMANIP/repos/shared/man/sections/loc_and_time_1input`

If the input object has time data or location data, it is transferred to the output object.

B.9. Explicit Location and Time Data - Two Input Objects (loc_and_time_2input)

include file `$DATAMANIP/repos/shared/man/sections/loc_and_time_2input`

This routine does not interpret location or time data. Rather, it operates on the implicit data space. Therefore, if more than one input object is supplied, and location or time data exist, the first input object will be the dominant object, and all attributes will be transferred from the first object to the output object (regardless of whether the other inputs have location or time data). Padding or truncation of explicit location data may occur if the resulting output object size is different than the size of input 1.

If the first input object does not contain location or time data, but another input object does contain it, the location or time data will not be propagated.

B.10. Failure Modes - Single Input (fail_1input)

include file `$DATAMANIP/repos/shared/man/sections/fail_1input`

This program fails if the input object lacks both map data and value data.

B.11. Failure Modes - Two Input Objects (fail_2input)

include file `$DATAMANIP/repos/shared/man/sections/fail_2input`

This program can only operate on input objects if both have value data (either or both can have optional map data) or if both *only* have map data.

B.12. Mapping Data (map_force)

include file `$DATAMANIP/repos/shared/man/sections/map_force`

If the source object contains map data as well as value data, then the value data will be mapped through the map (the value data type becomes that of the map) before the operation is performed. The destination object contains value data with no associated map.

If the data has a validity mask as well as a map, the mapped data is also masked. When mapping a value datum that has a mask value of zero (invalid), the mapping process will first look for an entry in the map that corresponds to the "invalid" value, if one does not exist (for example, the value is NaN), the mapped data is set to the pad value. In either case, the data remains masked. The mask is resized to correspond to the size of the mapped value data to properly mask it. For example, the value data may have the dimensions of

    width = 100,
    height = 50,
    elements = depth = time = 1.

If the map dimensions are

    map width = 3,
    map height = whatever,
    map element = map time = map spatial = 1,

the resulting value (and mask) size will be

    width = 100,
    height = 50,
    elements = 3,
    depth = time = 1.

C. Polymorphic Operators Organized by Category/Subcategory

The following table contains a listing of the operators that are available in the Datamanip toolbox. Table entries are organized by category and subcategory. A brief description of each operator is given, followed by name of the executable.

DATAMANIP — Polymorphic Data Processing Operators

Category Subcategory	Operator	Description	Executable
Arithmetic Bitwise Operators	AND	Output = Input 1 AND (Input 2 or Constant)	kbitand
	AND Inverted	Output = NOT(Input 1) AND (Input 2 or Constant)	kbitandinv
	AND Reverse	Output = Input 1 AND NOT(Input 2 or Constant)	kbitandrev
	CLEAR	Set All Bits to Zero	kbitclear
	Left Shift	Bitwise LEFT SHIFT of Input 1 by (Input 2 or Constant) Bits	kbitlshift
	NAND	Output = NOT(Input 1) OR NOT(Input 2 or Constant)	kbitnand
	NOR	Output = NOT(Input 1) AND NOT(Input 2 or Constant)	kbitnor
	NOT	Perform Bitwise NOT (Invert) Operation on Input	kbitnot
	OR	Output = Input 1 OR (Input 2 or Constant)	kbitor
	OR Inverted	Output = NOT(Input 1) OR (Input 2 or Constant)	kbitorinv
	OR Reverse	Output = Input 1 OR NOT(Input 2 or Constant)	kbitorrev
	Right Shift	Bitwise RIGHT SHIFT of Input 1 by (Input 2 or Constant) Bits	kbitrshift
	SET	Set All Bits to One	kbitset
	XOR	Output = Input 1 XOR (Input 2 or Constant)	kbitxor
Arithmetic Comparison Operators	!=	IF Input 1 is Not Equal to (Input 2 or Constant), Output = TRUE	kne
	<	IF Input 1 is Less Than (Input 2 or Constant), Output = TRUE	klt
	<=	IF Input 1 is Less or Equal to (Input 2 or Constant), Output = TRUE	kle
	==	IF Input 1 is Equal to (Input 2 or Constant), Output = TRUE	keq
	>	IF Input 1 is Greater Than (Input 2 or Constant), Output = TRUE	kgt
	>=	IF Input1 is Greater or Equal to (Input2 or Constant), Output = TRUE	kge
Arithmetic Complex Operators	Complex to Real	Output = Real, Imaginary, Phase, or Magnitude Component of Input	kcmplx2real
	Conjugate	Output(real, imaginary) = Input(real, -imaginary)	kconj
	Imaginary Part	Output = Imaginary Component of (Complex) Input	kimagpart
	Magnitudes	Output is a Function of the Magnitude of the Input	kmag
	Phase	Output = Phase Component of Input [atan2(imaginary, real)]	kphase
	Polar to Rect	Convert (r, theta) to (real, imaginary) Coordinates	kpol2rect
	Real Part	Output = Real Component of (Complex) Input	krealpart
	Real to Complex	Output = Input 1 + j(Input 2)	kreal2cmplx
	Rect to Polar	Convert (real, imaginary) to (r, theta) Coordinates	krect2pol
Arithmetic Linear Transforms	FFT	Fast Fourier Transform (Forward and Inverse)	kfft
	Linear Operator	Performs linear operations (convolution/correlation)	klinearop
Arithmetic NonLinear Functions	Bessel	Compute Bessel Functions	kbessel
	Error Function	Compute Error & Complement Error Functions	kerf

