Matrox Genesis – Cross-Reference Guide (MIL vs. Genesis Native Library commands)

MIL versus Matrox Genesis Native Library commands

When used with Matrox Genesis, MIL functions make calls to the Matrox Genesis Native Library, which executes operations on the processing node(s). For most MIL functions, there is a corresponding Native Library function. However, some functions are only available through MIL such as gauging and OCR. Also, the Native Library offers some board-specific functions that MIL does not provide. In general, MIL can be used to develop the entire application or at least the majority of the application. If required, MIL Native Mode Programming can be used to integrate Native Library functions. The objective in developing mainly with MIL functions is that the application can be as portable as possible. Moving the application later to a different platform will require changing only the board-dependent portion of the code.

To facilitate the use of the two libraries; this document lists and describes MIL commands and provides you with the equivalent or appropriate Native Library command(s). This document will serve as a quick cross-reference. For those developers who have already written a MIL application, this guide will assist in porting the application to Matrox Genesis. For developers who would like to build a MIL application for Matrox Genesis, this guide will assist in determining the right mix of MIL and Native Library commands. The commands have been listed under their respective MIL module for faster reference. This document is not intended to be used alone; but rather in conjunction with both the MIL and Native Library Command Reference manuals.

Differences in Data Management

MIL and Native Library functions operate and store information in buffers. While both libraries allocate buffers in basically the same way, there are some capabilities available with Native Library that are not available with MIL. With Native Library, users can allocate a buffer that contains control fields to store function options. This is done by adding the required control fields of those desired function options and then passing this control buffer to the function. With MIL, rather than storing options in a single control buffer the options are specified in the function's parameter. For copying data, both libraries can copy data between buffers, however with The Native Library, users can specify copying data over the PCI bus or VMchannel, and specify copy options not available with MIL (i.e. tag buffers, zooming and subsampling, extracting or swapping bytes, etc.).

MIL users will also note that the Native Library functions are identified according to the data type that they support. In other words, functions that can perform operations on packed binary buffers, integer buffers, and floating-point buffers are known as the imBin...(), imInt...(), and imFloat...() functions, respectively. For additional information, refer to the MIL and Native Library command reference manuals.

The Application Allocation and Control Module

MIL Command	MIL Description	Native Library Command	Comments
MappAlloc()	Allocate a MIL application.	Not applicable	
MappAllocDefault()	Allocate MIL application defaults.	imBufAlloc() or imCamAlloc() or imDevAlloc() or imThrAlloc()	With MIL, a user can separately call each individual allocation or use this macro to set up the environments using defaults. With the Native Library, a user will have to separately call each allocation. See individual allocations.
MappChild()	Allocate a child MIL application.	Not applicable	
MappControl()	Control an application environment setting.	imAppControl()	
MappControlThread()	Allocate/control MIL application thread(s) or events.	imThrControl()	
MappFree()	Free a MIL application.	Not applicable	
MappFreeDefault()	Free MIL application defaults.	imBufFree() or imCamFree() or imDevFree() or imThrFree()	With MIL, a user can separately free each individual allocation or use this macro to free the defaults that were allocated. With the Native Library, a user will have to separately free each allocation. See individual Free functions.
MappGetError()	Get error code and related information.	imAppGetError() or imAppCatchError()	With MIL, a user has only one way to get an error. With the

		or imThrGetError() or imSyncGetError()	Native Library, there are several ways.
MappGetHookInfo()	Get information about a hooked event.	Not available	
MappHookFunction()	Hook function to an event.	Not available	
MappInquire()	Inquire about the application.	imAppInquire()	
MappModify()	Modify MIL objects using specified operations.	Not applicable	
MappTimer()	Control the MIL timer.	imSysClock()	

The Blob Analysis Module

MIL Command	MIL Description	Native Library Command	Comments
MblobAllocFeatureList()	Allocate a feature list.	imBlobAllocFeatureList()	
MblobAllocResult()	Allocate a blob analysis result buffer.	imBlobAllocResult()	
MblobCalculate()	Perform blob analysis calculations.	imBlobCalculate()	
MblobControl()	Change the blob-analysis processing mode.	imBlobControl()	
MblobFill()	Fill blobs that meet a given criteria.	imBlobFill()	
MblobFree()	Free a blob-analysis result buffer or feature list.	imBlobFree()	
MblobGetLabel()	Get the label value of a blob at a specific position.	imBlobGetLabel()	
MblobGetNumber()	Get the number of currently included blobs.	imBlobGetNumber()	

