Image SXM Macro Programming Language

Image SXM's built-in macro programming language can be used to automate complex or repetitive tasks. Macros are similar to Pascal procedures, except the procedure keyword is replaced by 'macro' and the procedure heading is a text string that becomes a command in the Macro menu when the macro is loaded. Macros are stored in text files that may also contain Pascal-like functions and procedures. Macros files are created and edited using Image's built-in text editor. The best way to learn how to write your own macros is by studying and customising the many example macros distributed with Image in the Macros folder.

Here is an example macro file containing one macro and a function called by that macro. You can try the macro by copying the indented text to the Clipboard, pasting it into a blank text window, selecting 'Load Macros from Window' from the Macro menu, and then selecting 'Do Exponentiation... [P]' from the Macro menu.

```
{ Example macro file }                  { comments go between curly braces }

var
    x, y, z: integer;                   { global variables }


function Power(x, n: real): real;       { raise x to the nth power }
begin
    power := exp(ln(x) * n);
end;


macro 'Do Exponentiation... [P]';
var
    base, ex: real;
begin
    base := GetNumber('Base:', 2);
    ex := GetNumber('Exponent:', 5);
    PutMessage(power(base, ex) : 6 : 3);
end;
```

Global variables (optional) are defined at the beginning of the file before any functions, procedures or macros. Text within curly braces { } is treated as a comment. Macro files cannot be larger than 32K.

The text in quotes following the keyword Macro becomes a new command at the bottom of the Macro menu. Macros may be assigned to keys by enclosing the key character in brackets. For example, you can execute the 'Do Exponentiation...' macro by pressing the 'P' key. Use 'F1', 'F2', etc. to assign macros to function keys. A few characters (';', '^', '!', '<', '/', '(') have special meaning to the Menu Manager, and should normally be avoided. You can use / to assign a command key equivalent to a macro, e.g., 'Test Macro /6'. Use '(-' as the name of an empty macro to create a dividing line in the Macro menu.

Functions and procedures may not be nested inside other functions and procedures, or inside macros. However, unlike Pascal, variables declared in a calling procedure or macro are made available to a called procedure. Function and procedure names must be unique in the first twelve characters.

Key Words

MACRO FUNCTION PROCEDURE BEGIN END VAR FOR TO DO
IF THEN ELSE WHILE REPEAT UNTIL

Operators

\+ – * / DIV MOD :=
= < > <> <= >= AND OR NOT

Types

INTEGER REAL BOOLEAN STRING

Both integer and real variables are stored internally in extended precision real format, which has a range of 10^{-383} to 10^{384} and 16 digits precision. Real numbers are automatically converted (by rounding) to integer without warning as needed. Strings have a maximum length of 255 characters and string comparisons are case insensitive. Variable names must be unique within the first twelve characters.

Built-in Arrays

Several one-dimensional arrays are predefined. User-defined arrays, however, are not supported, Measurement results can be accessed using arrays named rArea, rMean, rStdDev, rX, rY, rMin, rMax, rLength, rMajor, rMinor, and rAngle. These arrays use indexes ranging from 1 to Max Measurements, where the value of Max Measurements can be changed in the Options dialog box. Use the rCount function to get the value of the current measurement counter and SetCounter(n) to change it.

Two predefined arrays (rUser1 and rUser2) can be used to record and display derived results. Unlike the other results arrays, rUser1 and rUser2 are reserved for use by macros, and are never written to by any of the commands in Image. The column headings used for User1 and User2 can be set from within macros using the routines SetUser1Label('Label') and SetUser2Label('Label'). After computing a derived result, use UpdateResults to redisplay the last line in the results table or ShowResults to redisplay the entire table. Use SetCounter to control the number of lines displayed by ShowResults. Several example macros distributed with Image (in 'Measurement Macros') use these arrays to display derived results.

Histogram values are available using the read-only array Histogram, which accepts indexes in the range 0 to 255. For example, after using Measure, histogram[0] returns the number of white pixels.