DATAMANIP — Polymorphic Data Processing Operators			
Category Subcategory	**Operator**	**Description**	**Executable**
	Exponential	Compute Exponential (antilog)	`kexp`
	Log Gamma	Compute Log Gamma Function	`kgamma`
	Logarithm	Compute Logarithm	`klog`
	ldexp	Output = Input 1 * 2**(Input 2 or Constant)	`kldexp`
Arithmetic	Absolute Value	Output = Absolute Value of Input	`kabs`
Single Operand Arithmetic	Ceiling	Output = Integer Ceiling of Input	`kceiling`
	Cube	Output = Input**3	`kcube`
	Cube Root	Output = Cube Root of Input	`kcbrt`
	Floor	Output = Integer Floor of Input	`kfloor`
	Fractional Part	Output = Fractional Part of Input	`kfraction`
	Negative	Output = 0.0 - Input	`kneg`
	Reciprocal	Output = 1/Input	`krecip`
	Square	Output = Input**2	`ksqr`
	Square Root	Output = Square Root of Input	`ksqrt`
	Truncate	Output = Integer Truncate of Input	`ktruncate`
Arithmetic	Atan2	Output = Arc Tangent[Input 1 / (Input 2 or Constant)]	`katan2`
Trigonometry	Cos & ArcCos	Compute Cosine or Arc Cosine Function	`kcos`
	Cosh & ArcCosh	Compute Hyperbolic Cosine or Arc Cosine Function	`kcosh`
	Degree to Radian	Output = Input * (pi/180)	`kdeg2rad`
	Hypotenuse	Output = sqrt[(Input 1)**2 + (Input 2 or Constant)**2]	`khypot`
	Radian to Degree	Output = Input * (180/pi)	`krad2deg`
	Sin & ArcSin	Compute Sine or Arc Sine Function	`ksin`
	Sinc	Output = sin(Input)/Input	`ksinc`
	Sinh & ArcSinh	Compute Hyperbolic Sine or Arc Sine Function	`ksinh`
	Tan & ArcTan	Compute Tangent or Arc Tangent Function	`ktan`
	Tanh & ArcTanh	Compute Hyperbolic Tangent or Arc Tangent Function	`ktanh`
Arithmetic	Absolute Diff	Output = Absolute Value of (Input 1 - (Input 2 or Constant))	`kabsdiff`
Two Operand Arithmetic	Add	Output = Input 1 + (Input 2 or Constant)	`kadd`
	Blend Data	Output = (Input 1 * Constant) + (Input 2 (1 - Constant))	`kblend`
	Divide	Output = Input 1 / (Input 2 or Constant)	`kdiv`
	Divide Into	Output = (Input 2 or Constant) / Input 1	`kdivinto`
	Gate Data	Output Assigned Value in Input 1 or Input 2 Based on Gating Element	`kgate`
	Maximum	Returns Higher Value between Input 1 and (Input 2 or Constant)	`kmaximum`
	Minimum	Returns Lower Value between Input 1 and (Input 2 or Constant)	`kminimum`
	Modulus	Output = Remainder of [Input 1 / (Input 2 or Constant)]	`kmod`
	Multiply	Output = Input 1 * (Input 2 or Constant)	`kmul`
	Power	Output = (Input 1) ** (Input 2 or Constant)	`kpow`
	Replace Value	Replace All Occurrences of X with Y	`kreplace`