MblobGetResult()	Read features values of the included blobs.	imBlobGetResult() or imBlobCopyResult()	imBlobGetResult() is equivalent, however imBlobCopyResult() can be more efficient.
MblobGetResultSingle()	Read the feature value of a single blob.	imBlobGetResultSingle()	
MblobGetRuns()	Get the blob length encoding information.	imBlobGetRuns() or imBlobCopyRuns()	imBlobGetRuns() is equivalent, however imBlobCopyRuns() can be more efficient.
MblobInquire()	Inquire about a blob-analysis result buffer.	imBlobInquire()	
MblobLabel()	Draw labeled image.	imBlobLabel()	
MblobReconstruct()	Reconstruct blobs in an image.	imBlobFill and imBlobSelect()	With the Native Library, a user can achieve same results with this combination of two functions.
MblobSelect()	Select blobs for calculations and result retrieval.	imBlobSelect()	
MblobSelectFeature()	Select feature(s) to be calculated.	ImBlobSelectFeature()	
MblobSelectFeret()	Add Feret angle to the feature list.	imBlobSelectFeret()	
MblobSelectMoment()	Add specified moment to the feature list.	imBlobSelectMoment()	

The Data Allocation and Access Module

MIL Command	MIL Description	Native Library Command	Comments
MbufAllocColor()	Allocate a color data buffer.	imBufAlloc()	Function is equivalent for processing buffers, however display buffers should be

			allocated with imBufChild().
MbufAlloc1d()	Allocate a 1D buffer.	imBufAlloc1d()	Function is equivalent, display buffer should be allocated with imBufChild().
MbufAlloc2d()	Allocate a 2D buffer.	imBufAlloc2d()	Function is equivalent, display buffer should be allocated with imBufChild().
MbufChildColor()	Allocate a child data buffer within a color parent buffer.	imBufChildBand()	
MbufChild1d()	Allocate a 1D child data buffer.	imBufChild()	Allocate a one-dimensional child buffer with the Xstart and Xsize parameters.
MbufChild2d()	Allocate a 2D child data buffer.	imBufChild()	Allocate a two-dimensional child buffer with the Xstart, Ystart, Xsize, and Ysize parameters.
MbufClear()	Clear buffer to a color.	imBufClear()	
MbufControl()	Control buffer features.	Not available	
MbufControlNeighborhood()	Change the value of an operation flag associated with a custom kernel or structuring element.	imBufPutField()	With the Native Library, this MIL function does not exist as a specific function. See individual neighborhood functions. (e.g. operation flag M_OVERSCAN's M_TRANSPARENT and M_REPLACE are specified with imIntConvolve's

			Control parameter IM_CTL_OVERSCAN)
MbufCopy()	Copy data from one buffer to another.	imBufCopy() or imBufCopyPCI() or imBufCopyVM()	With the Native Library, a user can specify the data path (over the PCI bus or VM channel), to copy data to a destination buffer.
MbufCopyClip()	Copy buffer-clipping data outside destination buffer.	imBufChild() and imBufCopy()	First, allocate a child buffer and then copy the buffer.
MbufCopyColor()	Copy one or all bands of an image buffer.	imBufChildBand() and imBufCopy()	First, allocate a one-band child buffer and then copy the buffer.
MbufCopyCond()	Copy conditionally the source buffer to the destination buffer.	imIntBinarize() and imIntTriadic()	Binarize a conditional buffer, then call imIntTriadic(), setting the operation parameter to: IM_PP_MERGE.
MbufCopyMask()	Copy buffer with mask.	imIntTriadic()	Call imIntTriadic(), setting the operation parameter to: IM_PP_MERGE.
MbufDiskInquire()	Inquire about the buffer data in a file.	Not available	
MbufExport()	Export a data buffer to a file using the specified output file format.	imBufSave()	
MbufFree()	Free a data buffer.	imBufFree()	
MbufGet()	Get data from a buffer and place it in a user-supplied array.	imBufGet()	
MbufGet1d()	Get data from a 1D area of a buffer and	imBufGet1d()	