Three built-in read/write arrays(RedLUT, GreenLUT, and BlueLUT) provide access to the video lookup table (LUT) associated with each open image. These arrays use indexes in the range 0-255 and return intensity values in the range 0-255. Use the UpdateLUT command to redraw the LUT window using newly modified LUT array values. Several macros (in 'LUT Macros') that use these arrays are distributed with Image, including a macro to export the current LUT as a text file, macros to load various functions into the LUT, a macro to plot the current LUT, and a macro to load a grayscale step function ('Posterize') into the LUT.

The built-in array LineBuffer provides access to the internal line buffer used by GetRow, PutRow, GetColumn and PutColumn. LineBuffer uses indexes in the range of 0-4095 and returns pixel values in the range 0-255. Using GetRow, LineBuffer[x] and PutRow to do pixel-by-pixel processing is up to twice as fast as using GetPixel and PutPixel, but is still at least 100 time slower than using compiled code.

The X-Y coordinates of the current selection are available from within macros using the built-in arrays xCoordinates[n] and yCoordinates[n]. The number of coordinates can be obtained using the macro function nCoordinates. The coordinates are relative to the upper left corner of the selection's bounding rectangle. For an example, look at the macro 'Plot X-Y Coordinates' in 'Plotting Macros'.

The data values generated by the PlotProfile and GetPlotData macro commands are stored in a built-in real array named PlotData, which uses indexes in the range 0-4095. The macro 'Plot Profile' in 'Plotting Macros' illustrates how to use GetPlotData and PlotData.

Here is a summary of the built-in arrays.

Name	**Index Range**	**Value Range**	**Permission**
Results arrays	1 - Max Meas	real	read/write
Histogram	0 - 255	+ integer	read only
RedLUT	0 - 255	0 - 255	read/write
GreenLUT	0 - 255	0 - 255	read/write
BlueLUT	0 - 255	0 - 255	read/write
LineBuffer	0 - 4095	0 - 255	read/write
xCoordinates	1 - 10000	real	read only
yCoordinates	1 - 10000	real	read only
PlotData	0 - 4095	real	read only

Built-in Commands and Functions

The macro language has more than 300 built-in commands and functions. Macro commands corresponding to menu commands are listed below under the heading for the appropriate menu. Note that ROI means Region of Interest.

File Menu

MakeNewWindow('Name')

Creates a new image window. Use SetNewSize to specify the size of the new window.

SetNewSize(width,height)

Specifies width and height of new image windows.

NewTextWindow('Name', w, h)

Creates a new text window with the title 'Name'. W and h (optional) specify the width and height of the new window.

Open('File Name')

Opens the specified image file.

SetImport('string')

Set various file Import options, where string contains some combination of: 'TIFF', 'DICOM', 'MCID', 'Palette', 'Text', 'Custom', '8-bits', '16-bits Unsigned', '16-bits Signed', 'Swap Bytes', 'Auto-Scale', Fixed Scale', 'Calibrate' and 'Open All'.

SetCustom(width,height,offset,slices)

Specifies the width, height, offset, and number of slices for imported files. The slices argument is optional.

SetImportMinMax(min, max)

Disables auto-scaling and fixes the range for imported 16-bit images and images in text (ASCII) format.

Import('File Name')

Imports the specified file using parameters specified by SetImport, SetCustom and SetImportMinMax.

Close

Closes the active image, text, profile plot, Histogram or Results window.

Dispose

Similar to Close, but user is never prompted to save changes.

DisposeAll

Closes all open image windows without prompting to ask if changes should be saved.

Save

Resaves the contents of the current image or text window.

SaveAll

Saves all open image windows.

SetSaveAs('mode')

Sets various file Save options, where 'mode' is one of: 'TIFF', 'RGB TIFF', 'PICT', MacPaint', 'PICS', 'LUT' or 'Outline'.

SaveAs('name')

Saves the current image using the specified file name. Use SetSave to specify the format. Uses the window title as the file name when saving a text file or if SaveAs is used with no argument. When saving images, the dialog box should only be displayed the first time SaveAs is called within a macro. For both image and text windows, 'name' can be a full folder path (e.g., 'HD400:Images:MyImage'). In this case, no dialog box is displayed.

SetExport('mode')

Sets various file Export options, where mode is one of: 'Raw', 'MCID', 'Text', 'LUT', 'Measurements', 'Plot Values', 'Histogram Values' or 'XY Coordinates'

Export('name')

Use SetExport to specify what to export. Similar to SaveAs, the dialog box is not displayed more than once per macro and full folder paths are allowed.