`DATAMANIP` — **Polymorphic Data Processing Operators**			
Category Subcategory	**Operator**	**Description**	**Executable**
	Subtract	Output = Input 1 - (Input 2 or Constant)	`ksub`
	Subtract From	Output = (Input 2 or Constant) - Input 1	`ksubfrom`
Data Manip Analysis & Information	Compare Objects	Compare Attributes & Data of Input 1 and Input 2	`ksegcmp`
	Print Value	Print Data Value to Parser and/or Concatenate to File	`kprval`
	Statistics	Compute Statistics of Data Object	`kstats`
Data Manip Clustering Operators	K-Means	Perform K-Means Clustering	`kkmeans`
Data Manip Data Conversion	Convert Type	Convert Data Storage Type (Optional Scale & Offset)	`kconvert`
	Normalize	Normalize Data Regions Using Minimum & Maximum of Each Region	`knormal`
Data Manip Frequency Filters	InvFilter	Inverse Filtering (Restoration) in Fourier Frequency Domain	`kinverse`
	Wiener	Wiener Filtering (Restoration) in Fourier Frequency Domain	`kwiener`
Data Manip Histogram Operators	Equalize	Perform Histogram Equalization	`kheq`
	Histogram	Compute Histogram for Data Object	`khisto`
	Stretch	Perform Histogram Stretching	`khstr`
Data Manip Introduce Noise	Exponential Noise	Introduce Exponential Noise in Input Object	`kexpon`
	Gaussian Noise	Introduce Gaussian Noise in Input Object	`kgauss`
	Poisson Noise	Introduce Poisson Noise in Input Object	`kpoisson`
	Rayleigh Noise	Introduce Rayleigh Noise in Input Object	`krayleigh`
	Shot Noise	Introduce Shot Noise in Input Object	`kshot`
	Uniform Noise	Introduce Uniform Noise in Input Object	`kuniform`
Data Manip Location Operators	Transform Location	Rotate, Translate, and Scale Location Data	`klocxform`
Data Manip Map Operators	Map Data	Map Value Data Through the Map	`kmapdata`
	Squish Map	Compress Map to One Column by Means of Average, RMS, or MAX	`kmsquish`
Data Manip Mask Operators	Unmask Data	Remove Mask from Data Object	`kunmask`
Data Manip Object Attributes	Comment Data	Change the Comment on a Data Object	`kcomment`
	Set Attribute	Modify Data Object Attributes	`ksetdattr`
Data Manip Reorganize Data	Flip	Reflect Data Along Specified Directions	`kflip`
	Reorient	Change Orientation of Data on Dimensions	`korient`
	Switch Axes	Reorient Data on the Width, Height, and Depth Axes	`kaxis`
	Translate	Translate Data in Object	`ktranslate`
Data Manip Segment Operators	Copy Segments	Copy Selected Data Segments from Input	`kcpseg`
	Copy from Value	Copy Value Segment Into One Segment of Output	`kcpfromval`
	Copy to Value	Copy One Segment Into Value Segment of Output	`kcptoval`
	Insert Segments	Insert Selected Data Segments from Input 1 into Input 2	`kinsertseg`
	Remove Segments	Remove Selected Data Segments from Input	`krmseg`

