

Matrox Imaging Library

Version 9.0 with Processing Pack 1

Reference

Manual no.Y10513-701-0910

July 29, 2009

Matrox® is a registered trademark of Matrox Electronic Systems Ltd.

Microsoft®, MSDN®, Visual Basic®, Visual C++®, Visual C#®, Visual Studio®, and Windows® are registered trademarks of Microsoft Corporation.

PCI-X® and PCI Express® are registered trademarks of PCI-SIG.

PCIe™ is a trademark of PCI-SIG.

CompactPCI® is a registered trademark of PCI Industrial Computer Manufacturers' Group.

Camera Link® is a registered trademark of the Automated Imaging Association (AIA).

GigE Vision™ is a trademark of the Automated Imaging Association (AIA).

Intel®, MMX®, Celeron®, and Pentium® are registered trademarks of Intel Corporation.

Texas Instruments® is a registered trademark of Texas Instruments Incorporated.

PowerPC® is a registered trademark of International Business Machines Corporation, used under license by Freescale Semiconductor Inc.

Freescale™ is a trademark of Freescale Semiconductor Incorporated.

Linux® is a registered trademark of Linus Torvalds.

Red Hat® is a registered trademark of Red Hat, Inc.

SUSE® is a registered trademark of Novell, Inc.

Ubuntu® is a registered trademark of Canonical, Ltd.

Portions of this software are copyright ©2007 The FreeType Project.

Parts of the MIL Driver for GigE Vision™ are copyrighted © and are the properties of StorageCraft, Inc.

Some scripts in this help file are implemented using the jQuery library version 1.2.6 copyright©2008 John Resig.

The ColorSpace tool is copyrighted © and is the property of Philippe Colantoni.

All other nationally and internationally recognized trademarks and tradenames are hereby acknowledged.

Protected by U.S. Patents 7,027,651; 7,319,791; 7,327,888; U.S. Patents Pending.

© Copyright Matrox Electronic Systems Ltd., 1992-2009.

All rights reserved. Limitation of Liabilities: In no event will Matrox or its suppliers be liable for any indirect, special, incidental, economic, cover or consequential damages arising out of the use of or inability to use the product, user documentation or related technical support, including without limitation, damages or costs relating to the loss of profits, business, goodwill, even if advised of the possibility of such damages. In no event will Matrox and its suppliers' liability exceed the amount paid by you, for the product.

Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation might not apply to you.

Disclaimer: Matrox Electronic Systems Ltd. reserves the right to make changes in specifications at any time and without notice. The information provided by this document is believed to be accurate and reliable. However, neither Matrox Electronic Systems Ltd. nor its suppliers assume any responsibility for its use; or for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent right of Matrox Electronic Systems Ltd.

M3dmap functions

Synopsis

3dmap module can be used to obtain corrected information from a laser line profiling setup. Such a setup consist of a device projecting a sheet of light, usually a laser diode, coupled with a usual camera. When an object moves under the laser plane, the camera takes images of the intersection of that plane with the object, thus resulting in a set of images containing a laser line, each line representing a "slice" of the object being scanned.

Functions

- [M3dmapAddScan](#)
- [M3dmapAlloc](#)
- [M3dmapAllocResult](#)
- [M3dmapCalibrate](#)
- [M3dmapControl](#)
- [M3dmapExtract](#)
- [M3dmapFree](#)
- [M3dmapGetResult](#)
- [M3dmapInquire](#)
- [M3dmapTriangulate](#)

M3dmapAddScan

Synopsis

Extract laser line information from an image and store data in a 3D reconstruction result buffer.

Syntax

```
void M3dmapAddScan(
    MIL_ID ContextId,
    MIL_ID ResultId,
    MIL_ID ImageId,
    MIL_ID LineIntensityImageId,
    MIL_ID ExtraInfoArrayId,
    MIL_INT Index,
    MIL_INT ControlFlag
)
```

Description

This function stores laser line data extracted from grabbed images into the specified result buffer. You can use this data to calibrate a laser reconstruction setup with [M3dmapCalibrate\(\)](#) or to generate a depth map with [M3dmapExtract\(\)](#).

If the number of times **M3dmapAddScan()** is called is greater than the maximum number of frames to be stored, specified using [M3dmapControl\(\)](#) with [M_MAX_FRAMES](#), the result buffer's oldest values are overwritten with the new ones.

The size and depth of the image buffer must not change when multiple calls are made to **M3dmapAddScan()** with the same result buffer.

You can clear the contents of the result buffer with [M_RESET](#) without having to free and reallocate it. For example, you can use the same result buffer to calibrate your 3D reconstruction setup and to accumulate scan lines.

Parameters

ContextId

Specifies the context identifier of the 3D reconstruction setup used to grab the image. The 3D reconstruction context must have been previously allocated on the required system using [M3dmapAlloc\(\)](#).

Note that this parameter must be set to **M_NULL** if [M_RESET](#) is specified.

ResultId

Specifies the identifier of the 3D reconstruction result buffer in which to store or clear scan line data. The 3D reconstruction result buffer must have been previously allocated with [M3dmapAllocResult\(\)](#).

ImageId

Specifies the identifier of the image buffer containing the grabbed image of the laser line. This buffer must have been previously allocated with [MbufAlloc2d\(\)](#). The image must be a 1 band, 8-bit or 16-bit, unsigned buffer and its size must not exceed 65535 pixels by 65535 pixels.

Note that this parameter must be set to **M_NULL** if [M_RESET](#) is specified.

LineIntensityImageId

Reserved for future expansion. This parameter must be set to **M_NULL**.

ExtraInfoArrayId

Reserved for future expansion. This parameter must be set to **M_NULL**.

Index

Reserved for future expansion. This parameter must be set to **M_DEFAULT**.

ControlFlag

Specifies the function's control flag. This parameter must be set to one of the following values:

● To specify the function's control flag	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Accumulates laser line data into the result buffer specified by ResultId .
<input type="checkbox"/> M_RESET	Clears all scan lines from the 3D reconstruction result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

M3dmapAlloc

Synopsis

Allocate a 3D reconstruction context.

Syntax

```
MIL_ID M3dmapAlloc(
    MIL_ID SystemId,
    MIL_INT ContextType,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr
)
```

Description

This function allocates a 3D reconstruction context on the specified system. When the 3D reconstruction context is no longer required, you should release its memory, using [M3dmapFree\(\)](#).

Parameters

SystemId

Specifies the identifier of the system on which to allocate the 3D reconstruction context.

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ContextType

Specifies the context type. This parameter must be set to the value below:

● For specifying the type of context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_LASER	Specifies a 3D reconstruction context that will be used to perform laser line profiling.

ControlFlag

Specifies the type of depth map (partially corrected or fully corrected) to be generated. This parameter must be set to one of the following values:

● For specifying the laser line profiling mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CALIBRATED_CAMERA_LINEAR_MOTION	Specifies that the 3D reconstruction context will be used to generate a fully corrected depth map. That is, the shape and the gray level of the depth map will be corrected. A cloud of 3D points can also be obtained. Note that the calibration object of the camera setup must have been allocated with M_TSAI_BASED or M_3D_ROBOTICS . (summarize)
<input type="checkbox"/> M_DEPTH_CORRECTION	Specifies that the 3D reconstruction context will be used to generate a partially corrected depth map. That is, only the depth (gray level) of the depth map will be corrected. The geometric shape of the depth map will not be corrected.

	(summarize)
--	-----------------------------

ContextIdPtr

Specifies the address of the variable in which to write the 3D reconstruction context identifier. Since the **M3dmapAlloc()** function also returns the 3D reconstruction context identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the 3D reconstruction context identifier if allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

M3dmapAllocResult

Synopsis

Allocate a 3D reconstruction result buffer.

Syntax

```
MIL_ID M3dmapAllocResult (
    MIL_ID SystemId,
    MIL_INT ResultType,
    MIL_INT ControlFlag,
    MIL_ID *ResultIdPtr
)
```

Description

This function allocates a result buffer, on the specified system, to be used with a 3D reconstruction context. When the result buffer is no longer required, release its memory, using [M3dmapFree\(\)](#).

Parameters

SystemId

Specifies the identifier of the system on which to allocate the result buffer.

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ResultType

Specifies the type of result buffer to allocate. This parameter must be set to the following value:

● For specifying the type of result buffer to allocate	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_LASER_DATA	Specifies to allocate a 3D reconstruction result buffer which will be used to store laser line profiling results.

ControlFlag

Specifies the function's control flag. This parameter must be set to **M_DEFAULT**.

ResultIdPtr

Specifies the address of the variable in which to write the 3D reconstruction result buffer identifier. Since **M3dmapAllocResult()** also returns the 3D reconstruction result buffer identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the 3D reconstruction result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
--------	----------------

Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

M3dmapCalibrate

Synopsis

Perform calibration of the 3D reconstruction setup.

Syntax

```
void M3dmapCalibrate(
    MIL_ID ContextId,
    MIL_ID ResultId,
    MIL_ID CalibrationId,
    MIL_INT ControlFlag
)
```

Description

This function is used to calibrate your 3D reconstruction setup.

If the 3D reconstruction context is set to generate fully corrected depth maps ([M_CALIBRATED_CAMERA_LINEAR_MOTION](#) 3D reconstruction mode), the 3D reconstruction setup calibration is done using the calibration object specified by **CalibrationId** and the extracted laser line data stored in the result buffer. In this mode, the following setup constraints must be met:

- The laser plane must be vertical (no tilt).
- The laser plane must be perpendicular to the conveyor (no rotation).
- The camera must have a sufficient angle with respect to the Z-axis to have accurate results; that is, the camera must be able to grab meaningful displacements of the laser line for different depths. For more information, see the [Camera angle requirement](#) subsection in the [Laser line profiling requirements](#) section in [Chapter 16: 3D Reconstruction](#).
- The calibration object must have been allocated in a 3D mode such as [M_TSAI_BASED](#) or [M_3D_ROBOTICS](#).

If the 3D reconstruction context specified is set to generate partially corrected depth maps ([M_DEPTH_CORRECTION](#) 3D reconstruction mode), the reconstruction setup calibration will be done with the extracted laser lines data stored in the result buffer. With this type of context, there is no constraint on the orientation of the laser plane; that is, it doesn't need to be perfectly vertical or perpendicular to the conveyor motion. However, the geometric shape of the depth map generated with [M3dmapExtract\(\)](#) will not be corrected; only the depth (gray level) is adjusted to reflect real depth. You do not need to provide a calibration object and the **CalibrationId** parameter must be set to **M_NULL**.

For the steps required to calibrate your 3D reconstruction setup in either mode, see the [Calibrating your 3D reconstruction setup](#) section in [Chapter 16: 3D Reconstruction](#).

Parameters

ContextId

Specifies the identifier of the 3D reconstruction setup's context used to grab images of the laser line. The context must have been previously allocated on the required system using [M3dmapAlloc\(\)](#).

ResultId

Specifies the identifier of the 3D reconstruction result buffer holding laser line calibration data. The 3D reconstruction result buffer must have been previously allocated with [M3dmapAllocResult\(\)](#) and contain data for at least one laser line.

CalibrationId

Specifies the identifier of the calibration object holding the 3D camera calibration data or **M_NULL**.

For fully corrected depth maps, this parameter must be set to the identifier of a calibration object allocated using [McalAlloc\(\)](#) with [M_TSAI_BASED](#) or [M_3D_ROBOTICS](#).

For partially corrected depth maps, this parameter must be set to **M_NULL**.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

M3dmapControl

Synopsis

Control a 3D reconstruction context or result buffer.

Syntax

```
void M3dmapControl(
    MIL_ID ContextOrResultId,
    MIL_INT Index,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function is used to control settings for the 3D reconstruction context or the 3D reconstruction result buffer.

Parameters

- ContextOrResultId
- Specifies the identifier of the 3D reconstruction context or the result buffer to control.
- Index
- Reserved for future expansion. This parameter must be set to **M_DEFAULT**.
- ControlType
- Specifies the type of control to set.
- See the [Parameter associations](#) section for possible values.
- ControlValue
- Specifies the required value for the control.
- See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For a 3D reconstruction context](#)
- [For a 3D reconstruction context in M_CALIBRATED_CAMERA_LINEAR_MOTION 3D reconstruction mode](#)
- [For a 3D reconstruction context in M_DEPTH_CORRECTION 3D reconstruction mode](#)
- [For a 3D reconstruction result buffer](#)

The possible ControlType and corresponding ControlValue parameter settings for a 3D reconstruction context are:

For a 3D reconstruction context	
ControlType	Description
ControlValue	

<input type="checkbox"/> M_MIN_INTENSITY	Sets the minimum average intensity level for a group of pixels to be considered a laser line. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 100.0. (summarize)
<input type="checkbox"/> Value >= 0.0	Specifies the minimum average intensity level.
<input type="checkbox"/> M_ORIENTATION	Sets whether laser line detection is performed vertically or horizontally. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_1D_COLUMNS .
<input type="checkbox"/> M_1D_COLUMNS	Specifies that laser line detection is performed vertically. In this case, the laser line should be horizontal. (summarize)
<input type="checkbox"/> M_1D_ROWS	Specifies that laser line detection is performed horizontally. In this case, the laser line should be vertical. (summarize)
<input type="checkbox"/> M_PEAK_WIDTH	Sets the approximate width of the laser line. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 10 pixels. (summarize)
<input type="checkbox"/> Value > 0	Specifies the approximate width of the laser line, in pixels.

The possible ControlType and corresponding ControlValue parameter settings for a 3D reconstruction context of type [M_CALIBRATED_CAMERA_LINEAR_MOTION](#) are:

For a 3D reconstruction context in M_CALIBRATED_CAMERA_LINEAR_MOTION 3D reconstruction mode	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_SCAN_SPEED	Sets the speed of the object being scanned, in the Y direction, in world units per frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value	Specifies the speed in world units per frame. Note that this value can be negative if the object is going in the negative Y direction. (summarize)

The possible ControlType and corresponding ControlValue parameter settings for a 3D reconstruction context of type [M_DEPTH_CORRECTION](#) are:

For a 3D reconstruction context in M_DEPTH_CORRECTION 3D reconstruction mode	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_CORRECTED_DEPTH	Sets the gray level to use to represent the depth established from the next laser line (M3dmapAddScan()) when calibrating your 3D reconstruction setup. This only applies when calibrating for a partially corrected depth map. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 <= Value <= 254	Specifies the gray level for 8-bit depth maps. Note that 255 is used to indicate an invalid value (no data). (summarize)
<input type="checkbox"/> 0 <= Value <= 65534	Specifies the gray level for 16-bit maps. Note that 65535 is used to indicate an invalid value (no data). (summarize)

The possible ControlType and corresponding ControlValue parameter settings for a 3D reconstruction result buffer are:

|--|

For a 3D reconstruction result buffer	
ControlType	Description
ControlValue	
<input type="checkbox"/> M_FILL_MODE	Sets the mode in which to fill gaps in depth maps obtained using M3dmapExtract() . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Disables the filling of gaps.
<input type="checkbox"/> M_X_THEN_Y	Specifies that each depth map row is analyzed so that each row is filled, then each column is analyzed to fill the remaining columns. The filling operation used depends on the M_FILL_SHARP_ELEVATION_DEPTH and M_FILL_SHARP_ELEVATION control types. Invalid data that is on the boundary of the image (outside of the scanned object) will not be filled. (summarize)
<input type="checkbox"/> M_FILL_SHARP_ELEVATION	Sets which boundary will be used to fill sharp elevation gaps in depth maps obtained using M3dmapExtract() . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Disables the filling of sharp elevation gaps.
<input type="checkbox"/> M_MAX	Specifies that the destination pixel will be set to the maximum value of the two pixels on either side of the gap.
<input type="checkbox"/> M_MIN	Specifies that the destination pixel will be set to the minimum value of the two pixels on either side of the gap.
<input type="checkbox"/> M_FILL_SHARP_ELEVATION_DEPTH	Sets the threshold used to differentiate between gradual elevation gaps and sharp elevation gaps in the depth map. If the difference between a gap's boundary values is less than the specified threshold, the gap is considered a gradual elevation gap, otherwise it is considered a sharp elevation gap. Gradual elevation gaps are filled using linear interpolation; sharp elevation gaps are either filled by propagating the value of one of the boundaries or are left unfilled, depending on the M_FILL_SHARP_ELEVATION control type. It is often better to fill gradual elevation gaps using linear interpolation since the underlying surface of such a gap is considered to be part of the same surface as its boundaries. Whereas for sharp elevation gaps, the underlying surface is considered to be discontinuous at least at one of its boundaries, so propagating the value of one of the boundaries is recommended. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_INFINITE .
<input type="checkbox"/> M_INFINITE	Specifies that the threshold is infinite. When M_INFINITE is specified, all gaps are considered gradual elevation gaps, so linear interpolation is always used and M_FILL_SHARP_ELEVATION is ignored. (summarize)
<input type="checkbox"/> Value >= 0.0	Specifies the threshold, in world units specified using M_GRAY_LEVEL_SIZE_Z . When the threshold is set to 0.0, all gaps are considered sharp elevation gaps, so linear interpolation is never used and all gaps are filled according to M_FILL_SHARP_ELEVATION . (summarize)
<input type="checkbox"/> M_GRAY_LEVEL_SIZE_Z	Sets the length, in world units, of one gray level, in the Z-direction. In M_CALIBRATED_CAMERA_LINEAR_MOTION 3D reconstruction mode, this control type only affects how M3dmapExtract() generates and draws the depth map in the specified image buffer. In M_DEPTH_CORRECTION 3D reconstruction mode, this control type must be set to M_DEFAULT . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value != 0.0	Specifies the length of one gray level. A negative value indicates that brighter depth map pixels reflect world points with lower Z values. (summarize)
<input type="checkbox"/> M_MAX_FRAMES	Sets the maximum number of scan lines that the result buffer should keep internally. Note that, if the value of this control type is changed after it has already been set, scan lines can be lost. (summarize)

<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1024. (summarize)
<input type="checkbox"/> Value > 0	Specifies the maximum number of scan lines to keep.
<input type="checkbox"/> M_PIXEL_SIZE_X	Sets the length, in world units, of one pixel, in the X-direction. In M_CALIBRATED_CAMERA_LINEAR_MOTION 3D reconstruction mode, this control type only affects how M3dmapExtract() generates and draws the depth map in the specified image buffer. In M_DEPTH_CORRECTION 3D reconstruction mode, this control type must be set to M_DEFAULT . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0.0	Specifies the length of one pixel.
<input type="checkbox"/> M_PIXEL_SIZE_Y	Sets the length, in world units, of one pixel, in the Y-direction. In M_CALIBRATED_CAMERA_LINEAR_MOTION 3D reconstruction mode, this control type only affects how M3dmapExtract() generates and draws the depth map in the specified image buffer. In M_DEPTH_CORRECTION 3D reconstruction mode, this control type must be set to M_DEFAULT . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0.0	Specifies the length of one pixel.
<input type="checkbox"/> M_WORLD_OFFSET_X	Sets the X-world coordinate of the depth map's top left corner. In M_DEPTH_CORRECTION 3D reconstruction mode, this control type must be set to M_DEFAULT . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the X-offset.
<input type="checkbox"/> M_WORLD_OFFSET_Y	Sets the Y-world coordinate of the depth map's top left corner. In M_DEPTH_CORRECTION 3D reconstruction mode, this control type must be set to M_DEFAULT . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the Y-offset.
<input type="checkbox"/> M_WORLD_OFFSET_Z	Sets the Z-world coordinate for gray level 0 of the depth map. In M_DEPTH_CORRECTION 3D reconstruction mode, this control type must be set to M_DEFAULT . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the Z-offset.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

M3dmapExtract

Synopsis

Generate a depth map image and, optionally, an intensity image from accumulated scan lines.

Syntax

```
void M3dmapExtract(
    MIL_ID ResultId,
    MIL_ID DepthMapImageBufId,
    MIL_ID IntensityMapImageBufId,
    MIL_INT Operation,
    MIL_INT Index,
    MIL_INT ControlFlag
)
```

Description

This function takes the accumulated scan line data stored inside a result buffer and generates a depth map image, as well as an intensity map image, if required. The gray level of the depth map image indicates depth whereas the gray level in the intensity map image indicates luminosity.

Gaps (holes) can appear in the resulting depth map due to different phenomena. To fill these gaps, two filling operations are available: linear interpolation between boundary values and the propagation of one of the boundary values. To set how gaps should be filled, use [M3dmapControl\(\)](#) with [M_FILL_MODE](#).

Parameters

ResultId

Specifies the identifier of the result buffer which contains the laser line data.

DepthMapImageBufId

Specifies the identifier of the image buffer in which to store the depth map. The image buffer must be a 1-band, 8-bit or 16-bit unsigned buffer.

In [M_DEPTH_CORRECTION](#) 3D reconstruction mode, the X-size of the output depth map buffer should match either the X-size or Y-size of the buffer given to the [ImageId](#) parameter of [M3dmapAddScan\(\)](#), depending on whether the [M3dmapControl\(\)](#) with [M_ORIENTATION](#) is set to [M_1D_COLUMNS](#) or [M_1D_ROWS](#), respectively. The Y-size of the depth map buffer should match either the number of times [M3dmapAddScan\(\)](#) was called or the value of the [M_MAX_FRAMES](#) control, whichever is smaller. You can determine the required size of the depth map buffer using [M3dmapGetResult\(\)](#) with [M_CORRECTED_DEPTH_MAP_SIZE_X](#) or [M_CORRECTED_DEPTH_MAP_SIZE_Y](#). Similarly, you can determine the required buffer type with [M_CORRECTED_DEPTH_MAP_BUFFER_TYPE](#).

In [M_CALIBRATED_CAMERA_LINEAR_MOTION](#), there are no restrictions to the X-size, Y-size or type of the depth map buffer. However, to get a depth map which is meaningful, the buffer size and type should take into consideration the [M_PIXEL_SIZE_X](#), [M_PIXEL_SIZE_Y](#), [M_GRAY_LEVEL_SIZE_Z](#), [M_WORLD_OFFSET_X](#), [M_WORLD_OFFSET_Y](#), and [M_WORLD_OFFSET_Z](#) controls.

IntensityMapImageBufId



Specifies the identifier of the intensity image buffer. The image buffer must have the same dimensions as the buffer specified for [DepthMapImageBufId](#). In addition, the image buffer must be a 1-band, 8- or 16-bit unsigned buffer; the type and pixel depth must be the same those of the image buffer used to extract laser line data ([M3dmapAddScan\(\)](#) with [ImageId](#)). You can call [M3dmapGetResult\(\)](#) with [M_INTENSITY_MAP_BUFFER_TYPE](#) to determine the expected type and pixel depth.

If you don't need to generate the intensity map, set this parameter to [M_NULL](#).

Operation

Specifies the type of operation to perform. This parameter must always be set to the following value:

• For specifying the operation

 Value	Description
 M_CORRECTED_DEPTH_MAP	Generates a corrected depth map image and, optionally, an intensity map image.

Index

Reserved for future expansion. This parameter must be set to **M_DEFAULT**.

ControlFlag

Reserved for future expansion. This parameter must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

M3dmapFree

Synopsis

Free a 3D reconstruction context or a 3D reconstruction result buffer.

Syntax

```
void M3dmapFree(  
    MIL_ID ContextOrResultId  
)
```

Description

This function deletes the specified 3D reconstruction context or 3D reconstruction result buffer identifier, and releases any memory associated with it.

All 3D reconstruction contexts and all 3D reconstruction result buffers allocated on a particular system must be freed before the system can be freed.

Parameter

ContextOrResultId

Specifies the identifier of the 3D reconstruction context or 3D reconstruction result buffer to free. These must have been successfully allocated (with [M3dmapAlloc\(\)](#) or [M3dmapAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

M3dmapGetResult

Synopsis

Get the specified type of result(s) from a 3D reconstruction result buffer.

Syntax

```
void M3dmapGetResult(
    MIL_ID ResultId,
    MIL_INT Index,
    MIL_INT ResultType,
    void *ResultArrayPtr
)
```

Description

This function retrieves the result(s) of the specified type from a 3D reconstruction result buffer. Results are available after calibrating the 3D reconstruction setup using [M3dmapCalibrate\(\)](#).

Parameters

ResultId

Specifies the identifier of the 3D reconstruction result buffer from which to retrieve results.

Index

Reserved for future expansion. This parameter must be set to **M_DEFAULT**.

ResultType

Specifies the type of result to get.

To retrieve a result about fully corrected depth maps, the **ResultType** parameter can be set to one of the values specified in the table below:

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to the number of points obtained using [M3dmapGetResult\(\)](#) with [M_NUMBER_OF_3D_POINTS](#).

● For retrieving results about fully corrected depth maps		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_3D_POINTS_I +	Retrieves the intensities (gray level) of all extracted 3D points.	
<input type="checkbox"/> M_3D_POINTS_X +	Retrieves the X-coordinates of all extracted 3D points.	
<input type="checkbox"/> M_3D_POINTS_Y +	Retrieves the Y-coordinates of all extracted 3D points.	
<input type="checkbox"/> M_3D_POINTS_Z +	Retrieves the Z-coordinates of all extracted 3D points.	
<input type="checkbox"/> M_NUMBER_OF_3D_POINTS +	Retrieves the number of 3D points extracted from laser line data and stored in the result buffer. 0 is returned if no corrected points are available. (summarize)	
	<i>ResultArrayPtr info</i>	
	Data type: <code>MIL_DOUBLE</code>	

To retrieve a result about partially corrected depth maps, the **ResultType** parameter can be set to one of the values specified in the table below:

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a `MIL_DOUBLE`.

● For retrieving results about partially corrected depth maps	
☐ Value	Description
☐ M_CORRECTED_DEPTH_MAP_BUFFER_TYPE +	Retrieves the type (8+ M_UNSIGNED or 16+ M_UNSIGNED) that the partially corrected depth map should be allocated with.
☐ M_CORRECTED_DEPTH_MAP_SIZE_X +	Retrieves the X-size value that the depth map image buffer should be allocated with. Depending on the value of M_ORIENTATION , this is equal to the X- or Y-size of the image buffer passed to M3dmapAddScan() . (summarize)
☐ M_CORRECTED_DEPTH_MAP_SIZE_Y +	Retrieves the Y-size value that the depth map image buffer should be allocated with. This is equal to the number of times M3dmapAddScan() was called or the value of the M_MAX_FRAMES control, whichever is smaller. (summarize)

To retrieve the buffer type with which the intensity image buffer should be allocated, set the **ResultType** parameter to the following value:

Unless otherwise specified, the following values require that you pass the **ResultArrayPtr** parameter the address of a **MIL_DOUBLE**.

● For retrieving the intensity map buffer type of a partially or fully corrected depth map	
☐ Value	Description
☐ M_INTENSITY_MAP_BUFFER_TYPE +	Retrieves the depth and data type (8+ M_UNSIGNED or 16+ M_UNSIGNED) with which the intensity image buffer should be allocated. The returned depth and data type are the same as those of the image buffer, passed to M3dmapAddScan() , containing the grabbed image of the laser line (ImageId parameter). (summarize)

Combination constant for any of the possible values of the **ResultType** parameter

You can add the following value to the above-mentioned values to determine whether a result is available.

● For any result	
☐ Value	Description
☐ M_AVAILABLE	Returns whether a result is available. (summarize)
	<i>ResultType info</i> Return values: <div> <div>M_FALSE</div> <div>The result is not available to be retrieved.</div> </div> <div> <div>M_TRUE</div> <div>The result is available to be retrieved.</div> </div>

Combination constants for any of the possible values of the **ResultType** parameter

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

● For specifying the data type	
☐ Value	Description
☐ M_TYPE_CHAR	Casts the requested results to a <i>char</i> . (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> Data type: array of type <i>char</i> Array size: Depends on the result type. Note: When multiple results.

	<ul style="list-style-type: none"> • Data type: char <p>Note: When a single result.</p>
M_TYPE_DOUBLE	<p>Casts the requested results to a <i>double</i>. This is the default value.</p> <p>(summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type double <p>Array size: Depends on the result type. Note: When multiple results.</p> <ul style="list-style-type: none"> • Data type: double <p>Note: When a single result.</p>
M_TYPE_FLOAT	<p>Casts the requested results to a <i>float</i>.</p> <p>(summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type float <p>Array size: Depends on the result type. Note: When multiple results.</p> <ul style="list-style-type: none"> • Data type: float <p>Note: When a single result.</p>
M_TYPE_LONG	<p>Casts the requested results to a <i>long</i>.</p> <p>(summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type long <p>Array size: Depends on the result type. Note: When multiple results.</p> <ul style="list-style-type: none"> • Data type: long <p>Note: When a single result.</p>
M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>.</p> <p>(summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE <p>Array size: Depends on the result type. Note: When multiple results.</p> <ul style="list-style-type: none"> • Data type: MIL_DOUBLE <p>Note: When a single result.</p>
M_TYPE_MIL_ID	<p>Casts the requested results to a <i>MIL_ID</i>.</p> <p>(summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_ID <p>Array size: Depends on the result type. Note: When multiple results.</p> <ul style="list-style-type: none"> • Data type: MIL_ID <p>Note: When a single result.</p>
M_TYPE_MIL_INT	<p>Casts the requested results to a <i>MIL_INT</i>.</p> <p>(summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT <p>Array size: Depends on the result type. Note: When multiple results.</p> <ul style="list-style-type: none"> • Data type: MIL_INT <p>Note: When a single result.</p>
M_TYPE_MIL_INT32	<p>Casts the requested results to a <i>MIL_INT32</i>.</p> <p>(summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 <p>Array size: Depends on the result type. Note: When multiple results.</p>

	<ul style="list-style-type: none"> • Data type: MIL_INT32 Note: When a single result.
<input type="checkbox"/> M_TYPE_MIL_INT64	Casts the requested results to a <i>MIL_INT64</i> . (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT64 Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_INT64 Note: When a single result.
<input type="checkbox"/> M_TYPE_SHORT	Casts the requested results to a <i>short</i> . (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type short Array size: Depends on the result type. Note: When multiple results. • Data type: short Note: When a single result.

ResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type char
- array of type double
- array of type float
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_ID
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- array of type short
- char
- double
- float
- long
- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT32
- MIL_INT64
- short

Specifies the address in which to write the results.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

M3dmapInquire

Synopsis

Inquire about a 3D reconstruction context or result buffer setting.

Syntax

```
MIL_INT M3dmapInquire(
    MIL_ID ContextOrResultId,
    MIL_INT Index,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function is used to inquire about the setting of a 3D reconstruction context or result buffer.

Parameters

ContextOrResultId

Specifies the identifier of the 3D reconstruction context or result buffer about which to inquire information.

Index

Reserved for future expansion. This parameter must be set to **M_DEFAULT**.

InquireType

Specifies the setting to inquire.

For a partially or fully corrected depth map, the **InquireType** parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For a partially or fully corrected depth map	
☐ Value	Description
☐ M_CALIBRATION_STATUS +	<div>Returns the status of the calibration. (summarize)</div> <div>UserVarPtr info</div> <div>Return values:</div> <div>M_CALIBRATEDA successful call to M3dmapCalibrate() has been made and the context can now be used to produce calibrated data.</div> <div>M_LASER_LINE_NOT_DETECTEDThe laser line could not be extracted from the calibration data.</div> <div>M_MATHEMATICAL_EXCEPTIONM3dmapCalibrate() could not compute calibration parameters. Verify if all control type settings and images are consistent.</div> <div>M_NOT_ENOUGH_MEMORYThere was not enough memory for M3dmapCalibrate() to complete its task.</div> <div>M_NOT_INITIALIZEDContext is not calibrated. M3dmapCalibrate() has not been called.</div>
☐ M_MIN_INTENSITY +	<div>Returns the minimum average intensity level for a group a pixels to be considered a laser line. (summarize)</div>

	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value >= 0.0; (details)
<input type="checkbox"/> M_ORIENTATION +	Returns whether laser line detection is performed vertically or horizontally. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_1D_COLUMNS; M_1D_ROWS; (details)
<input type="checkbox"/> M_PEAK_WIDTH +	Returns the approximate width of the laser line, in pixels. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)

For a fully calibrated depth map, the **InquireType** parameter can be set to the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a fully corrected depth map	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SCAN_SPEED +	Returns the speed of the object being scanned in the Y-direction, in world unit per frame. Note that this value can be negative if the object is going in the negative Y-direction. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)

For a partially corrected depth map, the **InquireType** parameter can be set to the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a partially corrected depth map	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CORRECTED_DEPTH +	Returns the gray level which will be used to represent the depth established from the next laser line (M3dmapAddScan()) when calibrating your 3D reconstruction setup. This only applies when calibrating for a partially corrected depth map. (summarize)
	<i>UserVarPtr info</i> Return values: 0 <= Value <= 254; 0 <= Value <= 65534; M_DEFAULT; (details)

For a 3D reconstruction result buffer, the **InquireType** parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a result buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILL_MODE +	Returns the mode used to fill gaps in depth maps obtained using M3dmapExtract() . (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_X_THEN_Y; (details)
<input type="checkbox"/> M_FILL_SHARP_ELEVATION +	Returns the mode used to fill sharp elevation gaps in depth maps obtained using M3dmapExtract() . (summarize)

	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_MAX; M_MIN; (details)
<input type="checkbox"/> M_FILL_SHARP_ELEVATION_DEPTH +	Returns the threshold, in world units, used to differentiate between gradual elevation gaps and sharp elevation gaps. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_INFINITE; Value >= 0.0; (details)
<input type="checkbox"/> M_GRAY_LEVEL_SIZE_Z +	Returns the depth, in world units, of one gray level in the Z direction. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value != 0.0; (details)
<input type="checkbox"/> M_MAX_FRAMES +	Returns the maximum number of scan lines that the laser result buffer keeps internally. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)
<input type="checkbox"/> M_PIXEL_SIZE_X +	Returns the length in world units, in the X-direction, of one pixel. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0.0; (details)
<input type="checkbox"/> M_PIXEL_SIZE_Y +	Returns the length in world units, in the Y-direction, of one pixel. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0.0; (details)
<input type="checkbox"/> M_WORLD_OFFSET_X +	Returns the X-world coordinate of a depth map's top left corner. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_WORLD_OFFSET_Y +	Returns the Y-world coordinate of a depth map's top left corner. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_WORLD_OFFSET_Z +	Returns the Z-world coordinate for gray level 0 of the depth map. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)

Combination constant for any of the possible values of the [InquireType](#) parameter

You can add the following value to the above-mentioned values to get the default value of an inquire type.

● For inquiring the value used by M_DEFAULT	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Returns the default value of the specified inquire type. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE

Combination constant for any of the possible values of the [InquireType](#) parameter

You can add the following value to the above-mentioned values to determine whether an inquire type is supported.

● For inquiring whether an inquire type is supported	
Value	Description
M_SUPPORTED	Returns whether the specified inquire type is supported. (summarize)
	<div> <div>UserVarPtr info</div> <div>Return values:</div> <div> <div>M_FALSE</div> <div>The inquire type is not available.</div> </div> <div> <div>M_TRUE</div> <div>The inquire type is available.</div> </div> </div>

Combination constants for any of the possible values of the [InquireType](#) parameter

You can add one of the following values to the above-mentioned values to cast the requested information to the required data type.

● For specifying the data type	
Value	Description
M_TYPE_CHAR	Casts the requested information to a <i>char</i> . (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: char</div> </div>
M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: double</div> </div>
M_TYPE_FLOAT	Casts the requested information to a <i>float</i> . (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: float</div> </div>
M_TYPE_LONG	Casts the requested information to a <i>long</i> . (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: long</div> </div>
M_TYPE_MIL_DOUBLE	Casts the requested information to a <i>MIL_DOUBLE</i> . (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> </div>
M_TYPE_MIL_ID	Casts the requested information to a <i>MIL_ID</i> . (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_ID</div> </div>
M_TYPE_MIL_INT	Casts the requested information to a <i>MIL_INT</i> . (summarize)

	<div>UserVarPtr info</div> <div>Data type: MIL_INT</div>
<div><div></div>M_TYPE_MIL_INT32</div>	<div>Casts the requested information to a <i>MIL_INT32</i>.</div> <div>(summarize)</div>
	<div>UserVarPtr info</div> <div>Data type: MIL_INT32</div>
<div><div></div>M_TYPE_MIL_INT64</div>	<div>Casts the requested information to a <i>MIL_INT64</i>.</div> <div>(summarize)</div>
	<div>UserVarPtr info</div> <div>Data type: MIL_INT64</div>
<div><div></div>M_TYPE_SHORT</div>	<div>Casts the requested information to a <i>short</i>.</div> <div>(summarize)</div>
	<div>UserVarPtr info</div> <div>Data type: short</div>

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):
<ul style="list-style-type: none">• char• double• float• long• MIL_DOUBLE• MIL_ID• MIL_INT• MIL_INT32• MIL_INT64• short

Specifies the address in which to write the requested information.

Return value

The return value is the required information, cast to a *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

M3dmapTriangulate

Synopsis

Triangulate the 3D world coordinates of points from their pixel location in the image plane of two or more cameras.

Syntax

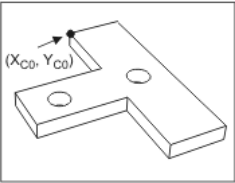
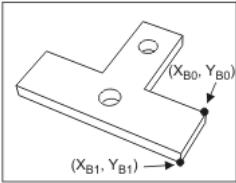
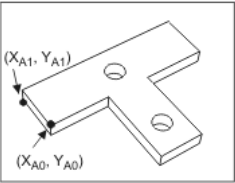
```
void M3dmapTriangulate(
    const MIL_ID *CalibrationIdArrayPtr,
    const MIL_DOUBLE *XPixelArrayPtr,
    const MIL_DOUBLE *YPixelArrayPtr,
    MIL_DOUBLE *XWorldArrayPtr,
    MIL_DOUBLE *YWorldArrayPtr,
    MIL_DOUBLE *ZWorldArrayPtr,
    MIL_DOUBLE *RMSErrorArrayPtr,
    MIL_INT NbCalibrations,
    MIL_INT NbPoints,
    MIL_INT CoordinateSystem,
    MIL_INT ControlFlag
)
```

Description

This function uses triangulation to calculate the X-, Y-, and Z-world coordinates of points (features) that can be seen by two or more cameras. You must identify the pixel location of the points in the image plane of the different cameras. Each of the camera setups must be calibrated in 3D mode so that the position of the cameras in the world is known. To calculate each point's location in the world, the point's location in each camera's image plane and the camera's location in the world are used.

With a single call to **M3dmapTriangulate()**, you can calculate the world coordinates of multiple points, even if a point does not appear in the image plane of all cameras. Each point must be seen by at least two cameras. To indicate that a point is not seen by a camera, use **M_INVALID_POINT**. For example, to determine the world coordinates of the following two points on an object that is seen by three cameras, you would pass the following arguments to **M3dmapTriangulate()**:

Feature matching



Object as seen by camera A

Object as seen by camera B

Object as seen by camera C

Point number	Array index	XPixelArrayPtr	YPixelArrayPtr
0	0	XA0	YA0
	1	XB0	YB0
	2	XC0	YC0
1	3	XA1	YA1
	4	XB1	YB1
	5	M_INVALID_POINT	M_INVALID_POINT

CalibrationIdArrayPtr
CalibrationObjectA
CalibrationObjectB
CalibrationObjectC

The calibration objects of the camera setups must have been allocated in 3D mode using `McalAlloc()` with `M_TSAI_BASED` or `M_3D_ROBOTICS` and they must share a common world coordinate system. For more information about camera calibration, see [Chapter 5: Camera calibration](#).