	place it in a user-supplied array.		
MbufGet2d()	Get data from a 2D area of a buffer and place it in a user-supplied array.	imBufGet2d()	
MbufGetColor()	Get data from one or all bands of a buffer and place it in a user-supplied array.	imBufChild() and imBufGet()	First, allocate a child buffer in a certain band of color, and then get the data.
MbufGetLine()	Read a series of pixels within specified coordinates, count them, and store them in a user-defined array.	imBufMap()	Create a pointer to the buffer data using imBufMap(), then use the pointer to read the pixels along the line.
MbufImport()	Import data from a file into a data buffer taking into account its file format.	imBufRestore() or imBufLoad()	imBufRestore() loads data from a file into an automatically allocated buffer while imBufLoad() loads data into a specified buffer.
MbufInquire()	Inquire about a data buffer	imBufInquire()	
MbufLoad()	Load data from a file into a data buffer assuming it is in a MIL file format.	imBufLoad()	
MbufPut()	Transfer data from Host memory to a buffer.	imBufPut()	
MbufPutColor()	Put data from a user-supplied array into one or all bands of a data buffer.	imChild() and imBufPut()	First, allocate a child buffer, and then transfer data to the buffer.
MbufPutLine()	Write a specified series of pixels within specified	imBufMap()	Create a pointer to the buffer data using imBufMap, and then

	coordinates on a line.		use the pointer to write the pixels along the line.
MbufPut1d()	Put data from a user-supplied array into a 1D area of a buffer.	imBufPut1d()	
MbufPut2d()	Put data from a user-supplied array into a 2D area of a buffer.	imBufPut2d()	
MbufRestore()	Restore MIL file format data from a file into an automatically allocated data buffer.	imBufRestore()	
MbufSave()	Save a data buffer in a file using the MIL output file format.	imBufSave()	
MbufFree()	Free a data buffer.	imBufFree()	
MbufGet()	Get data from a buffer and place it in a user-supplied array.	imBufGet()	
MbufGet1d()	Get data from a 1D area of a buffer and place it in a user-supplied array.	imBufGet1d()	
MbufGet2d()	Get data from a 2D area of a buffer and place it in a user-supplied array.	imBufGet2d()	
MbufGetColor()	Get data from one or all bands of a buffer and place it in a user-supplied array.	imBufChild() and imBufGet()	First, allocate a child buffer in a certain band of color, and then get the data.
MbufGetLine()	Read a series of pixels within specified coordinates, count	imBufMap()	Create a pointer to the buffer data using imBufMap(), then use the pointer to

	them, and store them in a user-defined array.		read the pixels along the line.
MbufImport()	Import data from a file into a data buffer taking into account its file format.	imBufRestore() or imBufLoad()	imBufRestore() loads data from a file into an automatically allocated buffer while imBufLoad() loads data into a specified buffer.
MbufInquire()	Inquire about a data buffer.	imBufInquire()	
MbufLoad()	Load data from a file into a data buffer assuming it is in a MIL file format.	imBufLoad()	
MbufPut()	Transfer data from Host memory to a buffer.	imBufPut()	
MbufPutColor()	Put data from a user-supplied array into one or all bands of a data buffer.	imChild() and imBufPut()	First, allocate a child buffer, and then transfer data to the buffer.
MbufPutLine()	Write a specified series of pixels within specified coordinates on a line.	imBufMap()	Create a pointer to the buffer data using imBufMap, and then use the pointer to write the pixels along the line.
MbufPut1d()	Put data from a user-supplied array into a 1D area of a buffer.	imBufPut1d()	
MbufPut2d()	Put data from a user-supplied array into a 2D area of a buffer.	imBufPut2d()	
MbufRestore()	Restore MIL file format data from a file into an automatically	imBufRestore()	