`DATAMANIP` — Polymorphic Data Processing Operators			
Category Subcategory	**Operator**	**Description**	**Executable**
Data Manip	Append	Append (or Prepend) Input Objects to Base Object	`kappend`
Size & Region Operators	Expand	Expand Object Via Pixel Replication	`kexpand`
	Extract	Extract Rectangular Region from Object	`kextract`
	Inset	Inset Object 2 into Object 1	`kinset`
	Pad	Pad Data with a Constant Value	`kpad`
	Resample	Resample Object Data	`kresample`
	SampleLine	Sample a data object along an arbitrary line	`ksampline`
	Shrink	Shrink Object Via Pixel Subsampling	`kshrink`
Data Manip	Clip Above	Clip Data Values that are Above Specified Cutoff	`kclipabove`
Threshold & Clip Operators	Clip Below	Clip Data Values that are Below Specified Cutoff	`kclipbelow`
	Clip Inside	Clip Data Values Inside the Specified Range	`kclipin`
	Clip Outside	Clip Data Values Outside of the Specified Range	`kclipout`
	MegaClip	Clip the Range of Values in Data Object	`kmegaclip`
	MegaThresh	Threshold the Range of Data Values in Data Object	`kmegathresh`
	Thresh Above	Threshold Data Values that are Above Specified Cutoff	`kthreshabove`
	Thresh Below	Threshold Data Values that are Below Specified Cutoff	`kthreshbelow`
	Thresh Inside	Threshold Data Values Inside the Specified Range	`kthreshin`
	Thresh Outside	Threshold Data Values Outside of the Specified Range	`kthreshout`
Input/Output Export Data	Supported Formats	Conversion Between the Formats Supported in Khoros 2	`kformats`
Input/Output	Constant	Generate Object Containing Constant Value Data	`kgconst`
Generate Data	Generate Location	Generate Location Data	`kgenloc`
	Impulse	Generate Object Containing Impulse Value Data	`kimpulse`
	Sinusoid	Generate Object Containing Sinusoidal Value Data	`kgsin`
Input/Output	Exponential Noise	Generate Exponential Noise Data	`kgexpon`
Generate Noise	Gaussian Noise	Generate Gaussian Noise Data	`kggauss`
	Poisson Noise	Generate Poisson Noise Data	`kgpoisson`
	Rayleigh Noise	Generate Rayleigh Noise Data	`kgrayleigh`
	Uniform Noise	Generate Uniform Noise Data	`kguniform`
Input/Output	ASCII to Location	Import ASCII Data File to Location Segment	`kasc2loc`
Import Data	ASCII to Map	Import ASCII Data File to Map Segment	`kasc2map`
	ASCII to Mask	Import ASCII Data File to Mask Segment	`kasc2mask`
	ASCII to Time	Import ASCII Data File to Time Segment	`kasc2time`
	ASCII to Value	Import ASCII Data File to Value Segment	`kasc2val`
	Import ASCII	Import ASCII Data File to Data Segment	`kimportasc`
	Import Raw	Low Level Import of Raw Data	`kimportraw`
Input/Output	Data Object Info	Print File Information	`kfileinfo`
Information	Print Data	Print Data in Formatted ASCII	`kprdata`

`DATAMANIP` — **Polymorphic Data Processing Operators**			
Category Subcategory	**Operator**	**Description**	**Executable**
Program Utilities			
Command Icons	Generic Interface	Generic Routine Interface (glue)	`kgeneric`

D. Alphabetical Listing of Polymorphic Operators

The following table contains a listing of the data operators that are available in the Datamanip toolbox. Table entries are organized alphabetically by the binary name, followed by the name and a brief description of each operator.