Although results can be obtained with only two cameras, rounding and extraction errors can affect the accuracy of the calculations. Using more than two cameras will increase the robustness of the function.

Parameters

CalibrationIdArrayPtr

Specifies the address of an array containing the identifiers of the camera calibration objects for each of the camera setups. Alternatively, the array can contain the identifiers of calibrated or corrected images; the operation uses the camera calibration objects associated with the images.

XPixelArrayPtr

Specifies the address of the array containing the X-pixel coordinates of the common points in the image plane of each camera. This array must have a size equal to **(NbCalibrations x NbPoints)**.

Specify the location of a given point in the different image planes, before specifying the location of another point. Specify the location of a given point in the different image planes in the same order as you specify the calibration objects. If more than two cameras are used and a point is seen by only a subset of all the cameras, indicate that the point is not seen by a camera by setting the corresponding entry in the array to `M_INVALID_POINT`.

YPixelArrayPtr

Specifies the address of the array containing the Y-pixel coordinates of the common points in the image plane of each camera. This array must have a size equal to **(NbCalibrations x NbPoints)**.

Specify the location of a given point in the different image planes, before specifying the location of another point. Specify the location of a given point in the different image planes in the same order as you specify the calibration objects. If more than two cameras are used and a point is seen by only a subset of all the cameras, indicate that the point is not seen by a camera by setting the corresponding entry in the array to `M_INVALID_POINT`.

XWorldArrayPtr

Specifies the address of an array in which to write the calculated X-world coordinates. This array must have a size equal to **NbPoints**.

YWorldArrayPtr

Specifies the address of the array in which to write the calculated Y-world coordinates. This array must have a size equal to **NbPoints**.

ZWorldArrayPtr

Specifies the address of the array in which to write the calculated Z-world coordinates. This array must have a size equal to **NbPoints**.

RMSErrorArrayPtr

Specifies the address of the array in which to write the root mean square (RMS) error for the calculated world coordinates of each point. This array must have a size equal to **NbPoints**.

If the RMS error is not needed, set this parameter to `M_NULL`.

NbCalibrations

Specifies the number of calibration objects. This should correspond to the number of cameras. This parameter must be greater or equal to 2.

NbPoints

Specifies the number of points to compute using triangulation. This parameter must be greater or equal to 1.

CoordinateSystem

Specifies the world coordinate system that is common to all the specified camera calibration objects.

• For specifying the common coordinate system

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ABSOLUTE_COORDINATE_SYSTEM	Specifies that the calibration objects have the absolute coordinate system in common.

<input type="checkbox"/> M_RELATIVE_COORDINATE_SYSTEM	Specifies that the calibration objects have the relative coordinate system in common.
<input type="checkbox"/> M_ROBOT_BASE_COORDINATE_SYSTEM	Specifies that the calibration objects have the robot-base coordinate system in common.
<input type="checkbox"/> M_TOOL_COORDINATE_SYSTEM	Specifies that the calibration objects have the tool coordinate system in common.

ControlFlag

Specifies how to handle the coordinates of invalid points.

● For specifying how invalid points are handled	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies to return a MIL error if the coordinates of any point could not be computed. The output arrays will contain undefined values. (summarize)
<input type="checkbox"/> M_ALLOW_INVALID_POINT_OUTPUT	Specifies not to return a MIL error if the coordinates of a point could not be computed. For points that are successfully computed, the calculated coordinates will be saved in their corresponding entries in the output arrays. For points that could not be computed, their corresponding entries in the coordinate output arrays will be set to M_INVALID_POINT ; their corresponding entries in the RMSErrorArrayPtr array will be set to 0.0. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; mil3dmap.lib.
DLL	Requires mil.dll; mil3dmap.dll.

Mapp functions

Synopsis

The functions prefixed with Mapp make up the Application module. The Application module allows you to initialize and control the execution environment for a MIL application. The module provides, among other things, integrated debugging services, a high-performance timer for benchmarking, portable threads and event handling, and automatic system resource compensation for processing and memory operations.

Functions

- [MappAlloc](#)
- [MappAllocDefault](#)
- [MappControl](#)
- [MappFree](#)
- [MappFreeDefault](#)
- [MappGetError](#)
- [MappGetHookInfo](#)
- [MappHookFunction](#)
- [MappInquire](#)
- [MappTimer](#)

MappAlloc

Synopsis

Allocate a MIL application.

Syntax

```
MIL_ID MappAlloc(
    MIL_INT InitFlag,
    MIL_ID *ApplicationIdPtr
)
```

Description

This function allocates a MIL application. A MIL application must be allocated prior to using any other MIL functions.

In multi-thread environments, if an application is allocated in the main thread, it is shared by all threads. When this is the case, **Mapp...()** function calls from any thread apply to all threads, unless specifically localized to that thread by specifying **M_THREAD_CURRENT** when calling the function. However, if a new MIL application is allocated within a secondary thread, this thread will be isolated from the shared application and all application settings and hooks belonging to that thread will be independent from those in other threads. For example, turning off error printing in a secondary thread, using **MappControl()**, will not affect the printing of errors in the original shared application. Likewise, calling **MappControl()** with **M_PRINT_DISABLE** from the main thread, will not affect the application running on a secondary thread. You must free the allocated application (**MappFree()**) in the thread in which it was allocated.

Note, upon the first allocation of a MIL application in a thread, a default system (**M_SYSTEM_HOST**) is automatically allocated. This default Host system can be used in MIL function calls by specifying **M_DEFAULT_HOST** wherever a system identifier is required.

In addition, a default graphics context is also allocated upon allocation of a MIL application. This default graphics context can be used in MIL graphic function calls by specifying **M_DEFAULT** wherever a graphics context identifier is required. In multi-thread applications, a default graphics context is allocated for each thread to avoid inter-thread interference.

Parameters

InitFlag

Specifies the type of initialization to perform on the MIL application. This parameter should be set to one of the following values:

● For specifying the type of initialization	
☐ Value	Description
☐ M_DEFAULT +	Specifies the default initialization. During the allocation of the application, reported error messages are displayed. (summarize)
☐ M_QUIET +	Suppresses the displaying of error messages during the allocation of the application.

Combination constants for the values listed in [For specifying the type of initialization](#)

You can add one of the following values to the above-mentioned values to set which hardware-acceleration display mode to use.

By default, MIL uses the mode set using MILConfig.

● For specifying the hardware-acceleration display mode	
☐ Value	Description
☐ M_DX_VERSION(MIL_INT Version)	<i>[This is only applicable to Windows]</i> Specifies the version of DirectX that MIL can use for display purposes. This determines the hardware-acceleration display mode. For advantages and restrictions of each mode, see the Supported hardware acceleration modes section in Chapter 20: Displaying an image .

ApplicationIdPtr

Specifies the address of the variable in which to write the application identifier. Since the **MappAlloc()** function also returns the application identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier. After allocating an application, we recommend that you check if the operation was successful, using [MappGetError\(\)](#), or by verifying that the application identifier returned is not **M_NULL**.

Return value

The returned value is the application identifier. If allocation fails, **M_NULL** is returned.

Remark

- If you are creating a DLL that includes a call to **MappAlloc()**, ensure that the call is not made from the **DIIMain()** function, because **MappAlloc()** might load a required DLL and you cannot load a DLL from **DIIMain()**. If necessary, call **MappAlloc()** from an initialization function in your DLL instead.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MappAllocDefault

Synopsis

Allocate default MIL objects.

Syntax

```
void MappAllocDefault (
    MIL_INT  InitFlag,
    MIL_ID *ApplicationIdPtr,
    MIL_ID *SystemIdPtr,
    MIL_ID *DisplayIdPtr,
    MIL_ID *DigIdPtr,
    MIL_ID *ImageBufIdPtr
)
```

Description

MappAllocDefault() is implemented as a macro. This macro can allocate and set up the requested MIL objects using the defaults specified during allocation and using MilConfig. This macro can allocate and initialize a MIL application, and allocate the default MIL system. On this system, it can allocate the default digitizer and display, and allocate and clear a default displayable image buffer.

The installation customizes the default setup. After installation, if you want to review or change the default settings of **MappAllocDefault()**, use MilConfig.

Note, if allocating the default digitizer and the default DCF, selected using MilConfig, is for a 3-band color (RGB) camera, then the default image buffer will be 3-band if allocated; otherwise, it will be 1-band if allocated. The size of a default image buffer, unless the user specifies, is 640 x 480, in pixels.

Note, upon execution of this function, a default graphics context is automatically allocated. This default graphics context can be used in MIL function calls by specifying **M_DEFAULT** wherever a graphics context identifier is required.

When you have finished using the allocated MIL objects, you must free them using **MappFreeDefault()**. With multiple threads, you must free the allocated application (using **MappFreeDefault()**) in the thread in which it was allocated.

Parameters

InitFlag

Specifies the type of initialization setup to perform and is used principally to initialize the default system. This parameter can be set to one of the following:

● For specifying the type of initialization setup	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_COMPLETE	Performs a complete initialization of the MIL environment: initializes MIL to its default state and downloads on the default system any required resident software. At least one complete initialization is necessary after you power-up your system. (summarize)
<input type="checkbox"/> M_PARTIAL	Initializes MIL to its default state, but does not download any resident software on the default system. M_PARTIAL should only be selected if the required resident software has already been downloaded. This option is particularly useful when debugging since resident software generally needs to be downloaded once after power-up (or rebooting the system) and the downloading process can take a substantial amount of initialization time on certain systems. (summarize)
<input type="checkbox"/> M_SETUP	Sets InitFlag to one of the above, based on the default state requested during installation or using MilConfig.

ApplicationIdPtr

Specifies the address of the variable in which to write the application identifier. Upon execution of this function, the application is allocated and its identifier returned. Instead of using **MappAllocDefault()**, you can use **MappAlloc()** to allocate an application. Note, an application must be allocated to allocate any other object in MIL.

SystemIdPtr

Specifies the address of the variable in which to write the system identifier. Upon execution of this function, the default system is allocated and its identifier returned. Instead of using **MappAllocDefault()**, you can use [MsysAlloc\(\)](#) to allocate a system. [MappAlloc\(\)](#) will also allocate a default Host system. Note, a system must be allocated to allocate any other objects on it (display, digitizer or data buffers).

DisplayIdPtr

Specifies the address of the variable in which to write the identifier of the default display. If this parameter is set to **M_NULL**, a display is not allocated; otherwise, the default display is allocated and its identifier returned.

DigIdPtr

Specifies the address of the variable in which to write the identifier of the default digitizer. If this parameter is set to **M_NULL**, a digitizer is not allocated; otherwise, the default digitizer is allocated and its identifier returned.

ImageBufIdPtr

Specifies the address of the variable in which to write the identifier of the default image buffer. If this parameter is set to **M_NULL**, an image buffer is not allocated; otherwise, the default image buffer is allocated and its identifier returned. It is then cleared and displayed on the system's display screen.

Examples

For example, a typical setup for Matrox Odyssey board in its power-up state with one input device (RS-170 camera) and one default image buffer (full-screen size) selected on the default display is:

```
MappAllocDefault(M_COMPLETE, &Application, &System, &Display, &Digitizer, &ImageBuffer);
```

If, for example, you don't need to acquire data from the camera but want to perform the rest of the above setup, you would make the following call:

```
MappAllocDefault(M_COMPLETE, &Application, &System, &Display, M_NULL, &ImageBuffer);
```

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MappControl

Synopsis

Control an application environment setting.

Syntax

```
void MappControl (
    MIL_INT ControlType,
    MIL_INT ControlValue
)
```

Description

This function controls the settings of a MIL application environment. These setting include printing error messages, function, and function parameters to the screen, parameter checking, as well as on-board memory and processing compensation modes.

In multi-thread environments, an **MappControl()** call applies to all application threads running MIL, unless specifically limited to the calling thread by adding [M_THREAD_CURRENT](#) to the [ControlType](#) parameter. When you override settings for a specific thread, a subsequent call to change those settings from any other thread will not change the settings. For example, if you enable trace printing for a particular thread, that thread will ignore calls from other threads that try to change trace printing setting. Note that any thread created by the current thread will automatically inherit all the current thread's settings, but it will not inherit this exclusivity.

Parameters

- ControlType
- Specifies the type of application environment setting to control.
- See the [Parameter associations](#) section for possible values.
- ControlValue
- Specifies the setting's new value.
- See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the table below.

- [For specifying the type of application environment](#)

For specifying the type of application environment	
ControlType	Description
ControlValue	
M_ERROR +	Sets whether the printing of error messages to screen is enabled. (summarize)
M_PRINT_DISABLE	Disables printing of error messages. If error printing is disabled, you can still check for error, using MappGetError() . (summarize)
M_PRINT_ENABLE	Enables printing of error messages. This is the default value. (summarize)

<input type="checkbox"/> M_MEMORY +	<p>Sets whether on-board memory compensation on the Host is enabled. On-board memory compensation allows memory to be allocated on Host when a system has insufficient memory to perform that memory allocation.</p> <p>(summarize)</p>
<input type="checkbox"/> M_COMPENSATION_DISABLE	<p>Disables on-board memory compensation. If the memory available is insufficient for the allocation, an error is generated.</p> <p>(summarize)</p>
<input type="checkbox"/> M_COMPENSATION_ENABLE	<p>Enables on-board memory compensation.</p> <p>This is the default value.</p> <p>(summarize)</p>
<input type="checkbox"/> M_MP_MAX_CORES_PER_THREAD +	<p>Sets the absolute maximum number of CPU cores to use for each thread when MP processing is enabled using MilConfig or MappControl() with M_MP_USE.</p> <p>To set a different number of CPU cores for a specific thread, use MthrControl() with M_MP_MAX_CORES.</p> <p>As long as the sum of the maximum number of CPU cores available for each thread does not exceed the number of CPU cores available to your MIL application, MIL tries to ensure that the threads use different cores.</p> <p>Note that this control type overwrites the value stored in MILConfig.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the number of CPU cores available to a process, according to the limits set by your operating system.</p>
<input type="checkbox"/> 2 <= Value <= 65535	<p>Specifies the maximum number of CPU cores to use per thread. The effective number of available CPU cores is limited to the number of CPU cores installed in your computer and any limits imposed by the operation system.</p> <p>(summarize)</p>
<input type="checkbox"/> M_MP_USE +	<p>Sets whether multi-core processing (MP) is enabled. Note that this control type affects all threads.</p> <p>To use multi-core processing with a specific thread, use MthrControl() with M_MP_USE.</p> <p>The default value for this control type is set during MIL installation and using MILConfig.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DISABLE	<p>Disables multi-core processing.</p>
<input type="checkbox"/> M_ENABLE	<p>Enables multi-core processing.</p>
<input type="checkbox"/> M_PARAMETER +	<p>Sets whether the checking of parameters is enabled.</p> <p>(summarize)</p>
<input type="checkbox"/> M_CHECK_DISABLE	<p>Disables checking of parameters. Note, if parameter checking is disabled (for example, to accelerate an application), unpredictable behavior can be expected when passing invalid parameters to a function.</p> <p>(summarize)</p>
<input type="checkbox"/> M_CHECK_ENABLE	<p>Enables checking of parameters.</p> <p>This is the default value.</p> <p>(summarize)</p>
<input type="checkbox"/> M_PROCESSING +	<p>Sets whether the Host should compensate for processes that cannot be performed on-board due to board-specific limitations.</p> <p>(summarize)</p>
<input type="checkbox"/> M_COMPENSATION_DISABLE	<p>Disables on-board processing compensation. If an operation cannot be performed by the system's on-board processor, an error is generated.</p> <p>(summarize)</p>
<input type="checkbox"/> M_COMPENSATION_ENABLE	<p>Enables on-board processing compensation. If an operation cannot be performed by the system's on-board processor, it will be performed by the Host.</p> <p>This is the default value.</p> <p>(summarize)</p>
<input type="checkbox"/> M_TRACE +	<p>Sets whether the printing of function names and parameters is enabled. When trace printing is enabled, a pop-up window containing this information appears at the start and end of each MIL function executed. Note that, if enabled, this ControlValue is synchronous.</p> <p>(summarize)</p>
<input type="checkbox"/> M_PRINT_DISABLE	<p>Disables printing of function names and parameters.</p> <p>This is the default value.</p>

	(summarize)
<input type="checkbox"/> M_PRINT_ENABLE	Enables printing of function names and parameters.

Combination constant for the values listed in [For specifying the type of application environment](#)

You can add the following value to the above-mentioned value to set whether the control should be limited to the current thread.

ⓘ For limiting effects to the current thread	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_THREAD_CURRENT	Limits the effect of the ControlType setting to the current thread.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MappFree

Synopsis

Free a MIL application.

Syntax

```
void MappFree (
    MIL_ID ApplicationId
)
```

Description

This function deallocates a MIL application previously allocated with [MappAlloc\(\)](#).

When freeing a MIL application, you must free it in the thread in which it was allocated. Also, in a given thread, **MappFree()** must be the last MIL function called; no other MIL function can be executed in a thread after a call to this function. Prior to freeing a MIL application, ensure that all systems, buffers, displays, and digitizers allocated in the thread are freed.

Note, if you use [MappAllocDefault\(\)](#) to allocate the default MIL application, you must use [MappFreeDefault\(\)](#) to free the application.

Parameter

ApplicationId

Specifies the application to free.

Remark

- If you are creating a DLL that includes a call to **MappFree()**, ensure that the call is not made from the **DllMain()** function, because **MappFree()** might unload any DLL loaded with [MappAlloc\(\)](#) and you cannot unload a DLL from **DllMain()**. If necessary, call **MappFree()** from a clean-up function in your DLL instead.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MappFreeDefault

Synopsis

Free default MIL objects.

Syntax

```
void MappFreeDefault (
    MIL_ID ApplicationId,
    MIL_ID SystemId,
    MIL_ID DisplayId,
    MIL_ID DigId,
    MIL_ID ImageBufId
)
```

Description

MappFreeDefault() is implemented as a macro. This macro frees the default MIL objects that were allocated with the [MappAllocDefault\(\)](#) macro. You must free the allocated application ([MappAllocDefault\(\)](#)) in the thread in which it was allocated.

Parameters

ApplicationId

Specifies the identifier of the application to free.

SystemId

Specifies the identifier of the system to free.

DisplayId

Specifies the identifier of the display to free. If set to **M_NULL**, no display is freed.

DigId

Specifies the identifier of the digitizer to free. If set to **M_NULL**, no digitizer is freed.

ImageBufId

Specifies the identifier of the image buffer to free. If set to **M_NULL**, no buffer is freed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MappGetError

Synopsis

Get error codes and related information.

Syntax

```
MIL_INT MappGetError (
    MIL_INT ErrorType,
    void *ErrorPtr
)
```

Description

This function obtains current or global application error information, for example, error codes, error subcodes, error messages, error submessages, function codes, and function names. This function allows you to check for errors after each MIL function call or to get the first error that occurred after a series of MIL function calls.

A typical use of this function is to check whether a buffer allocation call was successful ([MbufAllocColor\(\)](#), [MbufAlloc1d\(\)](#), and [MbufAlloc2d\(\)](#)) or to see if a function returns an error.

In multi-thread environments, an **MappGetError()** call returns information either with the first error-returning function of the current thread or, if there no errors in the current thread, it checks for errors in the other MIL running threads. To return only errors in the current thread, see the combination constants, described below.

This function can also be used to obtain information about a detected error when error-reporting to the screen has been disabled using [MappControl\(\)](#).

Parameters

ErrorType

Specifies the type of error information to read. This parameter can be set to one of the values below.

Unless otherwise specified, the following values require that you pass the [ErrorPtr](#) parameter the address of a MIL_INT.

● For specifying the type of error information to read	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CURRENT +	Returns the error code returned by the last function call. The current-error code is reset to M_NULL_ERROR before each MIL function call and is set to a specific error code when an error occurs during the execution of a function. (summarize)
<input type="checkbox"/> M_CURRENT_FCT +	Returns the opcode associated with the last function called, when it returns an error.
<input type="checkbox"/> M_CURRENT_SUB_1 +	Returns the first error subcode returned by the last function call. Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
<input type="checkbox"/> M_CURRENT_SUB_2 +	Returns the second error subcode returned by the last function call. Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
<input type="checkbox"/> M_CURRENT_SUB_3 +	Returns the third error subcode returned by the last function call. Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
<input type="checkbox"/> M_CURRENT_SUB_NB	Returns the number of error subcodes associated with the last function called, when it returns an error.
<input type="checkbox"/> M_GLOBAL +	Returns the error code of the first error that has occurred since the last call to MappGetError() (with ErrorType set to M_GLOBAL). This value is reset to M_NULL_ERROR after the next call to a MIL function (other than MappGetError()).

	(summarize)
<input type="checkbox"/> M_GLOBAL_FCT +	Returns the opcode associated with the first error-generating function since the last call to MappGetError() (with ErrorType set to M_GLOBAL).
<input type="checkbox"/> M_GLOBAL_SUB_1 +	Returns the first error subcode of the first error that has occurred since the call to MappGetError() (with ErrorType set to M_GLOBAL). Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
<input type="checkbox"/> M_GLOBAL_SUB_2 +	Returns the second error subcode of the first error that has occurred since the call to MappGetError() (with ErrorType set to M_GLOBAL). Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
<input type="checkbox"/> M_GLOBAL_SUB_3 +	Returns the third error subcode of the first error that has occurred since the call to MappGetError() (with ErrorType set to M_GLOBAL). Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
<input type="checkbox"/> M_GLOBAL_SUB_NB	Returns the number of error subcodes associated with the first error that occurred since the last call to MappGetError() (with ErrorType set to M_GLOBAL).

Combination constants for **M_CURRENT**; **M_CURRENT_FCT**; **M_CURRENT_SUB_1**; **M_CURRENT_SUB_2**; **M_CURRENT_SUB_3**; **M_GLOBAL**; **M_GLOBAL_FCT**; **M_GLOBAL_SUB_1**; **M_GLOBAL_SUB_2**; **M_GLOBAL_SUB_3**;

You can add one or more of the following values to the above-mentioned values to set whether to return a message, make the inquire synchronous, or limit it to the current thread.

● For error types	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MESSAGE	Writes the error message associated with the specified error type, or the function name associated with a function code, to the address specified by the ErrorPtr parameter. The error/function code is still returned by MappGetError() , but it is not written to the address specified by ErrorPtr . Note that this value cannot be added to M_CURRENT_SUB_NB or M_GLOBAL_SUB_NB . (summarize) <i>ErrorPtr info</i> Data type: array of type MIL_TEXT_CHAR Array size: When reporting an error message or function name, the array must be at least M_ERROR_MESSAGE_SIZE (128) characters in size.
<input type="checkbox"/> M_SYNCHRONOUS	Forces the function to wait for all pending calls to complete before continuing.
<input type="checkbox"/> M_THREAD_CURRENT	Returns only errors in the current thread.

ErrorPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type **MIL_TEXT_CHAR**
- **MIL_INT**

Specifies the address in which to write the requested information.

The **MappGetError()** function also returns the error/function code; if you don't want to get an error message or function name, you can set this parameter to **M_NULL**.

Return value

The returned value is the requested error/function code. If the error code is read and it is equal to **M_NULL_ERROR**, no error has occurred.

Compilation information

Header	Include mil.h .
Library	Use mil.lib .
DLL	Requires mil.dll .

MappGetHookInfo

Synopsis

Get information about a hooked event.

Syntax

```
MIL_INT MappGetHookInfo (
    MIL_ID eventId,
    MIL_INT infoType,
    void *userVarPtr
)
```

Description

This function retrieves information about the event that caused the hook-handler function to be called. This function should only be called within the scope of an application hook-handler function call (see [MappHookFunction\(\)](#)).

Note that functions hooked to an event execute on a distinct thread. This permits the functions to run asynchronously from the operation that fired the event and from functions hooked to other events. Although there is a small queue to permit a certain amount of overlap, hooked functions should not take longer to execute than the period in which two of their associated events can occur. You cannot determine the instance of the event that fired the function, and even if this were possible, this information would generally not be very useful. Typically, a hooked function performs the minimum number of operations required and, if necessary, performs longer processes by launching other threads.

Parameters

eventId

Specifies the application event identifier received by the hook-handler function. See [MappHookFunction\(\)](#) for more information.

infoType

Specifies the type of information to get.

If the hook-handler function was called due to an [M_ERROR_CURRENT](#) event type, the [infoType](#) parameter can be set to one of the values below.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_INT](#).

● For specifying the information type (with an M_ERROR_CURRENT event type)	
☐ Value	Description
☐ M_CURRENT +	Returns the error code returned by the last function call. The current-error code is reset to M_NULL_ERROR before each MIL function call and is set to a specific error code if an error occurs while trying to execute the function. (summarize)
☐ M_CURRENT_SUB_1 +	Returns the first error subcode returned by the last function call. Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
☐ M_CURRENT_SUB_2 +	Returns the second error subcode returned by the last function call. Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
☐ M_CURRENT_SUB_3 +	Returns the third error subcode returned by the last function call. Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
☐ M_CURRENT_SUB_NB	Returns the number of error subcodes associated with the last function called, when it returns an error.

If the hook-handler function was called due to an [M_ERROR_GLOBAL](#) event type, the [InfoType](#) parameter can be set to one of the values below.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_INT](#).

● For specifying the information type (with an M_ERROR_GLOBAL event type)	
☐ Value	Description
☐ M_GLOBAL +	Returns the error code of the first error that has occurred since the last call to MappGetError() (with ErrorType set to M_GLOBAL). This value is reset to M_NULL_ERROR after each MappGetError() call with this setting. (summarize)
☐ M_GLOBAL_FCT +	Returns the opcode associated with the first error-generating function since the last call to MappGetError() (with ErrorType set to M_GLOBAL).
☐ M_GLOBAL_SUB_1 +	Returns the first error subcode of the first error that has occurred since the call to MappGetError() (with ErrorType set to M_GLOBAL). Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
☐ M_GLOBAL_SUB_2 +	Returns the second error subcode of the first error that has occurred since the call to MappGetError() (with ErrorType set to M_GLOBAL). Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
☐ M_GLOBAL_SUB_3 +	Returns the third error subcode of the first error that has occurred since the call to MappGetError() (with ErrorType set to M_GLOBAL). Note, when there is no error, the error subcode is set to M_NULL_ERROR . (summarize)
☐ M_GLOBAL_SUB_NB	Returns the number of error subcodes associated with the first error that occurred since the last call to MappGetError() (with ErrorType set to M_GLOBAL).

If the hook-handler function was called due to an [M_ERROR_CURRENT](#), [M_TRACE_START](#), or [M_TRACE_END](#) event type, the [InfoType](#) parameter can be set to the value below.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_INT](#).

● For specifying the information type (with an M_ERROR_CURRENT , M_TRACE_START , or M_TRACE_END event type)	
☐ Value	Description
☐ M_CURRENT_FCT +	Returns the opcode associated to the last function called when it returns an error (for M_ERROR_CURRENT), just started (for M_TRACE_START), or just ended (for M_TRACE_END).

Combination constant for the values listed in [For specifying the information type \(with an \[M_ERROR_CURRENT\]\(#\), \[M_TRACE_START\]\(#\), or \[M_TRACE_END\]\(#\) event type\)](#); and for the following values: [M_CURRENT](#); [M_CURRENT_SUB_1](#); [M_CURRENT_SUB_2](#); [M_CURRENT_SUB_3](#); [M_GLOBAL](#); [M_GLOBAL_FCT](#); [M_GLOBAL_SUB_1](#); [M_GLOBAL_SUB_2](#); [M_GLOBAL_SUB_3](#);

You can add the following value to the above-mentioned values to get the associated message instead of the numeric code.

● For all values except M_CURRENT_SUB_NB , M_GLOBAL_SUB_NB , M_PARAM_NB , M_PARAM_TYPE , and M_PARAM_VALUE	
☐ Value	Description
☐ M_MESSAGE	Returns the error message instead of the error code, or the function name instead of the function code. (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: array of type MIL_TEXT_CHAR Array size: The array must be at least M_ERROR_MESSAGE_SIZE (128) characters in size.</div> </div>

If the hook-handler function was called due to an [M_TRACE_START](#) or [M_TRACE_END](#) event type, the [InfoType](#) parameter can be set to one of the values below.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_INT](#).

|--|--|

● For specifying the information type (with an M_TRACE_START or M_TRACE_END event type)																													
☐ Value	Description																												
☐ M_PARAM_NB	Returns the number of parameters associated with the MIL function called.																												
☐ M_PARAM_TYPE +	<p>Returns the data type of the specified parameter of the MIL function called. You must specify a combination value from the table below. (summarize)</p> <p><i>UserVarPtr Info</i> Return values:</p> <table> <tr> <td>M_TYPE_ARRAY_ID_PTR</td><td>The specified parameter is an array of type <i>MIL_ID</i>. Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.</td></tr> <tr> <td>M_TYPE_CHAR</td><td>The specified parameter is of type char.</td></tr> <tr> <td>M_TYPE_DOUBLE</td><td>The specified parameter is of type <i>double</i>.</td></tr> <tr> <td>M_TYPE_FILENAME</td><td>The specified parameter is of type filename. Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.</td></tr> <tr> <td>M_TYPE_INT64</td><td>The specified parameter is of type integer.</td></tr> <tr> <td>M_TYPE_LONG</td><td>The specified parameter is of type <i>long</i>.</td></tr> <tr> <td>M_TYPE_MIL_DOUBLE</td><td>The specified parameter is of type <i>MIL_DOUBLE</i>.</td></tr> <tr> <td>M_TYPE_MIL_ID</td><td>The specified parameter is of type <i>MIL_ID</i>.</td></tr> <tr> <td>M_TYPE_MIL_INT</td><td>The specified parameter is of type <i>MIL_INT</i>.</td></tr> <tr> <td>M_TYPE_MIL_INT32</td><td>The specified parameter is of type <i>MIL_INT32</i>.</td></tr> <tr> <td>M_TYPE_MIL_INT64</td><td>The specified parameter is of type <i>MIL_INT64</i>.</td></tr> <tr> <td>M_TYPE_PTR</td><td>The specified parameter is a pointer. Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.</td></tr> <tr> <td>M_TYPE_SHORT</td><td>The specified parameter is of type short.</td></tr> <tr> <td>M_TYPE_STRING</td><td>The specified parameter is an array of type <i>MIL_TEXT_CHAR</i>. Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.</td></tr> </table>	M_TYPE_ARRAY_ID_PTR	The specified parameter is an array of type <i>MIL_ID</i> . Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.	M_TYPE_CHAR	The specified parameter is of type char.	M_TYPE_DOUBLE	The specified parameter is of type <i>double</i> .	M_TYPE_FILENAME	The specified parameter is of type filename. Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.	M_TYPE_INT64	The specified parameter is of type integer.	M_TYPE_LONG	The specified parameter is of type <i>long</i> .	M_TYPE_MIL_DOUBLE	The specified parameter is of type <i>MIL_DOUBLE</i> .	M_TYPE_MIL_ID	The specified parameter is of type <i>MIL_ID</i> .	M_TYPE_MIL_INT	The specified parameter is of type <i>MIL_INT</i> .	M_TYPE_MIL_INT32	The specified parameter is of type <i>MIL_INT32</i> .	M_TYPE_MIL_INT64	The specified parameter is of type <i>MIL_INT64</i> .	M_TYPE_PTR	The specified parameter is a pointer. Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.	M_TYPE_SHORT	The specified parameter is of type short.	M_TYPE_STRING	The specified parameter is an array of type <i>MIL_TEXT_CHAR</i> . Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.
M_TYPE_ARRAY_ID_PTR	The specified parameter is an array of type <i>MIL_ID</i> . Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.																												
M_TYPE_CHAR	The specified parameter is of type char.																												
M_TYPE_DOUBLE	The specified parameter is of type <i>double</i> .																												
M_TYPE_FILENAME	The specified parameter is of type filename. Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.																												
M_TYPE_INT64	The specified parameter is of type integer.																												
M_TYPE_LONG	The specified parameter is of type <i>long</i> .																												
M_TYPE_MIL_DOUBLE	The specified parameter is of type <i>MIL_DOUBLE</i> .																												
M_TYPE_MIL_ID	The specified parameter is of type <i>MIL_ID</i> .																												
M_TYPE_MIL_INT	The specified parameter is of type <i>MIL_INT</i> .																												
M_TYPE_MIL_INT32	The specified parameter is of type <i>MIL_INT32</i> .																												
M_TYPE_MIL_INT64	The specified parameter is of type <i>MIL_INT64</i> .																												
M_TYPE_PTR	The specified parameter is a pointer. Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.																												
M_TYPE_SHORT	The specified parameter is of type short.																												
M_TYPE_STRING	The specified parameter is an array of type <i>MIL_TEXT_CHAR</i> . Note that this data type is invalid after exiting the hook-handler function. For future use, copy and save it.																												
☐ M_PARAM_VALUE +	<p>Returns the value of the specified parameter of the MIL function called. You must specify a combination value from the table below. (summarize)</p> <p><i>UserVarPtr Info</i></p> <ul style="list-style-type: none"> • Data type: long • Data type: MIL_INT • Data type: MIL_INT32 • Data type: MIL_INT64 • Data type: short • Data type: char • Data type: double • Data type: MIL_DOUBLE • Data type: MIL_ID • Data type: MIL_INT64 • Data type: MIL_TEXT_PTR • Data type: void pointer 																												

Combination constant for [M_PARAM_TYPE](#); [M_PARAM_VALUE](#);

You must add the following value to the above-mentioned values to set the number of the parameter for which to receive the information.

● For M_PARAM_TYPE and M_PARAM_VALUE	
☐ Value	Description
☐ Value >= 1	Specifies the number of the parameter for which to retrieve the information. This value must be a positive integer, and should not exceed the number of parameters associated with the MIL function call (M_PARAM_NB).

[\(summarize\)](#)

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type MIL_TEXT_CHAR
- char
- double
- long
- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT32
- MIL_INT64
- MIL_TEXT_PTR
- short
- a pointer of type void

Specifies the address in which to write the requested information. Note that, when getting parameter values (**MappGetHookInfo()** with [M_PARAM_VALUE](#)), the [UserVarPtr](#) parameter should be of the same data type as value of the selected parameter.

Return value

The returned value is **M_NULL** if successful; a non **M_NULL** value on error, without logging any errors in the application.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MappHookFunction

Synopsis

Hook a function to an event.

Syntax

```
void MappHookFunction(
    MIL_INT HookType,
    MIL_APP_HOOK_FUNCTION_PTR HookHandlerPtr,
    void *UserDataPtr
)
```

Description

This function allows you to attach or detach a user-defined function to a specified application event. Once a hook-handler function is defined and hooked to an event, it is automatically called when the event occurs.

Note that functions hooked to an event execute on a distinct thread. This permits the functions to run asynchronously from the operation that fired the event and from functions hooked to other events. Although there is a small queue to permit a certain amount of overlap, hooked functions should not take longer to execute than the period in which two of their associated events can occur. You cannot determine the instance of the event that fired the function, and even if this were possible, this information would generally not be very useful. Typically, a hooked function performs the minimum number of operations required and, if necessary, performs longer processes by launching other threads.

You can hook more than one function to an event by making separate calls to **MappHookFunction()** for each function that you want to hook. MIL automatically chains and keeps an internal list of all these hooked functions. When a function is hooked, this new function is added to the end of the list. When the event happens, all user-defined functions in the list will be executed in the same order that they were hooked to the event. You can also remove any function from the list; in this case, MIL preserves the order of the remaining functions in the list.

You can obtain information concerning the event, using **MappGetHookInfo()**, and take appropriate action before returning control to the application.

In multi-thread environments, an **MappHookFunction()** call hooks or unhooks the function specified by **HookHandlerPtr** to all application threads running MIL, unless specifically limited to the calling thread.

This function is typically used to trap errors that occur in an application, without checking every MIL function execution with **MappGetError()**, or to detect the start or end of certain MIL functions.

Parameters

HookType

Specifies the event type. This parameter can be set to one of the following values.

● For specifying the event type	
☐ Value	Description
☐ M_ERROR_CURRENT +	Calls the hook-handler function each time an error occurs. Note that you could end up with infinite recursion if the MIL function generating an error is called within the hook function. (summarize)
☐ M_ERROR_FATAL +	Calls the hook-handler function before a fatal-error occurs.
☐ M_ERROR_GLOBAL +	Calls the hook-handler function when the first error occurs in a series of MIL calls. Note that you could end up with infinite recursion if the MIL function generating an error is called within the hook function. (summarize)
☐ M_TRACE_END +	Calls the hook-handler function at the end of each MIL function. When setting the HookType parameter to M_TRACE_END , the only MIL function that can be called in a hook function is MappGetHookInfo() , otherwise infinite recursion will occur. (summarize)
☐ M_TRACE_START +	Calls the hook-handler function at the start of each MIL function.

When setting the **HookType** parameter to [M_TRACE_START](#), the only MIL function that can be called in a hook function is [MappGetHookInfo\(\)](#), otherwise infinite recursion will occur.
([summarize](#))

Combination constant for any of the possible values of the [HookType](#) parameter
You can add the following value to the above-mentioned values to set whether to detach the hook-handler function.

● For the HookType parameter	
▢ Value	Description
▢ M_UNHOOK	Detaches (that is, unhooks) the hook-handler function.

Combination constant for any of the possible values of the [HookType](#) parameter
You can add the following value to the above-mentioned values to set whether to limit the hooking or unhooking of the function to the calling thread.

● For hooking or unhooking the function to application threads	
▢ Value	Description
▢ M_THREAD_CURRENT	Limits the hooking or unhooking of the function to the calling thread. For example, to call the hook-handler function only for errors occurring in the current thread, specify M_ERROR_CURRENT + M_THREAD_CURRENT as the HookType parameter. (summarize)

HookHandlerPtr

Specifies the address of the function that should be called when an event occurs.

The hook-handler function, pointed to by **HookHandlerPtr**, must be declared as follows:

```
MIL_INT MFTYPE HookHandler(  
    MIL_INT HookType,  
    MIL_ID EventId,  
    void MPTYPE *UserDataPtr  
)  
Parameters:  
  
    HookType  
        Type of event hooked.  
  
    EventId  
        Event identifier to pass to MappGetHookInfo\(\) when inquiring about the hooked event.  
  
    UserDataPtr  
        Specifies the user data pointer passed to MappHookFunction\(\).
```

Upon successful completion, the hook-handler function should return **M_NULL**. Note, MFTYPE and MIL_APP_HOOK_FUNCTION_PTR are reserved MIL predefined types for functions and data pointers, respectively.

UserDataPtr

Specifies the address of the user data that you want to make available to the hook-handler function. This address is passed to the hook-handler function, through its **UserDataPtr** parameter, when the specified event occurs. Set this parameter to **M_NULL** if user data is not required.

Return value

The original prototype of this function has been kept for backwards compatibility. However, because of the current chaining method, the function always returns null.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MappInquire

Synopsis

Inquire about an application setting.