RevertToSaved

Restores the the previously saved version of the current image from disk.

Duplicate('Window Title')

Creates a new image window using the specified name and copies the contents of the current selection to the new window.

GetInfo

Creates a new image window that displays information about the current image window.

Print

Prints the active image (or selection), text, Plot, Results or Histogram window.

Edit Menu

Undo
: Reverses the effect of the last undoable operation.

Copy
: Copies contents of the current ROI to the Clipboard.

CopyResults
: Copies measurement results to Clipboard.

Paste
: Pastes into current ROI if Clipboard object and ROI have the same dimensions, otherwise, pastes into center of image.

PasteLive
: Pastes 'live' from video (frame grabber) source into a selection. Note that the destination window cannot be larger that the Camera window.

Clear
: Erases current ROI to background color.

Fill
: Fills current ROI with foreground color.

Invert
: Inverts image or ROI, i.e., value = 255 - value for all pixels in the image or ROI.

DrawBoundary
: Outlines current ROI using foreground color. Use SetLineWidth to control width of outline.

DrawScale
: Draws a grayscale ramp in the current rectangular ROI.

SelectAll
: Creates a rectangular ROI consisting of the entire image.

SetScaling('string')
: Sets ScaleAndRotate options, where string contains some combination of: 'Nearest', 'Bilinear', 'New Window', 'Same Window' or 'Interactive'.

ScaleAndRotate(xscale, yscale, angle)
: Scales and/or rotates the current rectangular ROI, where 0.05 <= xscale, yscale <= 25.0 and -180 <= angle <= 180.

RotateLeft(b)
: Rotates the current image or rectangular ROI counter-clockwise 90 degrees. Creates a new window if b is true.

RotateRight(b)
: Rotates the current image or rectangular ROI clockwise 90 degrees. Creates a new window if b is true.

FlipVertical
: Vertically inverts the current rectangular image or ROI.

FlipHorizontal
: Horizontally inverts the current rectangular image or ROI.

Options Menu

InvertLUT

Inverts the video look-up table.

SetPalette(string, ExtraColors)

Loads a new look-up table, where string is one of: 'Grayscale', 'PseudoColor', 'System Palette', 'Rainbow' or 'Spectrum'. ExtraColors (optional) is the number (0-6) of LUT entries reserved for extra colors.

SetFont('Font name')

Specifies the typeface used for drawing text, where 'Font Name' is 'Geneva', 'Monoco', 'Helvetica', 'Times', etc.

SetFontSize(size)

Sets font size in points, where 6 <= size <= 720.

SetText(string)

Specifies text style, where string contains some combination of: 'Bold', 'Italic', 'Underline', 'Outline', 'Shadow', 'Left Justified', 'Right Justified', 'Centered', 'No Background' or 'With Background'.

ScaleConvolutions(b)

Sets or resets Scale Convolutions flag in Preferences, where b = true or false.

InvertY(b)

Invert Y-coordinates if b is true.

SetPlotLabels(b)

Specifies whether or not profile plots are to be labeled, where b = true or false.

SetPlotScale(min, max)

Set min and max to zero for auto-scaling.

SetPlotSize(width, height)

Set width and height to zero for auto-sizing of plots.

SetThreshold(level)

Sets the threshold, where 0 <= level <= 255. SetThreshold(-1) disables thresholding.

AutoThreshold

Set threshold level to an automatically determined value.

SetDensitySlice(lower, upper)

Sets the lower and upper threshold levels, where 1 <= lower,upper <= 254. SetDensitySlice (255,255) enables density slicing without changing the levels. SetDensitySlice(0,0) disables density slicing.

GetThresholds(lower, upper)

In density slicing mode, returns the lower and upper thresholds. In thresholding mode, lower is set to the threshold and upper is set to 255. Otherwise, both are set to zero.

PropagateLUT

Propagates current LUT to all other open images.

PropagateSpatial

Propagates current spatial calibration to all other open images.

PropagateDensity

Propagates current density calibration to all other open images.

Process Menu

Filter('name')

Runs the specified filter, where 'name' is one of the following: 'smooth', 'smooth more', 'sharpen', 'sharpen more', 'find edges' (or 'sobel'), 'median', 'max', 'min' or 'dither'.