	allocated data buffer.		
MbufSave()	Save a data buffer in a file using the MIL output file format.	imBufSave()	
MbufInquire()	Inquire about a data buffer	imBufInquire()	
MbufLoad()	Load data from a file into a data buffer assuming it is in a MIL file format.	imBufLoad()	
MbufPut()	Transfer data from Host memory to a buffer.	imBufPut()	
MbufPutColor()	Put data from a user-supplied array into one or all bands of a data buffer.	imChild() and imBufPut()	First, allocate a child buffer, and then transfer data to the buffer.
MbufPutLine()	Write a specified series of pixels within specified coordinates on a line.	imBufMap()	Create a pointer to the buffer data using imBufMap, and then use the pointer to write the pixels along the line.
MbufPut1d()	Put data from a user-supplied array into a 1-d area of a buffer.	imBufPut1d()	
MbufPut2d()	Put data from a user-supplied array into a 2-d area of a buffer.	imBufPut2d()	
MbufRestore()	Restore MIL file format data from a file into an automatically allocated data buffer.	imBufRestore()	
MbufSave()	Save a data buffer in a file using the	imBufSave()	

	MIL output file format.		

The Data Allocation and Access Module

MIL Command	MIL Description	Native Library Command	Comments
MdigAlloc()	Allocate a digitizer.	imCamAlloc() or imDigAlloc()	Generally imCamAlloc() will be used, imDigAlloc() is used when there is more than one digitizer (and necessary to specify which digitizer a function should use).
MdigAverage()	Frame sequence averaging from an input device.	imDigGrab() and processing function	First grab into a buffer, and then call processing function depending on the kind of averaging needed to be performed.
MdigChannel()	Select the active input channel of a digitizer.	imCamControl()	imCamControl() has an item parameter (IM_DIG_CHANNEL).
MdigControl()	Control the specified digitizer.	imCamControl() or imDigControl()	Generally most digitizer attributes can be set using imCamControl(), imDigControl() programs the digitizer directly and will interfere with other applications using the digitizer.
MdigFree()	Free a digitizer.	imCamFree() or imDigFree()	
MdigGrab()	Grab data from an input device into a buffer.	imDigGrab()	
MdigGrabContinuous()	Grab data continuously from an input device.	imDigGrab()	imDigGrab() has the Count parameter that can be set

			IM_CONTINUOUS which will grabs continuously until imThrHalt() is called.
MdigGrabWait()	Wait for the end of the grab in progress.	imSyncHost() or imSyncThread()	imSyncHost() and imSyncThread() both have the State parameter that can be set to IM_COMPLETED (wait until the function is completed).
MdigHalt()	Halt a continuous grab from an input device.	imThrHalt()	
MdigHookFunction()	Hook a function to a digitizer event.	Not available	
MdigInquire()	Inquire about a digitizer.	imCamInquire() or imDigInquire()	imCamInquire() should be used to inquire about most digitizer attributes, imDigInquire() can be used to inquire about the input line attribute.
MdigLut()	Copy a LUT buffer to a digitizer.	imCmControl() a	
MdigReference()	Select digitization reference level.	imCamControl()	With Item parameter: IM_DIG_REF_BLACK and IM_DIG_REF_WHITE.

The Display Control Module

MIL Command	**MIL Description**	**Native Library Command**	**Comments**
MdispAlloc()	Allocate a display.	imDispAlloc()	Not needed unless there is more than one display.
MdispControl()	Set display attributes.	imDispControl()	

MdispDeselect()	Stop displaying an image buffer.	Not applicable	
MdispHookFunction()	Hook a function to a display event.	Not available	
MdispFree()	Free a display.	imDispFree()	
MdispInquire()	Inquire about a display.	imDispInquire()	
MdispLut()	Copy a LUT buffer to the display output LUT.	imDispControl()	imDispControl() has a Control parameter that can be set to IM_DISP_LUT_BUF field.
MdispOverlayKey()	Enable overlay keying.	imDispCdontrol()	imDispControl() has a Control parameter that has a IM_KEY_MODE field, which you can enable overlay keying.
MdispPan()	Pan and scroll a display.	imDispControl()	imDispControl() has a Control parameter that has a IM_DISP_PAN_X or _Y field, which allows image displacement.
MdispSelect()	Select an image buffer to display.	imBufChild() and imBufCopy()	First, allocate a buffer for display, then grab or copy the image into that buffer.
MdispSelectWindow()	Select an image buffer to display in a user-defined window.	Not available	
MdispZoom()	Zoom a display.	imDispControl()	mDispControl() has a Control parameter that has a IM_DISP_ZOOM field, in which you can zoom the display by a specified factor.