Syntax

```
MIL_INT MappInquire(
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires about the specified application setting.

Parameters

InquireType

Specifies the type of application setting about which to inquire. This parameter can be set to one of the values below.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_INT`.

For specifying the type of application setting	
Value	Description
<input checked="" type="checkbox"/> M_CURRENT_APPLICATION	<p>Returns the identifier of the current MIL application, if any. (summarize)</p> <p><i>UserVarPtr info</i> Data type: MIL_ID Return values: MIL application identifier Identifier of current MIL application. 0 No application is allocated. Note that no error is generated.</p>
<input checked="" type="checkbox"/> M_DIRECTX_VERSION	<p>[<i>This is only applicable to Windows</i>] Returns the version of DirectX that MIL can use for display purposes under Windows. This determines the hardware acceleration display mode. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0, 7 or 9. Please see MappAlloc() with M_DX_VERSION. (details)</p>
<input checked="" type="checkbox"/> M_ERROR +	<p>Returns the error printing mode. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_PRINT_DISABLE; M_PRINT_ENABLE; (details)</p>
<input checked="" type="checkbox"/> M_INSTALLED_SYSTEM_COUNT	<p>Returns the number of different system types installed (during MIL installation).</p>
<input checked="" type="checkbox"/> M_INSTALLED_SYSTEM_DESCRIPTOR_SIZE	<p>Returns the size of the descriptor string of a system.</p>
<input checked="" type="checkbox"/> M_INSTALLED_SYSTEM_DESCRIPTOR + n	<p>Returns the descriptor of system type n, where n is a number between 0 and the value returned by M_INSTALLED_SYSTEM_COUNT. The descriptor can be passed to MsysAlloc() to allocate the system. (summarize)</p> <p><i>UserVarPtr info</i></p>

	<p>Data type: array of type MIL_TEXT_CHAR</p> <p>Array size: The size of the array must be large enough to store the entire system descriptor.</p> <p>Return values: M_SYSTEM_1394; M_SYSTEM_CORONA_II; M_SYSTEM_CRONOSPLUS; M_SYSTEM_DEFAULT; M_SYSTEM_GIGE_VISION; M_SYSTEM_GPU; M_SYSTEM_HELIOS; M_SYSTEM_HOST; M_SYSTEM_IRIS; M_SYSTEM_METEOR_II; M_SYSTEM_METEOR_II_CL; M_SYSTEM_METEOR_II_DIG; M_SYSTEM_MORPHIS; M_SYSTEM_MORPHISQXT; M_SYSTEM_NEXIS; M_SYSTEM_ODYSSEY; M_SYSTEM_SOLIOS; M_SYSTEM_SOLIOS_GIGE; M_SYSTEM_VIO; (details)</p> <p>"dmlltcp://RemoteComputerId:Port/MILSystemType" A remote system where <i>RemoteComputerId</i> is the remote computer's name or IP address, <i>Port</i> is the port number, and <i>MILSystemType</i> is a valid MIL system type.</p>
<input type="checkbox"/> M_INSTALLED_SYSTEM_PRINT_NAME_SIZE	Returns the size of the printable name of a system.
<input type="checkbox"/> M_INSTALLED_SYSTEM_PRINT_NAME + n	<p>Returns a string with the printable name of system type n (for example, Matrox Helios), where n is a number between 0 and the value returned by M_INSTALLED_SYSTEM_COUNT.</p> <p>(summarize)</p> <p><i>UserVarPtr Info</i></p> <p>Data type: array of type MIL_TEXT_CHAR</p> <p>Array size: The size of the array must be large enough to store the entire printable name of the system.</p>
<input type="checkbox"/> M_MEMORY +	<p>Returns the memory compensation mode.</p> <p>(summarize)</p> <p><i>UserVarPtr Info</i></p> <p>Return values: M_COMPENSATION_DISABLE; M_COMPENSATION_ENABLE; (details)</p>
<input type="checkbox"/> M_MP_CORES_NUM	<p>Returns the maximum number of CPU cores available to the process, according to the limits set by your operating system. Note that if the operating system restricts the number of CPU cores, the number of CPU cores returned differs from the number of CPU cores installed in your computer.</p> <p>(summarize)</p> <p><i>UserVarPtr Info</i></p> <p>Return values: 2 <= Value <= 65535; (details)</p>
<input type="checkbox"/> M_MP_MAX_CORES_PER_THREAD	<p>Returns the absolute maximum number of cores to use for each thread.</p> <p>(summarize)</p> <p><i>UserVarPtr Info</i></p> <p>Return values: 2 <= Value <= 65535; (details)</p>
<input type="checkbox"/> M_MP_USE	<p>Returns whether multi-core processing (MP) is enabled for all threads used by your application.</p> <p>(summarize)</p> <p><i>UserVarPtr Info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_NON_PAGED_MEMORY_FREE	Returns the amount of free non-paged memory, in bytes.
<input type="checkbox"/> M_NON_PAGED_MEMORY_SIZE	Returns the amount of the non-paged memory (DMA) reserved for MIL, in bytes.
<input type="checkbox"/> M_NON_PAGED_MEMORY_USED	Returns the amount of non-paged memory being used, in bytes.
<input type="checkbox"/> M_PARAMETER +	<p>Returns the parameter checking mode.</p> <p>(summarize)</p> <p><i>UserVarPtr Info</i></p> <p>Return values: M_CHECK_DISABLE; M_CHECK_ENABLE; (details)</p>
<input type="checkbox"/> M_PROCESSING +	<p>Returns the processing compensation mode.</p> <p>(summarize)</p> <p><i>UserVarPtr Info</i></p> <p>Return values: M_COMPENSATION_DISABLE; M_COMPENSATION_ENABLE; (details)</p>
<input type="checkbox"/> M_TRACE +	<p>Returns the trace printing mode.</p> <p>(summarize)</p> <p><i>UserVarPtr Info</i></p> <p>Return values: M_PRINT_DISABLE; M_PRINT_ENABLE; (details)</p>
<input type="checkbox"/> M_VERSION	<p>Returns the version of the MIL library.</p> <p>(summarize)</p>

MappTimer

Synopsis

Control a setting of a MIL timer and read the MIL timer.

Syntax

```
MIL_DOUBLE MappTimer (
    MIL_INT  ControlType,
    MIL_DOUBLE *TimePtr
)
```

Description

This function sets the specified control for a MIL timer or reads the MIL timer. There is a different MIL timer for each thread of the application and one global MIL timer for the application. By default, the MIL timer that is set and read is the timer of the current thread. However, you can set and read the global timer by adding [M_GLOBAL](#) to the [ControlType](#) parameter.

This function is useful for benchmarking operations in a MIL application. The timer resolution varies according to the hardware and operating system used.

Note that MIL functions run at a slower speed during debug mode. Therefore, once you are done debugging your application, you should compile in release mode to achieve maximum processing speed.

Parameters

ControlType

Specifies the control to exert on the MIL timer. It can be set to one of the following:

● For MIL timers	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TIMER_READ +	Reads the time, in sec, of the MIL timer since the last reset. Note that the time read takes into account (subtracts) the μ sec (microseconds) needed to execute M_TIMER_READ . (summarize)
<input type="checkbox"/> M_TIMER_RESET +	Resets the MIL timer to zero.
<input type="checkbox"/> M_TIMER_RESOLUTION +	Reads the MIL timer resolution, in seconds.
<input type="checkbox"/> M_TIMER_WAIT +	Waits for the specified period of time, in sec, before returning.

Combination constant for the values listed in [For MIL timers](#)

You can add the following value to the above-mentioned values to set whether the function will be called synchronously.

● For all ControlType parameters	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SYNCHRONOUS	Forces the function to wait for all pending calls of the current thread to complete before continuing.

Combination constant for the values listed in [For MIL timers](#)

You can add the following value to the above-mentioned values to set whether to control and access the global MIL timer.

Note that, in a MIL application, there is only one global MIL timer that is accessible by all MIL threads.

● For controlling and accessing the global MIL timer	
▢ Value	Description
▢ M_GLOBAL	Executes the requested timer operation using the global MIL timer. If used with M_SYNCHRONOUS , the global MIL timer will wait for all pending calls of the current thread to complete before continuing. (summarize)

TimePtr

Specifies the address of the variable in which to store the timer information produced by the [M_TIMER_READ](#) or [M_TIMER_RESOLUTION](#) controls. Since the **MappTimer()** also returns the requested information when [M_TIMER_READ](#) or [M_TIMER_RESOLUTION](#) are selected, you can set this parameter to **M_NULL**.

For [M_TIMER_RESET](#), set [TimePtr](#) to **M_NULL**.

For [M_TIMER_WAIT](#), [TimePtr](#) specifies the address of the variable from which to read the timer information.

Return value

For [M_TIMER_READ](#) and [M_TIMER_RESOLUTION](#), the returned value is cast to *MIL_DOUBLE*. For [M_TIMER_RESET](#) and [M_TIMER_WAIT](#), this function returns 0.0.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

Mblob functions

Synopsis

The functions prefixed with Mblob make up the Blob Analysis module. The Blob Analysis module allows you to identify connected regions of pixels (blobs) within an image, and then calculate selected features of these blobs. The Blob Analysis module can measure a wide assortment of blob features, such as the blob area, perimeter, Feret diameter at a given angle, minimum bounding box, and center of gravity. The module also allows you to perform some image processing operations, such as reconstructing or eliminating blobs.

Functions

- [MblobAllocFeatureList](#)
- [MblobAllocResult](#)
- [MblobCalculate](#)
- [MblobControl](#)
- [MblobDraw](#)
- [MblobFill](#)
- [MblobFree](#)
- [MblobGetLabel](#)
- [MblobGetNumber](#)
- [MblobGetResult](#)
- [MblobGetResultSingle](#)
- [MblobGetRuns](#)
- [MblobInquire](#)
- [MblobLabel](#)
- [MblobMerge](#)
- [MblobReconstruct](#)
- [MblobSelect](#)
- [MblobSelectFeature](#)
- [MblobSelectFeret](#)
- [MblobSelectMoment](#)

MblobAllocFeatureList

Synopsis

Allocate a blob analysis feature list.

Syntax

```
MIL_ID MblobAllocFeatureList (
    MIL_ID SystemId,
    MIL_ID *FeatureListIdPtr
)
```

Description

This function allocates a feature list. The feature list holds the feature(s) to be calculated by [MblobCalculate\(\)](#). You must specify which feature(s) to calculate, using [MblobSelectFeature\(\)](#), [MblobSelectFeret\(\)](#), and [MblobSelectMoment\(\)](#). Immediately after allocation, no features are selected in the feature list. When the feature list is no longer required, release it, using [MblobFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the feature list.

This parameter should be set to one of the following values:

For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

FeatureListIdPtr

Specifies the address of the variable in which the feature list identifier will be written. Since the **MblobAllocFeatureList()** function also returns the feature list identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the feature list identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobAllocResult

Synopsis

Allocate a blob analysis result buffer.

Syntax

```
MIL_ID MblobAllocResult (
    MIL_ID SystemId,
    MIL_ID *BlobResIdPtr
)
```

Description

This function allocates a result buffer used to store blob analysis results.

Each blob creates a separate result entry in the blob analysis result buffer. You can retrieve blob analysis results from a result buffer, using [MblobGetResult\(\)](#) or [MblobGetResultSingle\(\)](#). Use the latter to obtain results for a single blob. For more specific results, you can call [MblobGetLabel\(\)](#) and [MblobGetRuns\(\)](#). When the result buffer is no longer required, release it, using [MblobFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the result buffer.

This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

BlobResIdPtr

Specifies the address of the variable in which the blob analysis result buffer identifier is to be written. Since the **MblobAllocResult()** function also returns the blob analysis result buffer identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobCalculate

Synopsis

Perform blob analysis calculations.

Syntax

```
void MblobCalculate (
    MIL_ID BlobIdentImageId,
    MIL_ID GrayImageId,
    MIL_ID FeatureListId,
    MIL_ID BlobResId
)
```

Description

This function calculates the features specified in the given feature list for all currently included blobs in the blob identifier image and stores results in the specified result buffer. Features are added to the feature list with [MblobSelectFeature\(\)](#), [MblobSelectFeret\(\)](#), and [MblobSelectMoment\(\)](#). Specific blobs can be selected using [MblobSelect\(\)](#).

Calculations on binary features (such as [M_AREA](#)) are performed using only the [BlobIdentImageId](#) parameter. If a grayscale feature (such as [M_MAX_PIXEL](#)) is to be calculated, an image buffer must be specified for the [GrayImageId](#) parameter. In this case, the blob identifier image will be used to identify the blobs and the pixel values in the grayscale image are used to calculate the features. Note that the blob identifier image and the grayscale image must be the same size.

If several calls are made to [MblobCalculate\(\)](#) with the same image and result buffer, features calculated in one call remain in the result buffer and are not recalculated in subsequent calls. However, if you then use a result buffer with different images or if you change its processing mode with [MblobControl\(\)](#), any results already in the buffer become invalid and will be discarded. Therefore, it is more efficient to use a result buffer exclusively in one processing mode with one blob identifier image (or one identifier/grayscale image pair if grayscale features are needed).

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image. When calculated in pixel units, the pixel aspect ratio, specified with [MblobControl\(\)](#), is taken into account horizontally. Results are returned in units of "pixel height" since the pixel width is adjusted to be equal to the pixel height.

If you are calculating chain features (selected using [MblobSelectFeature\(\)](#) or [MblobSelect\(\)](#) with [M_NUMBER_OF_CHAINED_PIXELS](#), [M_CHAINS](#), [M_CHAIN_X](#), [M_CHAIN_Y](#), [M_CHAIN_INDEX](#) or [M_ALL_FEATURES](#)), you cannot use [MblobControl\(\)](#) with [M_BLOB_IDENTIFICATION](#) set to [M_LABELED_TOUCHING](#).

Parameters

BlobIdentImageId

Specifies the blob identifier image that will be used in calculations. The blob identifier image identifies each blob as a group of touching pixels in the current foreground state (zero or non-zero, depending on the value assigned to [M_FOREGROUND_VALUE](#) processing mode in [MblobControl\(\)](#)). The current [M_LATTICE](#) processing mode (also set with [MblobControl\(\)](#)), determines when to consider pixels as touching. The blob identifier image must be an unsigned, single band, packed binary, 8-bit or 16-bit grayscale image buffer.

If the identifier image has previously been binarized so that it contains only two extreme values (0 and 1 for 1-bit images, 0 and 0xff for 8-bit images, and 0 and 0xffff for 16-bit images), blob analysis can be performed a little faster. However, you must first change the [M_IDENTIFIER_TYPE](#) to [M_BINARY](#), using [MblobControl\(\)](#), to let MIL know that the identifier image has only two states.

Depending on the [M_BLOB_IDENTIFICATION](#) mode (set with [MblobControl\(\)](#)), [MblobCalculate\(\)](#) either treats each blob individually ([M_INDIVIDUAL](#)), groups all blobs together ([M_WHOLE_IMAGE](#)), groups blobs according to their actual pixel value in the blob identifier image ([M_LABELED](#)), or treats touching blobs that have different labels individually ([M_LABELED_TOUCHING](#)).

GrayImageId

Specifies the grayscale image (not a binary buffer) that will be used to calculate grayscale features. If this parameter is set to [M_NULL](#), grayscale features, such as [M_SUM_PIXEL](#), cannot be calculated. This parameter is ignored when calculating only binary type features. The grayscale image must be a single band, 8-bit or 16-bit unsigned grayscale image buffer.

FeatureListId

Specifies the identifier of the feature list buffer, previously allocated with [MblobAllocFeatureList\(\)](#), that specifies the feature(s) to calculate.

BlobResId

Specifies the identifier of a blob analysis result buffer, previously allocated with [MblobAllocResult\(\)](#), in which to store calculated results.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The blob analysis result buffer ([BlobResId](#)) must be allocated on the same system as the feature list buffer ([FeatureListId](#)). If it is not, an error will occur.

Remark

- Note that this function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobControl

Synopsis

Control a blob analysis processing control setting.

Syntax

```
void MblobControl (
    MIL_ID BlobResId,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function changes a processing control associated with the specified blob analysis result buffer. This function is normally called immediately after allocating a blob analysis result buffer, using [MblobAllocResult\(\)](#), but can be called later (in which case, already calculated results are discarded). If not called, default processing controls and values are used in calculations.

Note that you can use [MblobInquire\(\)](#) to inquire about specific processing controls associated with a blob analysis result buffer.

Parameters

- BlobResId

Specifies the identifier of the blob analysis result buffer.
- ControlType

Specifies the processing setting to control.

See the [Parameter associations](#) section for possible values.
- ControlValue

Specifies the setting required for the control for the processing setting.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the table below.

- For calculating the features of individual or groups of blobs

For calculating the features of individual or groups of blobs	
ControlType	Description
ControlValue	
M_BLOB_IDENTIFICATION	Sets whether to calculate features for each blob, or to treat groups of blobs as single blobs and calculate the features of the blobs. (summarize)
<input checked="" type="checkbox"/> M_INDIVIDUAL	Specifies that all blobs are measured individually.
	This is the default value. (summarize)

<input type="checkbox"/> M_LABELED	<p>Specifies that blobs with the same label are grouped together, and that touching blobs with different labels are also grouped together.</p> <p>When using M_LABELED, M_FOREGROUND_VALUE cannot be set to M_ZERO, and the identifier image cannot be binary (1-bit). However, you can have the same behavior as M_LABELED with a binary (1-bit) identifier image, by setting M_BLOB_IDENTIFICATION to M_WHOLE_IMAGE and M_IDENTIFIER_TYPE to M_BINARY. (summarize)</p>
<input type="checkbox"/> M_LABELED_TOUCHING	<p>Specifies that blobs with the same label are grouped together, and that touching blobs with different labels are measured individually.</p> <p>When using M_LABELED_TOUCHING, M_FOREGROUND_VALUE cannot be set to M_ZERO, and the identifier image cannot be binary (1-bit). However, you can have the same behavior as M_LABELED_TOUCHING with a binary (1-bit) identifier image, by setting M_BLOB_IDENTIFICATION to M_WHOLE_IMAGE and M_IDENTIFIER_TYPE to M_BINARY.</p> <p>M_LABELED_TOUCHING is not supported with the following:</p> <ul style="list-style-type: none"> • MblobCalculate(), when chain features are selected (MblobSelectFeature() or MblobSelect() with M_NUMBER_OF_CHAINED_PIXELS, M_CHAINS, M_CHAIN_X, M_CHAIN_Y, M_CHAIN_INDEX or M_ALL_FEATURES). • MblobDraw() with M_DRAW_BLOBS_CONTOUR or M_DRAW_HOLES_CONTOUR. • MblobFill() with M_CONTOUR. • MblobSelect() with M_MERGE. <p>(summarize)</p>
<input type="checkbox"/> M_WHOLE_IMAGE	Specifies that all blobs are grouped together.
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_X	<p>Sets the X-coordinate of the top left corner of the region in the blob analysis result buffer to use when drawing in the destination image buffer. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>The default is 0.0 pixels. (summarize)</p>
<input type="checkbox"/> Value	Specifies the relative X-offset, in pixels.
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_Y	<p>Sets the Y-coordinate of the top left corner of the region in the blob analysis result buffer to use when drawing in the destination image buffer. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>The default is 0.0 pixels. (summarize)</p>
<input type="checkbox"/> Value	Specifies the relative Y-offset, in pixels.
<input type="checkbox"/> M_FOREGROUND_VALUE	<p>Sets which pixel values are considered to be in the foreground. (summarize)</p>
<input type="checkbox"/> M_NONZERO	<p>Specifies the blobs consisting of non-zero pixels.</p> <p>This is the default value. (summarize)</p>
<input type="checkbox"/> M_ZERO	Specifies the blobs consisting of zero pixels.
<input type="checkbox"/> M_IDENTIFIER_TYPE	<p>Sets the values that non-zero pixels in the image can have.</p> <p>If the identifier image is already binarized (for example, pixel values for an 8-bit image are either 0 or 0xff), you can change the identifier type to M_BINARY to calculate features faster. (summarize)</p>
<input type="checkbox"/> M_BINARY	Specifies that non-zero pixels must have the maximum value of the buffer (for example, 0xff for an 8-bit image).
<input type="checkbox"/> M_GRAYSCALE	<p>Specifies that non-zero pixels can have any value.</p> <p>This is the default value. (summarize)</p>
<input type="checkbox"/> M_INPUT_SELECT_UNITS	<p>Sets the units in which the CondLow and CondHigh parameters of MblobSelect() expect the limits of the blob selection condition. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_PIXEL .
<input type="checkbox"/> M_PIXEL	Specifies that the limit values passed to MblobSelect() are in pixels units.

<input type="checkbox"/> M_WORLD	Specifies that the limit values passed to MblobSelect() are in world units.
<input type="checkbox"/> M_LATTICE	Sets the image lattice for blob analysis. (summarize)
<input type="checkbox"/> M_4_CONNECTED	Specifies that each pixel has 4 neighbors.
<input type="checkbox"/> M_8_CONNECTED	Specifies that each pixel has 8 neighbors. This is the default value. (summarize)
<input type="checkbox"/> M_MAX_BLOBS	Sets the maximum number of blobs. If this number is reached, processing is stopped and results available in the result buffer become invalid and are discarded. (summarize)
<input type="checkbox"/> M_DISABLED	Specifies to disable the maximum limit on the number of blobs.
<input type="checkbox"/> Value >= 0	Specifies the maximum number of blobs.
<input type="checkbox"/> M_NUMBER_OF_FERETS	Sets the number of Feret angles to be used when calculating a Feret feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 8. (summarize)
<input type="checkbox"/> M_MIN_FERETS	Specifies the minimum number of Feret angles. The minimum number of Feret angles is 2. (summarize)
<input type="checkbox"/> Value > M_MIN_FERETS	Specifies the number of Feret angles. The first Feret angle used is always 0°, and the difference between successive angles is 180° / number of Ferets. (summarize)
<input type="checkbox"/> M_PIXEL_ASPECT_RATIO	Sets the pixel aspect ratio of the image(s). (summarize)
<input type="checkbox"/> Value	Specifies the pixel width/pixel height. The default value is 1.0. (summarize)
<input type="checkbox"/> M_SAVE_RUNS	Sets whether to save run information when calling MblobCalculate() . (summarize)
<input type="checkbox"/> M_DISABLE	Does not save run information. Disabling saves time when performing the first call to MblobCalculate() and reduces the memory requirements for the result buffer. However, you cannot use MblobFill() , MblobGetLabel() , MblobGetRuns() , or MblobLabel() . In addition, you cannot calculate the chained pixels feature (M_CHAINS), using MblobSelectFeature() . (summarize)
<input type="checkbox"/> M_ENABLE	Saves run information. Calls to MblobCalculate() will save run information from the blob identifier image in the result buffer. This is the default value. (summarize)
<input type="checkbox"/> M_STOP_CALCULATE	Stops the current blob analysis calculation. You must call MblobControl() with M_STOP_CALCULATE from another thread (typically of higher priority). Results already available in the result buffer become invalid and will be discarded. (summarize)
<input type="checkbox"/> M_DEFAULT	This parameter must be set to M_DEFAULT .
<input type="checkbox"/> M_TIMEOUT	Sets the maximum processing time. If the timeout limit is exceeded, processing is stopped and results already available in the result buffer become invalid and will be discarded. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that there is no maximum processing time. This is the default value. (summarize)
<input type="checkbox"/> Value >= 0	Specifies the maximum processing time, in msec.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobDraw

Synopsis

Draw specific blob features in the destination image buffer.

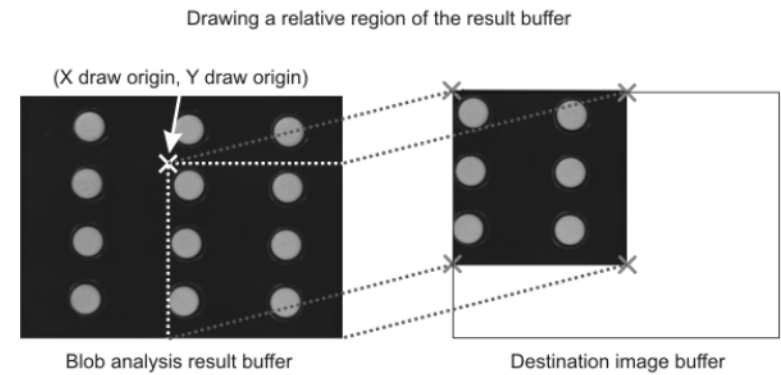
Syntax

```
void MblobDraw(  
    MIL_ID GraphContId,  
    MIL_ID ResultId,  
    MIL_ID DestImageId,  
    MIL_INT Operation,  
    MIL_INT Label,  
    MIL_INT ControlFlag  
)
```

Description

This function draws specific blob features, which were calculated with [MblobCalculate\(\)](#), in the destination buffer. This function can draw multiple features at a time.

You can also draw a sub-region of the blob results by specifying the appropriate values for [MblobControl\(\)](#) with the [M_DRAW_RELATIVE_ORIGIN_X](#) and [M_DRAW_RELATIVE_ORIGIN_Y](#) control types. The relative origin values set the coordinates of the top left corner of the region in the blob analysis result buffer to use when drawing in the destination image buffer. If the destination image buffer is not large enough to contain the entire region, the region will be truncated.



Parameters

GraphContId

Specifies the graphics context to use when drawing. This parameter must be set to one of the following values:

For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.

<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .
----------------------------------------------------------	-------------------------------------------------------------------------------------------------------------

ResultId

Specifies the identifier of the blob analysis result buffer from which to retrieve the features to draw.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The blob analysis result buffer ([ResultId](#)) must be allocated on the same system as the graphics context ([GraphContId](#)). If it is not, an error will occur.

DestImageId

Specifies the identifier of the destination image buffer in which to draw. You can also choose to annotate an image non-destructively, by drawing into its display's overlay buffer. If a calibration object is associated with the image buffer, it is ignored.

Operation

Specifies the type of operation to perform.

The [Operation](#) parameter values in the table below can be added together to draw multiple features at a time. For example, to draw the contour, holes, and position of the blobs, you would specify [M_DRAW_BLOBS_CONTOUR](#) + [M_DRAW_HOLES](#) + [M_DRAW_POSITION](#).

● For specifying the type of operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_AXIS	Draws a cross at the blobs' center of gravity, respecting the principal and secondary axes. Note that to perform this operation, one feature among M_AXIS_PRINCIPAL_ANGLE or M_AXIS_SECONDARY_ANGLE must have been calculated using the binary definition. (summarize)
<input type="checkbox"/> M_DRAW_BLOBS	Draws the blobs. Note that to perform this operation, runs must have been previously saved by using MblobControl() with M_SAVE_RUNS set to M_ENABLE . (summarize)
<input type="checkbox"/> M_DRAW_BLOBS_CONTOUR	Draws the external outline of the blobs. Note that to perform this operation, runs must have been previously saved using MblobControl() with M_SAVE_RUNS set to M_ENABLE . Also note that when using MblobControl() with M_BLOB_IDENTIFICATION set to either M_WHOLE_IMAGE or M_LABELED , both the blob and hole contours will be drawn when performing this operation. If you perform blob analysis calculations with M_BLOB_IDENTIFICATION set to M_LABELED_TOUCHING , then you cannot draw the external outlines of the blobs with M_DRAW_BLOBS_CONTOUR . (summarize)
<input type="checkbox"/> M_DRAW_BOX	Draws the bounding box of each blob. Note that to perform this operation, the M_BOX feature must have been calculated. (summarize)
<input type="checkbox"/> M_DRAW_BOX_CENTER	Draws a cross at the center of the blobs' bounding box. Note that to perform this operation, the M_BOX feature must have been calculated. (summarize)
<input type="checkbox"/> M_DRAW_CENTER_OF_GRAVITY	Same as M_DRAW_POSITION .
<input type="checkbox"/> M_DRAW_FERET_MAX	Draws the blobs' maximum Feret diameter, using an H-type line (-), centered at the blobs' position, at the maximum Feret diameter's angle. Note that to perform this operation, the following features must have been calculated: M_BOX , M_FERET_MAX_DIAMETER , and M_FERET_MAX_ANGLE . (summarize)
<input type="checkbox"/> M_DRAW_FERET_MIN	Draws the blobs' minimum Feret diameter, using an H-type line (-), centered at the blobs' position, at the minimum Feret diameter's angle. Note that to perform this operation, the following features must have been calculated: M_BOX , M_FERET_MIN_DIAMETER , and M_FERET_MIN_ANGLE . (summarize)
<input type="checkbox"/> M_DRAW_HOLES	Draws the holes of the blobs. Note that to perform this operation, runs must have been previously saved by using MblobControl() with M_SAVE_RUNS set to M_ENABLE and one feature among M_BOX_X_MIN , M_BOX_X_MAX , M_BOX_Y_MIN , or M_BOX_Y_MAX must have been calculated. (summarize)
<input type="checkbox"/> M_DRAW_HOLES_CONTOUR	Draws the outline of the blobs' holes.

	Note that to perform this operation, runs must have been previously saved using MblobControl() with M_SAVE_RUNS set to M_ENABLE . Also note that when using MblobControl() with M_BLOB_IDENTIFICATION set to either M_WHOLE_IMAGE or M_LABELED , both the blob and hole contours will be drawn when performing this operation. If you perform blob analysis calculations with M_BLOB_IDENTIFICATION set to M_LABELED_TOUCHING , then you cannot draw the blobs' holes with M_DRAW_HOLES_CONTOUR . (summarize)
<input type="checkbox"/> M_DRAW_POSITION	Draws a cross at the center of gravity of the blobs. Note that to perform this operation, M_CENTER_OF_GRAVITY must have been calculated using the binary definition. (summarize)

Label

Specifies the label of the blob or blobs to draw.

This parameter should be set to one of the following:

● For specifying the label of the blob or blobs to draw	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ALL_BLOBS .
<input type="checkbox"/> M_ALL_BLOBS	Draws all blobs.
<input type="checkbox"/> M_EXCLUDED_BLOBS	Draws all currently excluded blobs.
<input type="checkbox"/> M_INCLUDED_BLOBS	Draws all currently included blobs.
<input type="checkbox"/> Value	Draws the blob with the specified blob label value.

ControlFlag

Reserved for future expansion and must be set to [M_DEFAULT](#).

Compilation information

Header	Include mil.h .
Library	Use mil.lib ; milblob.lib .
DLL	Requires mil.dll ; milblob.dll .

MblobFill

Synopsis

Draw blobs that meet a specified fill criterion.

Syntax

```
void MblobFill(
    MIL_ID BlobResId,
    MIL_ID DestImageBufId,
    MIL_INT Criterion,
    MIL_INT Value
)
```

Description

This function draws, in an image, those blobs that meet a specified fill criterion with the specified fill value.

This function is often used to remove unwanted (excluded or deleted) blobs from the identifier image (by drawing them with the background color), or to highlight included blobs in a different color. Therefore, an appropriate destination image is the blob identifier image (or a copy of it) associated with the result buffer, or another image buffer that has been cleared.

[MblobCalculate\(\)](#) must have been called prior to using this function.

Parameters

BlobResId

Specifies the identifier of the blob analysis result buffer.

DestImageBufId

Specifies the identifier of the destination image buffer. This must be a single band, packed binary, 8, or 16-bit unsigned buffer. Note, this buffer need not be the same size as the original identifier image used to calculate the blobs.

Criterion

Specifies which blobs to draw with the specified value.

The status of the blobs (included or excluded) is taken from the blob analysis result buffer. By default, all blobs in the result buffer are included for future operations. To change the status of a blob, use [MblobSelect\(\)](#).

This parameter can be set to one of the following values:

For specifying which blobs to draw	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL_BLOBS +	Draws all blobs with the specified fill value.
<input type="checkbox"/> M_EXCLUDED_BLOBS +	Draws all currently excluded blobs with the specified fill value.
<input type="checkbox"/> M_INCLUDED_BLOBS +	Draws all currently included blobs with the specified fill value.

Combination constant for any of the possible values of the [Criterion](#) parameter

You can add the following value to the above-mentioned values to set whether the blob's border should be filled.



● For the Criterion parameter	
☐ Value	Description
☐ M_CONTOUR	<p>Specifies that only the blob's borders should be filled.</p> <p>M_CONTOUR uses the coordinates obtained from chained pixels to draw the borders; therefore, if you have already calculated M_CHAINS with MblobCalculate(), M_CONTOUR will operate faster.</p> <p>If you are using M_CONTOUR, you cannot use MblobControl() with M_BLOB_IDENTIFICATION set to M_LABELED_TOUCHING. (summarize)</p>

Value

Specifies the value with which to fill the blobs that meet the specified criterion. If the destination buffer is binary, this value must be 0 or 1.

Remark

- Note that this function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobFree

Synopsis

Free the blob analysis result buffer or the feature list.

Syntax

```
void MblobFree(  
    MIL_ID BlobId  
)
```

Description

This function deletes the specified blob analysis result buffer or feature list and releases any memory associated with it.

Parameter

BlobId

Specifies the identifier of the blob analysis result buffer or feature list buffer to free. The buffer must have been successfully allocated, using [MblobAllocResult\(\)](#) or [MblobAllocFeatureList\(\)](#), prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobGetLabel

Synopsis

Get the label value of a blob at a specified position.

Syntax

```
MIL_INT MblobGetLabel(  
    MIL_ID BlobResId,  
    MIL_INT XPos,  
    MIL_INT YPos,  
    MIL_INT *LabelVarPtr  
)
```

Description

This function gets the label value of a specified blob. Label values are used, for example, to obtain calculation results for single blobs ([MblobGetResultSingle\(\)](#)). Blob label values must have been generated by calling [MblobCalculate\(\)](#).

Parameters

- BlobResId**
- Specifies the identifier of the blob analysis result buffer where the labels are stored. Note, this function cannot obtain the label values of blobs that have been deleted from the result buffer with [MblobSelect\(\)](#).
- XPos**
- Specifies the X-coordinate of the blob.
- YPos**
- Specifies the Y-coordinate of the blob.
- LabelVarPtr**
- Specifies the address of the variable in which to write the label value. If there is no blob at the specified location, the blob has been deleted, or if the blob's [XPos](#) and [YPos](#) lie outside the original identifier image, **M_NULL** is returned instead of the label value. Also, since **MblobGetLabel()** returns the label value, you can set this parameter to **M_NULL**.

Return value

The returned value is the label value of the specified blob.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobGetNumber

Synopsis

Get the number of currently included blobs.

Syntax

```
MIL_INT MblobGetNumber(  
    MIL_ID BlobResId,  
    MIL_INT *CountVarPtr  
)
```

Description

This function reads the number of currently included blobs from the specified blob analysis result buffer. All blobs are included unless their status is changed, using [MblobSelect\(\)](#). Included blobs will be included in future operations and result retrievals.

This function must be used to determine the number of blob results that will be returned by [MblobGetResult\(\)](#).

A call to [MblobCalculate\(\)](#) must have been made prior to using this function.

Parameters

- BlobResId**
Specifies the identifier of the blob analysis result buffer.
- CountVarPtr**
Specifies the address of the variable in which to write the count. Since **MblobGetNumber()** also returns the number of selected blobs, you can set this parameter to **M_NULL**.

Return value

The returned value is the number of included blobs in the specified result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobGetResult

Synopsis

Get results for a feature of the included blobs, from a blob result buffer.

Syntax

```
void MblobGetResult(
    MIL_ID BlobResId,
    MIL_INT Feature,
    void *TargetArrayPtr
)
```

Description

This function retrieves the results for a specified feature for all included blobs, from the blob analysis result buffer.

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image. When calculated in pixel units, the pixel aspect ratio, specified with [MblobControl\(\)](#), is taken into account horizontally. Results are returned in units of "pixel height" since the pixel width is adjusted to be equal to the pixel height.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set **TargetArrayPtr** to **M_NULL**. When this parameter is set to **M_NULL**, the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call **MblobGetResult()** again and you pass an array to the parameter **TargetArrayPtr**. You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared. Note that the requests for **MblobGetResult()** and [MblobGetResultSingle\(\)](#) are put in the same queue.

By setting the **Feature** parameter to [M_INTERACTIVE](#), you can view the results interactively.

Note that trying to retrieve a result which is not available generates an error.

Parameters

BlobResId

Specifies the identifier of the blob analysis result buffer from which to get results. The specified feature(s) must have already been calculated with [MblobCalculate\(\)](#).

Feature

Specifies the feature for which to retrieve the results or opens a dialog box that displays the features stored in the result buffer interactively.

To display the features currently stored in the result buffer in an interactive dialog box, select the following.

Unless otherwise specified, the following values require that you pass the [TargetArrayPtr](#) parameter **M_NULL**.

• For displaying features in an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	<div>[This is only applicable to Windows]</div> <div>Opens a dialog box containing a spreadsheet that displays the types of results stored in the result buffer. The results are displayed in separate columns. If there are queued requests, only those results are displayed.</div> <div>(summarize)</div>

To retrieve the result for a binary feature, select one of the following.

Unless otherwise specified, the following values require that you pass the [TargetArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to the number of currently included blobs. This number can be obtained, using [MblobGetNumber\(\)](#).

● For retrieving results for a binary feature	
Value	Description
M_AREA +	Returns the number of foreground pixels in each blob (holes are not counted).
M_BOX_X_MAX +	Returns the extreme right X-coordinate of each blob.
M_BOX_X_MIN +	Returns the extreme left X-coordinate of each blob.
M_BOX_Y_MAX +	Returns the extreme bottom Y-coordinate of each blob.
M_BOX_Y_MIN +	Returns the extreme top Y-coordinate of each blob.
M_BREADTH +	Returns a measure of the true breadth of each blob.
M_COMPACTNESS +	Returns the compactness of each blob.
M_CONVEX_PERIMETER +	Returns an approximation of the perimeter of the convex hull of each blob.
M_ELONGATION +	Returns a value that is equal to M_LENGTH / M_BREADTH of each blob.
M_EULER_NUMBER +	Returns the number of blobs - number of holes. This value is more useful if calculated using the M_WHOLE_IMAGE processing mode instead of the M_INDIVIDUAL processing mode. (summarize)
M_FERET_ELONGATION +	Returns a measure of the shape of each blob. It is equal to M_FERET_MAX_DIAMETER / M_FERET_MIN_DIAMETER . (summarize)
M_FERET_MAX_ANGLE +	Returns the angle at which the maximum Feret diameter is found for each blob. The value is in degrees, with positive values indicating a counter-clockwise displacement from the positive X-axis. (summarize)
M_FERET_MAX_DIAMETER +	Returns the largest Feret diameter found after checking a certain number of angles.
M_FERET_MEAN_DIAMETER +	Returns the average of the Feret diameters at the angles checked for each blob.
M_FERET_MIN_ANGLE +	Returns the angle at which the minimum Feret diameter is found. The value is in degrees, with positive values indicating a counter-clockwise displacement from the positive X-axis. (summarize)
M_FERET_MIN_DIAMETER +	Returns the smallest Feret diameter found after checking a certain number of angles.
M_FERET_X +	Returns the dimension of the minimum bounding box of each blob, in the horizontal direction.
M_FERET_Y +	Returns the dimension of the minimum bounding box of each blob, in the vertical direction.
M_FIRST_POINT_X +	Returns (along with M_FIRST_POINT_Y) a unique point for each blob, that is always on the perimeter of the blob. The X-coordinate is that of the left-most pixel on the top-most line of the blob. (summarize)
M_FIRST_POINT_Y +	Returns (along with M_FIRST_POINT_X) a unique point for each blob, that is always on the perimeter of the blob. The Y-coordinate is that of the top-most line of the blob. (summarize)
M_GENERAL_FERET +	Returns the Feret diameter at the specified angle.
M_GENERAL_MOMENT +	Returns the moment calculation.
M_INTERCEPT_0 +	Returns the number of times a transition from background to foreground (not vice versa) occurs in the horizontal direction for the entire blob. In other words, it is equal to the number of times the neighborhood configuration [B, F] occurs in a blob, where B is a background pixel and F is a foreground pixel. (summarize)
M_INTERCEPT_45 +	Returns the number of times that the neighborhood configuration $\begin{bmatrix} \bullet & F \\ B & \bullet \end{bmatrix}$ occurs in a blob, where F is a foreground pixel, B is a background pixel and a dot can be any pixel value.
M_INTERCEPT_90 +	

	Returns the number of times that the neighborhood configuration $\begin{bmatrix} F \\ B \end{bmatrix}$ occurs in a blob.
<input type="checkbox"/> M_INTERCEPT_135 +	Returns the number of times that the neighborhood configuration $\begin{bmatrix} F & \bullet \\ \bullet & B \end{bmatrix}$ occurs in a blob.
<input type="checkbox"/> M_LABEL_VALUE +	Returns the label value for each blob in an image. This is a positive integer (>= 1) that is unique for each blob. (summarize)
<input type="checkbox"/> M_LENGTH +	Returns a measure of the true length of each blob, although it is only accurate for certain object types (for example, long thin ones) because it is derived from the perimeter (P) and area (A) assuming that $P = 2(\text{length} + \text{breadth})$ and $A = \text{length} \times \text{breadth}$.
<input type="checkbox"/> M_NUMBER_OF_CHAINED_PIXELS +	Returns the number of chained pixels for each blob.
<input type="checkbox"/> M_NUMBER_OF_HOLES +	Returns the number of holes in each blob. Holes that intersect the edge of the image are not counted (they might not be holes). This value is equal to $1 - \text{M_EULER_NUMBER}$ and is therefore a true hole count if calculated using the M_INDIVIDUAL processing mode. (summarize)
<input type="checkbox"/> M_NUMBER_OF_RUNS +	Returns the total number of runs in each blob.
<input type="checkbox"/> M_PERIMETER +	Returns the total length of edges in each blob (including the edges of any holes), with an allowance made for the staircase effect that is produced when diagonal edges are digitized (inside corners are counted as 1.414, rather than 2.0). A single pixel blob (area = 1) has a perimeter of 4.0. (summarize)
<input type="checkbox"/> M_ROUGHNESS +	Returns a measure of how rough a blob is and is equal to $\text{M_PERIMETER} / \text{M_CONVEX_PERIMETER}$. A smooth convex blob will have the minimum roughness of 1.0. (summarize)
<input type="checkbox"/> M_X_MAX_AT_Y_MAX +	Returns the maximum X-coordinate at the maximum Y-coordinate of the blob.
<input type="checkbox"/> M_X_MAX_AT_Y_MIN +	Returns the maximum X-coordinate at the minimum Y-coordinate of the blob.
<input type="checkbox"/> M_X_MIN_AT_Y_MAX +	Returns the minimum X-coordinate at the maximum Y-coordinate of the blob.
<input type="checkbox"/> M_X_MIN_AT_Y_MIN +	Returns the minimum X-coordinate at the minimum Y-coordinate of the blob.
<input type="checkbox"/> M_Y_MAX_AT_X_MAX +	Returns the maximum Y-coordinate at the maximum X-coordinate of the blob.
<input type="checkbox"/> M_Y_MAX_AT_X_MIN +	Returns the maximum Y-coordinate at the minimum X-coordinate of the blob.
<input type="checkbox"/> M_Y_MIN_AT_X_MAX +	Returns the minimum Y-coordinate at the maximum X-coordinate of the blob.
<input type="checkbox"/> M_Y_MIN_AT_X_MIN +	Returns the minimum Y-coordinate at the minimum X-coordinate of the blob.