Shadow('direction')

Set 'direction' (optional) to 'N', 'NE', 'E', 'SE', 'S', 'SW', 'W' or 'NW'.

Convolve('Kernel file name')

Note that the name can be a directory path such as 'HD80:Image:Kernels:Smooth'.

CallFilter('name')

Runs the specified filter plug-in, which is assumed to be in the Plug-ins folder.

MakeBinary

Converts the current grayscale image to binary.

Erode

Removes pixels from the edges of objects.

Dilate

Adds pixels to the edges of objects.

SetBinaryCount(n)

See description of Set Count command.

Outline

Generates one pixel wide object outlines.

Skeletonize

Reduces objects to single pixel wide skeletons.

AddConstant(n)

Adds n to the current image or rectangular selection, where -255 <= n <= 255.

MultiplyByConstant(n)

Multiplies the current image or rectangular selection by n, where 0.0 <= n <= 255.0.

ImageMath('op', pic1, pic2, scale, offset, result)

Where 'op' is one of 'add', 'sub', 'mul', 'div', 'and', 'or', 'xor', 'min', 'max' or 'copy'. Add the keyword 'real' (e.g. 'add real') to generate a 32-bit real result. Pic1 and pic2 are pic numbers or pid numbers. ImageMath performs the specified operation, the result is multiplied by scale, offset is added and, if real wasn't specified, the final result is clipped to 8-bits. The arithmetic operation is performed in the upper left corner of each image using the largest common rectangle. Result can be either a string or a pid number. If it's a string, a window with that name is created to store the result, otherwise the result is stored in the image specified by the pid number.

FFT('foreward')

Generates a Fourier transform of a square, power of two size image. The image can be either 8-bit or real. For real images, the transformation is done in-place.

FFT('inverse')

Does an in-place inverse transform with black or white regions in the 8-bit power spectrum used as a mask to generate a filter. Duplicates the behaviour of the 'Inverse FFT' menu command.

FFT('inverse with mask')

Same as FFT('inverse').

FFT('inverse with filter')

Uses the 8-bit grayscale component of the FFT window as a filter that the frequency domain image is multiplied by prior to retransformation. This means that the power spectrum must be replaced by a grayscale filter before retransformation. The 'High Pass' and 'Low Pass' macros in 'FFT Macros' use this variation of the FFT command.

FFT('inverse without filter')

No masking or filtering is done before retransformation.

FFT('Display Power Spectrum')

Recomputes the power spectrum.

FFT('Swap Quadrants')

Swaps quadrants 1 and 3 and quadrants 2 and 4 of the current 8-bit image.

SubtractBackground('str',radius)

Where 'str' is one of: '1D Horizontal', '1D Vertical', '2D Rolling Ball' or '2D Remove Streaks'. Add 'faster' to 'str' (e.g. '2D Rolling Ball (faster)') for faster operation.

ApplyLUT

Transforms the pixel data using the current look-up table.

EnhanceContrast

Does a histogram stretch on the LUT. Does not alter pixel values.

ChangeValues(v1, v2, v3)

Changes pixels with a value in the range v1–v2 to v3.

Analyze Menu

Measure

Results are stored in the results arrays rArea[], rMean[], etc. and in the Histogram[] array.

GetResults(n, mean, mode, min, max)

Use after Measure. Returns the pixel count, the mean pixel value, the most frequently occurring pixel value, and the minimum and maximum pixel values. Values are always uncalibrated. Use cValue function to calibrate them.

AnalyzeParticles('options')

Does particle analysis, where 'options' (optional) contains some combination of 'label', 'outline', 'ignore', 'include' and 'reset'. Any option not listed is disabled.

Use AnalyzeParticles('dialog') to display the dialog box using the existing settings.

SetParticleSize(min, max)

Particles smaller than minimum (pixels) and larger than maximum (pixels) will be ignored by AnalyzeParticles.

ShowResults

Displays the Results window.

ShowHistogram

Generates a density histogram and displays it in the Histogram window.

RestoreRoi

Same as the Restore Selection menu command.

MarkSelection

Same as the Label Selection menu command.