`DATAMANIP` — **Polymorphic Data Processing Operators**		
Executable	**Operator**	**Description**
`kabs`	Absolute Value	Output = Absolute Value of Input
`kabsdiff`	Absolute Diff	Output = Absolute Value of (Input 1 - (Input 2 or Constant))
`kadd`	Add	Output = Input 1 + (Input 2 or Constant)
`kappend`	Append	Append (or Prepend) Input Objects to Base Object
`karith1`	1-Operand Arith	Single Operator Pointwise Arithmetic
`karith2`	2-Operand Arith	Dual Operator Pointwise Arithmetic
`kasc2loc`	ASCII to Location	Import ASCII Data File to Location Segment
`kasc2map`	ASCII to Map	Import ASCII Data File to Map Segment
`kasc2mask`	ASCII to Mask	Import ASCII Data File to Mask Segment
`kasc2time`	ASCII to Time	Import ASCII Data File to Time Segment
`kasc2val`	ASCII to Value	Import ASCII Data File to Value Segment
`katan2`	Atan2	Output = Arc Tangent[Input 1 / (Input 2 or Constant)]
`kaxis`	Switch Axes	Reorient Data on the Width, Height, and Depth Axes
`kbessel`	Bessel	Compute Bessel Functions
`kbitand`	AND	Output = Input 1 AND (Input 2 or Constant)
`kbitandinv`	AND Inverted	Output = NOT(Input 1) AND (Input 2 or Constant)
`kbitandrev`	AND Reverse	Output = Input 1 AND NOT(Input 2 or Constant)
`kbitclear`	CLEAR	Set All Bits to Zero
`kbitlshift`	Left Shift	Bitwise LEFT SHIFT of Input 1 by (Input 2 or Constant) Bits
`kbitnand`	NAND	Output = NOT(Input 1) OR NOT(Input 2 or Constant)
`kbitnor`	NOR	Output = NOT(Input 1) AND NOT(Input 2 or Constant)
`kbitnot`	NOT	Perform Bitwise NOT (Invert) Operation on Input
`kbitor`	OR	Output = Input 1 OR (Input 2 or Constant)
`kbitorinv`	OR Inverted	Output = NOT(Input 1) OR (Input 2 or Constant)
`kbitorrev`	OR Reverse	Output = Input 1 OR NOT(Input 2 or Constant)
`kbitrshift`	Right Shift	Bitwise RIGHT SHIFT of Input 1 by (Input 2 or Constant) Bits

DATAMANIP — **Polymorphic Data Processing Operators**		
Executable	**Operator**	**Description**
kbitset	SET	Set All Bits to One
kbitwise	Bitwise Ops	Dual Operand Bitwise Operations
kbitxor	XOR	Output = Input 1 XOR (Input 2 or Constant)
kblend	Blend Data	Output = (Input 1 * Constant) + (Input 2 (1 - Constant))
kcbrt	Cube Root	Output = Cube Root of Input
kceiling	Ceiling	Output = Integer Ceiling of Input
kclip	Clip & Thresh	Clip or Threshold the Range of Values in Object
kclipabove	Clip Above	Clip Data Values that are Above Specified Cutoff
kclipbelow	Clip Below	Clip Data Values that are Below Specified Cutoff
kclipin	Clip Inside	Clip Data Values Inside the Specified Range
kclipout	Clip Outside	Clip Data Values Outside of the Specified Range
kcmplx	Complex Arith	Single Operand Complex Arithmetic
kcmplx2real	Complex to Real	Output = Real, Imaginary, Phase, or Magnitude Component of Input
kcomment	Comment Data	Change the Comment on a Data Object
kcompare	Comparison Ops	Comparison Operators
kconj	Conjugate	Output(real, imaginary) = Input(real, -imaginary)
kconvert	Convert Type	Convert Data Storage Type (Optional Scale & Offset)
kcos	Cos & ArcCos	Compute Cosine or Arc Cosine Function
kcosh	Cosh & ArcCosh	Compute Hyperbolic Cosine or Arc Cosine Function
kcpfromval	Copy from Value	Copy Value Segment Into One Segment of Output
kcpseg	Copy Segments	Copy Selected Data Segments from Input
kcptoval	Copy to Value	Copy One Segment Into Value Segment of Output
kcube	Cube	Output = Input**3
kdeg2rad	Degree to Radian	Output = Input * (pi/180)
kdiv	Divide	Output = Input 1 / (Input 2 or Constant)
kdivinto	Divide Into	Output = (Input 2 or Constant) / Input 1
keq	==	IF Input 1 is Equal to (Input 2 or Constant), Output = TRUE
kerf	Error Function	Compute Error & Complement Error Functions
kexp	Exponential	Compute Exponential (antilog)
kexpand	Expand	Expand Object Via Pixel Replication
kexpon	Exponential Noise	Introduce Exponential Noise in Input Object
kextract	Extract	Extract Rectangular Region from Object
kfft	FFT	Fast Fourier Transform (Forward and Inverse)
kfileinfo	Data Object Info	Print File Information
kflip	Flip	Reflect Data Along Specified Directions
kfloor	Floor	Output = Integer Floor of Input
kformats	Supported Formats	Conversion Between the Formats Supported in Khoros 2
kfraction	Fractional Part	Output = Fractional Part of Input
kgamma	Log Gamma	Compute Log Gamma Function
kgate	Gate Data	Output Assigned Value in Input 1 or Input 2 Based on Gating Element