To retrieve the result for a grayscale feature, select one of the following. Results for these features are only available if both a blob identifier image and a grayscale image were passed to [MblobCalculate\(\)](#) (and the features were selected for calculation).

Unless otherwise specified, the following values require that you pass the [TargetArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to the number of currently included blobs. This number can be obtained using [MblobGetNumber\(\)](#).

● For retrieving results for a grayscale feature with blob id and grayscale image	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MAX_PIXEL +	Returns the maximum pixel value found in each blob.
<input type="checkbox"/> M_MEAN_PIXEL +	Returns the mean pixel value in each blob.
<input type="checkbox"/> M_MIN_PIXEL +	Returns the minimum pixel value found in each blob.
<input type="checkbox"/> M_SIGMA_PIXEL +	Returns the standard deviation of pixel values in each blob.
<input type="checkbox"/> M_SUM_PIXEL +	Returns the sum of all pixel values in each blob.
<input type="checkbox"/> M_SUM_PIXEL_SQUARED +	Returns the sum of the squares of each pixel value in each blob.

To retrieve the result for a feature that has two different definitions (a binary and a grayscale definition), select one of the following values. If you did not provide both a blob identifier image and a grayscale image, only the binary version was calculated. If you did provide a grayscale image, both versions were calculated, unless otherwise specified. If both versions were calculated and no version is specified, then the grayscale version of the feature is retrieved.

Unless otherwise specified, the following values require that you pass the [TargetArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to the number of currently included blobs ([MblobGetNumber\(\)](#)).

● For retrieving results for a feature that has two different definitions	
☐ Value	Description
☐ M_AXIS_PRINCIPAL_ANGLE +	Returns the angle at which each blob has the least moment of inertia. For elongated blobs, it is aligned with the longest axis. The result is always between -90° and $+90^\circ$, measured in a counter-clockwise direction from the positive X-axis. When the blob identifier image is calibrated, this feature is calculated in calibrated units, otherwise it is calculated in pixel units. (summarize)
☐ M_AXIS_SECONDARY_ANGLE +	Returns the angle perpendicular to M_AXIS_PRINCIPAL_ANGLE of each blob. It is always between -90° and $+90^\circ$. (summarize)
☐ M_CENTER_OF_GRAVITY_X +	Returns the X-position of the center of gravity of each blob.
☐ M_CENTER_OF_GRAVITY_Y +	Returns the Y-position of the center of gravity of each blob.
☐ M_MOMENT_CENTRAL_Xn_Ym +	Returns the calculation of central moments for each blob. Results are only available in pixel units. (summarize)
☐ M_MOMENT_Xn_Ym +	Returns the calculation of ordinary moments for each blob. Results are only available in pixel units. (summarize)

Combination constants for the values listed in [For retrieving results for a feature that has two different definitions](#)

You can add one of the following values to the above-mentioned values to set whether the results should be returned for the binary or grayscale version of the selected feature.

● For feature parameters that have two definitions	
☐ Value	Description
☐ M_BINARY +	Returns the result for the binary version of the selected feature.
☐ M_GRAYSCALE +	Returns the result for the grayscale version of the selected feature. This is the default value. (summarize)

To retrieve a result that specifies whether a limit is reached, select one of the values specified in the table below:

Unless otherwise specified, the following values require that you pass the [TargetArrayPtr](#) parameter the address of a `MIL_DOUBLE`.

● For retrieving results of reaching limits	
☐ Value	Description
☐ M_MAX_BLOBS_END	Returns whether the maximum number of blobs was reached. This limit is set using MblobControl() with M_MAX_BLOBS . (summarize) <i>TargetArrayPtr info</i> Return values: M_FALSE Specifies that the maximum number of blobs was not reached. M_TRUE Specifies that the maximum number of blobs was reached.
☐ M_TIMEOUT_END	Returns whether the timeout limit was reached. You can see the timeout limit using MblobControl() with M_TIMEOUT .

	(summarize)
	<i>TargetArrayPtr info</i> Return values: M_FALSE Specifies that the timeout limit was not reached. M_TRUE Specifies that the timeout limit was reached.

Combination constants for [the values listed in](#) all tables **except For displaying features in an interactive dialog box, For retrieving results of reaching limits**

You can add one of the following values to the above-mentioned values to set the requested results to a data type.

● For specifying a data type	
▢ Value	Description
▢ M_TYPE_CHAR	Casts the requested results to a <i>char</i> . (summarize) <i>TargetArrayPtr info</i> Data type: array of type char Array size: Number of currently included blobs (MblobGetNumber())
▢ M_TYPE_DOUBLE	Casts the requested results to a <i>double</i> . This is the default value. (summarize) <i>TargetArrayPtr info</i> Data type: array of type double Array size: Number of currently included blobs (MblobGetNumber())
▢ M_TYPE_FLOAT	Casts the requested results to a <i>float</i> . (summarize) <i>TargetArrayPtr info</i> Data type: array of type float Array size: Number of currently included blobs (MblobGetNumber())
▢ M_TYPE_LONG	Casts the requested results to a <i>long</i> . (summarize) <i>TargetArrayPtr info</i> Data type: array of type long Array size: Number of currently included blobs (MblobGetNumber())
▢ M_TYPE_MIL_DOUBLE	Casts the requested results to a <i>MIL_DOUBLE</i> . (summarize) <i>TargetArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE • Array size: Number of currently included blobs (MblobGetNumber()) • Data type: MIL_DOUBLE • Note: When retrieving timing results with M_TIMEOUT_END.
▢ M_TYPE_MIL_INT	Casts the requested results to a <i>MIL_INT</i> . (summarize) <i>TargetArrayPtr info</i> Data type: array of type MIL_INT Array size: Number of currently included blobs (MblobGetNumber())
▢ M_TYPE_MIL_INT32	Casts the requested results to a <i>MIL_INT32</i> . (summarize) <i>TargetArrayPtr info</i> Data type: array of type MIL_INT32

	Array size: Number of currently included blobs (MblobGetNumber())
<input type="checkbox"/> M_TYPE_MIL_INT64	<div>Casts the requested results to a <i>MIL_INT64</i>. (summarize)</div> <div><i>TargetArrayPtr info</i> Data type: array of type MIL_INT64 Array size: Number of currently included blobs (MblobGetNumber())</div>
<input type="checkbox"/> M_TYPE_SHORT	<div>Casts the requested results to a <i>short</i>. (summarize)</div> <div><i>TargetArrayPtr info</i> Data type: array of type short Array size: Number of currently included blobs (MblobGetNumber())</div>

TargetArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• array of type char• array of type double• array of type float• array of type long• array of type MIL_DOUBLE• array of type MIL_INT• array of type MIL_INT32• array of type MIL_INT64• array of type short• M_NULL• MIL_DOUBLE

Specifies the address of the array in which to write results. Each blob creates a separate result entry. Only results for blobs that are currently included are obtained.

You must set this parameter to **M_NULL** if you are setting the **Feature** parameter to **M_INTERACTIVE** or if you would like to put the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobGetResultSingle

Synopsis

Get results for a feature of a single blob.

Syntax

```
void MblobGetResultSingle(
    MIL_ID BlobResId,
    MIL_INT LabelVal,
    MIL_INT Feature,
    void *TargetVarPtr
)
```

Description

This function retrieves the result for a specified feature, for a specific blob, from the blob analysis result buffer. The blob for which to obtain the result is determined by its label value.

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image. When calculated in pixel units, the pixel aspect ratio, specified with [MblobControl\(\)](#), is taken into account horizontally. Results are returned in units of "pixel height" since the pixel width is adjusted to be equal to the pixel height.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [TargetVarPtr](#) to [M_NULL](#). When this parameter is set to [M_NULL](#), the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [MblobGetResult\(\)](#) again and you pass an array to the parameter [TargetVarPtr](#). You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared. Note that the requests for [MblobGetResult\(\)](#) and [MblobGetResultSingle\(\)](#) are put in the same queue.

By setting the [Feature](#) parameter to [M_INTERACTIVE](#), you can view the results, for a specified blob, interactively.

Note that trying to retrieve a result which is not available generates an error.

Parameters

BlobResId

Specifies the identifier of the blob analysis result buffer from which to get the result.

LabelVal

Specifies the label value of the blob for which to get the result. The label value can be obtained, using [MblobGetLabel\(\)](#) or [MblobGetResult\(\)](#). Note, you cannot obtain results for blobs that have been deleted from the result buffer, using [MblobSelect\(\)](#).

Feature

Specifies the feature for which to retrieve results or opens a dialog box that displays the results stored in the result buffer. The specified feature(s) must have already been calculated with [MblobCalculate\(\)](#).

To display the results in an interactive dialog box, select the following.

Unless otherwise specified, the following values require that you pass the [TargetVarPtr](#) parameter [M_NULL](#).

● For displaying results in an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows]

Opens a dialog box containing a spreadsheet that displays the types of results stored in the result buffer. The results are displayed in separate columns. If there are queued requests, only those results are displayed.
([summarize](#))

To retrieve the result for a binary feature, select one of the following.

Unless otherwise specified, the following values require that you pass the *TargetVarPtr* parameter the address of a *MIL_DOUBLE*.

● For retrieving results for a binary feature	
Value	Description
<input type="checkbox"/> M_AREA +	Returns the number of foreground pixels in the specified blob (holes are not counted).
<input type="checkbox"/> M_BOX_X_MAX +	Returns the extreme right coordinate of the specified blob.
<input type="checkbox"/> M_BOX_X_MIN +	Returns the extreme left coordinate of the specified blob.
<input type="checkbox"/> M_BOX_Y_MAX +	Returns the extreme bottom coordinate of the specified blob.
<input type="checkbox"/> M_BOX_Y_MIN +	Returns the extreme top coordinate of the specified blob.
<input type="checkbox"/> M_BREADTH +	Returns a measure of the true breadth of the specified blob.
<input type="checkbox"/> M_CHAIN_INDEX +	<p>Returns the indices which differentiate each chain's pixels within the specified blob. The blob's bordering chain is identified as index 1. Chained pixels that delimit holes in blobs are identified by subsequent indices for each chain.</p> <p>(summarize)</p> <p><i>TargetVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINED_PIXELS to determine the size.</p>
<input type="checkbox"/> M_CHAIN_X +	<p>Returns the X-coordinate of each chained pixel in the specified blob, for all chains contained within the blob.</p> <p>Results are only available in pixel units.</p> <p>(summarize)</p> <p><i>TargetVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINED_PIXELS to determine the size.</p>
<input type="checkbox"/> M_CHAIN_Y +	<p>Returns the Y-coordinate of each chained pixel in the specified blob, for all chains contained within the blob.</p> <p>Results are only available in pixel units.</p> <p>(summarize)</p> <p><i>TargetVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINED_PIXELS to determine the size.</p>
<input type="checkbox"/> M_COMPACTNESS +	Returns the compactness of the specified blob.
<input type="checkbox"/> M_CONVEX_PERIMETER +	Returns an approximation of the perimeter of the convex hull of the specified blob.
<input type="checkbox"/> M_ELONGATION +	Returns a value that is equal to M_LENGTH / M_BREADTH of the specified blob.
<input type="checkbox"/> M_EULER_NUMBER +	<p>Returns the number of blobs - number of holes. This value is more useful for M_WHOLE_IMAGE than for M_INDIVIDUAL processing mode.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_ELONGATION +	<p>Returns a measure of the shape of the specified blob. It is equal to M_FERET_MAX_DIAMETER / M_FERET_MIN_DIAMETER.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_MAX_ANGLE +	<p>Returns the angle at which the maximum Feret diameter is found for the specified blob. The value is in degrees, with positive values indicating a counter-clockwise displacement from the positive X-axis.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_MAX_DIAMETER +	Returns the largest Feret diameter found after checking a certain number of angles.
<input type="checkbox"/> M_FERET_MEAN_DIAMETER +	Returns the average of the Feret diameters at the angles checked for the specified blob.
<input type="checkbox"/> M_FERET_MIN_ANGLE +	Returns the angle at which the minimum Feret diameter is found. The value is in degrees, with positive values indicating a counter-clockwise displacement from

	the positive X-axis. (summarize)
<input type="checkbox"/> M_FERET_MIN_DIAMETER +	Returns the smallest Feret diameter found after checking a certain number of angles.
<input type="checkbox"/> M_FERET_X +	Returns the dimension of the minimum bounding box of the specified blob, in the horizontal direction.
<input type="checkbox"/> M_FERET_Y +	Returns the dimension of the minimum bounding box of the specified blob, in the vertical direction.
<input type="checkbox"/> M_FIRST_POINT_X +	Returns (along with M_FIRST_POINT_Y) a unique point for the specified blob, that is always on the perimeter of the blob. The X-coordinate is that of the left-most pixel on the top-most line of the blob. (summarize)
<input type="checkbox"/> M_FIRST_POINT_Y +	Returns (along with M_FIRST_POINT_X) a unique point for the specified blob, that is always on the perimeter of the blob. The Y-coordinate is that of the topmost line of the blob. (summarize)
<input type="checkbox"/> M_GENERAL_FERET +	Returns the Feret diameter at the specified angle.
<input type="checkbox"/> M_GENERAL_MOMENT +	Returns the moment calculation.
<input type="checkbox"/> M_INTERCEPT_0 +	Returns the number of times a transition from background to foreground (not vice versa) occurs in the horizontal direction for the entire blob. In other words, it is equal to the number of times the neighborhood configuration [B, F] occurs in the specified blob, where B is a background pixel and F is a foreground pixel. (summarize)
<input type="checkbox"/> M_INTERCEPT_45 +	Returns the number of times that the neighborhood configuration $\begin{bmatrix} \bullet & F \\ B & \bullet \end{bmatrix}$ occurs in the specified blob, where F is a foreground pixel, B is a background pixel and a dot can be any pixel value.
<input type="checkbox"/> M_INTERCEPT_90 +	Returns the number of times that the neighborhood configuration $\begin{bmatrix} F \\ B \end{bmatrix}$ occurs in the specified blob.
<input type="checkbox"/> M_INTERCEPT_135 +	Returns the number of times that the neighborhood configuration $\begin{bmatrix} F & \bullet \\ \bullet & B \end{bmatrix}$ occurs in the specified blob.
<input type="checkbox"/> M_LABEL_VALUE +	Returns the label value for the specified blob in an image. This is a positive integer (≥ 1) that is unique for the specified blob. (summarize)
<input type="checkbox"/> M_LENGTH +	Returns a measure of the true length of an object, although it is only accurate for certain object types (for example, long thin ones) because it is derived from the perimeter (P) and area (A) assuming that $P = 2(\text{length} + \text{breadth})$ and $A = \text{length} \times \text{breadth}$.
<input type="checkbox"/> M_NUMBER_OF_CHAINED_PIXELS +	Returns the number of chained pixels for the specified blob.
<input type="checkbox"/> M_NUMBER_OF_HOLES +	Returns the number of holes in the specified blob. Holes that intersect the edge of the image are not counted (they might not be holes). This value is equal to $1 - \text{M_EULER_NUMBER}$ and is therefore a true hole count if calculated using the M_INDIVIDUAL processing mode. (summarize)
<input type="checkbox"/> M_NUMBER_OF_RUNS +	Returns the total number of runs in the specified blob.
<input type="checkbox"/> M_PERIMETER +	Returns the total length of edges in the specified blob (including the edges of any holes), with an allowance made for the staircase effect that is produced when diagonal edges are digitized (inside corners are counted as 1.414, rather than 2.0). A single pixel blob (area = 1) has a perimeter of 4.0. (summarize)
<input type="checkbox"/> M_ROUGHNESS +	Returns a measure of how rough the specified blob is and is equal to $\text{M_PERIMETER} / \text{M_CONVEX_PERIMETER}$. A smooth convex blob will have the minimum roughness of 1.0. (summarize)
<input type="checkbox"/> M_X_MAX_AT_Y_MAX +	Returns the maximum X-coordinate at the maximum Y-coordinate of the specified blob.
<input type="checkbox"/> M_X_MAX_AT_Y_MIN +	Returns the maximum X-coordinate at the minimum Y-coordinate of the specified blob.
<input type="checkbox"/> M_X_MIN_AT_Y_MAX +	Returns the minimum X-coordinate at the maximum Y-coordinate of the specified blob.
<input type="checkbox"/> M_X_MIN_AT_Y_MIN +	Returns the minimum X-coordinate at the minimum Y-coordinate of the specified blob.
<input type="checkbox"/> M_Y_MAX_AT_X_MAX +	Returns the maximum Y-coordinate at the maximum X-coordinate of the specified blob.

<input type="checkbox"/> M_Y_MAX_AT_X_MIN +	Returns the maximum Y-coordinate at the minimum X-coordinate of the specified blob.
<input type="checkbox"/> M_Y_MIN_AT_X_MAX +	Returns the minimum Y-coordinate at the maximum X-coordinate of the specified blob.
<input type="checkbox"/> M_Y_MIN_AT_X_MIN +	Returns the minimum Y-coordinate at the minimum X-coordinate of the specified blob.

The following features require grayscale pixel values, and were calculated only if you provided a grayscale image.

Unless otherwise specified, the following values require that you pass the [TargetVarPtr](#) parameter the address of a MIL_DOUBLE.

● For grayscale pixel values	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MAX_PIXEL +	Returns the maximum pixel value found in the specified blob.
<input type="checkbox"/> M_MEAN_PIXEL +	Returns the mean pixel value in the specified blob.
<input type="checkbox"/> M_MIN_PIXEL +	Returns the minimum pixel value found in the specified blob.
<input type="checkbox"/> M_SIGMA_PIXEL +	Returns the standard deviation of pixel values in the specified blob.
<input type="checkbox"/> M_SUM_PIXEL +	Returns the sum of all pixel values in the specified blob.
<input type="checkbox"/> M_SUM_PIXEL_SQUARED +	Returns the sum of the squares of each pixel value in the specified blob.

To retrieve the result for a feature that has two different definitions (a binary and a grayscale definition), select one of the following values.

If you did not provide both the specified blob identifier image and a grayscale image, only the binary version was calculated. If you did provide a grayscale image, both versions were calculated, unless otherwise specified. To specify which result to retrieve, see the combination values below. If both versions were calculated and no version is specified, then the grayscale version of the feature is retrieved.

Unless otherwise specified, the following values require that you pass the [TargetVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a feature that has two different definitions	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_AXIS_PRINCIPAL_ANGLE +	Returns the angle at which the specified blob has the least moment of inertia. For elongated blobs, it is aligned with the longest axis. The result is always between -90° and +90°, measured in a counter-clockwise direction from the positive X-axis. When the blob identifier image is calibrated this feature is calculated in calibrated units; otherwise they are calculated in pixel units. (summarize)
<input type="checkbox"/> M_AXIS_SECONDARY_ANGLE +	Returns the angle perpendicular to M_AXIS_PRINCIPAL_ANGLE . It is always between -90° and +90°. (summarize)
<input type="checkbox"/> M_CENTER_OF_GRAVITY_X +	Returns the X-position of the center of gravity of the specified blob.
<input type="checkbox"/> M_CENTER_OF_GRAVITY_Y +	Returns the Y-position of the center of gravity of the specified blob.
<input type="checkbox"/> M_MOMENT_CENTRAL_Xn_Ym +	Returns the calculation of central moments for the specified blob. Results are only available in pixel units. (summarize)
<input type="checkbox"/> M_MOMENT_Xn_Ym +	Returns the calculation of ordinary moments for the specified blob. Results are only available in pixel units. (summarize)

Combination constants for [the values listed in](#) all tables **except** For displaying results in an interactive dialog box

You can add one of the following values to the above-mentioned values to set whether the results should be returned for the binary or grayscale version of the selected feature.

● For a feature that has two different definitions and both versions have been calculated	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_BINARY +	Returns the result for the binary version of the selected feature.
<input type="checkbox"/> M_GRAYSCALE +	Returns the result for the grayscale version of the selected feature. This is the default value. (summarize)

Combination constants for the values listed in [For a feature that has two different definitions](#)

You can add one of the following values to the above-mentioned values to set the requested results to a data type.

● For specifying the data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_CHAR	Casts the requested results to a <i>char</i> . (summarize)
	<i>TargetVarPtr info</i> Data type: char
<input type="checkbox"/> M_TYPE_DOUBLE	Casts the requested results to a <i>double</i> . This is the default value. (summarize)
	<i>TargetVarPtr info</i> Data type: double
<input type="checkbox"/> M_TYPE_FLOAT	Casts the requested results to a <i>float</i> . (summarize)
	<i>TargetVarPtr info</i> Data type: float
<input type="checkbox"/> M_TYPE_LONG	Casts the requested results to a <i>long</i> . (summarize)
	<i>TargetVarPtr info</i> Data type: long
<input type="checkbox"/> M_TYPE_MIL_DOUBLE	Casts the requested results to a <i>MIL_DOUBLE</i> . (summarize)
	<i>TargetVarPtr info</i> Data type: MIL_DOUBLE
<input type="checkbox"/> M_TYPE_MIL_INT	Casts the requested results to a <i>MIL_INT</i> . (summarize)
	<i>TargetVarPtr info</i> Data type: MIL_INT
<input type="checkbox"/> M_TYPE_MIL_INT32	Casts the requested results to a <i>MIL_INT32</i> . (summarize)
	<i>TargetVarPtr info</i> Data type: MIL_INT32
<input type="checkbox"/> M_TYPE_MIL_INT64	Casts the requested results to a <i>MIL_INT64</i> . (summarize)
	<i>TargetVarPtr info</i> Data type: MIL_INT64
<input type="checkbox"/> M_TYPE_SHORT	Casts the requested results to a <i>short</i> .

	(summarize)
	<i>TargetVarPtr info</i> Data type: short

TargetVarPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• array of type MIL_DOUBLE• char• double• float• long• M_NULL• MIL_DOUBLE• MIL_INT• MIL_INT32• MIL_INT64• short

Specifies the address in which to write the result retrieved from the blob analysis result buffer.

You must set this parameter to **M_NULL** if you are setting the [Feature](#) parameter to [M_INTERACTIVE](#) or if you would like to put the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobGetRuns

Synopsis

Get the blob run-length encoding information.

Syntax

```
void MblobGetRuns(
    MIL_ID BlobResId,
    MIL_INT LabelVal,
    MIL_INT ArrayType,
    void *RunXPtr,
    void *RunYPtr,
    void *RunLengthPtr
)
```

Description

This function obtains the coordinate and length of each run (unbroken horizontal sequence of foreground pixels) in a specified blob, from the blob analysis result buffer. Prior to using this function, [M_NUMBER_OF_RUNS](#) must have been added to the feature list, using [MblobSelectFeature\(\)](#), and a call to [MblobCalculate\(\)](#) must have been made.

The coordinate and length arrays must be large enough to hold information for all runs in the specified blob. The number of runs for a blob can be obtained, using [MblobGetResult\(\)](#) or [MblobGetResultSingle\(\)](#). The number of runs, as well as the run-length encoding results, are given in raw pixel values and are not affected by the pixel aspect ratio.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set the [RunXPtr](#), [RunYPtr](#) and [RunLengthPtr](#) parameters to **M_NULL**. When all these result parameters are set to **M_NULL**, the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued results are retrieved when at least one result parameter is passed an array. You must ensure that the coordinate and length arrays are sufficiently large to hold all the requested results. Once the results are retrieved, the queue associated with the result buffer is cleared.

Parameters

BlobResId
Specifies the identifier of the blob analysis result buffer.

LabelVal
Specifies the label value of a blob.

For the upper limit of the selected condition	
Value	Description
M_ALL	Specifies to read consecutively the runs of all blobs selected by MblobSelect() . The runs are placed in the array in an increasing order of the labels of blobs. The array should be large enough to hold all the runs. The total size of the required array is the sum of all results obtained by MblobGetResult() or MblobGetResultSingle() with M_NUMBER_OF_RUNS for all blobs. (summarize)
Value	Specifies the label value of the blob for which to get run information. The label value for a blob can be obtained, using MblobGetLabel() or MblobGetResult() . You cannot obtain run-encoding information for blobs that have been deleted from the result buffer. (summarize)

ArrayType
Sets the type of the arrays in which the coordinate and length of the runs for a blob will be returned. This parameter can be set to one of the following values:

The following values require that you pass the value listed in the data-type area of the specified parameter.

● For specifying the type of the arrays	
☐ Value	Description
☐ M_TYPE_CHAR	<p>Returns an array of type <i>char</i>. (summarize)</p> <p><i>RunXPtr, RunYPtr, and RunLengthPtr info</i> Data type: array of type <i>char</i> Array size: Large enough to hold the number of currently included blobs.</p>
☐ M_TYPE_LONG	<p>Returns an array of type <i>long</i>. (summarize)</p> <p><i>RunXPtr, RunYPtr, and RunLengthPtr info</i> Data type: array of type <i>long</i> Array size: Large enough to hold the number of currently included blobs.</p>
☐ M_TYPE_MIL_INT	<p>Returns an array of type <i>MIL_INT</i>. (summarize)</p> <p><i>RunXPtr, RunYPtr, and RunLengthPtr info</i> Data type: array of type <i>MIL_INT</i> Array size: Large enough to hold the number of currently included blobs.</p>
☐ M_TYPE_MIL_INT32	<p>Returns an array of type <i>MIL_INT32</i>. (summarize)</p> <p><i>RunXPtr, RunYPtr, and RunLengthPtr info</i> Data type: array of type <i>MIL_INT32</i> Array size: Large enough to hold the number of currently included blobs.</p>
☐ M_TYPE_MIL_INT64	<p>Returns an array of type <i>MIL_INT64</i>. (summarize)</p> <p><i>RunXPtr, RunYPtr, and RunLengthPtr info</i> Data type: array of type <i>MIL_INT64</i> Array size: Large enough to hold the number of currently included blobs.</p>
☐ M_TYPE_SHORT	<p>Returns an array of type <i>short</i>. (summarize)</p> <p><i>RunXPtr, RunYPtr, and RunLengthPtr info</i> Data type: array of type <i>short</i> Array size: Large enough to hold the number of currently included blobs.</p>

RunXPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type *char*
- array of type *long*
- array of type *MIL_INT*
- array of type *MIL_INT32*
- array of type *MIL_INT64*
- array of type *short*

Specifies the address of the array in which to write the X-coordinate of the start (left-most pixel) of each run in the specified blob.

Note, you can set [RunXPtr](#) to **M_NULL** if the X-coordinate is not required.

RunYPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type *char*

- array of type long
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- array of type short

Specifies the address of the array in which to write the Y-coordinate of the start of each run in the specified blob.

Note, you can set [RunYPtr](#) to **M_NULL** if the Y-coordinate is not required.

RunLengthPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type char
- array of type long
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- array of type short

Specifies the address of the array in which to write the length of each run in the specified blob.

Note, you can set [RunLengthPtr](#) to **M_NULL** if the length of each run is not required.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobInquire

Synopsis

Inquire about a processing setting associated with a blob analysis result buffer.

Syntax

```
MIL_INT MblobInquire(  
    MIL_ID BlobResId,  
    MIL_INT InquireType,  
    void *UserVarPtr  
)
```

Description

This function inquires about a specified processing setting associated with a blob analysis result buffer.

You can use [MblobControl\(\)](#) to change a processing setting associated with a result buffer.

Parameters

BlobResId

Specifies the identifier of a blob analysis result buffer.

InquireType

Specifies the type of processing setting about which to inquire. This parameter can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

For specifying the type of processing setting	
Value	Description
<input type="checkbox"/> M_BLOB_IDENTIFICATION +	<div>Returns the blob identification mode that was selected. (summarize)</div> <div>UserVarPtr info Return values: M_INDIVIDUAL; M_LABELED; M_LABELED_TOUCHING; M_WHOLE_IMAGE; (details)</div>
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_X +	<div>Returns the X-coordinate of the top left corner of the region in the blob analysis result buffer to use when drawing in the destination image buffer. (summarize)</div> <div>UserVarPtr info Return values: M_DEFAULT; Value; (details)</div>
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_Y +	<div>Returns the Y-coordinate of the top left corner of the region in the blob analysis result buffer to use when drawing in the destination image buffer. (summarize)</div> <div>UserVarPtr info Return values: M_DEFAULT; Value; (details)</div>
<input type="checkbox"/> M_FOREGROUND_VALUE +	<div>Returns the pixel values that are considered to be in the foreground. (summarize)</div> <div>UserVarPtr info Return values: M_NONZERO; M_ZERO; (details)</div>

<input type="checkbox"/> M_IDENTIFIER_TYPE +	<p>Returns the values that the non-zero pixels in the image have. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_BINARY; M_GRAYSCALE; (details)</p>
<input type="checkbox"/> M_INPUT_SELECT_UNITS +	<p>Returns the units in which the CondLow and CondHigh parameters of MblobSelect() expect the limits of the blob selection condition. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_PIXEL; M_WORLD; (details)</p>
<input type="checkbox"/> M_LATTICE +	<p>Returns the image lattice. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_4_CONNECTED; M_8_CONNECTED; (details)</p>
<input type="checkbox"/> M_MAX_LABEL +	<p>Returns the maximum label value given to any included blob in the result buffer. Label values for blobs are generated when a call to MblobCalculate() is made. The maximum label value is not necessarily the same as the total number of blobs because some label values might not be used, depending on the blob's shape. One of the uses for obtaining the maximum label value is to determine if an 8 or 16-bit image buffer is needed for MblobLabel(). (summarize)</p>
<input type="checkbox"/> M_NUMBER_OF_FERETS +	<p>Returns the number of Feret angles set to calculate a Feret feature. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_MIN_FERETS; Value > M_MIN_FERETS; (details)</p>
<input type="checkbox"/> M_OWNER_SYSTEM +	<p>Returns the identifier of the system on which the result buffer is allocated.</p>
<input type="checkbox"/> M_PIXEL_ASPECT_RATIO +	<p>Returns the pixel aspect ratio that you set for the image(s). (summarize)</p> <p><i>UserVarPtr info</i> Return values: Please see MblobControl() with M_PIXEL_ASPECT_RATIO. (details)</p>
<input type="checkbox"/> M_SAVE_RUNS +	<p>Returns whether the run information will be saved when calling MblobCalculate(). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_TIMEOUT +	<p>Returns the maximum processing time for MblobCalculate(), in msec. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DISABLE; Value >= 0; (details)</p>

Combination constants for any of the possible values of the [InquireType](#) parameter

You can add one of the following values to the above-mentioned values to set the requested results to a data type.

● For specifying the data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_DOUBLE	<p>Casts the requested information to a <i>double</i>. This is the default. (summarize)</p> <p><i>UserVarPtr info</i> Data type: double</p>
<input type="checkbox"/> M_TYPE_LONG	<p>Casts the requested information to a <i>long</i>.</p>

	(summarize)
	<div>UserVarPtr info</div> Data type: long
<input type="checkbox"/> M_TYPE_MIL_DOUBLE	<div>Casts the requested information to a <i>MIL_DOUBLE</i>.</div> (summarize)
	<div>UserVarPtr info</div> Data type: MIL_DOUBLE
<input type="checkbox"/> M_TYPE_MIL_INT	<div>Casts the requested information to a <i>MIL_INT</i>.</div> (summarize)
	<div>UserVarPtr info</div> Data type: MIL_INT
<input type="checkbox"/> M_TYPE_MIL_INT32	<div>Casts the requested information to a <i>MIL_INT32</i>.</div> (summarize)
	<div>UserVarPtr info</div> Data type: MIL_INT32
<input type="checkbox"/> M_TYPE_MIL_INT64	<div>Casts the requested information to a <i>MIL_INT64</i>.</div> (summarize)
	<div>UserVarPtr info</div> Data type: MIL_INT64

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• double• long• MIL_DOUBLE• MIL_INT• MIL_INT32• MIL_INT64

Specifies the address in which the inquiry result will be written.

Return value

Returns the requested information, cast to a *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobLabel

Synopsis

Draw a labeled image.

Syntax

```
void MblobLabel(
    MIL_ID BlobResId,
    MIL_ID DestImageBufId,
    MIL_INT Mode
)
```

Description

This function draws a labeled image in which each blob existing in the specified result buffer is represented with its own unique label value.

The label values are taken from the blob analysis result buffer. A call to [MblobCalculate\(\)](#) must have been made to generate label values for an image. Blobs that have been deleted from the result buffer are not drawn.

Parameters

- BlobResId**
- Specifies the identifier of the blob analysis result buffer.
- DestImageBufId**
- Specifies the identifier of the destination (labeled) image buffer. This must be a single band, 8 or 16-bit unsigned buffer.
- Note, this buffer need not be the same size as the original identifier image used to calculate the blobs, but must be 16 bits deep if the maximum label value exceeds 255. To determine if a 16-bit buffer is necessary, perform an [M_MAX_LABEL](#) inquiry, using [MblobInquire\(\)](#). The number of blobs alone does not tell you the maximum label value (label values are not necessarily contiguous).
- Mode**
- Specifies whether or not to clear the destination image buffer before drawing the labeled image into it. This parameter can be set to one of the following:

● For the destination image buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CLEAR	Clears the destination image buffer before placing the labeled image into it.
<input type="checkbox"/> M_NO_CLEAR	Does not clear the destination image buffer before placing the labeled image into it (background pixels will be unchanged).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobMerge

Synopsis

Merge the results of two blob result buffers.

Syntax

```
void MblobMerge (
    MIL_ID SourceResult1,
    MIL_ID SourceResult2,
    MIL_ID DestResult,
    MIL_INT ControlFlag
)
```

Description

This function merges the results of two blob result buffers into a single result buffer.

The source results are merged as if they were taken from two vertically adjacent child buffers of one image. As such, the destination result buffer uses the coordinate system of the first result buffer and positional results from the second result buffer are translated in the y-direction to take into account this coordinate system change. Border blobs that would touch if they were in one image are grouped into one grouped blob, assigned a single label, and recalculated as if the grouped blobs were one blob. Only features that were calculated for all the individual blobs in the group are recalculated for the new grouped blob; calculations are made using the results of the individual blobs in the group. See the [Merging results](#) section in [Chapter 6: Blob analysis](#) for more detailed explanations.

Note that after merging the results, you can perform a [MblobSelect\(\)](#) operation to further include or exclude blobs in your destination result buffer, but if you perform an [MblobCalculate\(\)](#) operation after the merge, all the results in the destination result buffer will be discarded.

The source blob results must:

- Both have the their run information saved or unsaved ([M_SAVE_RUNS](#) set to [M_ENABLE](#) or [M_DISABLE](#)).
- Have the same pixel aspect ratio ([M_PIXEL_ASPECT_RATIO](#)).
- Have the same lattice ([M_LATTICE](#)).
- Have the same blob identification mode ([M_BLOB_IDENTIFICATION](#)). If the blob result buffers have [M_BLOB_IDENTIFICATION](#) set to [M_LABELED](#), they cannot be merged.
- Have the same number of Ferets ([M_NUMBER_OF_FERETS](#)).

Furthermore, if you specify a sorting key for result retrieval, the sorting order will not be respected after the merge.

Parameters

SourceResult1

Specifies the identifier of the first source result buffer. [SourceResult1](#) and [SourceResult2](#) cannot be the same.

SourceResult2

Specifies the identifier of the second source result buffer. [SourceResult1](#) and [SourceResult2](#) cannot be the same.

DestResult

Specifies the identifier of the destination result buffer. [DestResult](#) cannot be the same as [SourceResult1](#) or [SourceResult2](#).

ControlFlag

Specifies the control settings for the merge operation. This parameter can be set to any combination of the following.

● For specifying the type of merge operation	

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default control operation. The default combination is M_TOP_BOTTOM + M_COPY , in which the source results are copied into the destination result buffer, and the orientation of the merged result will be as if the source results were taken from two vertically adjacent child buffers of one image. (summarize)
<input type="checkbox"/> M_COPY	Copies the results of SourceResult1 and SourceResult2 into the destination result buffer. After the merge, the original results of SourceResult1 and SourceResult2 remain intact. (summarize)
<input type="checkbox"/> M_MOVE	Moves the results of the source result buffers into the destination result buffer. After the merge, the original results of SourceResult1 and SourceResult2 are emptied. (summarize)
<input type="checkbox"/> M_TOP_BOTTOM	Specifies that the orientation of the merged result will be as if the source results were taken from two vertically adjacent child buffers of one image.

Remark

- Note that if you add a specified moment to the feature list (using [MblobSelectMoment\(\)](#) with [M_GRAYSCALE](#) + [M_CENTRAL](#), or with [M_BINARY](#) + [M_CENTRAL](#) and [M_SAVE_RUNS](#) disabled), [M_MERGE](#) will not be able to perform the moment calculation.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobReconstruct

Synopsis

Reconstruct blobs (or blob holes) in an image buffer.