The Data Generation Module

MIL Command	MIL Description	Native Library Command	Comments
MgenLutFunction()	Generate data into a LUT buffer.	imGen1d()	
MgenLutRamp()	Generate ramp data into a LUT buffer.	imGen1d()	

The Graphics Module

MIL Command	MIL Description	Native Library Command	Comments
MgraAlloc()	Allocate a graphics context.	imBufAllocControl()	A graphics context can be set in an ordinary buffer through desired graphic fields.
MgraArc()	Draw an arc.	imGraArc()	
MgraArcFill()	Draw a filled elliptical arc.	imGraArcFill()	
MgraBackColor()	Associate a background color with a graphics context.	imBufPutField()	See imGraText(): IM_GRA_BACK_COLOR Field.
MgraClear()	Clear an image buffer.	imBufClear()	
MgraColor()	Associate a foreground color with a graphics context.	imBufPutField()	See imGraText(): IM_GRA_COLOR Field.
MgraControl()	Control the specified graphic context.	imBufPutField()	See imGraText(): IM_GRA_BACK_MODE Field.
MgraDot()	Draw a dot.	imGraLine() or imGraRect()	A dot can be created by drawing a line or rectangle of one pixel.
MgraFill()	Perform a boundary-type seed fill.	imGraFill()	
MgraFont()	Associate a text font with a graphics context.	imBufPutField()	See imGraText(): IM_GRA_FONT Field.

MgraFontScale()	Associate a font scale with a graphics context.	imBufPutField()	See imGraText(): IM_GRA_FONT_SCALE_X or _Y Field.
MgraFree()	Free a graphics context.	imBdufFree()	A graphics context can be freed on an ordinary buffer for desired graphic fields.
MgraInquire()	Inquire about the graphic parameters.	imBufGetField() or imBufGetFieldDouble()	imBufGetField() returns a field value as a type long while imBufGetFieldDouble() returns as a type double.
MgraLine()	Draw a line.	imGraLine() or imGraPlot()	imGraPlot() is a faster method to draw a series of lines.
MgraRect()	Draw a rectangle.	imGraRect()	
MgraRectFill()	Draw a filled rectangle.	imGraRectFill()	
MgraText()	Write text.	imGraText()	

The Image Processing Module

MIL Command	**MIL Description**	**Native Library Command**	**Comments**
MimAllocResult()	Allocate an image processing result buffer.	imBufAlloc()	
MimArith()	Perform a point-to-point arithmetic operation.	binary: imBinTriadic() integer: imIntMonadic(), imIntDyadic() or imIntTriadic() floating point: imFloatDyadic(), imFloatMonadic(), or imFloatUnary() Mix binary with integer or integer with floating point, must first convert with	See also Optimization with Matrox Genesis Native Library and Data Management.

		imFloatConvert() or imBinConvert()	
MimBinarize()	Perform a point-to-point binary-thresholding operation.	imIntBinarize()	
MimClip()	Perform a point-to-point clipping operation.	imIntLutMap() or other processing functions.	Depends on type of clipping.
MimClose()	Perform a closing-type morphological operation.	binary: imBinMorphic() grayscale: imIntErodeDilate()	Perform a closing operation by performing a dilation followed by an erosion. With both imBinMorphic() and imIntErodeDilate(), a user specifies erosion or dilation with OP parameter.
MimConnectMap()	Perform a 3x3 connectivity mapping.	imIntConnectMap()	
MimConvert()	Perform a color conversion.	imIntConvertColor()	
MimConvolve()	Perform a general convolution operation.	imIntConvolve()	
MimCountDifference()	Count image differences.	imIntCountDifference()	
MimDilate()	Perform a dilation type morphological operation.	binary: imBinMorphic() grayscale: imIntErodeDilate()	With the Native Library, depends on data type.
MimDistance	Perform a distance transform.	imIntDistance()	
MimEdgeDetect()	Perform a specific edge detection operation and produce a gradient intensity and/or gradient angle image.	imIntConvolve() and other processing functions.	Specify edge detection identifier (horizontal, vertical, laplacian, etc.) with IntConvolve's Kernel parameter.
MimErode()	Perform an erosion type	binary: imBinMorphic()	Depends on data type.