SetOptions('string')

Specifies measurement options as listed in the Options dialog box. 'String' should contain some combination of 'Area', 'Mean', 'Std. Dev.', 'X-Y Center', 'Mode', 'Perimeter' (or 'Length'), 'Major', 'Minor', 'Angle', 'Int. Den.', 'Min/Max', 'User1' or 'User2'. Any variable not listed is disabled.

Redirect(b)

Enables/disables redirected sampling. b = true or false.

WandAutoMeasure(b)

Sets the 'Wand Auto-Measure' flag in Measurement/Options. b = true or false.

AdjustAreas(b)

Sets the 'Adjust Areas' flag in Measurement/Options. b = true or false.

SetPrecision(digits, fwidth)

Specifies the format of displayed results, were digits is the number of digits to the right of the decimal point and fwidth (optional) is the field width.

GetScale(scale, unit, AspectRatio)

Returns the number of pixels per unit of measurement in the real variable scale, the unit of measurement in the string variable unit, and (optional) the pixel aspect ratio in the real variable AspectRatio. For uncalibrated images, scale and AspectRatio are set to 1.0 and unit to 'pixel'.

SetScale(scale, 'unit', AspectRatio)

Scale is the number of pixels per unit of measurement. Set 'Unit' to 'nm', 'µm', 'mm', 'cm', 'meter', 'km', inch', 'ft', 'mile' or 'pixel' or use an arbitrary unit up to 11 characters in length. AspectRatio (optional) is the x/y pixel aspect ratio. Us SetScale(0,'pixel') to disable spatial calibration and SetScale(0,'') to activate the Set Scale dialog box.

Calibrate('fit', 'unit', m1, k1, m2, k2, ...)

'fit' is one of 'straight', 'poly2', 'poly3', 'poly4', 'exp', 'power', 'log', 'rodbard', 'uncalibrated' or 'uncalibrated od'. 'unit' is the unit of measurement, m1, m2, etc. are the measured values and k1, k2, etc. are the known values. For example, 'Calibrate('Straight', 'Invert', 0, 255, 255, 0)' sets up a simple inverting function. Use 'Calibrate('Uncalibrated OD')' to enable uncalibrated OD and 'Calibrate('Uncalibrated')' to disable calibration.

PlotProfile

Generates a gray scale profile plot of the current rectangular selection or line selection.

GetPlotData(count, ppv, min, max)

Performs the equivalent of PlotProfile and returns the results in the built-in PlotData array. Count is the number of values, ppv is the number of pixels averaged for each value, and min and max are the minimum and maximum values.

SurfacePlot

Creates a surface plot of the current image. Use SetNewSize to specify the size of the plot.

ResetCounter

Sets the measurement counter to zero.

Stacks Menu

AddSlice
: Adds a slice following the current slice.

DeleteSlice
: Deletes the current slice.

MakeMovie('str', frames, interval)
: Captures a sequence of video frames to a stack. 'str' is some combination of 'blind', 'buffer', 'time stamp', 'existing', 'trigger first', 'trigger each', and 'dialog'. Frames is the number of frames to capture, and interval is the interval between frames in seconds. See the 'Movie Making' macro file for examples.

AverageSlices
: Averages all the slices in the current stack.

Capture Color
: Capures a 24-bit RGB color image.

RGBToIndexed('string')
: Converts a 3-slice RGB stack to an 8-bit image with color LUT. 'String' contains some combination of 'System LUT', 'Existing LUT', 'Custom LUT' and 'Dither'. Custom LUT and Dither are the defaults.

IndexedToRGB
: Converts an 8-bit color image to a 3-slice RGB stack.

SetProjection('string', n)
: Specifies 3D projection variables, where n is integer and 'string' is one of the following: 'Initial Angle', 'Total Rotation', 'Rotation Increment', 'Surface Opacity', 'Surface Depth-Cueing' or 'Interior Depth-Cueing'. Use SetDensitySlice to set the transparency bounds.

SetProjection('string', b)
: Set projections flags, where b is boolean (true or false) and 'string' is either 'Save Projections' or 'Minimize Size'.

SetProjection('string')
: Set projection options, where 'string' is one of: 'X-Axis', 'Y-Axis', Z-Axis, 'Nearest', 'Brightest', 'Mean Value'

Project
: Does 3D projection of current stack. Dialog box is not displayed if SetProjection() has been called.