Syntax

```
void MblobReconstruct(
    MIL_ID SrcImageBufId,
    MIL_ID SeedImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT Operation,
    MIL_INT ProcMode
)
```

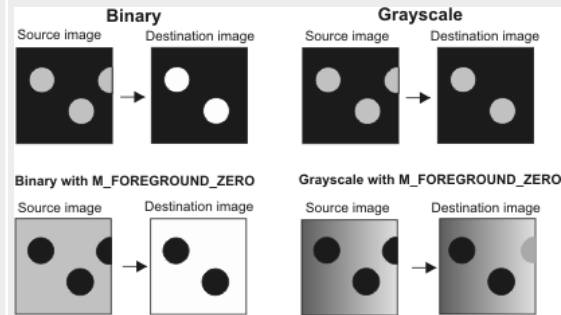
Description

This function copies (or reconstructs) blobs or blob holes from the source to the destination buffer, according to the specified operation and processing mode. By default, all non-zero pixels in the source buffer are considered to be part of a blob. Use the [M_FOREGROUND_ZERO](#) processing mode to inverse this behavior.

Parameters

- SrcImageBufId
- Specifies the identifier of the source image buffer. The source buffer must be a single band, packed binary, 8- or 16-bit unsigned buffer.
- SeedImageBufId
- Specifies the identifier of the image buffer to use as a seed image. The seed image buffer must be a single band, packed binary, 8- or 16-bit unsigned buffer.
- A seed image is needed to perform an [M_RECONSTRUCT_FROM_SEED](#) type of operation. For any other operation type, set this parameter to [M_NULL](#).
- DestImageBufId
- Specifies the identifier of the destination (processed blobs) image buffer. The destination buffer must be a single band, packed binary, 8- or 16-bit unsigned buffer.
- Operation
- Specifies the type of operation to perform. This parameter can be set to one of the following values:

● For specifying the type of operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ERASE_BORDER_BLOBS	All blobs that do not touch the borders of the source image are copied to the destination image buffer according to the selected processing mode. Pixels of the blobs that touch the border are replaced according to the selected processing mode.

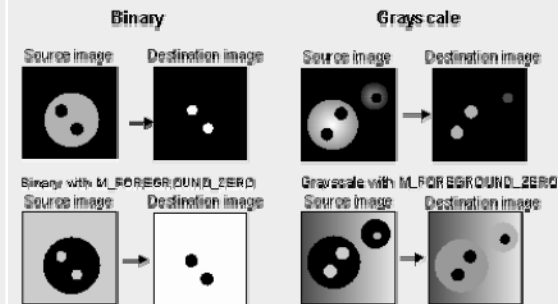


Note that when the [M_GRAYSCALE](#) and [M_FOREGROUND_ZERO](#) processing modes are selected, the border blobs are filled with the average grayscale pixel value of the background. This operation is similar to a "border kill".

([summarize](#))

☐ [M_EXTRACT_HOLES](#)

All holes within the blobs of the source buffer are copied to the destination buffer with their pixel values set to their corresponding blob's pixel values, according to the selected processing mode. A hole cannot touch any image border to be considered a hole.

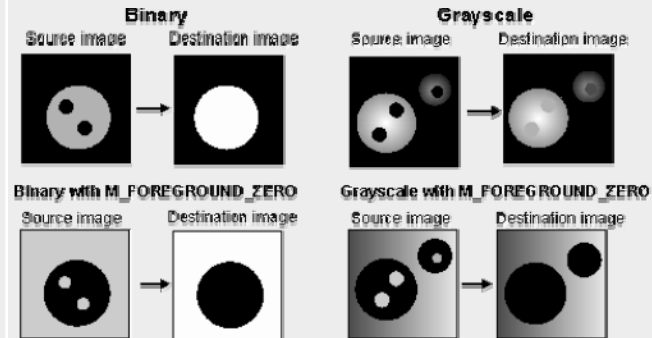


Note that, in the [M_GRAYSCALE](#) processing mode, the pixel values of holes copied to the destination buffer are set to the blob's average grayscale pixel value in the source buffer. Also, when the [M_GRAYSCALE](#) and [M_FOREGROUND_ZERO](#) processing modes are selected, the blobs are filled with the average grayscale pixel value of the background.

([summarize](#))

☐ [M_FILL_HOLES](#)

All blobs in the source buffer are copied to the destination buffer according to the selected processing mode, and those blobs with holes are filled according to the processing mode. A hole must not touch the border of the image to be considered a hole.



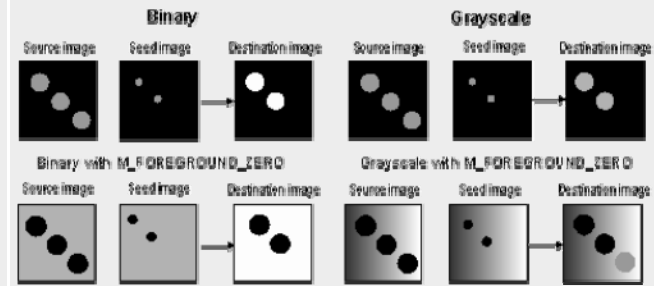
Note that when the [M_GRAYSCALE](#) processing mode is selected, holes are filled with the average grayscale pixel value of the corresponding blob.

([summarize](#))

☐ [M_RECONSTRUCT_FROM_SEED](#)

All blobs in the source buffer that have at least one corresponding foreground seed pixel in the seed buffer are copied to the destination buffer, according to the

selected processing mode. Blobs that are not seeded are replaced according to the selected processing mode.



Note that when the [M_GRAYSCALE](#) and [M_FOREGROUND_ZERO](#) processing modes are selected, blobs in the source image that are not seeded are filled with the average grayscale value of the background. Also, the value of the seed pixels must be strictly zero for this operation to be performed properly.

Note that the seed image and the destination image must have identical dimensions.

([summarize](#))

ProcMode

Specifies the processing mode to use.

This parameter should be set to one of the following values:

● For specifying the processing mode to use	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT +	Specifies the default processing mode (which corresponds to M_BINARY + M_8_CONNECTED).
<input type="checkbox"/> M_BINARY +	Specifies that non-zero pixel values will be treated as ones (1) during processing, and the resulting non-zero pixels copied to the destination buffer will be set to the maximum value of that buffer (for example, 0xff for an 8-bit buffer). Note, in general, the M_BINARY processing mode is faster. (summarize)
<input type="checkbox"/> M_GRAYSCALE +	Specifies that the values of pixels copied to the destination buffer will be changed to the values of corresponding pixels in the source buffer.

Combination constants for the values listed in [For specifying the processing mode to use](#)

You can add one of the following values to the above-mentioned values to set the type of connectivity (lattice).

● For selecting the type of connectivity	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_4_CONNECTED	Specifies that blobs are computed on a four connected lattice.
<input type="checkbox"/> M_8_CONNECTED	Specifies that blobs are computed on an eight connected lattice. This is the default value. (summarize)

Combination constant for the values listed in [For specifying the processing mode to use](#)

You can add the following value to the above-mentioned values to set the value of the foreground pixels.

● For selecting the foreground pixels	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_FOREGROUND_ZERO	Specifies that the pixel values of blobs will consist of zero values and the pixels of the background will consists of non-zero values; that is, the inverse of the usual blob pixel value definition.
--------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Combination constant for the values listed in [For specifying the processing mode to use](#)
You can add the following value to the above-mentioned values to set whether to accelerate blob reconstruction with seed images.

● For accelerating blob reconstruction with seed images	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SEED_PIXELS_ALL_IN_BLOBS	Optimizes the reconstruction process. Use this value in cases when all seed pixels in the seed buffer have corresponding blob pixels in the source buffer. This condition often exists when the seed buffer is an eroded (see MimErode()) version of the source buffer. (summarize)

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobSelect

Synopsis

Select blobs for calculations and result retrieval.

Syntax

```
void MblobSelect(
    MIL_ID BlobResId,
    MIL_INT Operation,
    MIL_INT SelectionCriterion,
    MIL_INT Condition,
    MIL_DOUBLE CondLow,
    MIL_DOUBLE CondHigh
)
```

Description

This function selects or merges blobs that meet a specified criterion. These blobs will be included in or excluded from future operations (calculations or result retrieval), or deleted entirely from the result buffer. Selection criterion can be based on a calculated feature or on the current status of the blobs. [MblobCalculate\(\)](#) must have been called at least once to call this function.

If this function is not called at least once, all blobs are included by default. If there is more than one call to this function, the effect of the calls is cumulative unless [M_INCLUDE_ONLY](#) or [M_EXCLUDE_ONLY](#) is specified as the operation to perform.

Once a blob has been excluded, it can normally be re-included only by specifying [M_INCLUDE](#) or [M_INCLUDE_ONLY](#) in a future call to this function (with the correct criterion). However, if you change the processing mode of a result buffer (with [MblobControl\(\)](#)), or use the result buffer with different images (in a call to [MblobCalculate\(\)](#)), all results in the buffer are discarded and all blobs are re-included.

The limits of the blob selection criterion are set in [CondLow](#) and [CondHigh](#) parameters. If the blobs were taken from a calibrated image, these parameters can be set in pixel units or world units depending on [M_INPUT_SELECT_UNITS](#) control type setting in [MblobControl\(\)](#).

Parameters

BlobResId

Specifies the identifier of the blob analysis result buffer to use in the blob selection process.

Operation

Specifies the operation to perform on the specified blobs. This parameter can be set to one of the following.

● For operations on blobs	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DELETE	Deletes blobs that meet the specified condition. M_DELETE affects only included blobs, unless otherwise stated by the SelectionCriterion parameter. M_DELETE removes blobs permanently from the result buffer and, consequently, prevents these blobs from being re-included. (summarize)
<input type="checkbox"/> M_EXCLUDE	Excludes all blobs that meet the specified condition. M_EXCLUDE affects only the status of currently included blobs. (summarize)
<input type="checkbox"/> M_EXCLUDE_ONLY	Excludes only those blobs that meet the specified condition and includes all others. The exclusion does not consider the present status of blobs (whether they are excluded), except for blobs that have been deleted (M_DELETE), unless otherwise stated by the SelectionCriterion parameter. (summarize)

<input type="checkbox"/> M_INCLUDE	Includes all blobs that meet the specified condition. M_INCLUDE affects only the status of currently excluded blobs. (summarize)
<input type="checkbox"/> M_INCLUDE_ONLY	Includes only those blobs that meet the specified condition and excludes all others. The inclusion <i>does not</i> consider the present status of blobs (whether they are included), except for blobs that have been deleted (M_DELETE), unless otherwise stated by the SelectionCriterion parameter. (summarize)
<input type="checkbox"/> M_MERGE	Groups all the blobs that meet the specified condition. Once grouped, the blobs are treated as one blob (a grouped blob). The grouped blob is assigned a unique blob label. Only features that were calculated for all the individual blobs in the group are re-calculated for the new grouped blob; calculations are made using the results of the individual blobs in the group. Note that if you add a specified moment to the feature list (using MblobSelectMoment() with M_GRAYSCALE + M_CENTRAL , or with M_BINARY + M_CENTRAL and M_SAVE_RUNS disabled), M_MERGE will not be able to perform the moment calculation. Except when using M_EXCLUDED_BLOBS as the feature for selection, this grouping operation only applies to included blobs. If you are using M_MERGE , you cannot use MblobControl() with M_BLOB_IDENTIFICATION set to M_LABELED_TOUCHING . (summarize)

SelectionCriterion

Specifies the feature to use as part of the selection criterion or the status of the blobs to affect. See [MblobSelectFeature\(\)](#), [MblobSelectFeret\(\)](#), and [MblobSelectMoment\(\)](#) for a full description of the features. The specified result buffer must already contain the results for the specified feature.

To use as part of the criterion a feature that has only a binary definition, select one of the following.

● For specifying the feature to use	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_AREA	Uses the number of foreground pixels in a blob (holes are not counted).
<input type="checkbox"/> M_BOX_X_MAX	Uses the extreme right coordinate of the bounding box of a blob.
<input type="checkbox"/> M_BOX_X_MIN	Uses the extreme left coordinate of the bounding box of a blob.
<input type="checkbox"/> M_BOX_Y_MAX	Uses the extreme bottom coordinate of the bounding box of a blob.
<input type="checkbox"/> M_BOX_Y_MIN	Uses the extreme top coordinate of the bounding box of a blob.
<input type="checkbox"/> M_BREADTH	Uses the breadth of a blob.
<input type="checkbox"/> M_COMPACTNESS	Uses the compactness feature.
<input type="checkbox"/> M_CONVEX_PERIMETER	Uses the approximation of the perimeter of the convex hull of a blob.
<input type="checkbox"/> M_ELONGATION	Uses the elongation of a blob.
<input type="checkbox"/> M_EULER_NUMBER	Uses the number of blobs - number of holes.
<input type="checkbox"/> M_FERET_ELONGATION	Uses the measure of the shape of a blob. It is equal to $\text{M_FERET_MAX_DIAMETER} / \text{M_FERET_MIN_DIAMETER}$. (summarize)
<input type="checkbox"/> M_FERET_MAX_ANGLE	Uses the angle at which the maximum Feret diameter is found, in degrees, with positive values indicating a counter-clockwise displacement from the positive X-axis.
<input type="checkbox"/> M_FERET_MAX_DIAMETER	Uses the largest Feret diameter found after checking a certain number of angles.
<input type="checkbox"/> M_FERET_MEAN_DIAMETER	Uses the average Feret diameter at all the angles checked.
<input type="checkbox"/> M_FERET_MIN_ANGLE	Uses the angle at which the minimum Feret diameter is found, in degrees, with positive values indicating a counter-clockwise displacement from the positive X-axis.
<input type="checkbox"/> M_FERET_MIN_DIAMETER	Uses the smallest Feret diameter found after checking a certain number of angles.
<input type="checkbox"/> M_FERET_X	Uses the dimension of the minimum bounding box of a blob in the horizontal direction.
<input type="checkbox"/> M_FERET_Y	Uses the dimension of the minimum bounding box of a blob in the vertical direction.
<input type="checkbox"/> M_FIRST_POINT_X	Uses the x-coordinate of the top left-most pixel along the perimeter of a blob.

<input type="checkbox"/> M_FIRST_POINT_Y	Uses the y-coordinate of the top left-most pixel along the perimeter of a blob.
<input type="checkbox"/> M_GENERAL_FERET	Uses the Feret diameter feature.
<input type="checkbox"/> M_GENERAL_MOMENT	Uses the moment calculation.
<input type="checkbox"/> M_INTERCEPT_0	Uses the number of times a transition from background to foreground (not vice versa) occurs in the horizontal direction for the entire blob. In other words, it is equal to the number of times the neighborhood configuration [B, F] occurs in a blob, where B is a background pixel and F is a foreground pixel. (summarize)
<input type="checkbox"/> M_INTERCEPT_45	Uses the number of times that the neighborhood configuration $\begin{bmatrix} \bullet & F \\ B & \bullet \end{bmatrix}$ occurs in a blob, where F is a foreground pixel, B is a background pixel and a dot can be any pixel value.
<input type="checkbox"/> M_INTERCEPT_90	Uses the number of times that the neighborhood configuration $\begin{bmatrix} F \\ B \end{bmatrix}$ occurs in a blob.
<input type="checkbox"/> M_INTERCEPT_135	Uses the number of times that the neighborhood configuration $\begin{bmatrix} F & \bullet \\ \bullet & B \end{bmatrix}$ occurs in a blob.
<input type="checkbox"/> M_LABEL_VALUE	Uses the label feature.
<input type="checkbox"/> M_LENGTH	Uses the measure of the true length of a blob.
<input type="checkbox"/> M_NUMBER_OF_CHAINED_PIXELS	Uses the number of chained pixels in a blob.
<input type="checkbox"/> M_NUMBER_OF_HOLES	Uses the number of holes in a blob.
<input type="checkbox"/> M_NUMBER_OF_RUNS	Uses the total number of runs in a blob.
<input type="checkbox"/> M_PERIMETER	Uses the total length of edges in a blob (including the edges of any holes), with an allowance made for the staircase effect that is produced when diagonal edges are digitized.
<input type="checkbox"/> M_ROUGHNESS	Uses the roughness feature.
<input type="checkbox"/> M_X_MAX_AT_Y_MAX	Uses the maximum X-coordinate at the maximum Y-coordinate of the blob.
<input type="checkbox"/> M_X_MIN_AT_Y_MIN	Uses the minimum X-coordinate at the minimum Y-coordinate of the blob.
<input type="checkbox"/> M_Y_MAX_AT_X_MIN	Uses the maximum Y-coordinate at the minimum X-coordinate of the blob.
<input type="checkbox"/> M_Y_MIN_AT_X_MAX	Uses the minimum Y-coordinate at the maximum X-coordinate of the blob.

To use as part of the criterion a feature that has only a grayscale definition, select one of the following. You can specify one of these features as part of your selection criterion only if both a blob identifier image and a grayscale image were passed to [MblobCalculate\(\)](#) (and the features were selected for calculation).

● For features that have only a grayscale definition	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MAX_PIXEL	Uses the maximum pixel value in a blob.
<input type="checkbox"/> M_MEAN_PIXEL	Uses the mean pixel value in a blob.
<input type="checkbox"/> M_MIN_PIXEL	Uses the minimum pixel value in a blob.
<input type="checkbox"/> M_SIGMA_PIXEL	Uses the standard deviation of pixel values in a blob.
<input type="checkbox"/> M_SUM_PIXEL	Uses the sum of all pixel values in a blob.
<input type="checkbox"/> M_SUM_PIXEL_SQUARED	Uses the sum of the squares of the pixel values in a blob.

To use as part of the criterion a feature that has two different definitions (a binary and a grayscale definition), select one of the following.

If you did not provide both a blob identifier image and a grayscale image to [MblobCalculate\(\)](#), only the binary version was calculated. If you did provide a grayscale image, both versions were calculated, unless otherwise specified. In the latter case, see combination values below to specify which to use as part of the criterion. If both versions were calculated and no version is specified, then the grayscale version of the feature is selected as criterion.

● For a feature that has two different definitions	
Value	Description
<input type="checkbox"/> M_AXIS_PRINCIPAL_ANGLE +	Uses the feature specifying the angle at which a blob has the least moment of inertia.
<input type="checkbox"/> M_AXIS_SECONDARY_ANGLE +	Uses the feature specifying the angle perpendicular to M_AXIS_PRINCIPAL_ANGLE .
<input type="checkbox"/> M_CENTER_OF_GRAVITY_X +	Uses the feature specifying the X-position of the center of gravity of a blob.
<input type="checkbox"/> M_CENTER_OF_GRAVITY_Y +	Uses the feature specifying the Y-position of the center of gravity of a blob.
<input type="checkbox"/> M_MOMENT_CENTRAL_Xn_Ym +	<p>Uses the feature specifying the calculation of central moments.</p> <p>Central moments have the syntax M_MOMENT_CENTRAL_Xn_Ym, and are defined as $\sum_i x_i^n y_i^m p_i$, where p_i = value of a pixel (always 1 for binary moments), x_i = its X-coordinate and y_i = its Y-coordinate. For central moments, coordinates are relative to each blob's center of gravity, as opposed to ordinary moments, which use coordinates that are relative to the image origin (top-left corner). Calculate higher moments by calling MblobSelectMoment(). (summarize)</p>
<input type="checkbox"/> M_MOMENT_Xn_Ym +	<p>Uses the feature specifying the calculation of ordinary moments.</p> <p>Ordinary moments have the syntax M_MOMENT_Xn_Ym, and are defined as $\sum_i x_i^n y_i^m p_i$, where p_i = value of a pixel (always 1 for binary moments), x_i = its X-coordinate and y_i = its Y-coordinate. For ordinary moments, coordinates are relative to the image origin (top-left corner), as opposed to central moments, which use coordinates that are relative to each blob's center of gravity. Calculate higher moments by calling MblobSelectMoment(). (summarize)</p>

Combination constants for the values listed in [For a feature that has two different definitions](#)

You can add one of the following values to the above-mentioned values to set whether to use the binary or grayscale version of the selected feature.

Note that the selected feature must have both a binary definition and a grayscale definition, and both versions must have been calculated.

● For features with a binary and grayscale definition (both have been calculated)	
Value	Description
<input type="checkbox"/> M_BINARY	Uses the binary version of the selected feature.
<input type="checkbox"/> M_GRAYSCALE	<p>Uses the grayscale version of the selected feature.</p> <p>This is the default value.</p> <p>(summarize)</p>

To use as part of the selection criterion the current status of the blobs, set the [SelectionCriterion](#) parameter to one of the following values. For example, you can delete all currently excluded blobs from the list of blobs on which to operate.

● For using as part of the selection criterion	
Value	Description
<input type="checkbox"/> M_ALL_BLOBS	Uses all blobs.
<input type="checkbox"/> M_EXCLUDED_BLOBS	Uses all excluded blobs.
<input type="checkbox"/> M_INCLUDED_BLOBS	Uses all included blobs.

Condition

Specifies the condition for the blob selection. This parameter must be set to one of the values below.

When the selection criterion is based on the current status of blobs, set the [Condition](#) parameter to **M_NULL**.

When the selection criterion is based on a feature, you can specify one of the following conditions that use two limits ([CondLow](#) and [CondHigh](#)).

● For the blob selection	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN_RANGE	Specifies that blobs with values for the specified feature in the range CondLow to CondHigh , inclusive, are included, excluded, or deleted from future operations on the specified result buffer.
<input type="checkbox"/> M_OUT_RANGE	Specifies that blobs with values for the specified feature less than CondLow , or greater than CondHigh , are included, excluded, or deleted from future operations on the specified result buffer.

When the selection criterion is based on a feature, you can specify one of the following conditions that use only one limit ([CondLow](#)).

● For selection criterion based on a feature	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EQUAL	Specifies that blobs with values for the specified feature equal to CondLow are included, excluded, or deleted from future operations on the specified result buffer.
<input type="checkbox"/> M_GREATER	Specifies that blobs with values for the specified feature greater than CondLow are included, excluded, or deleted from future operations on the specified result buffer.
<input type="checkbox"/> M_GREATER_OR_EQUAL	Specifies that blobs with values for the specified feature greater than or equal to CondLow are included, excluded, or deleted from future operations on the specified result buffer.
<input type="checkbox"/> M_LESS	Specifies that blobs with values for the specified feature less than CondLow are included, excluded, or deleted from future operations on the specified result buffer.
<input type="checkbox"/> M_LESS_OR_EQUAL	Specifies that blobs with values for the specified feature less than or equal to CondLow are included, excluded, or deleted from future operations on the specified result buffer.
<input type="checkbox"/> M_NOT_EQUAL	Specifies that blobs with values for the specified feature not equal to CondLow are included, excluded, or deleted from future operations on the specified result buffer.

CondLow

Specifies the lower limit of the selected condition.

● For the lower limit of the selected condition	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NULL	Specifies that this parameter is not applicable. Specify this value when and only when the selection criterion is based on the status of blobs. (summarize)
<input type="checkbox"/> Value	Specifies the lower limit of the condition. If the blobs were taken from a calibrated image, specify this value in in pixel units or world units depending on M_INPUT_SELECT_UNITS control type setting in MblobControl() . (summarize)

CondHigh

Specifies the upper limit of the selected condition.

● For the upper limit of the selected condition	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NULL	Specifies that this parameter is not applicable. Specify this value when the condition uses only one limit or when the selection criterion is based on the status of blobs. (summarize)

<input type="checkbox"/> Value	Specifies the upper limit of the condition. If the blobs were taken from a calibrated image, specify this value in in pixel units or world units depending on M_INPUT_SELECT_UNITS control type setting in MblobControl() . (summarize)
--------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobSelectFeature

Synopsis

Select feature(s) to be calculated.

Syntax

```
void MblobSelectFeature(
    MIL_ID FeatureListId,
    MIL_INT Feature
)
```

Description

This function selects the feature(s) to be calculated by [MblobCalculate\(\)](#) when using the specified feature list.

Calculations for binary features are performed using the blob identifier image. Grayscale feature calculations are performed using both the blob identifier image and grayscale image. Features that have both binary and grayscale definitions are calculated using both the blob identifier image and the grayscale image (see [MblobCalculate\(\)](#)).

You can get the feature results of a specific blob using [MblobGetResultSingle\(\)](#). To get the feature results of all blobs, use [MblobGetResult\(\)](#). The results of the features [M_CHAIN_INDEX](#), [M_CHAIN_X](#), and [M_CHAIN_Y](#) are only available with [MblobGetResultSingle\(\)](#). All other feature results are available with both [MblobGetResultSingle\(\)](#) and [MblobGetResult\(\)](#).

If your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image. When calculated in pixel units, the pixel aspect ratio, specified with [MblobControl\(\)](#), is taken into account horizontally. Results are returned in units of "pixel height" since the pixel width is adjusted to be equal to the pixel height.

Note that you can change the number of Feret angles, for those features requiring them in calculations, using [MblobControl\(\)](#).

Parameters

FeatureListId

Specifies the identifier of the feature list buffer.

Feature

Specifies the feature to add to the feature list. To add several features, you must call this function for each feature you want to add to the list (certain commonly used groups of features can be selected in a single call).

The following group of features do not use grayscale pixel values; they are calculated using only the blob identifier image. Note that, unless otherwise stated, features can be calculated in either pixel or calibrated units.

● For specifying the feature to add	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_AREA +	Determines the number of foreground pixels in a blob (holes are not counted).
<input type="checkbox"/> M_BOX_X_MAX +	Determines the extreme right coordinate of a blob.
<input type="checkbox"/> M_BOX_X_MIN +	Determines the extreme left coordinate of a blob.
<input type="checkbox"/> M_BOX_Y_MAX +	Determines the extreme bottom coordinate of a blob.
<input type="checkbox"/> M_BOX_Y_MIN +	Determines the extreme top coordinate of a blob.
<input type="checkbox"/> M_BREADTH +	Determines a measure of the true breadth of an object, with the same advantages and disadvantages as M_LENGTH .
<input type="checkbox"/> M_CHAIN_INDEX +	Determines the indices which differentiate each chain's pixels within a specified blob. The index of the blob's outermost chain is 1. Chains that delimit holes in blobs are identified by subsequent indices for each chain.

	(summarize)
<input type="checkbox"/> M_CHAIN_X +	<p>Determines the X-coordinate of each chained pixel in the specified blob, for all chains contained within the blob (both the pixels bordering a blob and those delimiting its holes). Chained pixels always form a closed chain. This implies that the starting pixel in the chain is also the closing one. If your blob has regions which are 1 pixel wide, these pixels are chained twice, once in the forward direction and then in the opposite direction.</p> <p>Blobs with holes have multiple chains. To determine the chain to which this pixel's X-coordinate belongs, use M_CHAIN_INDEX.</p> <p>(summarize)</p>
<input type="checkbox"/> M_CHAIN_Y +	<p>Determines the Y-coordinate of each chained pixel in the specified blob, for all chains contained within the blob (both the pixels bordering a blob and those delimiting its holes). Chained pixels always form a closed chain. This implies that the starting pixel in the chain is also the closing one. If your blob has regions which are 1 pixel wide, these pixels are chained twice, once in the forward direction and then in the opposite direction.</p> <p>Blobs with holes have multiple chains. To determine the chain to which this pixel's Y-coordinate belongs, use M_CHAIN_INDEX.</p> <p>(summarize)</p>
<input type="checkbox"/> M_COMPACTNESS +	<p>Determines the minimum value for a circle (1.0) and is derived from the perimeter (p) and area (A). The more convoluted the shape, the greater the value. It is equal to $\frac{p^2}{(4\pi A)}$.</p> <p>(summarize)</p>
<input type="checkbox"/> M_CONVEX_PERIMETER +	<p>Determines the approximation of the perimeter of the convex hull of a blob. It is derived from several Feret diameters; so, a larger number of Ferets gives a more accurate result.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ELONGATION +	<p>Determines a value that is equal to M_LENGTH / M_BREADTH. It is similar to M_FERET_ELONGATION, except that it should be used for long thin objects.</p> <p>(summarize)</p>
<input type="checkbox"/> M_EULER_NUMBER +	<p>Determines the number of blobs - number of holes. This value is more useful for M_WHOLE_IMAGE than for M_INDIVIDUAL processing mode.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_ELONGATION +	<p>Determines the measure of the shape of a blob. It is equal to M_FERET_MAX_DIAMETER / M_FERET_MIN_DIAMETER.</p> <p>It is accurate for reasonably compact objects, but becomes less accurate for very elongated objects (because M_FERET_MIN_DIAMETER becomes less accurate). For very elongated objects, you should probably use M_ELONGATION.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_MAX_ANGLE +	<p>Determines the angle at which the maximum Feret diameter is found. The value is in degrees, with positive values indicating a counter-clockwise displacement from the positive X-axis.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_MAX_DIAMETER +	<p>Determines the largest Feret diameter found after checking a certain number of angles. More angles will give a more accurate result, but will take longer to calculate. However, the maximum Feret diameter is not very sensitive to the number of angles, and 8 usually gives an accurate result.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_MEAN_DIAMETER +	<p>Determines the average Feret diameter at all the angles checked.</p>
<input type="checkbox"/> M_FERET_MIN_ANGLE +	<p>Determines the angle at which the minimum Feret diameter is found. The value is in degrees, with positive values indicating a counter-clockwise displacement from the positive X-axis.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_MIN_DIAMETER +	<p>Determines the smallest Feret diameter found after checking a certain number of angles. More angles will give a more accurate result, but will take longer to calculate. Note that this feature will not be very accurate for long thin blobs. However, you can get an accurate measure of the breadth of long thin blobs more quickly by using M_BREADTH.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_X +	<p>Determines the dimension of the minimum bounding box of a blob in the horizontal direction; that is, M_BOX_X_MAX - M_BOX_X_MIN + 1.</p>
<input type="checkbox"/> M_FERET_Y +	<p>Determines the dimension of the minimum bounding box of a blob in the vertical direction; that is, M_BOX_Y_MAX - M_BOX_Y_MIN + 1.</p>
<input type="checkbox"/> M_FIRST_POINT_X +	<p>Determines (along with M_FIRST_POINT_Y) a unique point for each object, that is always on the perimeter of the object. The X-coordinate is that of the left-most pixel on the top-most line of the object.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FIRST_POINT_Y +	<p>Determines (along with M_FIRST_POINT_X) a unique point for each object, that is always on the perimeter of the object. The Y-coordinate is that of the topmost line of the object.</p> <p>(summarize)</p>
<input type="checkbox"/> M_INTERCEPT_0 +	<p>Determines the number of times a transition from background to foreground (not vice versa) occurs in the horizontal direction for the entire blob. In other words, it is equal to the number of times the neighborhood configuration [B, F] occurs in a blob, where B is a background pixel and F is a foreground pixel.</p> <p>(summarize)</p>

<input type="checkbox"/> M_INTERCEPT_45 +	Determines the number of times that the neighborhood configuration $\begin{bmatrix} \bullet & F \\ B & \bullet \end{bmatrix}$ occurs in a blob, where F is a foreground pixel, B is a background pixel and a dot can be any pixel value.
<input type="checkbox"/> M_INTERCEPT_90 +	Determines the number of times that the neighborhood configuration $\begin{bmatrix} F \\ B \end{bmatrix}$ occurs in a blob.
<input type="checkbox"/> M_INTERCEPT_135 +	Determines the number of times that the neighborhood configuration $\begin{bmatrix} F & \bullet \\ \bullet & B \end{bmatrix}$ occurs in a blob.
<input type="checkbox"/> M_LABEL_VALUE +	Determines the label value for each blob in an image. This is a positive integer ($>= 1$) that is unique for each blob. This feature is always calculated; you do not need to select it. (summarize)
<input type="checkbox"/> M_LENGTH +	Determines the measure of the true length of an object, although it can only be applied to certain object types because it is derived from the perimeter (P) and area (A) assuming that $P = 2(\text{length} + \text{breadth})$ and $A = \text{length} \times \text{breadth}$. It complements M_FERET_MAX_DIAMETER because it is accurate for different blob types (for example, long thin ones). Note, it is calculated much faster than the maximum Feret diameter. (summarize)
<input type="checkbox"/> M_NUMBER_OF_CHAINED_PIXELS +	Determines the number of chained pixels for all blobs or a specified blob.
<input type="checkbox"/> M_NUMBER_OF_HOLES +	Determines a value that is equal to the number of holes in a blob. Holes that intersect the edge of the image are not counted (they might not be holes). This value is equal to $1 - \text{M_EULER_NUMBER}$ and is therefore a true hole count only in M_INDIVIDUAL processing mode. (summarize)
<input type="checkbox"/> M_NUMBER_OF_RUNS +	Determines a value that is equal to the total number of runs in a blob. A run is defined as a horizontal string of consecutive foreground pixels. (summarize)
<input type="checkbox"/> M_PERIMETER +	Determines the total length of edges in a blob (including the edges of any holes), with an allowance made for the staircase effect that is produced when diagonal edges are digitized (inside corners are counted as 1.414, rather than 2.0). A single pixel blob (area = 1) has a perimeter of 4.0. (summarize)
<input type="checkbox"/> M_ROUGHNESS +	Determines a measure of how rough a blob is and is equal to $\text{M_PERIMETER} / \text{M_CONVEX_PERIMETER}$. A smooth convex object will have the minimum roughness of 1.0. (summarize)
<input type="checkbox"/> M_X_MAX_AT_Y_MAX +	Determines the maximum X-coordinate at the maximum Y-coordinate of the blob. Together with M_BOX_Y_MAX , determines one of the contact points on the convex perimeter of the object. (summarize)
<input type="checkbox"/> M_X_MAX_AT_Y_MIN +	Determines the maximum X-coordinate at the minimum Y-coordinate of the blob. Together with M_BOX_Y_MIN , determines one of the contact points on the convex perimeter of the object. (summarize)
<input type="checkbox"/> M_X_MIN_AT_Y_MAX +	Determines the minimum X-coordinate at the maximum Y-coordinate of the blob. Together with M_BOX_Y_MAX , determines one of the contact points on the convex perimeter of the object. (summarize)
<input type="checkbox"/> M_X_MIN_AT_Y_MIN +	Determines the minimum X-coordinate at the minimum Y-coordinate of the blob. Together with M_BOX_Y_MIN , determines one of the contact points on the convex perimeter of the object. (summarize)
<input type="checkbox"/> M_Y_MAX_AT_X_MAX +	Determines the maximum Y-coordinate at the maximum X-coordinate of the blob. Together with M_BOX_X_MAX , determines one of the contact points on the convex perimeter of the object. (summarize)
<input type="checkbox"/> M_Y_MAX_AT_X_MIN +	Determines the maximum Y-coordinate at the minimum X-coordinate of the blob. Together with M_BOX_X_MIN , determines one of the contact points on the convex perimeter of the object. (summarize)
<input type="checkbox"/> M_Y_MIN_AT_X_MAX +	Determines the minimum Y-coordinate at the maximum X-coordinate of the blob. Together with M_BOX_X_MAX , determines one of the contact points on the convex perimeter of the object. (summarize)
<input type="checkbox"/> M_Y_MIN_AT_X_MIN +	Determines the minimum Y-coordinate at the minimum X-coordinate of the blob. Together with M_BOX_X_MIN , determines one of the contact points on the

convex perimeter of the object.
([summarize](#))

The following features require grayscale pixel values, and can only be calculated if you provide a grayscale image.

● For features with grayscale pixel values	
☐ Value	Description
☐ M_MAX_PIXEL +	Determines the maximum pixel value found in a blob.
☐ M_MEAN_PIXEL +	Determines the mean pixel value in a blob. It is equal to M_SUM_PIXEL / M_AREA . (summarize)
☐ M_MIN_PIXEL +	Determines the minimum pixel value found in a blob.
☐ M_SIGMA_PIXEL +	Determines the standard deviation of pixel values in a blob. It is equal to $\sqrt{\frac{\sum p_i^2 - \left(\frac{\sum p_i}{N}\right)^2}{N}}$, where N = number of pixels and p = pixel value. (summarize)
☐ M_SUM_PIXEL +	Determines the sum of all pixel values in a blob.
☐ M_SUM_PIXEL_SQUARED +	Determines the sum of the squares of each pixel value in a blob.

The following features have two different definitions: a binary definition, where all pixels are considered equal, and a grayscale, where pixels are weighted by their value in the gray image (the grayscale version is much slower to calculate). If you do not provide a grayscale image, only the binary version can be calculated. If you do provide a grayscale image, both versions are calculated.

● For features with two definitions	
☐ Value	Description
☐ M_AXIS_PRINCIPAL_ANGLE +	Determines the angle at which a blob has the least moment of inertia. For elongated blobs, it is aligned with the longest axis. The result is always between -90° and +90°, measured in a counter-clockwise direction from the positive X-axis. It is calculated as: $-0.5 + \arctan \frac{(2 * M_MOMENT_CENTRAL_X1_Y1)}{M_MOMENT_CENTRAL_X2_Y0 - M_MOMENT_CENTRAL_X0_Y2}$ When the blob identifier image is calibrated this feature is calculated in calibrated units; otherwise they are calculated in pixel units. (summarize)
☐ M_AXIS_SECONDARY_ANGLE +	Determines the angle perpendicular to M_AXIS_PRINCIPAL_ANGLE . It is always between -90° and +90°. (summarize)
☐ M_CENTER_OF_GRAVITY_X +	Determines the X-position of the center of gravity of a blob. The grayscale version is M_MOMENT_X1_Y0 / M_SUM_PIXEL . The binary version uses M_AREA instead of M_SUM_PIXEL . (summarize)
☐ M_CENTER_OF_GRAVITY_Y +	Determines the Y-position of the center of gravity of a blob. The grayscale version is M_MOMENT_X0_Y1 / M_SUM_PIXEL . The binary version uses M_AREA instead of M_SUM_PIXEL . (summarize)
☐ M_MOMENT_CENTRAL_Xn_Ym +	Determines the calculation of central moments, where n and m, being components of the pair (n, m), can have one of the following combinations only: (0, 2), (2, 0), or (1,1). Central moments have the syntax M_MOMENT_CENTRAL_Xn_Ym , and are defined as $\sum_i x_i^n y_i^m p_i$, where p_i = value of a pixel (always 1 for binary moments), x_i = its X-coordinate and y_i = its Y-coordinate. For central moments, coordinates are relative to each blob's center of gravity, as opposed to ordinary moments, which use coordinates that are relative to the image origin (top-left corner). Calculate higher moments by calling MblobSelectMoment() . (summarize)

<input type="checkbox"/> M_MOMENT_Xn_Ym +	<p>Determines the calculation of ordinary moments, where n and m, being components of the pair (n, m), can have one of the following combinations only: (0, 1), (1, 0), (1, 1), (0, 2), or (2, 0).</p> <p>Ordinary moments have the syntax <code>M_MOMENT_Xn_Ym</code>, and are defined as $\sum_i x_i^n y_i^m p_i$, where p_i = value of a pixel (always 1 for binary moments), x_i = its X-coordinate and y_i = its Y-coordinate. For ordinary moments, coordinates are relative to the image origin (top-left corner), as opposed to central moments, which use coordinates that are relative to each blob's center of gravity. Calculate higher moments by calling <code>MblobSelectMoment()</code>. (summarize)</p>
-------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Combination constants for the values listed in [For features with two definitions](#)

You can add one of the following values to the above-mentioned values to set whether to use the binary or grayscale version of the selected feature.

● For specifying whether to use the binary or grayscale version of the selected feature	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BINARY	Calculates the selected feature in the binary version only.
<input type="checkbox"/> M_GRAYSCALE	Calculates the selected feature in the grayscale version only.

The following values allow you to select groups of features in a single call.

For features that have both grayscale and binary definitions, the grayscale image is used as the sorting key by default. To override this default behavior, you must add `M_BINARY` to the feature selected as a sorting key (described earlier in this topic) prior to selecting the sorting key.