	morphological operation.	grayscale: imIntErodeDilate()	
MimFindExtreme()	Find an image buffer's extremes (min, max).	imIntFindExtreme()	
MimFree()	Free an image processing result buffer.	imBufFree()	
MimGetResult()	Get values from an image processing result buffer.	imBufGet()	Normal buffer functions.
MimGetResult1d()	Get values from an image processing 1D region of a result buffer.	imBufGet1d()	Normal buffer functions.
MimHistogram()	Generate the intensity histogram of an image buffer.	imIntHistogram()	
MimHistogramEqualize()	Perform a histogram equalization of an image.	imIntHistogramEqualize()	
MimInquire()	Inquire about an image processing result buffer.	imBufInquire()	
MimLabel()	Label objects in an image buffer.	imIntLabel()	
MimLocateEvent()	Locate event of a specified type in an image.	imIntLocateEvent()	
MimLutMap()	Perform a point-to-point LUT-mapping operation.	imIntLutMap()	
MimMorphic()	Morphological transformation.	binary: imBinMorphic() grayscale: imIntErodeDilate() imIntThickThin()	Depends on data type.
MimOpen()	Perform an opening-type	binary: imBinMorphic()	Depending on data type, perform an opening operation

	morphological operation.	grayscale: imIntErodeDilate()	by performing an erosion followed by dilation. With both, specify erosion or dilation with OP parameter.
MimProject()	Project a 2D image into 1D.	imIntProject()	
MimRank()	Perform a rank filter on the pixels in an image.	imIntRank()	User-defined kernels not supported.
MimResize()	Resize an image.	generally: imIntScale() or imIntWarpPolynomial()zoom up by an x and y factor: imIntZoom()zoom down by an x and y factor: imIntSubsample()	
MimRotate()	Rotate an image.	imIntWarpPolynomial() or imIntFlip()	imIntFlip() is faster, but can be used for 90 degree increments only.
MimShift()	Perform a point-to-point bit shift.	imIntMonadic()	Set operation (op) parameter to: IM_SHIFT.
MimThick()	Thicken blobs in an image.	binary: imBinMorphic() grayscale: imIntThickThin()	Depends on data type.
MimThin()	Thin blobs in an image.	binary: imBinMorphic() grayscale: imIntThickThin()	Depends on data type.
MimTranslate()	Translate an image in x and/or y.	imIntWarpPolynomial()	Equivalent to linear translating.
MimZoneOfInfluence()	Perform a zone of influence detection.	Not available	

The Measurement Module

MIL Command	MIL Description	Native Library Command	Comments
MmeasAllocContext()	Allocate a measurement context.	Not available	
MmeasAllocMarker()	Allocate a measurement marker.	Not available	
MmeasAllocResult()	Allocate a measurement result buffer.	Not available	
MmeasCalculate()	Calculate measurements using two markers.	Not available	
MmeasControl()	Set a measurement context control parameter.	Not available	
MmeasFindMarker()	Find a marker in an image and take its measurements.	Not available	
MmeasFree()	Free a measurement buffer (marker, result, or context).	Not available	
MmeasGetResult()	Get the results of measurements taken.	Not available	
MmeasInquire()	Inquire about a marker, result or context buffer.	Not available	
MmeasRestoreMarker()	Restore a marker from disk.	Not available	
MmeasSaveMarker()	Save a marker to disk.	Not available	
MmeasSetMarker()	Set a marker parameter.	Not available	