Reslice
: Reslice stack along current line selection.

Windows Menu

NextWindow
: Switches to the next image window.

TileWindows
: Repositions and resizes all open image windows so they don't overlap.

ShowPasteControl
: Activates the Paste Control window.

Miscellaneous Macro Commands

AutoOutline(x, y)
: Equivalent to clicking with the wand tool at location x,y. The outline was successfully created if GetRoi returns a width greater than zero.

Beep
: Makes a short sound.

ChangeValues(v1, v2, v3)
: Changes the value of all pixels with a value in the range v1-v2 to v3.

ChoosePic(n)
: Selects then Nth image window without activating it. Faster than SelectPic but changes, if any, are not displayed. Also accepts PidNumbers (see description of PidNumber function).

ChooseSlice(n)
: Selects the Nth slice in a stack without displaying it. ChooseSlice is faster than SelectSlice but changes, if any, are not displayed. Use SelectSlice before exiting macro to make sure the stack is correctly displayed.

Exit
: Terminates execution of the macro.

GetColumn(x, y, length)
: Copies a column of pixels from active image to the built-in LineBuffer array.

GetHistogram(left, top, w, h)
: Generates a density histogram of the specified rectangular ROI and stores it in the built-in histogram array.

GetLine(x1, y1, x2, y2, LineWidth)
: Returns the starting coordinates, ending coordinates and width of current straight line selection. Sets x1 = -1 if there is no line selection.

GetMouse(x, y)
: Returns the current cursor location in local pixel coordinates.

GetPicSize(width, height)
: Returns, in pixels, the width and height of active image.

GetRoi(left, top, width, height)
: Returns ROI location and size in pixels. Sets width=0 if no ROI. Returns location and size of bounding rectangle for non-rectangular ROIs and for line selections.

GetRow(x, y, length)
: Copies a row of pixels from the active image to the built-in LineBuffer array.

GetTime(year, month, day, hour, minute, second, dayofweek)
: Returns the current date and time.

InsetRoi(delta)
: Shrinks or expands (if delta<0) the current ROI by delta.

KillRoi
: Disables the current 'marching ants' selection.

LineTo(x, y)
: Draws a line from current location to x, y.

MakeLineRoi(x1, y1, x2, y2)
: Creates a new straight line selection. The origin (0,0) is assumed to be the upper left corner of the image. Coordinates are in pixels but they can be real numbers.

MakeNewStack('name')
: Creates a new 1-slice stack. Use SetNewSize to specify the size.

MakeRoi(left, top, width, height)

Creates a new rectangular selection. Left and top are the coordinates (in pixels) of the upper left corner of the ROI. The origin (0,0) of the coordinate system is the upper left corner of the image.

MakeOvalRoi(left, top, width, height)

Creates an elliptical ROI, where left and top define the upper left corner of the bounding rectangle.

MoveRoi(dx,dy)

Moves ROI right dx pixels and down dy pixels.

MoveTo(x, y)

Sets the current drawing location. The origin is always assumed to be the upper left corner of the image.

MoveWindow(x, y)

Moves current window to global screen coordinates x,y.

PlotXYZ

Plots XYZ coordinate data stored in a text file. See the example macro in 'Plotting Macros'.

PutColumn(x, y, length)

Copies length pixels from the built-in LineBuffer array to a column starting at x,y in the current image.

PutMessage('message')

Displays 'message' in a dialog box. Like the Write routine, accepts multiple string and numeric arguments.

PutPixel(x, y, value)

Stores value (0-255) at location x,y. When storing a sequence of pixels, it's usually faster to store them in the built-in LineBuffer array and use PutRow to display the line.

PutRow(x, y, length)

Copies length pixels from the built-in LineBuffer array to a row in the current image starting at x, y.

ResetGrayMap

Equivalent to using the Grayscale menu command or clicking on the reset icon in Map window.

SaveState

Saves foreground and background color, new window width and height, status of Invert Y flag, text attributes, and various ScaleAndRotate and SetScaling parameters. Use Restore State to restore the saved settings.

ScaleMath(b)

b = true or false. Sets or resets Scale Math flag in the Paste Control dialog box.