● For selecting groups of features in a single call	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL_FEATURES +	Adds all features (except general Feret and general moment).
<input type="checkbox"/> M_BOX +	Adds all 4 box features plus X- and Y-Ferets.
<input type="checkbox"/> M_CENTER_OF_GRAVITY +	Adds both X- and Y-coordinates of the center of gravity.
<input type="checkbox"/> M_CHAINS +	Adds all 4 chain features. Note that the chain code is computed depending on the lattice and foreground values set with <code>MblobControl()</code> . (summarize)
<input type="checkbox"/> M_CONTACT_POINTS +	Adds first point and other contact features (<code>M_X_MIN_AT_Y_MIN</code> , <code>M_X_MAX_AT_Y_MIN</code> , <code>M_X_MAX_AT_Y_MAX</code> , <code>M_X_MIN_AT_Y_MAX</code> , <code>M_Y_MIN_AT_X_MAX</code> , <code>M_Y_MAX_AT_X_MAX</code> , <code>M_Y_MAX_AT_X_MIN</code> , and <code>M_Y_MIN_AT_X_MIN</code>).
<input type="checkbox"/> M_NO_FEATURES +	Removes all features (except label value).

Combination constants for any of the possible values of the `Feature` parameter

You can add one of the following values to the above-mentioned values to set a sorting key for result retrieval.

Note that only one feature can be selected as the first, second, or third sorting key.

● For specifying a sorting key	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NO_SORT	Removes the specified sorting key.
<input type="checkbox"/> M_SORTn_DOWN	Specifies the feature as the <i>n</i> th sorting key (in descending order), where <i>n</i> stands for an integer between 1 and 3.
<input type="checkbox"/> M_SORTn_UP	Specifies the feature as the <i>n</i> th sorting key (in ascending order), where <i>n</i> stands for an integer between 1 and 3.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobSelectFeret

Synopsis

Add Feret angle to the feature list.

Syntax

```
void MblobSelectFeret(
    MIL_ID FeatureListId,
    MIL_DOUBLE Angle
)
```

Description

This function adds **M_GENERAL_FERET** at the specified angle to the feature list. The Feret diameter is then calculated at this angle, using [MblobCalculate\(\)](#). Results for this calculation can be obtained with [MblobGetResult\(\)](#) or [MblobGetResultSingle\(\)](#), specifying **M_GENERAL_FERET** as the feature.

Parameters

FeatureListId

Specifies the identifier of the feature list buffer.

Angle

Specifies the angle, in degrees, to be used for calculating the general Feret. It overrides any previous angle specified for the general Feret in this feature list. Note, the Feret diameters at 0° and 90° are calculated more efficiently with [MblobSelectFeature\(\)](#) (the features are called **M_FERET_X** and **M_FERET_Y** respectively).

To select the general Feret as a sorting key for the result retrieval, add the following sorting values to the [Angle](#) parameter:

For selecting a sorting key	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NO_SORT	Removes the specified sorting key.
<input type="checkbox"/> M_SORTn_DOWN	Specifies the feature as the <i>n</i> th sorting key (in descending order), where <i>n</i> stands for an integer between 1 and 3.
<input type="checkbox"/> M_SORTn_UP	Specifies the feature as the <i>n</i> th sorting key (in ascending order), where <i>n</i> stands for an integer between 1 and 3.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

MblobSelectMoment

Synopsis

Add specified moment to the feature list.

Syntax

```
void MblobSelectMoment(
    MIL_ID FeatureListId,
    MIL_INT MomType,
    MIL_INT XMomOrder,
    MIL_INT YMomOrder
)
```

Description

Moment calculations other than those supported by [MblobSelectFeature\(\)](#) must be specified with this function. This function adds the general moment with the specified parameters to the feature list. The general moment will be calculated by [MblobCalculate\(\)](#). Results for this calculation can be obtained by using [MblobGetResult\(\)](#) or [MblobGetResultSingle\(\)](#), specifying [M_GENERAL_MOMENT](#) as the feature. Moments directly supported by [MblobSelectFeature\(\)](#) (for example, [M_MOMENT_XO_Y2](#)) are calculated faster by selecting them through that function.

A call to this function overrides any previous values specified for [M_GENERAL_MOMENT](#) in the feature list.

Parameters

FeatureListId

Specifies the identifier of the feature list buffer.

MomType

Specifies the moment type. If the calculation is done on a binary image, only a binary version of the result is calculated. If you provide a grayscale image, both grayscale and binary versions are calculated.

This parameter must be set to one of the following values:

For specifying the moment type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CENTRAL +	Specifies a central moment type. Central moments use coordinates relative to each blob's center of gravity, and are therefore independent of a blob's position within an image. (summarize)
<input type="checkbox"/> M_ORDINARY +	Specifies an ordinary moment type. Ordinary moments are affected by the blob position because they use coordinates relative to the image origin (top-left corner). (summarize)

Combination constants for any of the possible values of the [MomType](#) parameter

You can add one of the following values to the above-mentioned values to set whether to use the binary or grayscale version of the selected feature.

If you provide a grayscale image, both grayscale and binary versions are calculated. The following limits the results to either binary or grayscale.

For specifying one version of the results	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_BINARY	Calculates only the binary version of the image.
<input type="checkbox"/> M_GRAYSCALE	Calculates only the grayscale version of the image.

Combination constants for any of the possible values of the [MomType](#) parameter

You can add one of the following values to the above-mentioned values to set the general moment as a sorting key for the result retrieval.

● For selecting a sorting key	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NO_SORT	Removes the specified sorting key.
<input type="checkbox"/> M_SORTn_DOWN	Specifies the feature as the n th sorting key (in descending order), where n stands for an integer between 1 and 3.
<input type="checkbox"/> M_SORTn_UP	Specifies the feature as the n th sorting key (in ascending order), where n stands for an integer between 1 and 3.

XMomOrder

Specifies the X-order of the moment. The X-order of the moment must be greater than or equal to 0.

YMomOrder

Specifies the Y-order of the moment. The Y-order of the moment must be greater than or equal to 0.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milblob.lib.
DLL	Requires mil.dll; milblob.dll.

Mbuf functions

Synopsis

The functions prefixed with Mbuf make up the Buffer module. The Buffer module allows you to allocate and control data buffers (storage areas) that are typically operated on by functions of several MIL modules. These buffers include image buffers and lookup table (LUT) buffers. The Buffer module allows you to isolate regions of a buffer using child buffers, as well as copy required sequential or non-sequential regions or bit-planes of a buffer to another buffer. The module's copy functions are optimized to make the best possible use of available system resources to maximize transferring speed. The module can archive and retrieve buffer data in common storage formats (for example, TIFF and JPEG). It can also create, save, and load sequences in AVI format. The module allows you to compress and decompress images and sequences, supporting both lossy and lossless JPEG and JPEG2000 compression algorithms. Furthermore, the module allows you to convert images grabbed from a camera with a Bayer filter, into 3-band color images.

Functions

- MbufAlloc1d
- MbufAlloc2d
- MbufAllocColor
- MbufBayer
- MbufChild1d
- MbufChild2d
- MbufChildColor
- MbufChildColor2d
- MbufChildMove
- MbufClear
- MbufControl
- MbufControlNeighborhood
- MbufControlRegion
- MbufCopy
- MbufCopyClip
- MbufCopyColor
- MbufCopyColor2d
- MbufCopyCond
- MbufCopyMask
- MbufCreate2d
- MbufCreateColor
- MbufDiskInquire
- MbufExport
- MbufExportSequence
- MbufFree
- MbufGet
- MbufGet1d
- MbufGet2d
- MbufGetArc
- MbufGetColor
- MbufGetColor2d
- MbufGetHookInfo
- MbufGetLine
- MbufHookFunction
- MbufImport
- MbufImportSequence
- MbufInquire
- MbufLoad
- MbufPut
- MbufPut1d
- MbufPut2d
- MbufPutColor
- MbufPutColor2d
- MbufPutLine
- MbufRestore
- MbufSave

MbufTransfer

MbufAlloc1d

Synopsis

Allocate a 1D data buffer.

Syntax

```
MIL_ID MbufAlloc1d(  
    MIL_ID SystemId,  
    MIL_INT SizeX,  
    MIL_INT Type,  
    MIL_INT64 Attribute,  
    MIL_ID *BufIdPtr  
)
```

Description

This function allocates a one-dimensional one-band data buffer on the specified system.

After allocating a buffer, we recommend that you check if the operation was successful, using [MappGetError\(\)](#) or by verifying that the buffer identifier returned is not **M_NULL**.

When a buffer is no longer required, release it, using [MbufFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the buffer. This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

SizeX

Specifies the buffer width. The units are determined by the selected buffer attribute. For example, if the buffer has a LUT buffer attribute, specify the number of LUT entries to allocate.

[Matrox GPU processing driver]
When dealing with monochrome (1-band) buffers, the buffer width must be a multiple of 4.

Type

Specifies a combination of two values: the depth and type of the data. Specify the depth of the buffer as one of the following:

● For the depth of the buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 1 +	Specifies a 1-bit (packed binary) buffer. Note that you cannot allocate a 1-bit LUT or kernel buffer. (summarize)
<input type="checkbox"/> 8 +	Specifies an 8-bit buffer.

<input type="checkbox"/> 16 +	Specifies a 16-bit buffer.
<input type="checkbox"/> 32 +	Specifies a 32-bit buffer.

Combination constants for any of the possible values of the [Type](#) parameter

You can add one of the following values to the above-mentioned values to set the data type.

Supported data types depend on the specified depth.

● For specifying the data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FLOAT	Specifies that the type of data is floating-point. The valid data depth is 32 bits. (summarize)
<input type="checkbox"/> M_SIGNED	Specifies that the type of data is signed. The valid data depths are 8, 16, or 32 bits. (summarize)
<input type="checkbox"/> M_UNSIGNED	Specifies that the type of data is unsigned. The valid data depths are 1, 8, 16, or 32 bits. This is the default value. (summarize)

Attribute

Specifies the buffer usage. MIL uses this information to determine where to allocate the buffer in physical memory.

Set this parameter to one of the following values:

● For specifying the buffer usage																												
<input type="checkbox"/> Value	Description	corona-II (a)	gige vision (c)	gpu processing (d)	hellios ea/xa (e)	hellios ec/xd (f)	hellios ed/xd (g)	hellios i394 i1dc (h)	iris (i)	met-II /d (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	vio (w)					
<input type="checkbox"/> M_ARRAY +	Allocates a buffer to store array type data. Note that some functions take an M_ARRAY buffer rather than a user-defined array. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
<input type="checkbox"/> M_IMAGE +	Allocates a buffer to store image data. You must specify a combination value from the table below . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
	<i>Board specific</i>																											
	Only single-band (monochrome) buffers with a depth of 8 or 16 bits can have an M_IMAGE attribute.				d																							
<input type="checkbox"/> M_KERNEL +	[<i>For essential MIL-Lite information, see remarks.</i>] Allocates a kernel buffer to store a custom filter for convolution functions. The depth must be 8, 16, or 32-bit integer or floating-point. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
<input type="checkbox"/> M_LUT +	Allocates a buffer to store lookup table data. The valid data depths are 8, 16, or 32 bits. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
<input type="checkbox"/> M_STRUCT_ELEMENT +	[<i>For essential MIL-Lite information, see remarks.</i>] Allocates a buffer to store structuring element data for morphology functions.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				

You can add the following value to the above-mentioned value to specify that the intended purpose of this bufer is to hold audio data.

For audio buffers	
Value	Description
M_AUDIO	Specifies an array that can hold audio data. The width of the buffer must be at least equal to the audio buffering size returned by M_AUDIO_BUFFERING_SIZE . (summarize)

You must add one or more of the following values to the above-mentioned value to set the intended purpose of this buffer.

For image buffers																																																																																																			
Value	Description	corona-II (a)	cronoplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	leee t394 iildc (h)	lris (i)	met-11/cel (j)	met-11/dig (k)	met-11/mrc (l)	met-11/std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ec/xd (u)	solos gige (v)	vo (w)																																																																											
M_COMPRESS +	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Specifies an image buffer that can hold compressed data. Note that a buffer with this attribute cannot have the M_SIGNED data type.</p> <p>When allocating buffers for operations that require a source and destination buffer, and one of the buffers has an M_COMPRESS attribute, the data will be compressed or decompressed depending on the attributes of the destination buffer. If both the source and destination buffers have M_COMPRESS specifies but different compression types, the data will be re-compressed according to the settings of the destination buffer.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w																																																																											
M_DISP	Specifies an image buffer that can be displayed.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w																																																																											
M_GRAB	<p>Specifies an image buffer in which to grab data. This type of buffer is usually allocated in MIL-reserved, physically contiguous, non-paged memory.</p> <p>For Host buffers (M_HOST_MEMORY), the maximum (total) number of grab (M_GRAB) buffers that can be allocated is restricted by the total amount of MIL-reserved, non-paged memory (specified at installation time or using MilConfig).</p> <p>For on-board buffers (M_ON_BOARD), maximum (total) number of grab (M_GRAB) buffers that can be allocated is restricted by the total amount of on-board memory.</p> <p>(summarize)</p> <table><tr><td>Board specific</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>Only single-band (monochrome) buffers with a depth of 8 or 16 bits can have an M_GRAB attribute.</td><td>a</td><td></td><td>c</td><td></td><td></td><td>e</td><td>f</td><td>g</td><td></td><td></td><td></td><td>j</td><td>k</td><td></td><td></td><td></td><td></td><td>q</td><td>r</td><td>s</td><td>t</td><td>u</td><td>v</td><td></td></tr><tr><td>Only single-band (monochrome) buffers with a depth of 8 bits can have an M_GRAB attribute.</td><td></td><td>b</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>i</td><td></td><td></td><td>l</td><td>m</td><td>n</td><td>o</td><td>p</td><td></td><td></td><td></td><td></td><td></td><td></td><td>w</td></tr></table>	Board specific																									Only single-band (monochrome) buffers with a depth of 8 or 16 bits can have an M_GRAB attribute.	a		c			e	f	g				j	k					q	r	s	t	u	v		Only single-band (monochrome) buffers with a depth of 8 bits can have an M_GRAB attribute.		b								i			l	m	n	o	p							w	a	b	c		e	f	g			i		j	k	l	m	n	o	p					w
Board specific																																																																																																			
Only single-band (monochrome) buffers with a depth of 8 or 16 bits can have an M_GRAB attribute.	a		c			e	f	g				j	k					q	r	s	t	u	v																																																																												
Only single-band (monochrome) buffers with a depth of 8 bits can have an M_GRAB attribute.		b								i			l	m	n	o	p							w																																																																											

<input type="checkbox"/> M_PROC +	Specifies an image buffer that can be processed. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	For systems with on-board processors, the total number of M_PROC buffers is limited by the amount of on-board memory.					e	f	g										q	r	s				

Combination constant for [M_PROC](#);

You can add the following value to the above-mentioned value to set the allocation of an overscan region.

This overrides the [MsysControl\(\)](#) [M_ALLOCATION_OVERSCAN](#) system control setting and forces the allocation of an overscan region. When this attribute is set, the setting of the [M_ALLOCATION_OVERSCAN_SIZE](#) system control determines the size of the overscan region and the [M_ALLOCATION_OVERSCAN](#) setting is ignored.

Note that this attribute is only useful when added to a buffer on which you want to perform neighborhood processing operations.

● For image buffers that can be processed	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALLOCATION_OVERSCAN	Specifies that the buffer is allocated with an overscan region. Specify the size of the region using MsysControl() with M_ALLOCATION_OVERSCAN_SIZE . (summarize)

Combination constants for [M_COMPRESS](#);

You can add one of the following values to the above-mentioned value to set the compression type.

The image buffer's data depth dictates which compression type can be selected.

● For compressing buffers																								
<input type="checkbox"/> Value	Description	corona-II (a)	cronosplus (b)	glige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ea/xd (f)	helios ed/xd (g)	ieee 1394 i/dc (h)	iris (i)	met-II /d (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis qxt (n)	nextis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	vio (w)	
<input type="checkbox"/> M_JPEG2000_LOSSLESS	Specifies that the buffer will be used to hold JPEG2000 lossless data. The supported data formats are: 1-band, 8- or 16-bit data. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_JPEG2000_LOSSY	Specifies that the buffer will be used to hold JPEG2000 lossy data. The supported data formats are: 1-band, 8- or 16-bit data. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_JPEG_LOSSLESS	Specifies that the buffer will be used to hold JPEG lossless data. The supported data formats are: 1-band, 8- or 16-bit data. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_JPEG_LOSSLESS_INTERLACED	Specifies that the buffer will be used to hold JPEG lossless data in separate fields. The supported data formats are 1-band, 8- or 16-bit data. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_JPEG_LOSSY	Specifies that the buffer will be used to hold JPEG lossy data the supported data format is 1-band 8-bit data. If no attribute is specified, M_JPEG_LOSSY is assumed. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_JPEG_LOSSY_INTERLACED	Specifies that the buffer will be used to hold JPEG lossy data in separate fields. The supported data format is 1-band 8-bit data.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Library	Use mil.lib.
DLL	Requires mil.dll.

MbufAlloc2d

Synopsis

Allocate a 2D data buffer.

Syntax

```
MIL_ID MbufAlloc2d(  
    MIL_ID SystemId,  
    MIL_INT SizeX,  
    MIL_INT SizeY,  
    MIL_INT Type,  
    MIL_INT64 Attribute,  
    MIL_ID *BufIdPtr  
)
```

Description

This function allocates a two-dimensional one-band data buffer on the specified system.

After allocating a buffer, we recommend that you check if the operation was successful, using [MappGetError\(\)](#) or by verifying that the buffer identifier returned is not **M_NULL**.

When a buffer is no longer required, release it, using [MbufFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the buffer. This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

SizeX

Specifies the buffer width. The units are determined by the selected buffer attribute. For example, if the buffer has an image buffer attribute, specify the width in pixels.

[Matrox GPU processing driver]
When dealing with monochrome (1-band) buffers, the buffer width must be a multiple of 4.

SizeY

Specifies the buffer height. The units are determined by the selected buffer attribute. For example, if the buffer has an image buffer attribute, specify the height in pixels.

Type

Specifies a combination of two values: the depth and type of the data. Specify the depth of the buffer as one of the following:

● For the depth of the buffer	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> 1 +	Specifies a 1-bit (packed binary) buffer. Note that you cannot allocate a 1-bit LUT or kernel buffer. (summarize)
<input type="checkbox"/> 8 +	Specifies an 8-bit buffer.
<input type="checkbox"/> 16 +	Specifies a 16-bit buffer.
<input type="checkbox"/> 32 +	Specifies a 32-bit buffer.

Combination constants for any of the possible values of the [Type](#) parameter

You can add one of the following values to the above-mentioned values to set the data type.

Supported data types depend on the specified depth.

● For specifying the data type	
Value	Description
<input type="checkbox"/> M_FLOAT	Specifies that the type of data is floating-point. The valid data depth is 32 bits. (summarize)
<input type="checkbox"/> M_SIGNED	Specifies that the type of data is signed. The valid data depths are 8, 16, or 32 bits. (summarize)
<input type="checkbox"/> M_UNSIGNED	Specifies that the type of data is unsigned. The valid data depths are 1, 8, 16, or 32 bits. This is the default value. (summarize)

Attribute

Specifies the buffer usage. MIL uses this information to determine where to allocate the buffer in physical memory.

Set this parameter to one of the following values:

● For specifying the buffer usage																									
Value	Description	corona-II (a)	cronosplus (b)	glge vision (c)	gpu processing (d)	helios ea/xa (e)	helios ecl/xd (f)	helios ed/xd (g)	ieee 1394 iIac (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ecl/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecl/xd (u)	solios glge (v)	vio (w)	
M_ARRAY +	Allocates a buffer to store array type data. Note that some functions take an M_ARRAY buffer rather than a user-defined array. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_IMAGE +	Allocates a buffer to store image data. You must specify a combination value from the table below . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
	Board specific																								
	Only single-band (monochrome) buffers with a depth of 8 or 16 bits can have an M_IMAGE attribute.				d																				
M_KERNEL +	<i>[For essential MIL-Lite information, see remarks.]</i> Allocates a kernel buffer to store a custom filter for convolution functions. The depth must be 8, 16, or 32-bit integer or floating-point. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_LUT +	Allocates a buffer to store lookup table data.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

MIL automatically selects the most appropriate storage format according to the specified intended usage attribute. For general processing, MIL will convert the data when the function requires a different format. If the default storage format is not appropriate and you want to avoid conversion during a time critical operation, you can add a storage format and a location specifier.

For specifying a storage format		corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	hellios ea/xa (e)	hellios ecl/xc1 (f)	hellios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ecl/xc1 (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecl/xc1 (u)	solios gige (v)	vio (w)
Value	Description																							
<input type="checkbox"/> M_DIB	[This is only applicable to Windows] Forces the buffer to be a DIB buffer. This ensures your buffer is stored with a DIB header. Use MbufInquire() with M_BITMAPINFO to return a pointer (LPBITMAPINFO) to the header of the MIL buffer. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DIRECTX	[This is only applicable to Windows] Forces the buffer to be a DirectX surface.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_GDI	Forces the buffer to be compatible with GDI. When using a device context (using MbufControl() with M_DC_ALLOC) for drawing, the buffer should be internally stored in GDI format, and cannot be a child buffer. (summarize)	a	b	c	d	e	f	g	h		j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_LINUX_MXIMAGE	[This is only applicable to Linux.] Forces the buffer to be an X11 Ximage.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constant for [M_IMAGE](#); [M_LUT](#);

You can add the following value to the above-mentioned values to specify a location for a buffer that is FPGA accessible.

Note that the following value cannot be used with [M_COMPRESS](#) or values in the [For specifying a storage format](#) table.

For image and LUT buffers		corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	hellios ea/xa (e)	hellios ecl/xc1 (f)	hellios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ecl/xc1 (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecl/xc1 (u)	solios gige (v)	vio (w)
Value	Description																							
<input type="checkbox"/> M_FPGA_ACCESSIBLE +	Forces the buffer to be allocated in a bank of memory that is accessible from the Processing FPGA. The exact bank of on-board memory is determined by the other attributes used with this value. If combined with the M_HOST_MEMORY attribute, the memory allocated will be in non-paged Host memory. (summarize)																	q	r	s	t	u	v	
	<i>Board specific</i>																							
	If combined with the M_GRAB , or M_GRAB + M_PROC attribute, the Matrox Solios will allocate a buffer in memory bank 0.																				t	u		
	If combined with the M_PROC attribute, the Matrox Solios will allocate a buffer in memory bank 1.																							
	If combined with the M_PROC + M_FAST_MEMORY attribute, the Matrox Solios will allocate a buffer in memory bank 2.																							
	For Matrox Odyssey Xpro+, the fastest memory is the SRAM directly connected to the Processing FPGA. Specifying this value combined with M_FAST_MEMORY will allocate the buffer in SRAM directly connected to the Processing FPGA.																	q	r	s				

MbufAllocColor

Synopsis

Allocate a color data buffer.

Syntax

```
MIL_ID MbufAllocColor(
    MIL_ID SystemId,
    MIL_INT SizeBand,
    MIL_INT SizeX,
    MIL_INT SizeY,
    MIL_INT Type,
    MIL_INT64 Attribute,
    MIL_ID *BufIdPtr
)
```

Description

This function allocates a data buffer with multiple color bands on the specified system. This type of buffer allows for the representation of color images (for example, RGB).

This function allocates buffers that have a two-dimensional surface for each specified color band. You can use [MbufAlloc1d\(\)](#) and [MbufAlloc2d\(\)](#) to create single band one- or two-dimensional data buffers, respectively.

After allocating a buffer, we recommend that you check if the operation was successful, using [MappGetError\(\)](#), or by verifying that the buffer identifier returned is not **M_NULL**.

When a buffer is no longer required, release it, using [MbufFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the buffer. This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

SizeBand

Specifies the number of (x,y) surfaces (also called color bands) to allocate to the buffer. Specify one band for each of the color components that the buffer needs to store (for example, monochrome images require one band; RGB color images require three color bands). This parameter can be set to any non-zero integer value. However, in general, only 1- and 3-band buffers are allowed.

SizeX

Specifies the buffer width. The units are determined by the selected buffer attribute. For example, if the buffer has an image buffer attribute, width is specified in pixels.

[Matrox GPU processing driver]

When dealing with monochrome (1-band) buffers, the buffer width must be a multiple of 4.

SizeY

Specifies the buffer height. The units are determined by the selected buffer attribute. For example, if the buffer has an image buffer attribute, height is specified in pixels.

Type

Specifies a combination of two values: data type and data depth per band. Specify the depth of the buffer per band as one of the following:

For the depth of the buffer	
Value	Description
1 +	Specifies a 1-bit (packed binary) buffer. Note, you cannot allocate a 1-bit (binary) LUT buffer. (summarize)
8 +	Specifies an 8-bit buffer.
16 +	Specifies a 16-bit buffer.
32 +	Specifies a 32-bit buffer.

Combination constants for any of the possible values of the **Type** parameter

You can add one of the following values to the above-mentioned values to set the data type.

Supported data types depend on the specified depth.

For specifying the data type	
Value	Description
M_FLOAT	Specifies that the type of data is floating-point. The valid data depth is 32 bits. (summarize)
M_SIGNED	Specifies that the type of data is signed. The valid data depths are 8, 16, or 32 bits. (summarize)
M_UNSIGNED	Specifies that the type of data is unsigned. The valid data depths are 1, 8, 16, or 32 bits. This is the default value. (summarize)

Attribute

Specifies the buffer usage. MIL uses this information to determine where to allocate the buffer in physical memory.

Set this parameter to one of the following values:

[illegible]

<input type="checkbox"/> M_LUT +	Allocates a buffer to store lookup table data. The valid data depths are 8, 16, or 32 bits and only the planar internal storage format is valid. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
----------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Combination constants for **M_IMAGE**;

You must add one or more of the following values to the above-mentioned value to set the intended purpose of this buffer.

● For image buffers																								
Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ed/Xd (f)	helios ed/Xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ed/Xd (r)	odyssey ed/Xd (s)	solios ea/Xa (t)	solios ec/Xcl (u)	solios gige (v)	v1o (w)
<input type="checkbox"/> M_COMPRESS +	[For essential MIL-Lite information, see remarks .] Specifies an image buffer that can hold compressed data. Note that a buffer with this attribute cannot have the M_SIGNED data type. When allocating buffers for operations that require a source and destination buffer, and one of the buffers has an M_COMPRESS attribute, the data will be compressed or decompressed depending on the attributes of the destination buffer. If both the source and destination buffers have M_COMPRESS specifiers but different compression types, the data will be re-compressed according to the settings of the destination buffer. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISP	Specifies an image buffer that can be displayed.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_GRAB	Specifies an image buffer in which to grab data. This type of buffer is usually allocated in MIL-reserved, physically contiguous, non-paged memory. For Host buffers (M_HOST_MEMORY), the maximum (total) number of grab (M_GRAB) buffers that can be allocated is restricted by the total amount of MIL-reserved, non-paged memory (specified at installation time or using MilConfig). For on-board buffers (M_ON_BOARD), the maximum (total) number of grab (M_GRAB) buffers that can be allocated is restricted by the total amount of on-board memory. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	Only single-band (monochrome) buffers with a depth of 8 or 16 bits can have an M_GRAB attribute.	a	c			e	f	g			j	k						q	r	s	t	u	v	
	Only single-band (monochrome) buffers with a depth of 8 bits can have an M_GRAB attribute.		b							i			l	m	n	o	p							w
	Note that 3-band buffers are only available when using a Matrox Iris P300C.									i														
	Note that 3-band buffers are only available when using a Matrox Nexis S300CT.																p							
<input type="checkbox"/> M_PROC +	Specifies an image buffer that can be processed. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	For systems with on-board processors, the total number of M_PROC buffers is limited by the amount of on-board memory.					e	f	g										q	r	s				

Combination constant for **M_PROC**;

You can add the following value to the above-mentioned value to set the allocation of an overscan region.

This overrides the **MsysControl()** **M_ALLOCATION_OVERSCAN** system control setting and forces the allocation of an overscan region. When this attribute is set, the setting of the **M_ALLOCATION_OVERSCAN_SIZE** system control determines the size of the overscan region and the **M_ALLOCATION_OVERSCAN** setting is ignored.

● For specifying a storage format																									
Value	Description	corona-II (a)	coronoplus (b)	gige vision (c)	gpu processing (d)	hellios ea/xa (e)	hellios ec/xcl (f)	hellios ed/xd (g)	leee 1394 i4dc (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xcl (u)	solios gige (v)	vio (w)	
M_DIB	<div>[This is only applicable to Windows]</div> <div>Forces the buffer to be a DIB buffer. This ensures your buffer is stored with a DIB header. Use MbufInquire() with M_BITMAPINFO to return a pointer (LPBITMAPINFO) to the header of the MIL buffer.</div> <div>(summarize)</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_DIRECTX	<div>[This is only applicable to Windows]</div> <div>Forces the buffer to be a DirectX surface.</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_GDI	<div>Forces the buffer to be compatible with GDI.</div> <div>When using a device context (using MbufControl() with M_DC_ALLOC) for drawing, the buffer should be internally stored in GDI format, and cannot be a child buffer.</div> <div>(summarize)</div>	a	b	c	d	e	f	g	h		j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_LINUX_MXIMAGE	<div>[This is only applicable to Linux.]</div> <div>Forces the buffer to be an X11 Ximage.</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Combination constant for the values listed in [For specifying the buffer usage](#)

You can add the following value to the above-mentioned values to set a location for a buffer that is FPGA accessible.

Note that the following value cannot be used with [M_COMPRESS](#) or values in the [For specifying a storage format](#) table.

● For image and LUT buffers																								
Value	Description	corona-II (a)	coronoplus (b)	gige vision (c)	gpu processing (d)	hellios ea/xa (e)	hellios ec/xcl (f)	hellios ed/xd (g)	leee 1394 i4dc (h)	iris (i)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xcl (u)	solios gige (v)	vio (w)	
M_FPGA_ACCESSIBLE +	<p>Forces the buffer to be allocated in a bank of memory that is accessible from the Processing FPGA. The exact bank of on-board memory is determined by the other attributes used with this value.</p> <p>If combined with the M_HOST_MEMORY attribute, the memory allocated will be in non-paged Host memory.</p> <p>(summarize)</p>																q	r	s	t	u			
	<i>Board specific</i>																							
	<p>If combined with the M_GRAB, or M_GRAB + M_PROC attribute, the Matrox Solios will allocate a buffer in memory bank 0.</p> <p>If combined with the M_PROC attribute, the Matrox Solios will allocate a buffer in memory bank 1.</p> <p>If combined with the M_PROC + M_FAST_MEMORY attribute, the Matrox Solios will allocate a buffer in memory bank 2.</p>																				t	u		
	<p>For Matrox Odyssey Xpro+, the fastest memory is the SRAM directly connected to the Processing FPGA. Specifying this value combined with M_FAST_MEMORY will allocate the buffer in SRAM directly connected to the Processing FPGA.</p>																	q	r	s				

Combination constants for [M_IMAGE](#);

You can add one of the following values to the above-mentioned value to set a storage format for color buffers.

Note that the buffer must be a color buffer. Also note that it might be slower to use buffers that have been forced with one of these attributes. Although there is no right or wrong storage format to use, certain operations are optimized for some formats.

● For specifying a storage format for color buffers																			
Value	Description	corona-II (a)	gige vision (b)	gpu processing (d)	hellios ea/Xa (e)	hellios ed/Xd (f)	hellios ed/Xd (g)	hellios ed/Xd (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis qxt (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ed/Xd (r)	odyssey ed/Xd (s)
<input type="checkbox"/> M_PACKED +	Specifies that the buffer bands must be packed (color buffer only); that is, the pixel components are stored together (RGB RGB RGB...). Typically, packed buffers are used for display (M_DISP). Note that it might be slower to process buffers with the M_PACKED attribute. Also, note that you cannot allocate a packed LUT buffer. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
<input type="checkbox"/> M_PLANAR +	Specifies that the buffer bands must be planar (color buffer only); that is, each pixel is stored as three component planes (RRR... GGG... BBB...). Typically, planar buffers are used for processing (M_PROC). (summarize)	a	b		d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
																		s	t
																		u	v
																		w	

Combination constants for [M_PLANAR](#);

You can add one of the following values to the above-mentioned value to set a planar color buffer format.

Note that the values below cannot be used with grab buffers ([M_GRAB](#)), unless otherwise stated.

[Matrox Iris]

Note that the following color buffer formats are only available when using a Matrox Iris P300C.

[Matrox Nexis]

Note that the following color buffer formats are only available when using a Matrox Nexis S300CT.

● For specifying a planar color buffer format																			
Value	Description	corona-II (a)	gige vision (b)	gpu processing (d)	hellios ea/Xa (e)	hellios ed/Xd (f)	hellios ed/Xd (g)	hellios ed/Xd (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis qxt (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ed/Xd (r)	odyssey ed/Xd (s)
<input type="checkbox"/> M_RGB3	Specifies 3-bit color depth (RGB 1:1:1) planar pixels.	a	b		d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
<input type="checkbox"/> M_RGB96	Specifies 96-bit color depth (RGB 32:32:32) planar pixels.	a	b		d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
<input type="checkbox"/> M_YUV9	Specifies YUV9 planar pixels.	a	b		d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
<input type="checkbox"/> M_YUV12	Specifies YUV12 planar pixels.	a	b		d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
<input type="checkbox"/> M_YUV24	Specifies YUV24 planar pixels.	a	b		d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
																		s	t
																		u	v
																		w	

Combination constants for [M_PACKED](#);

You can add one of the following values to the above-mentioned value to set a packed color buffer format.

Note that the values below cannot be used with grab buffers ([M_GRAB](#)), unless otherwise stated.

[Matrox Iris]

Note that the following color buffer formats are only available when using a Matrox Iris P300C.

[Matrox Nexis]

Note that the following color buffer formats are only available when using a Matrox Nexis S300CT.

For specifying a packed color buffer format																												
Value	Description	corona-II (a)	gigavision (c)	gigaplus (b)	gpu processing (d)	helios ea/xa (e)	helios ea/xd (f)	helios ea/xd (g)	helios ea/xd (h)	helios ea/xd (i)	helios ea/xd (j)	helios ea/xd (k)	helios ea/xd (l)	helios ea/xd (m)	helios ea/xd (n)	helios ea/xd (o)	helios ea/xd (p)	helios ea/xd (q)	helios ea/xd (r)	helios ea/xd (s)	helios ea/xd (t)	helios ea/xd (u)	helios ea/xd (v)	helios ea/xd (w)	helios ea/xd (x)	helios ea/xd (y)	helios ea/xd (z)	
<input type="checkbox"/> M_BGR24	Specifies 24-bit color depth (BGR) packed pixels. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
	Board specific																											
	This value can be used with grab (M_GRAB) buffers.	a	b	c		e	f	g	h	i			l	m	n	o	p	q	r	s	t	u	v	w				
	This value can be used with grab (M_GRAB) buffers only if a color DCF is used during the grab.										j	k																
<input type="checkbox"/> M_BGR32	Specifies 32-bit color depth (BGR) packed pixels. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
	Board specific																											
	This value can be used with grab (M_GRAB) buffers.	a	b	c		e	f	g	h	i			l	m	n	o	p	q	r	s	t	u	v	w				
	This value can be used with grab (M_GRAB) buffers only if a color DCF is used during the grab.										j	k																
<input type="checkbox"/> M_RGB15	Specifies 16-bit color depth packed pixels (XRGB 1:5:5:5). Note that when accessing an M_RGB15 + M_PACKED buffer as a 3-band 8-bit buffer, the least significant bits are set to 0. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
<input type="checkbox"/> M_RGB16	Specifies 16-bit color depth packed pixels (RGB 5:6:5). Note that when accessing an M_RGB16 + M_PACKED buffer as a 3-band 8-bit buffer, the least significant bits are set to 0. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
	Board specific																											
	Only available on the PC/104-Plus form factor.												l	m														
<input type="checkbox"/> M_YUV16_UYVY	Specifies YUV16 packed (4:2:2) pixels, whereby the components of each pixel are stored in the UYVY order. For more information, see the YUV buffers section in Chapter 18: Specifying and managing your data buffers . (summarize)	a		c					h		j	k	l															
	Board specific																											
	This value can be used with grab (M_GRAB) buffers.			c																								
<input type="checkbox"/> M_YUV16_YUYV	Specifies YUV16 packed (4:2:2) pixels, whereby the components of each pixel are stored in the YUYV order. For more information, see the YUV buffers section in Chapter 18: Specifying and managing your data buffers . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
	Board specific																											
	This value can be used with grab (M_GRAB) buffers.	a	b			e	f	g	h	i				m	n	o	p	q	r	s	t	u	v	w				

<div><div></div><div>M_HOST_MEMORY</div></div>	<div>Forces the buffer in Host memory. (summarize)</div> <div><div>Board specific</div><div>This is a board specific default value.</div></div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_MAPPABLE</div></div>	<div>Forces the buffer to be allocated in non-paged memory that can be set in the address space of the process (Host memory). By default, however it is not mapped to Host memory (that is, by default the buffer will have a physical address, but not a Host address). Use MbufControl() with M_MAP to enable or disable the mapping to Host memory.</div> <div>This setting is useful when dealing with many large buffers that cannot all be mapped in Host memory at the same time. By allocating a buffer as M_MAPPABLE, it can be mapped and unmapped as needed.</div> <div>Note that, if used with the M_PAGED attribute, an error will be generated. (summarize)</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_OFF_BOARD</div></div>	<div>Ensures that the buffer is not in on-board memory. (summarize)</div> <div><div>Board specific</div><div>To ensure that an M_OFF_BOARD buffer can be used by the bus master transfer section of the board, the buffer must be allocated in non-paged memory (M_OFF_BOARD + M_NON_PAGED or M_HOST_MEMORY + M_NON_PAGED).</div></div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_ON_BOARD +</div></div>	<div>Forces the buffer in on-board memory. (summarize)</div> <div><div>Board specific</div><div>By default, all on-board buffers are allocated in memory that is not mapped on the PCI bus. This unshared memory cannot be accessed by the Host or any system other than the one on which it was allocated. See combination values below (M_SHARED) to change this default behavior.</div><div>This is only available for grab buffers (M_GRAB).</div><div>This is a board specific default value.</div></div>	a				e	f	g		i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_VIDEO_MEMORY</div></div>	<div>[<i>This is only applicable to Windows</i>]</div> <div>Forces the buffer in (off-screen) display memory of your graphics controller.</div> <div>If Direct3D hardware acceleration mode is being used for display, the buffer cannot be allocated in display memory when any of the following attributes are used: M_GRAB or M_NON_PAGED. This is also the case when using DirectDraw 7 hardware acceleration mode if also using a non-Matrox graphics controller.</div> <div>If Direct3D hardware acceleration mode is being used for display and you allocate the buffer in display memory, you must use the lock-unlock mechanism to access the buffer (MbufControl() with M_LOCK / M_UNLOCK). (summarize)</div> <div><div>Board specific</div><div>Buffers cannot be allocated in display memory when any of the following attributes are used: M_COMPRESS, M_DIB, M_GDI, M_MAPPABLE, M_FAST_MEMORY, M_SHARED, or M_NON_PAGED.</div><div>This is a board specific default value.</div></div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constants for the values listed in [For image and LUT buffers](#); and for the following value: `M_ON_BOARD`;

You can add one of the following values to the above-mentioned values to specify a specific bank of on-board memory.

Note that this value must not conflict with the attribute of the specified physical memory to which it is mapped. Such a conflict can result in errors that MIL cannot catch.