The Pattern Recognition Module

MIL Command	MIL Description	Native Library Command	Comments
MpatAllocAutomodel()	Automatically allocate a unique pattern-matching model from a source image.	Not available	
MpatAllocModel()	Allocate a pattern-matching model from a source image.	imPatAllocModel()	
MpatAllocResult()	Allocate a pattern matching result buffer.	imPatAllocResult()	
MpatAllocRotatedModel()	Rotate a pattern-matching model.	imPatAllocRotatedModel()	
MpatCopy()	Copy a pattern-matching model to an image buffer.	imPatCopy()	
MpatFindModel()	Find a pattern-matching model in the target image buffer.	imPatFindModel()	
MpatFindMultipleModel()	Find multiple pattern matching models in the target image buffer.	imPatFindModel()	Use imPatFindModel() several times, specifying the different models in each separate Model parameter.
MpatFindOrientation()	Find the orientation of an image or of an object in an image.	Not available	
MpatFree()	Free a pattern-matching buffer (model or result buffer).	imPatFree()	
MpatGetNumber()	Get the number of model occurrences in the target image.	imPatGetNumber()	

MpatGetResult()	Get the pattern matching result values.	imPatGetResult()	
MpatInquire()	Inquire about a pattern-matching model.	imPatInquire()	
MpatPreprocModel()	Preprocess a pattern-matching model.	ImPatPreprocModel()	
MpatRead()	Read a pattern-matching model from an open file.	imPatRead()	
MpatRestore()	Restore a pattern-matching model from a file.	imPatRestore()	
MpatSave()	Save a pattern-matching model to a file.	imPatSave()	
MpatSetAcceptance()	Set the pattern matching acceptance level.	imPatSetAcceptance()	
MpatSetAccuracy()	Set the pattern matching positional accuracy.	imPatSetAccuracy()	
MpatSetAngle()	Set the angular search control parameters of a model.	Not available	
MpatSetCenter()	Set the pattern matching model center.	imPatSetCenter()	
MpatSetCertainty()	Set the pattern matching certainty level.	imPatSetCertainty()	
MpatSetDontCare()	Set model pixels to the "don't care" state.	imPatSetDontCare()	
MpatSetNumber()	Set the expected number of matches.	imPatSetNumber()	
MpatSetPosition()	Set the pattern matching search position.	imPatSetPosition()	

MpatSetSearchParameter()	Set a pattern matching internal search parameter.	imPatSetSearchParameter()	
MpatSetSpeed()	Set the pattern matching search speed.	imPatSetSpeed()	
MpatWrite()	Write a pattern-matching model to an open file.	imPatWrite()	

The System Allocation and Inquiry Module

MIL Command	MIL Description	Native Library Command	Comments
MsysAlloc()	Allocate a system.	imDevAlloc()	
MsysControl()	Control a system behavior.	imDevControl() or imAppControl()	
MsysFree()	Free a system.	DevFree()	
MsysInquire()	Inquire about a system.	imSysInquire() or imDevInquire() or ImAppInquire	

The OCR Commands

MIL Command	MIL Description	Native Library Command	Comments
MocrAllocFont()	Allocate an OCR font buffer.	Not available	
MocrAllocResult()	Allocate an OCR result buffer.	Not available	
MocrCalibrateFont()	Automatically calibrate the target font's character size to match a sample image.	Not available	
MocrControl()	Set OCR processing controls.	Not available	
MocrCopyFont()	Copy a font character to or from an image buffer.	Not available	
MocrFree()	Free an OCR font or result buffer.	Not available	

MocrGetResult()	Read results from an OCR result buffer.	Not available	
MocrHookFunction()	Hook a custom checksum function.	Not available	
MocrImportFont()	Import font data from file on disk.	Not available	
MocrInquire()	Retrieves font character information.	Not available	
MocrModifyFont()	Invert or resize a font to match the target image character.	Not available	
MocrReadString()	Read an unknown string from an image.	Not available	
MocrRestoreFont()	Restore a font from disk.	Not available	
MocrSaveFont()	Save an existing font and its current setting to disk.	Not available	
MocrSetConstraint()	Set the valid character for each position in the string.	Not available	
MocrVerifyString()	Verify a known string in an image.	Not available	

Optimizing with Matrox Genesis Native Library

In most applications, improved performance can be achieved by reducing the number of function calls required to perform an operation. With the Native Library, a user can make use of various multiple operation functions like imIntTriadic() to perform arithmetic and logical operations on up to three operands. Additionally, there are several other commands that can be used to optimize an application that should be considered (for example, imBinTriadic(), imIntMac1(), imIntMac2(), imFloatMac1(), and imFloatMac2()).