RequiresVersion(n)

Aborts macro if Image version number is less than n, where n is a real number.

RestoreState

Restores settings saved by SaveState.

SelectPic(n)

Activates the Nth image window. Also accepts PidNumbers (see description of PidNumber function).

SelectSlice(n)

Displies the Nth slice of the current stack.

SelectTool(tool)

Selects a tool from the tool palette, where tool = 'magnifier', 'grabber', 'pencil', 'eraser', 'brush', 'drawline', 'paintbucket', 'profile', 'wand', 'angletool', 'rectangle', 'oval', 'polygon', 'freehand', 'straightline', 'freeline', 'segmentline', 'lut', 'text', 'spraycan', 'picker' or 'crosshair'.

SelectWindow('name')
Activates the window with the title 'name'.
SetCounter(n)
Sets the measurement counter to n.
SetCursor('shape')
Changes the cursor shape, where 'shape' is 'watch', 'cross', 'arrow' or 'finger'.
SetForegroundColor(c)
Sets the foregound color, where 0 <= c <= 255 (note that 0=white and 255=black).
SetBackgroundColor(c)
Sets the background color, where 0 <= c <= 255.
SetLineWidth(width)
Specifies the line width (in pixels) used by LineTo, DrawBoundary and MakeLineRoi.
SetUser1Label('Label')
Replaces the label used for User 1 column in Results window. The maximum length of the label is 9 characters.
SetUser2Label('Label')
Replaces the label used for User 2 column in Results window. The maximum length of the label is 9 characters.
SetPicName('Name')
Renames the active image window.
SetOption
Equivalent to holding down option key while executing the immediately following macro command.
SetSliceSpacing(n)
Sets the slice spacing (in pixels) used by the Reslice and Project commands.
ShowMessage('message')
Displays 'message' in the Info window. Accepts multiple arguments in the same way the Write routine does. Use a back-slash ('\') to start a new line.
SortPalette
Sorts the current LUT by hue.
UpdateResults
Redisplays the last measurement in Info and Results windows.
UpdateLUT
Redisplays the LUT window.
Wait(seconds)
Delays for seconds seconds. Fractions of a second are allowed, e.g., wait(1.5).
WaitForTrigger
Waits for an external trigger. Requires a QuickCapture or Scion frame grabber card. Use 'repeat until button' to wait for a mouse down event.
Write(e1, e2, ...)
Draws text, variables, or constants in the current image at the current location. Like the Writeln procedure in Pascal, expressions may have optional field width specifications in the form e:f1:f2 (e.g., write('M=',mean:8:3), where f1 is the field width, and f2 specifies the number of digits to the right of the decimal point.
Writeln(e1, e2, ...)
Similar to Write, but does the equivalent of a line feed and carriage return after displaying the specified values.

Miscellaneous Functions

AllSameSize
: Returns true if all open images have the same dimensions.

Button
: Returns true if mouse button is down.

Calibrated
: Returns true if current image is density calibrated.

cValue(PixelValue)
: Converts a raw pixel value (an integer in the range 0-255) to a density calibrated value.

Get('FreeMem')
: Returns the total amount of free memory, in bytes.

Get('MaxBlock')
: Returns the size (in bytes) of the largest free memory block.

Get('MaxMeasurements')
: Returns the value of 'Max Measurements'.

Get('RoiType')
: Returns a code that specifies the current ROI type, where 0 = no ROI, 1 = rectangle, 2 = oval, 3 = polygon, 4 = freehand, 5=traced, 6 = line, 7 = freehand line and 8 = segmented line.

Get('UndoBufSize')
: Returns size (in bytes) of the Undo and Clipboard buffers.

GetNumber('Prompt', default, d)
: Displays a dialog box and returns with the value entered. d (optional) = # decimal places.

GetPixel(x, y)
: Returns the value of the pixel at x,y.

GetSliceSpacing
: Returns slice spacing (in pixels) of current stack.

KeyDown('key')
: Returns the (boolean) state of the specified modifier key. 'key' = 'option', 'shift' or 'control'.

nCoordinates
: Returns the number of XY coordinates used to define the current selection. The coordinates are stored in the xCoordinates[] and yCoordinates[] built-in arrays.

nPics
: Returns number of image windows.

nSlices
: Returns number of slices in current stack.