🌐 For specifying a specific on-board memory bank

[illegible]

[\(summarize\)](#)

Combination constants for any of the possible values of the [Attribute](#) parameter

You can add one of the following values to the above-mentioned values to set a type of memory.

• For specifying a type of memory

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NON_PAGED	Forces the buffer in MIL-reserved, non-pageable memory.
<input type="checkbox"/> M_PAGED	Forces the buffer in pageable memory.

BufIdPtr

Specifies the address of the variable in which to write the buffer identifier. Since the **MbufAllocColor()** function also returns the buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the buffer identifier. If allocation fails, **M_NULL** is returned.

Remarks

- *[MIL-Lite]*
Note that MIL-Lite compression support, particularly for an **M_COMPRESS** buffer type, is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.
- If you are creating a DLL that includes a call to **MbufAllocColor()**, ensure that the call is not made from the **DIIMain()** function, because **MbufAllocColor()** might load a required DLL and you cannot load a DLL from **DIIMain()**. If necessary, call **MbufAllocColor()** from an initialization function in your DLL instead.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufBayer

Synopsis

Decode the color information of a single-band, Bayer color-encoded image.

Syntax

```
void MbufBayer(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_ID CoefOrExpId,
    MIL_INT ControlFlag
)
```

Description

This function converts a single-band, Bayer color-encoded image into a 1- or 3-band image. This function can also perform white balance correction and gamma correction on the image, if the appropriate coefficients and exponents are provided, respectively. In addition, the function can perform color correction on the image if the adaptive algorithm is used to decode the image (demaosaicing process).

Color correction, white balance correction, and then gamma correction operations are performed on the intermediate RGB values recovered by the demosaicing process (before being converted into the proper destination format).

Note that you can use **MbufBayer()** to automatically calculate the appropriate white balance correction coefficients. To do so, specify a Bayer color-encoded source image that in reality should be a uniform shade of gray that is not too saturated, and add **M_WHITE_BALANCE_CALCULATE** to the **ControlFlag** parameter. Alternatively, you could fill an array with custom values, and then load these values (in order) into the MIL array using **MbufPut1d()** or **MbufPut2d()**. For more information, see the [Correcting the white balance of your Bayer images](#) subsection in the [Using buffers with the Bayer color filter](#) section in [Chapter 18: Specifying and managing your data buffers](#). Ensure that the image was grabbed under the same lighting conditions as subsequent source images. After converting the image and calculating the appropriate coefficients, the coefficients are written to the specified MIL array buffer and the converted image is then corrected using these automatically calculated coefficients.

Note that you cannot calculate the white balance coefficients and specify an adaptive demosaicing in the same call to **MbufBayer()**.

For more information on Bayer color-encoded images, the conversion, and the actual equations used to perform white balance and gamma correction, see the [Using buffers with the Bayer color filter](#) section in [Chapter 18: Specifying and managing your data buffers](#).

Note that if the scale of the image is changed (using **MdigControl()** with **M_GRAB_SCALE...**) to a value other than 1 prior to grabbing a Bayer image, the Bayer image will not be converted properly; some of the Bayer pattern is lost during the scaling process, rendering color recovery impossible.

Parameters

SrcImageBufId

Specifies the identifier of the source image buffer. To achieve maximum conversion performance, the source image buffer should be an unsigned one-band image buffer. If you specify a 3-band buffer, only the first band will be used.

[All except Matrox GPU processing driver]
The source buffer must be an 8- or 16-bit buffer.

[Matrox GPU processing driver]
The source buffer must be an 8- or 16-bit monochrome buffer. The buffer's minimum size X and size Y must be a multiple of 2.

DestImageBufId

Specifies the identifier of the destination image buffer. The results of the white balance correction, gamma correction, and color correction are clipped, if necessary, to the maximum value of the source buffer.

[All except Matrox GPU processing driver]
The destination buffer should be either an 8- or 16-bit monochrome, or a 3-band color buffer in one of the following formats:

- Any RGB color buffer format + **M_PLANAR**.
- M_BGR32** + **M_PACKED**.

- [M_YUV16_YUYV](#) (equivalent to [M_YUV16](#) + [M_PACKED](#)).

When you convert the image using the adaptive algorithm, it is recommended that you use a planar RGB buffer (any RGB color buffer format + [M_PLANAR](#)). The other destination formats mentioned above can be used, but require an extra conversion operation that increases processing time.

The RGB result is saturated according to the source buffer's maximum value, set using [MbufControl\(\)](#) with [M_MAX](#). With YUV or 1-band destination buffers, the saturation is applied to the intermediate RGB result, before doing the color conversion.

[Matrox GPU processing driver]

The destination buffer must be an unsigned 8- or 16-bit monochrome buffer, or a packed BGR32 buffer. The buffer's minimum size X and size Y must be a multiple of 2.



CoefOrExpId

Specifies whether white balance correction and gamma correction are performed, and the coefficients and exponents used to perform these corrections, respectively. This parameter can be set to one of the following:

● For balance and gamma correction	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Performs no white balance or gamma correction on the source image; only the conversion is performed.
<input type="checkbox"/> MIL array buffer identifier	<p>Specifies the MIL array buffer which contains the coefficients and exponents to perform the corrections.</p> <p>The buffer must be allocated with MbufAlloc1d() or MbufAlloc2d(), as a single-band 32-bit floating-point buffer with an M_ARRAY attribute and a 3x1 or 6x1 dimension. If a 3x1 MIL array buffer is specified, no gamma correction is performed.</p> <p>The first 3 elements in the MIL array buffer specify the coefficients to perform the white balance correction on the intermediate red, green, and blue bands, respectively. If you add M_WHITE_BALANCE_CALCULATE to the ControlFlag parameter, these three elements are overwritten with the coefficients that are automatically calculated. Note that white balance correction coefficients must be positive.</p> <p>The next 3 elements, if any, specify the exponents to perform the gamma correction on the intermediate red, green, and blue bands, respectively. Note that the gamma correction exponents must be positive, are typically in the range of 0.0 to 1.0, and are most commonly specified as 0.4.</p> <p>(summarize)</p>

ControlFlag

Identifies the Bayer pattern to use in the conversion, and specifies whether the white balance correction coefficients will be calculated. This parameter can be set to one of the following:

● For the Bayer pattern	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BAYER_BG +	<p>Uses the Bayer pattern that has the top-left pixel as the blue component and the next pixel as the green component. That is, pixel [0,0] is blue, pixel [1,0] is green and pixel [1,1] is red in the source image.</p> <div>  </div> <p>(summarize)</p>
<input type="checkbox"/> M_BAYER_GB +	<p>Uses the Bayer pattern that has the top-left pixel as the green component and the next pixel as the blue component. That is, pixel [0,0] is green, pixel [1,0] is blue and pixel [0,1] is red in the source image.</p> <div>  </div> <p>(summarize)</p>
<input type="checkbox"/> M_BAYER_GR +	<p>Uses the Bayer pattern that has the top-left pixel as the green component and the next pixel as the red component. That is, pixel [0,0] is green, pixel [1,0] is red and pixel [0,1] is blue in the source image.</p>


[\(summarize\)](#)
☐ M_BAYER_RG +

Uses the Bayer pattern that has the top-left pixel as the red component and the next pixel as the green component. That is, pixel [0,0] is red, pixel [1,0] is green and pixel [1,1] is blue in the source image.


[\(summarize\)](#)

Combination constant for the values listed in [For the Bayer pattern](#)

You can add the following value to the above-mentioned values to set whether to white balance the source image.

● For white balancing the source image																								
<input type="checkbox"/> Value	Description	corona-II (a)	croscoplus (b)	glige vision (c)	gpu processing (d)	hellios ea/Xa (e)	hellios ed/Xcl (f)	hellios ed/Xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /ci (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ed/Xcl (r)	odyssey ed/Xd (s)	solios ea/Xa (t)	solios ed/Xcl (u)	solios glige (v)	vio (w)
<input type="checkbox"/> M_WHITE_BALANCE_CALCULATE	Determines the 3 coefficients required to white balance the source image, and writes these coefficients into the CoefOrExpId array. Note that to use M_WHITE_BALANCE_CALCULATE , the source image must be a uniform shade of gray that is not too saturated. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constant for the values listed in [For the Bayer pattern](#)

You can add the following value to the above-mentioned values to set whether to bit-shift the result.

● For bit shifting the result	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BAYER_BIT_SHIFT(MIL_INT NbBit)	<p>Indicates to shift the result by the specified number of bits.</p> <p>Note that this operation is applied to the RGB values resulting from the demosaicing process (after the color conversion); with YUV or 1-band destination buffers, the shift is applied to the intermediate RGB result, before doing the color conversion.</p> <p>Signed buffers are considered as unsigned.</p> <p>(summarize)</p> <div> <div>Parameters</div> <div> <div>NbBit</div> <div>This parameter specifies the number of bits by which to shift the result. You can set this parameter to the following:</div> <div> <div>Depth of source buffer - 1 > Value > 0</div> <div>Specifies the number of bits by which to shift the result.</div> </div> </div> </div>

Combination constants for the values listed in [For the Bayer pattern](#)

You can add one of the following values to the above-mentioned values to set whether to override the use of the bilinear interpolation demosaicing algorithm.

● For overriding the bilinear interpolation demosaicing algorithm																	
Value	Description	corona-II (a)	corosplus (b)	gige vision (c)	gpu processing (d)	hellor ea/xa (e)	hellor ec/xd (f)	hellor ed/xd (g)	hellor ee/xd (h)	hellor ef/xd (i)	hellor eg/xd (j)	hellor eh/xd (k)	hellor ei/xd (l)	hellor ej/xd (m)	hellor ek/xd (n)	hellor el/xd (o)	hellor em/xd (p)
M_ADAPTIVE +	Specifies to use the adaptive demosaicing algorithm for the conversion process. By using this algorithm, false colors and the zipper effect are minimized. Note you cannot use this constant in combination with M_WHITE_BALANCE_CALCULATE . (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p
M_AVERAGE_2X2	Specifies to use a 2x2 neighborhood kernel to perform the demosaicing. By using this algorithm, the zipper effect is minimized, but the resulting image is shifted upwards and to the left by a small amount. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p

Combination constant for [M_ADAPTIVE](#);

You can add the following value to the above-mentioned value to set whether to perform a color correction on the image.

● For M_ADAPTIVE	
Value	Description
M_COLOR_CORRECTION	Specifies that a color correction is performed to further eliminate false colors.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufChild1d

Synopsis

Allocate a 1D child data buffer within a specific region of the first row of a parent data buffer.

Syntax

```
MIL_ID MbufChild1d(
    MIL_ID ParentBufId,
    MIL_INT OffX,
    MIL_INT SizeX,
    MIL_ID *BufIdPtr
)
```

Description

This function allocates a one-dimensional child data buffer within a region of the first row of the specified, previously allocated, parent data buffer. If the parent buffer is multi-band, this function allocates a multi-band child buffer; the child is allocated within the specified one-dimensional region in the first row of each color band. To allocate a child in one specific band use [MbufChildColor2d\(\)](#) instead of **MbufChild1d()**.

The child buffer is not allocated its own memory space; it remains part of the parent buffer. Therefore, any modification to the child buffer affects the parent and vice versa. Note, a parent buffer can have several child buffers.

A child buffer is considered a data buffer in its own right, and can be used in the same circumstances as its parent buffer. A child buffer inherits its type and attributes from the parent buffer.

When this buffer is no longer required, it can be released using [MbufFree\(\)](#).

Parameters

ParentBufId

Specifies the identifier of the parent buffer.

OffX

Specifies the offset of the child buffer relative to the parent buffer's top-left pixel. The offset must be within the width of the parent buffer.

[Matrox GPU processing driver]

When dealing with monochrome (1-band) buffers, the buffer offset must be a multiple of 4.

SizeX

Specifies the width of the child buffer.

[Matrox GPU processing driver]

When dealing with monochrome (1-band) buffers, the buffer width must be a multiple of 4.

BufIdPtr

Specifies the address of the variable in which the child buffer identifier is to be written. Since the **MbufChild1d()** function also returns the child buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the child buffer identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufChild2d

Synopsis

Allocate a child buffer within a specific region of a parent buffer.

Syntax

```
MIL_ID MbufChild2d(
    MIL_ID ParentBufId,
    MIL_INT OffX,
    MIL_INT OffY,
    MIL_INT SizeX,
    MIL_INT SizeY,
    MIL_ID *BufIdPtr
)
```

Description

This function allocates a two-dimensional child buffer within a region of the specified, previously allocated data buffer. If the parent buffer is multi-band, this function allocates a multi-band child buffer; the child is allocated within the specified region in each color band. To allocate a child region in one specific band, or specifically in all bands, use [MbufChildColor2d\(\)](#) instead of **MbufChild2d()**.

The child buffer is not allocated its own memory space; it remains part of the parent buffer. Any modification to the child buffer affects the parent and vice versa. Note, a parent buffer can have several child buffers.

A child buffer is considered a data buffer in its own right, and can be used in the same circumstances as its parent buffer. A child buffer inherits its type and attributes from the parent buffer.

When this buffer is no longer required, it can be released, using [MbufFree\(\)](#).

Parameters

ParentBufId

Specifies the identifier of the parent buffer.

OffX

Specifies the horizontal pixel offset of the child buffer's top-left pixel, relative to the parent buffer's top-left pixel. The given offset must be within the width of the parent buffer.

[Matrox GPU processing driver]

When dealing with monochrome (1-band) buffers, the buffer offset must be a multiple of 4.

OffY

Specifies the vertical pixel offset of the child buffer's top-left pixel, relative to the parent buffer's top-left pixel. The given offset must be within the height of the parent buffer.

SizeX

Specifies the width of the child buffer.

[Matrox GPU processing driver]

When dealing with monochrome (1-band) buffers, the buffer width must be a multiple of 4.

SizeY

Specifies the height of the child buffer.

BufIdPtr

Specifies the address of the variable in which the child buffer identifier is to be written. Since the **MbufChild2d()** function also returns the child buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the child buffer identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufChildColor2d

Synopsis

Allocate a child data buffer within a color parent buffer.

Syntax

```
MIL_ID MbufChildColor2d(
    MIL_ID ParentBufId,
    MIL_INT Band,
    MIL_INT OffX,
    MIL_INT OffY,
    MIL_INT SizeX,
    MIL_INT SizeY,
    MIL_ID *BufIdPtr
)
```

Description

This function allocates a child data buffer within the specified, previously allocated, color parent data buffer. It selects a two-dimensional region in one or all of the color bands of the parent data buffer and allocates the region as a child of that buffer.

The child buffer is not allocated its own memory space; it remains part of the parent buffer. Therefore, any modification to the child buffer affects the parent and vice versa. Note, a parent buffer can have several child buffers.

A color child buffer is considered a data buffer in its own right. It can be used in the same circumstances as its parent buffer. A child buffer inherits its type and attributes from the parent buffer.

When this buffer is no longer required, release it, using [MbufFree\(\)](#).

Parameters

ParentBufId

Specifies the identifier of the parent buffer.

The parent buffer cannot have an [M_COMPRESS](#) attribute unless the **Band** parameter is set to [M_ALL_BANDS](#).

Band

Specifies the color band of the parent data buffer from which to allocate the child data buffer. The specified color band should be valid in the parent buffer.

The **Band** parameter can be set to one of the following values:

● For the color band	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL_BANDS	Specifies all color bands (for RGB, HSL, and YUV parent buffers). That is, M_ALL_BANDS allocates a child buffer with the same number of bands as the parent buffer. (summarize)
<input type="checkbox"/> M_BLUE	Specifies the blue color band (for RGB parent buffers).
<input type="checkbox"/> M_GREEN	Specifies the green color band (for RGB parent buffers).
<input type="checkbox"/> M_HUE	Specifies the hue band (for HSL parent buffers).
<input type="checkbox"/> M_LUMINANCE	Specifies the luminance band (for HSL parent buffers).

<input type="checkbox"/> M_RED	Specifies the red color band (for RGB parent buffers).
<input type="checkbox"/> M_SATURATION	Specifies the saturation band (for HSL parent buffers).
<input type="checkbox"/> M_U	Specifies the U band (for YUV parent buffers). Note that the dimensions of the child buffer must not exceed the dimensions of the U band of the parent. (summarize)
<input type="checkbox"/> M_V	Specifies the V band (for YUV parent buffers). Note that the dimensions of the child buffer must not exceed the dimensions of the V band of the parent. (summarize)
<input type="checkbox"/> M_Y	Specifies the Y band (for YUV parent buffers).
<input type="checkbox"/> Value	Specifies the index of the band to use. Valid index values are from 0 to (number of bands of the buffer - 1). Band 0 corresponds to: the red band (for RGB parent buffers), the hue band (for HSL parent buffers), and the Y band (for YUV parent buffers). Band 1 corresponds to: the green band (for RGB parent buffers), the saturation band (for HSL parent buffers), and the U band (for YUV parent buffers). Band 2 corresponds to: the blue band (for RGB parent buffers), the luminance band (for HSL parent buffers), and the V band (for YUV parent buffers). (summarize)

OffX

Specifies the horizontal offset of the child buffer, relative to the parent buffer's top-left pixel. The offset must be within the width of the parent buffer.

[Matrox GPU processing driver]
When dealing with monochrome (1-band) buffers, the buffer offset must be a multiple of 4.

OffY

Specifies the vertical offset of the child buffer, relative to the parent buffer's top-left pixel. The offset must be within the height of the parent buffer.

SizeX

Specifies the width of the child buffer.

[Matrox GPU processing driver]
When dealing with monochrome (1-band) buffers, the buffer width must be a multiple of 4.

SizeY

Specifies the height of the child buffer.

BufIdPtr

Specifies the address of the variable in which the child buffer identifier is to be written. Since the **MbufChildColor2d()** function also returns the child buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the child buffer identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufChildColor

Synopsis

Allocate a color-band child data buffer within a color parent buffer.

Syntax

```
MIL_ID MbufChildColor(  
    MIL_ID ParentBufId,  
    MIL_INT Band,  
    MIL_ID *BufIdPtr  
)
```

Description

This function allocates a child data buffer within the specified, previously allocated, color parent data buffer. It selects one of the color bands of the data buffer and allocates the band as a child of that buffer.

The child buffer is not allocated its own memory space; it remains part of the parent buffer. Therefore, any modification to the child buffer affects the parent and vice versa. Note, a parent buffer can have several child buffers.

A color child buffer is considered a data buffer in its own right. It can be any color band of its parent buffer, and can be used in the same circumstances as its parent buffer. A child buffer inherits its type and attributes from the parent buffer.

To allocate a child in one specific band, or specifically in all bands, use [MbufChildColor2d\(\)](#) instead of **MbufChildColor()**.

When this buffer is no longer required, release it, using [MbufFree\(\)](#).

Parameters

ParentBufId

Specifies the identifier of the parent buffer.




The parent buffer cannot have an [M_COMPRESS](#) attribute.

Band

Specifies the color band of the parent data buffer from which to allocate the child data buffer. The specified color band should be valid in the parent buffer.

The [Band](#) parameter can be set to one of the values below.

For the color band	
Value	Description
<input type="checkbox"/> M_BLUE	Specifies the blue color band (for RGB parent buffers).
<input type="checkbox"/> M_GREEN	Specifies the green color band (for RGB parent buffers).
<input type="checkbox"/> M_HUE	Specifies the hue band (for HSL parent buffers).
<input type="checkbox"/> M_LUMINANCE	Specifies the luminance band (for HSL parent buffers).
<input type="checkbox"/> M_RED	Specifies the red color band (for RGB parent buffers).
<input type="checkbox"/> M_SATURATION	Specifies the saturation band (for HSL parent buffers).
<input type="checkbox"/> M_U	Specifies the U band (for YUV parent buffers).

 M_V	Specifies the V band (for YUV parent buffers).
 M_Y	Specifies the Y band (for YUV parent buffers).
 Value	Specifies the index of the band to use. Valid index values are from 0 to (number of bands of the buffer - 1). Band 0 corresponds to: the red band (for RGB parent buffers), the hue band (for HSL parent buffers), and the Y band (for YUV parent buffers). Band 1 corresponds to: the green band (for RGB parent buffers), the saturation band (for HSL parent buffers), and the U band (for YUV parent buffers). Band 2 corresponds to: the blue band (for RGB parent buffers), the luminance band (for HSL parent buffers), and the V band (for YUV parent buffers). (summarize)

BufIdPtr

Specifies the address of the variable in which the child buffer identifier is to be written. Since the **MbufChildColor()** function also returns the child buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the child buffer identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufChildMove

Synopsis

Move and resize a child buffer within the parent buffer.

Syntax

```
void MbufChildMove(
    MIL_ID BufferId,
    MIL_INT OffsetX,
    MIL_INT OffsetY,
    MIL_INT SizeX,
    MIL_INT SizeY,
    MIL_INT ControlFlag
)
```

Description

This function moves and resizes a child buffer within the parent buffer.

Parameters

- BufferId
- Specifies the identifier of the child buffer to move and/or resize.
- OffsetX
- Specifies the new X-offset.

For the X-offset	
Value	Description
M_DEFAULT	Specifies the default value. The default is the current X-offset of the child buffer. (summarize)
Value	Specifies the X-coordinate, in pixels.

OffsetY




Specifies the new Y-offset.

For the Y-offset	
Value	Description
M_DEFAULT	Specifies the default value. The default is the current Y-offset of the child buffer. (summarize)
Value	Specifies the Y-coordinate, in pixels.

SizeX





Specifies the new width.

For the width

 Value	Description
 M_DEFAULT	Specifies the default value. The default is the current width of the child buffer. (summarize)
 Value	Specifies the number of pixels in the horizontal direction.

SizeY

Specifies the new height.

 For the height	
 Value	Description
 M_DEFAULT	Specifies the default value. The default is the current height of the child buffer. (summarize)
 Value	Specifies the number of pixels in the vertical direction.

ControlFlag

This parameter is reserved for future use. Set this parameter to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufClear

Synopsis

Clears a buffer to a specified color.

Syntax

```
void MbufClear(  
    MIL_ID DestImageBufId,  
    MIL_DOUBLE Color  
)
```

Description

This function clears the entire specified buffer to the specified color.

To clear a 16-bit or 32-bit multi-band buffer to a color value, use [MgraClear\(\)](#).

Parameters

DestImageBufId

Specifies the identifier of the buffer to clear.

Color

Specifies the grayscale or RGB color value with which to clear the buffer. Set this parameter to one of the values below.

● For the grayscale of RGB color value	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_RGB888(MIL_INT Red, MIL_INT Green, MIL_INT Blue)	Specifies an RGB value used to clear an 8-bit 3-band buffer to an RGB color. Each value (red, green or blue) must not exceed 8 bits. (summarize)
	<div>Parameters</div>
	<div>Red</div> <div>Specifies the red component, as a value between 0 and 255.</div>
	<div>Green</div> <div>Specifies the green component, as a value between 0 and 255.</div>
	<div>Blue</div> <div>Specifies the blue component, as a value between 0 and 255.</div>
<input type="checkbox"/> M_COLOR_BLACK	Specifies the color black.
<input type="checkbox"/> M_COLOR_BLUE	Specifies the color blue.
<input type="checkbox"/> M_COLOR_BRIGHT_GRAY	Specifies the color bright gray.
<input type="checkbox"/> M_COLOR_CYAN	Specifies the color cyan.
<input type="checkbox"/> M_COLOR_DARK_BLUE	Specifies the color dark blue.
<input type="checkbox"/> M_COLOR_DARK_CYAN	Specifies the color dark cyan.

<input type="checkbox"/> M_COLOR_DARK_GREEN	Specifies the color dark green.
<input type="checkbox"/> M_COLOR_DARK_MAGENTA	Specifies the color dark magenta.
<input type="checkbox"/> M_COLOR_DARK_RED	Specifies the color dark red.
<input type="checkbox"/> M_COLOR_DARK_YELLOW	Specifies the color dark yellow.
<input type="checkbox"/> M_COLOR_GRAY	Specifies the color gray.
<input type="checkbox"/> M_COLOR_GREEN	Specifies the color green.
<input type="checkbox"/> M_COLOR_LIGHT_BLUE	Specifies the color light blue.
<input type="checkbox"/> M_COLOR_LIGHT_GRAY	Specifies the color light gray.
<input type="checkbox"/> M_COLOR_LIGHT_GREEN	Specifies the color light green.
<input type="checkbox"/> M_COLOR_LIGHT_WHITE	Specifies the color light white.
<input type="checkbox"/> M_COLOR_MAGENTA	Specifies the color magenta.
<input type="checkbox"/> M_COLOR_RED	Specifies the color red.
<input type="checkbox"/> M_COLOR_WHITE	Specifies the color white.
<input type="checkbox"/> M_COLOR_YELLOW	Specifies the color yellow.
<input type="checkbox"/> Value	<p>Specifies a grayscale value used to clear the 1-band or multi-band buffer. This value is cast to the buffer type and depth. If the destination buffer is 3-band, this value will be replicated in each band.</p> <p>If the destination buffer is 3-band, this value will be replicated in each band.</p> <p>(summarize)</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufControl

Synopsis

Control a specified data buffer setting.

Syntax

```
void MbufControl(
    MIL_ID BufId,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function allows you to control a specified data buffer setting.

Parameters

BufId

Specifies the identifier of the buffer to control.

ControlType

Specifies the buffer setting to control.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the value needed for the setting.

See the [Parameter associations](#) section for possible values.


Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For specifying the buffer setting to control](#)
- [For MPEG-4 buffers](#)
- [For M_IMAGE + M_COMPRESS](#)

For specifying the buffer setting to control																									
ControlType	Description	corona-II (a) cronoplus (b) gige vision (c) gpu processing (d) helios ea/xa (e) helios ec/xcl (f) helios ed/xd (g) ieee 1394 i1dc (h) iris (i) met-II/d (j) met-II/dig (k) met-II/mc (l) met-II/std (m) morphis (n) morphis qxt (o) nexis (p) odyssey ea/xa (q) odyssey ec/xcl (r) odyssey ed/xd (s) solios ea/xa (t) solios ec/xcl (u) solios gige (v) vio (w)																							
ControlValue																									
M_ASSOCIATED_LUT	Sets whether to associate or disassociate a LUT buffer with the specified image buffer. The image buffer must be a 1-band 8-bit or 16-bit buffer. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_DEFAULT	Disassociates a LUT buffer from the image buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

<input type="checkbox"/> MIL LUT buffer identifier	Associates a LUT buffer with the image buffer. If and when the image buffer is selected to a display, the required changes occur to produce the display effect of the LUT, unless the display is also associated with a LUT (MdispLut()). If associated with a LUT buffer, the image buffer cannot be selected to a display whose view mode is set to M_AUTO_SCALE or M_MULTI_BYTES . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DC_ALLOC	[<i>This is only applicable to Windows</i>] Allocates a device context (DC) for drawing. Determine the DC handle (HDC) using MbufInquire() with the M_DC_HANDLE inquire type. When using this control type, the buffer should be internally stored in M_GDI format, and cannot be a child buffer. No MIL operation can be done between a device context allocation and free, therefore you should use the device context for a short period of time. (summarize)	a	b	c	d	e	f	g	h		j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.	a	b	c	d	e	f	g	h		j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DC_FREE	[<i>This is only applicable to Windows</i>] Frees a device context (DC). (summarize)	a	b	c	d	e	f	g	h		j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.	a	b	c	d	e	f	g	h		j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_LOCK +	Locks the specified buffer to allow a memory pointer to obtain a valid buffer address. The address is valid while the buffer is locked, but becomes invalid after it is unlocked. The calling thread will be blocked if the buffer is already locked by a different thread. To unlock the specified buffer, use M_UNLOCK . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_MAP	Maps or unmaps the buffer to the memory address space of the process. Note that, to use this control, the buffer must be allocated with the M_MAPPABLE attribute. (summarize)	a	b	c	d	e	f	g	h	i			l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISABLE	Unmaps the buffer from the address space of the process. This sets the buffer's Host address to M_NULL . This is the default value. (summarize)	a	b	c	d	e	f	g	h	i			l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ENABLE	Maps the buffer to the address space of the process. This associates the buffer with a logical Host address. (summarize)	a	b	c	d	e	f	g	h	i			l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_MAX	Sets the maximum pixel value possible in the buffer. M_MIN and M_MAX should specify the range of pixel values in the buffer. Values larger than the maximum value or smaller than the minimum value will result in undetermined effects upon display. Pixel values between M_MIN and M_MAX are remapped linearly to values between the minimum and maximum display values possible. If you set the minimum and maximum pixel values to a value outside the buffer-type range, the results of some computations might be undefined. Note that these two values are static and not updated automatically. For example, if you set M_MAX to 50, and during computation you multiplied the buffer by 2, M_MAX would still be 50 and not 100 and the result values will not be clipped or wrapped around if they exceed 50. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> Value	Specifies the maximum pixel value within the range of the buffer type. For example, for an 8-bit unsigned buffer, set this value to a value between 0 and 255, inclusive. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_MIN	Sets the minimum pixel value possible in the buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

 M_DEFAULT	Implements the default behavior.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
---------------------------------------------------------------------------------------------	----------------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

Combination constants for [M_LOCK](#);

You can add one of the following values to the above-mentioned value to set which processor can access the locked buffer.

🌐 For M_LOCK (Processor access privileges)

Value	Description
M_GPU_ACCESS	Locks the buffer and gives access to the GPU processor.
M_HOST_ACCESS	Locks the buffer and gives access to the Host processor.
	<p>This is the default value.</p> <p>(summarize)</p>

Combination constants for [M_LOCK](#);

You can add one or more of the following values to the above-mentioned value to set the buffer's read and write permission.

• For M_LOCK (Read and write permission)

Value	Description
M_READ	Locks the buffer to be read only.
M_READ+M_WRITE	Locks the buffer to be read and write. This is the default value. (summarize)
M_WRITE	Locks the buffer to be write only.

For MPEG-4 buffers, **ControlType** and **ControlValue** can also be set to one of the values below.

For MPEG-4 buffers

[illegible]

<input type="checkbox"/> MIL M_ARRAY buffer identifier	Specifies the table.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_HUFFMAN_AC_LUMINANCE	Associates an AC Huffman table to the Y band (luminance) of the buffer. This control type is only supported with JPEG lossy buffers in YUV format. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> MIL M_ARRAY buffer identifier	Specifies the table.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_HUFFMAN_DC	Associates a DC Huffman table to the buffer. This control type is only supported with JPEG buffers (both lossy and lossless). (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> MIL M_ARRAY buffer identifier	Specifies the table. If the buffer is 3-band, the same table is applied to all bands. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_HUFFMAN_DC_CHROMINANCE	Associates a DC Huffman table to the U and V bands (chrominance) of the buffer. This control type is only supported with JPEG lossy buffers in YUV format. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> MIL M_ARRAY buffer identifier	Specifies the table.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_HUFFMAN_DC_LUMINANCE	Associates a DC Huffman table to the Y band (luminance) of the buffer. This control type is only supported with JPEG lossy buffers in YUV format. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> MIL M_ARRAY buffer identifier	Specifies the table.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_PREDICTOR	Sets the type of predictor. This control type is only supported with JPEG lossless buffers. If the buffer is 3-band, the same predictor is applied to all bands. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> 0	Specifies predictor #0 (no prediction).	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> 1	Specifies predictor #1 (the "pixel-to-the-left" predictor). This is the default value. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> 2	Specifies predictor #2 (the "pixel-above" predictor).	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_Q_FACTOR +	Sets the quantization factor. This control type is only supported with lossy buffers. For JPEG lossy buffers, the Q factor is always applied to all bands. For JPEG2000 lossy buffers, the Q factor is applied to all bands by default. To change this default behavior, see the combination values below. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> 1 to 99	Specifies the quantization factor. Only integer values are accepted. The higher the factor, the more the compression, but the lower the image quality. The default value is 50. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_Q_FACTOR_CHROMINANCE	Sets the chrominance quantization factor. This control type is only supported with JPEG or JPEG2000 lossy buffers in YUV format. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> 1 to 99	Specifies the factor. Only integer values are accepted. The higher the factor, the more the compression, but the lower the image quality. The factor is applied to the U and V bands (chrominance). The default value is 50. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_Q_FACTOR_LUMINANCE	Sets the luminance quantization factor. This control type is only supported with JPEG or JPEG2000 lossy buffers in YUV format. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w

<input type="checkbox"/> 1 to 99	<p>Specifies the factor. Only integer values are accepted. The higher the factor, the more the compression, but the lower the image quality.</p> <p>The factor is applied only to the Y band (luminance).</p> <p>The default value is 50. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_QUANTIZATION +	<p>Associates a quantization table to the buffer. This control type is only supported with lossy buffers.</p> <p>For JPEG lossy buffers, this table is associated to all bands.</p> <p>For JPEG2000 lossy buffers, this table is associated to all bands by default.</p> <p>Note that setting this control type will reset the M_Q_... control types to their default value (50). If you set the M_Q_FACTOR control type after specifying a custom table with the M_QUANTIZATION control type, the custom table will be scaled accordingly. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> MIL M_ARRAY buffer identifier	Specifies the table.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_QUANTIZATION_CHROMINANCE	<p>Associates a quantization table to the U and V bands (chrominance) of the buffer. This control type is only supported with JPEG or JPEG2000 lossy buffers in YUV format.</p> <p>Note that setting this control type will reset the M_Q_FACTOR_CHROMINANCE control type to its default value (50). If you set a M_Q_... control type after specifying a custom table with the M_QUANTIZATION_CHROMINANCE control type, the custom table will be scaled accordingly. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> MIL M_ARRAY buffer identifier	Specifies the table.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_QUANTIZATION_LUMINANCE	<p>Associates a quantization table to the Y band (luminance) of the buffer. This control type is only supported with JPEG or JPEG2000 lossy buffers in YUV format.</p> <p>Note that setting this control type will reset the M_Q_FACTOR_LUMINANCE control type to its default value (50). If you set a M_Q_... control type after specifying a custom table with the M_QUANTIZATION_LUMINANCE control type, the custom table will be scaled accordingly. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> MIL M_ARRAY buffer identifier	Specifies the table.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_RESTART_INTERVAL	<p>Sets the number of rows or 8x8 blocks between restart markers in a compressed image. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> Value > 0	<p>Specifies after how many rows (for JPEG lossless buffers) or 8x8 blocks (for JPEG lossy buffers) of data to place restart markers. Only integer values are accepted.</p> <p>The default value is 8. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_TARGET_SIZE	<p>Sets the size of the buffer into which the source image is compressed. This control type is only supported with JPEG2000 lossy buffers.</p> <p>To ensure that the compressed data fits into the specified buffer size, less-significant data is discarded. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> Value > 0	<p>Specifies the size, in bytes. Only integer values are accepted. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_TILE_SIZE_X	Sets the horizontal size for the region which will be compressed/decompressed individually within the image.														n	o								

MbufControlRegion

Synopsis

Control a specified region of a buffer.

Syntax

```
void MbufControlRegion(
    MIL_ID BufId,
    MIL_INT OffsetX,
    MIL_INT OffsetY,
    MIL_INT SizeX,
    MIL_INT SizeY,
    MIL_INT Band,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function allows you to control a specified region of a buffer.

Parameters

- BufId
- Specifies the identifier of the buffer to control.
- OffsetX
- Specifies the horizontal pixel offset of the region.
- OffsetY
- Specifies the vertical pixel offset of the region.
- SizeX
- Specifies the width of the region, in pixels.
- SizeY
- Specifies the height of the region, in pixels.
- Band
- Specifies the band of the region. This parameter can be set to one of the following values.

● For the band of the region	
<input checked="" type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ALL_BANDS .
<input type="checkbox"/> M_ALL_BANDS	Specifies all color bands (for RGB, HSL, and YUV buffers).
<input type="checkbox"/> M_BLUE	Specifies the blue color band (for RGB buffers).

<input type="checkbox"/> M_GREEN	Specifies the green color band (for RGB buffers).
<input type="checkbox"/> M_RED	Specifies the red color band (for RGB buffers).

ControlType

Specifies the type of buffer setting to control.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the value required for the setting.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the table below.

- [For the buffer settings](#)

For the buffer settings																											
ControlType	Description																										
ControlValue		corona-II (a)	cronosplus (b)	glige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xcl (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xcl (u)	solios glige (v)	v10 (w)			
<input type="checkbox"/> M_CACHE_CONTROL	Controls the region's status in cache memory. (summarize)																	q	r	s							
<input type="checkbox"/> M_IN_CACHE	Signals MIL that the content in the specified region of the buffer has been cached in cache memory without using MIL.																	q	r	s							
<input type="checkbox"/> M_MODIFIED	Signals MIL that the content in the specified region of the buffer was modified without using MIL. This control should be used to ensure that MIL updates its internal information on this region of the buffer. For example, if a region of a buffer that is selected on a display is modified using a pointer to its memory, the display will not be updated automatically. You should use this control type to force a display update of this region. Note that this setting does the same thing as using MbufControl() with M_MODIFIED , but it is faster because it updates a specific region of the buffer, instead of updating the whole buffer. This setting increments the buffer's modification counter. You can inquire about the counter's current value using MbufInquire() with M_MODIFICATION_COUNT . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufControlNeighborhood

Synopsis

Change an operation control type setting of a kernel buffer or structuring element buffer.

Syntax

```
void MbufControlNeighborhood(
    MIL_ID BufId,
    MIL_INT OperationControlType,
    MIL_INT OperationValue
)
```

Description

This function changes the setting of an operation control type of the specified kernel buffer or structuring element buffer. The operation control type settings establish how to perform a neighborhood operation when using the specified kernel buffer or structuring element buffer.

Neighborhood operations that do not use a kernel buffer structuring element buffer typically use the default values; the default normalization factor is the exception. For predefined filters, the normalization factor is automatically set to use the full data range of the destination buffer without overflows, and the result is given in the number of bits of the destination buffer. To specify a different normalization factor, define a custom filter and set the required normalization factor, using the [M_NORMALIZATION_FACTOR](#) operation control type.

Parameters

BufId

Specifies the identifier of the kernel buffer or structuring element buffer. You must have already allocated this buffer, using [MbufAlloc1d\(\)](#) or [MbufAlloc2d\(\)](#).

OperationControlType

Specifies the operation to perform during a neighborhood operation when using the specified buffer.

See the [Parameter associations](#) section for possible values.