`DATAMANIP` — **Polymorphic Data Processing Operators**		
Executable	**Operator**	**Description**
`kgauss`	Gaussian Noise	Introduce Gaussian Noise in Input Object
`kgconst`	Constant	Generate Object Containing Constant Value Data
`kge`	>=	IF Input1 is Greater or Equal to (Input2 or Constant), Output = TRUE
`kgeneric`	Generic Interface	Generic Routine Interface (glue)
`kgenloc`	Generate Location	Generate Location Data
`kgexpon`	Exponential Noise	Generate Exponential Noise Data
`kggauss`	Gaussian Noise	Generate Gaussian Noise Data
`kgnoise`	Generate Noise	Generate Noise Data
`kgpoisson`	Poisson Noise	Generate Poisson Noise Data
`kgrayleigh`	Rayleigh Noise	Generate Rayleigh Noise Data
`kgsin`	Sinusoid	Generate Object Containing Sinusoidal Value Data
`kgt`	>	IF Input 1 is Greater Than (Input 2 or Constant), Output = TRUE
`kguniform`	Uniform Noise	Generate Uniform Noise Data
`kheq`	Equalize	Perform Histogram Equalization
`khisto`	Histogram	Compute Histogram for Data Object
`khistops`	Histogram Ops	Perform Histogram Equalization and Stretching
`khstr`	Stretch	Perform Histogram Stretching
`khypot`	Hypotenuse	Output = sqrt[(Input 1)**2 + (Input 2 or Constant)**2]
`kimagpart`	Imaginary Part	Output = Imaginary Component of (Complex) Input
`kimportasc`	Import ASCII	Import ASCII Data File to Data Segment
`kimportraw`	Import Raw	Low Level Import of Raw Data
`kimpulse`	Impulse	Generate Object Containing Impulse Value Data
`kinsertseg`	Insert Segments	Insert Selected Data Segments from Input 1 into Input 2
`kinset`	Inset	Inset Object 2 into Object 1
`kinverse`	InvFilter	Inverse Filtering (Restoration) in Fourier Frequency Domain
`kkmeans`	K-Means	Perform K-Means Clustering
`kldexp`	ldexp	Output = Input 1 * 2**(Input 2 or Constant)
`kle`	<=	IF Input 1 is Less or Equal to (Input 2 or Constant), Output = TRUE
`klinearop`	Linear Operator	Performs linear operations (convolution/correlation)
`klocxform`	Transform Location	Rotate, Translate, and Scale Location Data
`klog`	Logarithm	Compute Logarithm
`klogexp`	Log & Exponential	Compute Logarithm and Exponential (Antilogarithm)
`klt`	<	IF Input 1 is Less Than (Input 2 or Constant), Output = TRUE
`kmag`	Magnitudes	Output is a Function of the Magnitude of the Input
`kmapdata`	Map Data	Map Value Data Through the Map
`kmaximum`	Maximum	Returns Higher Value between Input 1 and (Input 2 or Constant)
`kmegaclip`	MegaClip	Clip the Range of Values in Data Object
`kmegathresh`	MegaThresh	Threshold the Range of Data Values in Data Object
`kminimum`	Minimum	Returns Lower Value between Input 1 and (Input 2 or Constant)
`kmod`	Modulus	Output = Remainder of [Input 1 / (Input 2 or Constant)]