PicNumber
: Returns number (used by SelectPic) of the active image.

PidNumber
: Returns a negative permanent ID number for the current image. This number can be passed at a later time to SelectPic or ChoosePic to activate this image.

PidExists(pid)
: Returns TRUE if image with this PidNumber is still open.

rCount
: Returns current measurement counter value.

SliceNumber
: Returns number of current slice in a stack.

TickCount
: Returns the number of ticks (sixtieths of a second) since system last started.

WindowTitle
: Returns a string containing the title of the active window.

Math Functions

abs(n)
: Returns absolute value of n.

arctan(n)
: Returns arctangent of n (radians).

BitAnd(n1, n2)
: Returns n1 AND n2.

BitOr(n1, n2)
: Returns n1 OR n2.

cos(n)
: Returns cosine of n (radians).

exp(n)
: Returns exponential of n

ln(n)
: Returns natural logarithm of n.

odd(n)
: Returns TRUE if integer n is odd.

random
: Returns a random number between 0 and 1.

round(n)
: Converts a real value to an integer with rounding.

sin(n)
: Returns sine of n (radians).

sqr(n)
: Returns square of n.

sqrt(n)
: Returns square root of n.

trunc(n)
: Converts a real value to an integer with truncation.

Paste Control Related Commands

DoCopy, DoAnd, DoOr, DoXor, DoReplace, DoBlend, Add, Subtract, Multiply, Divide

These commands are equivalent to clicking on the corresponding button in the Paste Control dialog box. The must be used immediately after the Paste command. Precede DoCopy, DoAnd, etc. with SetOption to switch paste transfer modes, otherwise the operation is performed and the paste operation terminated. The foreground color is set to black and the background color to white unless SetOption is used. See the 'More Macros' macro file for examples of how to use SetOption with these commands. Add, Subtract, Multiple and Divide only work with rectangular selections.

String Functions

str := GetString('Prompt','default')

Displays a dialog box and returns with the string entered. The first argument is the prompting message and the second argument (optional) is the default input string.

str := concat(str1, str2,...)

Concatenates a series of strings. Will also convert one or more numbers to a string.

str := chr(n)

Converts a positive integer in the range 0..255 to a one character string (e.g. chr(65) = 'A').

Delete(str,index,count)

Removes count characters from str, beginning at index.

GetFileInfo(FullPath, FileType, FileSize)

Returns information about the file specified by FullPath, a string containing a path name (e.g. 'HD500:Images:image01'). Returns the file type ('TIFF', 'PICT', 'TEXT' etc) in the string variable FileType and the file size in bytes in the integer variable FileSize. If the file isn't found, FileType is set to an empty string and FileSize is set to -1. See the 'File Paths Demo' macro in 'Input/Output Macros' for an example of how to use GetFileInfo.

path := GetPath('window')

Returns the folder path (e.g. 'HD500:Images:') of the current current image or text window. Returns an empty string if no window is open or the current window has no file associated it. The 'File Paths Demo' macro in 'Input/Output Macros' demonstrates how to use the GetPath functions.

path := GetPath('startup')

Returns the path of the folder from which NIH Image was started.

path := GetPath('pref')

Returns the path of the Preferences folder in the System Folder.

i := length(str)

Returns the length of str.

n := ord(str)

Returns the ordinal number of the first character in a string (e.g. ord('A')=65). Returns -1 if the string is empty.

i := pos(substr, str)

Searches for substr within str and returns an integer that is the index of the first character of substr within str. Returns zero if substr is not found.

n := StringToNum(str)

Converts a string to a real number. Returns zero if the string contains no digits.

str := WindowTitle

Returns the title of the currently active window.

Note that routines that require a file name or window title (MakeNewWindow, MakeNewStack, Open, SaveAs, Import, Export, Duplicate and SetPicName) accept multiple arguments similar to the Write routine, except that numeric fields are left filled with zeros rather than spaces. As an example, SetPicName('PIC', n:2) result in window titles in the form 'PIC01', 'PIC02', 'PIC03', etc. Several other routines (PutMessage, ShowMessage, PutSerial, Concat) also accept multiple arguments.