`DATAMANIP` — **Polymorphic Data Processing Operators**		
Executable	**Operator**	**Description**
`kmsquish`	Squish Map	Compress Map to One Column by Means of Average, RMS, or MAX
`kmul`	Multiply	Output = Input 1 * (Input 2 or Constant)
`kne`	!=	IF Input 1 is Not Equal to (Input 2 or Constant), Output = TRUE
`kneg`	Negative	Output = 0.0 - Input
`knoise`	Introduce Noise	Introduce Noise in Input Object
`knormal`	Normalize	Normalize Data Regions Using Minimum & Maximum of Each Region
`korient`	Reorient	Change Orientation of Data on Dimensions
`kpad`	Pad	Pad Data with a Constant Value
`kphase`	Phase	Output = Phase Component of Input [atan2(imaginary, real)]
`kpoisson`	Poisson Noise	Introduce Poisson Noise in Input Object
`kpol2rect`	Polar to Rect	Convert (r, theta) to (real, imaginary) Coordinates
`kpow`	Power	Output = (Input 1) ** (Input 2 or Constant)
`kprdata`	Print Data	Print Data in Formatted ASCII
`kprval`	Print Value	Print Data Value to Parser and/or Concatenate to File
`krad2deg`	Radian to Degree	Output = Input * (180/pi)
`krayleigh`	Rayleigh Noise	Introduce Rayleigh Noise in Input Object
`kreal2cmplx`	Real to Complex	Output = Input 1 + j(Input 2)
`krealpart`	Real Part	Output = Real Component of (Complex) Input
`krecip`	Reciprocal	Output = 1/Input
`krect2pol`	Rect to Polar	Convert (real, imaginary) to (r, theta) Coordinates
`kreplace`	Replace Value	Replace All Occurrences of X with Y
`kresample`	Resample	Resample Object Data
`krmseg`	Remove Segments	Remove Selected Data Segments from Input
`ksampline`	SampleLine	Sample a data object along an arbitrary line
`ksegcmp`	Compare Objects	Compare Attributes & Data of Input 1 and Input 2
`ksetdattr`	Set Attribute	Modify Data Object Attributes
`kshot`	Shot Noise	Introduce Shot Noise in Input Object
`kshrink`	Shrink	Shrink Object Via Pixel Subsampling
`ksin`	Sin & ArcSin	Compute Sine or Arc Sine Function
`ksinc`	Sinc	Output = sin(Input)/Input
`ksinh`	Sinh & ArcSinh	Compute Hyperbolic Sine or Arc Sine Function
`ksqr`	Square	Output = Input**2
`ksqrt`	Square Root	Output = Square Root of Input
`kstats`	Statistics	Compute Statistics of Data Object
`ksub`	Subtract	Output = Input 1 - (Input 2 or Constant)
`ksubfrom`	Subtract From	Output = (Input 2 or Constant) - Input 1
`ktan`	Tan & ArcTan	Compute Tangent or Arc Tangent Function
`ktanh`	Tanh & ArcTanh	Compute Hyperbolic Tangent or Arc Tangent Function
`kthreshabove`	Thresh Above	Threshold Data Values that are Above Specified Cutoff
`kthreshbelow`	Thresh Below	Threshold Data Values that are Below Specified Cutoff

DATAMANIP — Polymorphic Data Processing Operators		
Executable	**Operator**	**Description**
`kthreshin`	Thresh Inside	Threshold Data Values Inside the Specified Range
`kthreshout`	Thresh Outside	Threshold Data Values Outside of the Specified Range
`ktranslate`	Translate	Translate Data in Object
`ktruncate`	Truncate	Output = Integer Truncate of Input
`kuniform`	Uniform Noise	Introduce Uniform Noise in Input Object
`kunmask`	Unmask Data	Remove Mask from Data Object
`kwiener`	Wiener	Wiener Filtering (Restoration) in Fourier Frequency Domain

E. Installation Guide

This section is incomplete, and will be updated as we approach the cd release and become more familiar with what is involved with the installation process. For now, refer to the Khoros installation guide.

This page left intentionally blank

Table of Contents

This page left intentionally blank