OperationValue

Specifies the new value to assign to the operation setting specified by the [OperationControlType](#) parameter.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **OperationControlType** and **OperationValue** parameters are described in the following tables:

- For the operation settings
- For the operation settings of M_KERNEL data buffers

For the operation settings		Description
OperationControlType		
OperationValue		
		via (w) solios gige (v) solios ed/xd (u) solios ea/xa (t) odyssey ed/xd (s) odyssey ed/xd (r) odyssey ea/xa (q) nexis (p) morphis qxt (o) morphis (n) met-II/std (m) met-II/mc (l) met-II/dlg (k) met-II/lcl (j) iris (i) iiee 1394 ilic (h) helios ed/xd (g) helios ecl/xd (f) helios ea/xa (e) gpu processing (d) gige vision (c) cromopius (b) corona-II (a)

<input type="checkbox"/> M_DEFAULT	Sets all operation control types to their default operation value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_NULL	Implements the default behavior. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ABSOLUTE_VALUE	Sets whether to take the absolute value of the results. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISABLE	Does not take the absolute value of the result.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ENABLE	Takes the absolute value of the result. Note that for a structuring element buffer, you cannot take the absolute value of a result. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_OFFSET_CENTER_X	Sets the X-coordinate of the center pixel of the kernel or structuring element. M_OFFSET_CENTER_X is ignored if M_FILTER_MODE is set to M_RECURSIVE . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Specifies the X-coordinate of the top-left pixel of the central elements.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> 0 to (SizeX-1)	Specifies the value of the X-coordinate.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_OFFSET_CENTER_Y	Sets the Y-coordinate of the center of the kernel or structuring element. M_OFFSET_CENTER_Y is ignored if M_FILTER_MODE is set to M_RECURSIVE . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Specifies the Y-coordinate of the top-left pixel of the central elements.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> 0 to (SizeY-1)	Specifies the value of the Y-coordinate.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_OVERSCAN	Sets the type of overscan used to handle the border pixels of a source image. M_OVERSCAN is ignored if M_FILTER_MODE is set to M_RECURSIVE . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Specifies that MIL automatically selects the type of overscan that optimizes speed and logic according to the specified operation and the target system.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISABLE	Specifies that no overscan will be used. When overscan is disabled, the border pixels of the source image are not processed by the neighborhood operation if additional processing time is needed. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_FAST	Specifies that MIL will automatically select the overscan that optimizes speed according to the specified operation and the target system. The overscan could be hardware-specific thereby having a different behavior than the other supported overscan modes. Note that when using M_FAST , the destination pixels in the overscan area are undefined. The pixels can therefore contain different values from one function call to the next, even if the function's parameter values are the same. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_MIRROR	Processes the border pixels of a source image using overscan pixel values that mirror the source buffer pixel values. That is, the overscan pixel values will be a mirror copy of the source buffer's borders. For example:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

For `M_KERNEL` data buffers only, you can set this parameter to one of the following values.

For the operation settings of M_KERNEL data buffers	
OperationControlType	Description
OperationValue	
M_FILTER_MODE	Sets the mode in which to apply the filter. Use M_FILTER_TYPE to set the type of the filter. (summarize)
<input type="checkbox"/> M_DEFAULT <input type="checkbox"/> M_KERNEL <input type="checkbox"/> M_RECURSIVE	<p>Same as M_KERNEL.</p> <p>Specifies the use of a non-recursive implementation of the filter. In this mode, a kernel is used to perform the neighborhood operation. The kernel size is specified when allocating the kernel buffer using MbufAlloc1d() or MbufAlloc2d(), although the kernel size can be constrained by the available hardware resources. When using an IIR filter in this mode, the filtering is done using a kernel approximation (FIR) of the recursive filter.</p> <p>When using an IIR filter, this mode is not as efficient as M_RECURSIVE for large kernels. (summarize)</p> <p>Specifies the use of a recursive implementation of an Infinite Impulse Response (IIR) filter, when applicable. If this mode actually used a kernel, the kernel size would be theoretically infinite.</p> <p>M_RECURSIVE is not supported if M_FILTER_TYPE is set to M_USER_DEFINED. (summarize)</p>
M_FILTER_OPERATION	<p>Sets the type of neighborhood operation to perform using the selected filter.</p> <p>M_FILTER_OPERATION is ignored if M_FILTER_TYPE is set to M_USER_DEFINED. (summarize)</p>
<input type="checkbox"/> M_DEFAULT <input type="checkbox"/> M_SMOOTH <input type="checkbox"/> M_HORIZ_EDGE <input type="checkbox"/> M_VERT_EDGE <input type="checkbox"/> M_EDGE_DETECT <input type="checkbox"/> M_EDGE_DETECT_SQR <input type="checkbox"/> M_LAPLACIAN_EDGE <input type="checkbox"/> M_SHARPEN <input type="checkbox"/> M_FIRST_DERIVATIVE_X <input type="checkbox"/> M_FIRST_DERIVATIVE_Y <input type="checkbox"/> M_SECOND_DERIVATIVE_X <input type="checkbox"/> M_SECOND_DERIVATIVE_Y <input type="checkbox"/> M_SECOND_DERIVATIVE_XY	<p>Same as M_SMOOTH.</p> <p>Performs a smoothing operation on the image using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the absolute value of the horizontal derivative of the image using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the absolute value of the vertical derivative of the image using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the gradient of the image using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the square of the gradient of the image using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the Laplacian values of the image using the Shen-Castan or Canny-Deriche filter.</p> <p>Performs a sharpening operation on the image using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the first derivative of the image with respect to X using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the first derivative of the image with respect to Y using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the second derivative of the image with respect to X using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the second derivative of the image with respect to Y using the Shen-Castan or Canny-Deriche filter.</p> <p>Computes the second derivative of the image with respect to X and Y using the Shen-Castan or Canny-Deriche filter.</p>
M_FILTER_SMOOTHNESS	<p>Sets the degree of smoothness (strength of denoising) applied by the filter during the neighborhood operation.</p> <p>M_FILTER_SMOOTHNESS is ignored if M_FILTER_TYPE is set to M_USER_DEFINED. (summarize)</p>
<input type="checkbox"/> M_DEFAULT <input type="checkbox"/> 0 to 100	<p>Specifies the default value. The default value is 50.0. (summarize)</p> <p>Specifies the smoothness value.</p> <p>A value of 100 results in a strong noise reduction effect, while a value of 0 has almost no noise reduction effect. (summarize)</p>
M_FILTER_TYPE	<p>Sets the type of filter used to perform the neighborhood operation. The type of filter determines the distribution of the neighborhoods' influence.</p> <p>For IIR filter types, the content of the kernel buffer is ignored even if M_FILTER_MODE is set to M_KERNEL. M_FILTER_OPERATION and M_FILTER_SMOOTHNESS determine the filter values.</p>

	(summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_USER_DEFINED .
<input type="checkbox"/> M_DERICHE	<p>Specifies a Canny-Deriche Infinite Support filter. This is an exponential weighting function, of the general form:</p> $k(a n +1)e^{-a n }$ <p>This is an IIR filter.</p> <p>For the Canny-Deriche filter, the neighborhoods' influence decreases much slower as the distance from the central pixel increases, compared to the Shen-Castan filter (see M_SHEN control value).</p> <p>Use M_FILTER_MODE, M_FILTER_OPERATION, and M_FILTER_SMOOTHNESS to set the mode of the filter, the type of operation to perform using the filter, and the degree of smoothness (strength of denoising) applied to the images by the filter, respectively.</p> <p>(summarize)</p>
<input type="checkbox"/> M_SHEN	<p>Specifies a Shen-Castan Infinite Support Exponential filter. This is an exponential weighting function, of the general form:</p> $Ke^{-\beta n }$ <p>This is an IIR filter.</p> <p>For the Shen-Castan filter, the neighborhoods' influence decreases much faster as the distance from the central pixel increases, compared to the Canny-Deriche filter (see M_DERICHE control value).</p> <p>Use M_FILTER_MODE, M_FILTER_OPERATION, and M_FILTER_SMOOTHNESS to set the mode of the filter, the type of operation to perform using the filter, and the degree of smoothness (strength of denoising) applied to the images by the filter, respectively.</p> <p>(summarize)</p>
<input type="checkbox"/> M_USER_DEFINED	<p>Specifies a user-defined filter. This is a FIR filter.</p> <p>M_USER_DEFINED is not supported if M_FILTER_MODE is set to M_RECURSIVE.</p> <p>(summarize)</p>
<input type="checkbox"/> M_NORMALIZATION_FACTOR	<p>Sets the normalization factor to apply to the result.</p> <p>Note that for a structuring element buffer, you cannot specify a normalization factor.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.
<input type="checkbox"/> Value > 0	<p>Specifies the normalization factor.</p> <p>If the factor produces an overflow in the destination, saturation might occur, depending on the M_SATURATION control type.</p> <p>(summarize)</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufCopy

Synopsis

Copy data from one buffer to another.

Syntax

```
void MbufCopy(
    MIL_ID SrcBufId,
    MIL_ID DestBufId
)
```

Description

This function copies the specified source buffer data to the specified destination buffer. If the source and destination buffers are of different data types, MIL converts the data automatically.

If the source buffer depth is greater than that of the destination, the most significant bits are truncated when the data is copied into the destination. If the destination depth is greater than that of the source, the source data is zero or sign-extended (depending on the type of the source) when copied into the destination. If the destination is larger in size than the source, exceeding areas of the buffer are unaffected.

Note, when copying from a non-binary buffer to a binary buffer, all non-zero pixels in the source buffer are represented as ones (1) in the binary buffer. When copying a binary buffer to a buffer of a different depth, each bit is copied into the least-significant bit of a different destination pixel. The remaining bits of the destination pixel are set to 0; to propagate the bit value to all bits, use [MimBinarize\(\)](#).

When copying from a floating-point buffer to an integer buffer, the values are truncated.

If the source buffer is a 3-band YUV buffer and the destination buffer is a 1-band buffer, only the Y band (luminance) is copied. If the source buffer is a 3-band RGB buffer and the destination buffer is a 1-band buffer, only the red band is copied.

Note, if the source is a 1-band image buffer associated with a LUT buffer and the destination is a 3-band image buffer, the source data copied to destination is first mapped through the LUT.

Parameters

- SrcBufId
- Specifies the identifier of the source data buffer.
- DestBufId
- Specifies the identifier of the destination data buffer.

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufCopyClip

Synopsis

Copy buffer, clipping data outside the destination buffer.

Syntax

```
void MbufCopyClip(
    MIL_ID SrcBufId,
    MIL_ID DestBufId,
    MIL_INT DestOffX,
    MIL_INT DestOffY
)
```

Description

This function copies the source buffer data to the destination buffer, starting at the specified offset. Data outside of the destination buffer is not copied (it is clipped).

If the source buffer depth is greater than that of the destination, the most significant bits are truncated when the data is copied to the destination. If the destination depth is greater than that of the source, the source data is zero or sign-extended (depending on the type of the source) when copied to the destination.

Note, when copying from a non-binary buffer to a binary buffer, all non-zero pixels in the source buffer are represented as ones (1) in the binary buffer. When copying a binary buffer to a buffer of a different depth, each bit is copied into the least-significant bit of a different destination pixel. The remaining bits of the destination pixel are set to 0; to propagate the bit value to all bits, use [MimBinarize\(\)](#).

When copying from a floating-point buffer to an integer buffer, the decimal point values are ignored and the most significant bits are truncated.

Parameters

- SrcBufId
- Specifies the identifier of the source data buffer.
- DestBufId
- Specifies the identifier of the destination data buffer.
- DestOffX
- Specifies the horizontal pixel offset of the destination buffer area at which to start copying data. The offset is relative to the top-left corner of the destination buffer (0, 0). Also, this parameter can be set to a negative value, and can be specified anywhere outside the destination buffer. Data extending beyond the limits of the destination buffer is not copied (it is clipped).
- DestOffY
- Specifies the vertical pixel offset of the destination buffer area at which to start copying data. The offset is relative to the top-left corner of the destination buffer (0,0). Also, this parameter can be set to a negative value and can be specified anywhere outside the destination buffer. Data extending beyond the limits of the destination buffer is not copied (it is clipped).

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufCopyColor2d

Synopsis

Copy a two-dimensional region of one or all bands of an image buffer to another buffer.

Syntax

```
void MbufCopyColor2d(
    MIL_ID SrcBufId,
    MIL_ID DestBufId,
    MIL_INT SrcBand,
    MIL_INT SrcOffX,
    MIL_INT SrcOffY,
    MIL_INT DestBand,
    MIL_INT DestOffX,
    MIL_INT DestOffY,
    MIL_INT SizeX,
    MIL_INT SizeY
)
```

Description

This function copies a two-dimensional region of one or all color bands of the specified source buffer to the specified color band(s) of the destination buffer. It can also be used to retrieve or replace a color component from a color buffer.

If the source is a monochrome buffer and the destination is a multi-band (color) buffer, the unique source buffer band is inserted into the specified band of the destination buffer. If the source is a multi-band buffer and the destination is a monochrome buffer, the specified source buffer band is extracted from the source buffer and written to the destination buffer. If both are multi-band buffers, the specified band(s) is copied from the source to the destination.

If the source buffer depth is greater than that of the destination, the most significant bits are truncated when the data is copied to the destination. If the destination depth is greater than that of the source, the source data is zero or sign-extended (depending on the type of the source) when copied to the destination. Also, the buffers must have the same number of bands if all bands are to be copied.

Note that when copying from a non-binary buffer to a binary destination buffer, all non-zero pixels in the source buffer are represented as ones (1) in the binary destination buffer. When copying a binary buffer to a buffer of a different depth, each bit is copied into the least-significant bit of a different destination pixel. The remaining bits of the destination pixel are set to 0; to propagate the bit value to all bits, use [MimBinarize\(\)](#).

Note, if the source is a 1-band image buffer associated with a LUT buffer and the destination is a 3-band image buffer, the source data copied to destination is first mapped through the LUT.

Note that when copying into the U or V band of a YUV destination buffer, the dimensions of the U or V band of the destination buffer must not be smaller than the dimensions of the source buffer.

Parameters

SrcBufId

Specifies the identifier of the source data buffer.

DestBufId

Specifies the identifier of the destination data buffer.

SrcBand

Specifies the source color band from which to copy. [SrcBand](#) can be set to one of the following values:

● For the source color band	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 0 <= Value <= #bands - 1	Specifies the index of the band to copy.

	<p>The relationship between index value and band for RGB and HSL buffers is the following:</p> <table> <tr> <td>0</td><td>Corresponds to the red band for RGB buffers or the hue band for HSL buffers.</td></tr> <tr> <td>1</td><td>Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.</td></tr> <tr> <td>2</td><td>Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.</td></tr> </table> <p>(summarize)</p>	0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.	1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.	2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.
0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.						
1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.						
2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.						
<input type="checkbox"/> M_ALL_BANDS	Specifies all color bands (for RGB, HSL, and YUV buffers).						
<input type="checkbox"/> M_BLUE	Specifies the blue color band (for RGB buffers).						
<input type="checkbox"/> M_GREEN	Specifies the green color band (for RGB buffers).						
<input type="checkbox"/> M_HUE	Specifies the hue band (for HSL buffers).						
<input type="checkbox"/> M_LUMINANCE	Specifies the luminance band (for HSL buffers).						
<input type="checkbox"/> M_RED	Specifies the red color band (for RGB buffers).						
<input type="checkbox"/> M_SATURATION	Specifies the saturation band (for HSL buffers).						
<input type="checkbox"/> M_U	Specifies the U band (for YUV buffers).						
<input type="checkbox"/> M_V	Specifies the V band (for YUV buffers).						
<input type="checkbox"/> M_Y	Specifies the Y band (for YUV buffers).						

SrcOffX

Specifies the horizontal pixel offset of the region to read relative to the source buffer starting coordinate. The offset must be within the width of the source buffer.

SrcOffY

Specifies the vertical pixel offset of the region to read relative to the source buffer starting coordinate. The offset must be within the height of the source buffer.

DestBand

Specifies the destination color band in which to copy. [DestBand](#) can be set to the same values as [SrcBand](#).

DestOffX

Specifies the horizontal pixel offset of the region to write relative to the destination buffer starting coordinate. The offset must be within the width of the destination buffer.

DestOffY

Specifies the vertical pixel offset of the region to write relative to the destination buffer starting coordinate. The offset must be within the height of the destination buffer.

SizeX

Specifies the width of the region to be copied, starting from the specified offset ([SrcOffX](#), [DestOffX](#)).

SizeY

Specifies the height of the region to be copied, starting from the specified offset ([SrcOffY](#), [DestOffY](#)).

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include <code>mil.h</code> .
Library	Use <code>mil.lib</code> .
DLL	Requires <code>mil.dll</code> .

MbufCopyColor

Synopsis

Copy one or all bands of an image buffer.

Syntax

```
void MbufCopyColor(
    MIL_ID SrcBufId,
    MIL_ID DestBufId,
    MIL_INT Band
)
```

Description

This function copies one or all color bands of the specified source buffer to the specified destination buffer. It can also be used to retrieve or replace a color component from a color image.

If the source is a monochrome buffer and the destination is a multi-band (color) buffer, the source buffer band is copied into the specified band of the destination buffer. If the source is a multi-band buffer and the destination is a monochrome buffer, the specified source buffer band is copied from the source buffer and written to the destination buffer. If both are multi-band buffers, the specified band(s) is copied from the source to the destination.

If the source buffer depth is greater than that of the destination, the most significant bits are truncated when the data is copied to the destination. If the destination depth is greater than that of the source, the source data is zero or sign-extended (depending on the type of the source) when copied to the destination. Also, the buffers must have the same number of bands if all bands are to be copied.

Note, when copying from a non-binary buffer to a binary buffer, all non-zero pixels in the source buffer are represented as ones (1) in the binary buffer. When copying a binary buffer to a buffer of a different depth, each bit is copied into the least-significant bit of a different destination pixel. The remaining bits of the destination pixel are set to 0; to propagate the bit value to all bits, use [MimBinarize\(\)](#).

Note, if the source is a 1-band image buffer associated with a LUT buffer and the destination is a 3-band image buffer, the source data copied to destination is first mapped through the LUT.

Parameters

SrcBufId

Specifies the identifiers of the source data buffer.

DestBufId

Specifies the identifiers of the destination data buffer.

Band

Specifies the color band to copy. Set **Band** to one of the following values:

● For the color band							
<input type="checkbox"/> Value	Description						
<input type="checkbox"/> 0 <= Value <= #bands - 1	<p>Specifies the index of the band to copy.</p> <p>The relationship between index value and band for RGB and HSL buffers is the following:</p> <table> <tr> <td>0</td><td>Corresponds to the red band for RGB buffers or the hue band for HSL buffers.</td></tr> <tr> <td>1</td><td>Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.</td></tr> <tr> <td>2</td><td>Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.</td></tr> </table> <p>(summarize)</p>	0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.	1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.	2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.
0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.						
1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.						
2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.						
<input type="checkbox"/> M_ALL_BANDS	Specifies all color bands (for RGB, HSL, and YUV buffers).						

<input type="checkbox"/> M_BLUE	Specifies the blue color band (for RGB buffers).
<input type="checkbox"/> M_GREEN	Specifies the green color band (for RGB buffers).
<input type="checkbox"/> M_HUE	Specifies the hue band (for HSL buffers).
<input type="checkbox"/> M_LUMINANCE	Specifies the luminance band (for HSL buffers).
<input type="checkbox"/> M_RED	Specifies the red color band (for RGB buffers).
<input type="checkbox"/> M_SATURATION	Specifies the saturation band (for HSL buffers).
<input type="checkbox"/> M_U	Specifies the U band (for YUV buffers).
<input type="checkbox"/> M_V	Specifies the V band (for YUV buffers).
<input type="checkbox"/> M_Y	Specifies the Y band (for YUV buffers).

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufCopyCond

Synopsis

Copy conditionally the source buffer to the destination buffer.

Syntax

```
void MbufCopyCond(
    MIL_ID SrcBufId,
    MIL_ID DestBufId,
    MIL_ID CondBufId,
    MIL_INT Condition,
    MIL_DOUBLE CondValue
)
```

Description

This function copies the source buffer data to the destination buffer, modifying only those pixels of the destination buffer which have a corresponding pixel in the condition buffer that satisfies the specified condition. Other pixels are unchanged. If the source and destination buffers are of different data types, MIL converts the data automatically.

If the source buffer depth is greater than that of the destination, the most-significant bits are truncated when the data is copied to the destination. If the destination depth is greater than that of the source, the source data is zero or sign-extended (depending on the type of the source) when copied to the destination. For example, the data is zero-extended when copying an 8-bit unsigned buffer to a 16-bit unsigned buffer.

Note that if a one-band condition buffer is used with a three-band destination buffer, the one band of the condition buffer will be used for each destination band.

Parameters

SrcBufId

Specifies the identifier of the source data buffer.

DestBufId

Specifies the identifier of the destination data buffer.

CondBufId

Specifies the identifier of the condition buffer.

Condition

Specifies the condition for which the condition buffer is tested. This parameter can be set to one of the following:

For specifying the condition for which the buffer is tested	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Modify destination buffer pixels corresponding to condition buffer pixels that are non-zero.
<input type="checkbox"/> M_EQUAL	Modify destination buffer pixels corresponding to condition buffer pixels that are equal to CondValue .
<input type="checkbox"/> M_NOT_EQUAL	Modify destination buffer pixels corresponding to condition buffer pixels that are not equal to CondValue .

CondValue

Specifies the pixel value for the specified condition. Even though this value is of type *MIL_DOUBLE*, it is treated as if it had the same type and depth as the condition buffer. If the [Condition](#) parameter is set to **M_DEFAULT**, [CondValue](#) is ignored. This parameter must be set to one of the following values:

● For specifying the pixel value									
☐ Value	Description								
☐ M_RGB888(MIL_ID Red, MIL_ID Green, MIL_ID Blue)	<p>Specifies an RGB value when the source buffer, destination buffer, and condition buffer are 8-bit, and the condition buffer is a 3-band buffer. This allows you to compare each band of the condition buffer against a different value. (summarize)</p> <table> <tr> <td colspan="2"><i>Parameters</i></td></tr> <tr> <td><i>Red</i></td><td>Specifies the red component, as a value between 0 and 255.</td></tr> <tr> <td><i>Green</i></td><td>Specifies the green component, as a value between 0 and 255.</td></tr> <tr> <td><i>Blue</i></td><td>Specifies the blue component, as a value between 0 and 255.</td></tr> </table>	<i>Parameters</i>		<i>Red</i>	Specifies the red component, as a value between 0 and 255.	<i>Green</i>	Specifies the green component, as a value between 0 and 255.	<i>Blue</i>	Specifies the blue component, as a value between 0 and 255.
<i>Parameters</i>									
<i>Red</i>	Specifies the red component, as a value between 0 and 255.								
<i>Green</i>	Specifies the green component, as a value between 0 and 255.								
<i>Blue</i>	Specifies the blue component, as a value between 0 and 255.								
☐ M_COLOR_BLACK	Specifies the color black.								
☐ M_COLOR_BLUE	Specifies the color blue.								
☐ M_COLOR_BRIGHT_GRAY	Specifies the color bright gray.								
☐ M_COLOR_CYAN	Specifies the color cyan.								
☐ M_COLOR_DARK_BLUE	Specifies the color dark blue.								
☐ M_COLOR_DARK_CYAN	Specifies the color dark cyan.								
☐ M_COLOR_DARK_GREEN	Specifies the color dark green.								
☐ M_COLOR_DARK_MAGENTA	Specifies the color dark magenta.								
☐ M_COLOR_DARK_RED	Specifies the color dark red.								
☐ M_COLOR_DARK_YELLOW	Specifies the color dark yellow.								
☐ M_COLOR_GRAY	Specifies the color gray.								
☐ M_COLOR_GREEN	Specifies the color green.								
☐ M_COLOR_LIGHT_BLUE	Specifies the color light blue.								
☐ M_COLOR_LIGHT_GRAY	Specifies the color light gray.								
☐ M_COLOR_LIGHT_GREEN	Specifies the color light green.								
☐ M_COLOR_LIGHT_WHITE	Specifies the color light white.								
☐ M_COLOR_MAGENTA	Specifies the color magenta.								
☐ M_COLOR_RED	Specifies the color red.								
☐ M_COLOR_WHITE	Specifies the color white.								
☐ M_COLOR_YELLOW	Specifies the color yellow.								
☐ Value	<p>Specifies a pixel value. If the condition buffer is binary, this value must be set to 0 or 1. If the condition buffer is three bands, this value will be used for each band. (summarize)</p>								

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufCopyMask

Synopsis

Copy buffer with mask.

Syntax

```
void MbufCopyMask(
    MIL_ID SrcBufId,
    MIL_ID DestBufId,
    MIL_INT MaskValue
)
```

Description

This function copies the specified source buffer data to the specified destination buffer, modifying only the bits of the destination that have a non-zero corresponding bit in the mask.

Parameters

- SrcBufId
- Specifies the identifier of the source data buffer. Note that the source data buffer cannot be of type floating-point.
- DestBufId
- Specifies the identifier of the destination data buffer. Note that the destination data buffer cannot be of type floating-point.
- If the source buffer depth is greater than that of the destination, the most significant bits are truncated when the data is copied to the destination. If the destination depth is greater than that of the source, the source data is zero or sign-extended (depending on the type of the source) when copied to the destination.
- MaskValue
- Specifies the mask value. Even though this value is of type *MIL_INT*, it is treated as if it had the same depth as the destination buffer; the most-significant bits that are not required are ignored.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufCreate2d

Synopsis

Create a two-dimensional data buffer.

Syntax

```
MIL_ID MbufCreate2d(
    MIL_ID SystemId,
    MIL_INT SizeX,
    MIL_INT SizeY,
    MIL_INT Type,
    MIL_INT64 Attribute,
    MIL_INT64 ControlFlag,
    MIL_INT Pitch,
    void *DataPtr,
    MIL_ID *BufIdPtr
)
```

Description

This function creates a two-dimensional data buffer that maps to a user-specified data array, or another existing buffer, and associates it with a specific MIL system.

This function should be used with caution because, when using physical addresses, it provides direct access to any of your computer's memory mapped devices; when using logical addresses, memory protection errors could result.

It is generally better to leave buffer allocation, data loading, and memory control to MIL ([MbufAlloc2d\(\)](#), [MbufGet2d\(\)](#), [MbufPut2d\(\)](#)), since MIL might require special memory attributes (such as non-paged memory) or alignment to associate the buffer with a particular target system.

You must allocate the appropriate memory before calling **MbufCreate2d()** and you must free the created buffer when no longer required, using [MbufFree\(\)](#) before freeing the memory. Using [MbufFree\(\)](#) on a buffer created with **MbufCreate2d()** will free the internal structure required for a mapped buffer, but it will not free the memory used. [MbufInquire\(\)](#) can be used to get the pointer to the data of a MIL allocated buffer.

[Matrox Helios eA/XA; Matrox Helios eCL/XCL; Matrox Helios eD/XD; Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD; Matrox Solios GigE; Matrox Solios eA/XA; Matrox Solios eCL/XCL]
Note that to map a buffer to on-board memory with a different system, the on-board memory must be in shared memory, unless the created buffer is created on the same system. See your Board Specific Notes for more information regarding shared memory.

Parameters

SystemId

Specifies the MIL system on which to create the buffer.

This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

SizeX

Sets the buffer width.

[Matrox GPU processing driver]

When dealing with monochrome (1-band) buffers, the buffer width must be a multiple of 4.

● For the buffer width	
☐ Value	Description
☐ M_DEFAULT	Specifies that the buffer width is identical to that of the source buffer when the ControlFlag parameter is set to M_MIL_ID .
☐ Value	Specifies buffer width. The units are determined by the selected buffer attribute. For example, if the buffer has an image buffer attribute, width is specified in pixels. (summarize)

SizeY

Sets the buffer height.

● For the buffer height	
☐ Value	Description
☐ M_DEFAULT	Specifies that the buffer height is identical to that of the source buffer when the ControlFlag parameter is set to M_MIL_ID .
☐ Value	Specifies the buffer height. The units are determined by the selected buffer attribute. For example, if the buffer has an image buffer attribute, height is specified in pixels. (summarize)

Type

Sets a combination of two values: data type and data depth per band. Specify the depth of the buffer per band as one of the following:

● For the depth of the buffer	
☐ Value	Description
☐ M_DEFAULT +	Specifies that the buffer data depth and type are identical to that of the source buffer when the ControlFlag parameter is set to M_MIL_ID .
☐ 1 +	Specifies a 1-bit (packed binary) buffer. Note that you cannot create a 1-bit LUT buffer. (summarize)
☐ 8 +	Specifies an 8-bit buffer.
☐ 16 +	Specifies a 16-bit buffer.
☐ 32 +	Specifies a 32-bit buffer.

Combination constants for the values listed in [For the depth of the buffer](#)

You can add one of the following values to the above-mentioned values to set the data type.

Supported data types depend on the specified depth.

● For specifying the data type	
☐ Value	Description
☐ M_FLOAT	Specifies that the type of data is floating-point. The valid data depth is 32 bits. (summarize)
☐ M_SIGNED	Specifies that the type of data is signed. The valid data depths are 8, 16 or 32 bits. (summarize)
☐ M_UNSIGNED	Specifies that the type of data is unsigned. The valid data depths are 1, 8, 16 or 32 bits. This is the default value.

[\(summarize\)](#)

Attribute

Specifies the buffer usage. This allows you to specify the type of buffer, intended purpose, compression type, storage format, data direction, location, internal storage format, memory type, and overscan region of the created buffer; all of which provides additional information for other MIL functions.

This function should be used with caution because, when using physical addresses, it provides direct access to any of your computer's memory mapped devices; when using logical addresses, memory protection errors could result.

This parameter should be set to one of the values below.

● For specifying the buffer usage																											
<input type="checkbox"/> Value	Description	corona-II (a)	cronoplus (b)	glge vision (c)	gpu processing (d)	helios ea/xa (e)	helios ecd/xd (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ecd/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecd/xd (u)	solios glge (v)	vio (w)			
<input type="checkbox"/> M_ARRAY +	Specifies a buffer to store array type data. Note that some functions take an M_ARRAY buffer rather than a user-defined array. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_IMAGE +	Specifies a buffer to store image data. You must specify a combination value from the table below . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
	<i>Board specific</i>																										
	Only single-band (monochrome) buffers with a depth of 8 or 16 bits can have an M_IMAGE attribute.				d																						
<input type="checkbox"/> M_KERNEL +	[For essential MIL-Lite information, see remarks .] Specifies a kernel buffer to store a custom filter for convolution functions. The depth must be 8, 16, or 32-bit integer or floating-point. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_LUT +	Specifies a buffer to store lookup table data. The valid data depths are 8, 16, or 32-bit. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_STRUCT_ELEMENT +	[For essential MIL-Lite information, see remarks .] Specifies a buffer to store structuring element data for morphology functions. The data depth must be 32-bit. If signed, the range is -32768 to +32767. If unsigned, the range is 0 to +32767. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

Combination constants for [M_IMAGE](#);

You must add one or more of the following values to the above-mentioned value to set the intended purpose of this buffer.

Note that this value must not conflict with the attribute of the specified physical memory to which it is mapped. Such a conflict can result in errors that MIL cannot catch.

● For image buffers

<input type="checkbox"/> Value	Description	corona-II (a)	cronoplus (b)	glge vision (c)	gpu processing (d)	helios ea/xa (e)	helios ea/xd (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ea/xd (r)	odyssey ea/xd (s)	solios ea/xa (t)	solios ea/xd (u)	solios glge (v)	vio (w)
--------------------------------	-------------	---------------	---------------	-----------------	--------------------	------------------	------------------	------------------	--------------------	----------	----------------	-----------------	----------------	-----------------	-------------	-----------------	-----------	-------------------	-------------------	-------------------	------------------	------------------	-----------------	---------

<div><div></div><div>M_COMPRESS +</div></div>	<div><div>[For essential MIL-Lite information, see remarks.]</div><div>Specifies an image buffer that can hold compressed data. Note that a buffer with this attribute cannot have the M_SIGNED data type.</div><div>When mapping buffers for operations that require a source and destination buffer, and one of the buffers has an M_COMPRESS attribute, the data will be compressed or decompressed depending on the attributes of the destination buffer. If both the source and destination buffers have M_COMPRESS specifiers but different compression types, the data will be re-compressed according to the settings of the destination buffer.</div><div>(summarize)</div></div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_DISP</div></div>	<div><div>Specifies an image buffer that can be displayed.</div><div>(summarize)</div><div><div>Board specific</div><div>Single-band display buffers with a depth greater than 16 bits cannot be allocated. Three-band display buffers can only be allocated with 8 bits per band.</div></div></div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_GRAB</div></div>	<div><div>Specifies an image buffer in which to grab data. This type of buffer is usually allocated in MIL-reserved, physically contiguous, non-paged memory.</div><div>For Host buffers (M_HOST_MEMORY), the maximum (total) number of grab (M_GRAB) buffers is restricted by the total amount of MIL-reserved, non-paged memory (specified at installation time or using MilConfig).</div><div>For on-board buffers (M_ON_BOARD), maximum (total) number of grab (M_GRAB) buffers is restricted by the total amount of on-board memory.</div><div>The physical nature of the buffers (using the ControlFlag) must be set to M_PHYSICAL_ADDRESS.</div><div>(summarize)</div><div><div>Board specific</div><div>Only single-band (monochrome) buffers with a depth of 8 or 16 bits can have an M_GRAB attribute.</div><div>Only single-band (monochrome) buffers with a depth of 8 bits can have an M_GRAB attribute.</div></div></div>	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_PROC +</div></div>	<div><div>Specifies an image buffer that can be processed.</div><div>(summarize)</div><div><div>Board specific</div><div>For systems with on-board processors, the total number of M_PROC buffers is limited by the amount of on-board memory.</div></div></div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constant for [M_PROC](#);

You can add the following value to the above-mentioned value to set the allocation of an overscan region.

Note that this value must not conflict with the attribute of the specified physical memory to which it is mapped. Such a conflict can result in errors that MIL cannot catch.

● For image buffers that can be processed	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALLOCATION_OVERSCAN	Specifies that the buffer is allocated with an overscan region. Specify the size of the region using <code>MsysControl()</code> with M_ALLOCATION_OVERSCAN_SIZE . (summarize)

Combination constants for [M_COMPRESS](#);

You can add one of the following values to the above-mentioned value to set the compression type.

The image buffer's data depth dictates which compression type can be selected.

Note that this value must not conflict with the attribute of the specified physical memory to which it is mapped. Such a conflict can result in errors that MIL cannot catch.

● For compressing buffers																										
Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ed/Xd (f)	helios ed/Xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nextis (p)	odyssey ea/Xa (q)	odyssey ed/Xd (r)	odyssey ed/Xd (s)	solos ea/Xa (t)	solos ed/Xd (u)	solos gige (v)	vio (w)		
<input type="checkbox"/> M_JPEG2000_LOSSLESS	Specifies that the buffer will be used to hold JPEG2000 lossless data. The supported data formats are: 1-band, 8- or 16-bit data. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<input type="checkbox"/> M_JPEG2000_LOSSY	Specifies that the buffer will be used to hold JPEG2000 lossy data. The supported data formats are: 1-band, 8- or 16-bit data. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<input type="checkbox"/> M_JPEG_LOSSLESS	Specifies that the buffer will be used to hold JPEG lossless data. The supported data formats are: 1-band, 8- or 16-bit data. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<input type="checkbox"/> M_JPEG_LOSSLESS_INTERLACED	Specifies that the buffer will be used to hold JPEG lossless data in separate fields. The supported data formats are: 1-band, 8- or 16-bit data. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<input type="checkbox"/> M_JPEG_LOSSY	Specifies that the buffer will be used to hold JPEG lossy data. The supported data formats is 1-band 8-bit data. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<input type="checkbox"/> M_JPEG_LOSSY_INTERLACED	Specifies that the buffer will be used to hold JPEG lossy data in separate fields. The supported data formats is 1-band 8-bit data. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<input type="checkbox"/> M_MPEG4	Specifies that the buffer will be used to store MPEG4 data.															o										

Combination constants for **M_IMAGE**;

You can add one of the following values to the above-mentioned value to set a storage format and a location specifier.

MIL will interpret the created buffer according to the specified intended usage attribute. The source buffer's actual storage format can differ. If a specific internal format is required, it should be specified.

● For specifying a storage format																										
Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ed/Xd (f)	helios ed/Xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis qxt (n)	morphis qxt (o)	nextis (p)	odyssey ea/Xa (q)	odyssey ed/Xd (r)	odyssey ed/Xd (s)	solos ea/Xa (t)	solos ed/Xd (u)	solos gige (v)	vio (w)		
M_DIB	<div>[This is only applicable to Windows]</div> <div>Specifies that the buffer to be a DIB buffer. This ensures that your buffer is stored with a DIB header. Use <code>MbufInquire()</code> with <code>M_BITMAPINFO</code> to return a pointer (LPBITMAPINFO) to the header of the MIL buffer.</div> <div>(summarize)</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
M_DIRECTX	<div>[This is only applicable to Windows]</div> <div>Specifies that the buffer to be a DirectX surface.</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
M_GDI	<div>Specifies that the buffer to be compatible with GDI.</div> <div>When using a device context (using <code>MbufControl()</code> with <code>M_DC_ALLOC</code>) for drawing, the</div>	a	b	c	d	e	f	g	h		j	k	l	m	n	o	p	q	r	s	t	u	v	w		

<input type="checkbox"/> M_NON_PAGED	Specifies that the buffer is in MIL-reserved, non-pageable memory.
<input type="checkbox"/> M_PAGED	Specifies that the buffer is in pageable memory.

ControlFlag

Specifies the physical nature of the buffer. This parameter can be set to one of the values below.

● For specifying the physical nature of the buffer																													
<div><div></div>Value</div>	Description	<div><div>corona-II (a)</div><div>cronosplus (b)</div><div>glige vision (c)</div><div>gpu processing (d)</div><div>helios ea/Xa (e)</div><div>helios ec/Xcd (f)</div><div>helios ed/Xcd (g)</div><div>ieee 1394 i1dc (h)</div><div>iris (i)</div><div>met-II /cl (j)</div><div>met-II /dig (k)</div><div>met-II /mc (l)</div><div>met-II /std (m)</div><div>morphis (n)</div><div>morphis qxt (o)</div><div>nexis (p)</div><div>odyssey ea/Xa (q)</div><div>odyssey ec/Xcd (r)</div><div>odyssey ed/Xcd (s)</div><div>solos ea/Xa (t)</div><div>solos ec/Xcd (u)</div><div>solos glige (v)</div><div>vio (w)</div></div>																											
<div><div></div>M_HOST_ADDRESS +</div>	<div>Specifies that DataPtr is a logical address that points to the data.</div> <div>Note that when using logical addresses, memory protection errors could result.</div> <div>Note that you can use MbufInquire() with M_HOST_ADDRESS to get the logical address of a MIL allocated buffer.</div> <div>You must specify a combination value from the table below.</div> <div>(summarize)</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w					
<div><div></div>M_HOST_ADDRESS_REMOTE +</div>	<div>Specifies that DataPtr is the remote Host's logical address that points to the data.</div> <div>Note that when using logical addresses, memory protection errors could result.</div> <div>Note that you can use MbufInquire() with M_HOST_ADDRESS_REMOTE to get the logical address of a MIL allocated buffer.</div> <div>You must specify a combination value from the table below.</div> <div>(summarize)</div>																		q	r	s								
<div><div></div>M_MIL_ID +</div>	<div>Specifies that the new buffer maps to an existing source buffer. DataPtr points to the MIL identifier of the source buffer.</div> <div>You must specify a combination value from the table below.</div> <div>(summarize)</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w					
<div><div></div>M_PHYSICAL_ADDRESS +</div>	<div>Specifies that DataPtr is a physical address that points to the data.</div> <div>Note that using physical addresses provides direct access to any of your computer's memory mapped devices.</div> <div>Note that you can use MbufInquire() with M_PHYSICAL_ADDRESS to get the physical address of a MIL allocated buffer.</div> <div>This setting must be used for all buffers with the M_GRAB attribute.</div> <div>You must specify a combination value from the table below.</div> <div>(summarize)</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w					

Combination constants for the values listed in [For specifying the physical nature of the buffer](#)

You must add one of the following values to the above-mentioned values to set how the pitch is measured.

For specifying how the pitch is measured	
Value	Description
<input type="checkbox"/> M_PITCH	Specifies that the pitch is in pixels. This is the default value. (summarize)
<input type="checkbox"/> M_PITCH_BYTE	Specifies that the pitch is in bytes.

Pitch

Sets the size of the pitch. The pitch is the number of pixels or bytes (as specified by the **ControlFlag** parameter) between the start of any two adjacent rows of the buffer. The actual length of a row in the buffer, in physical memory, might be different from the value of the **SizeX** parameter (for example, due to use of buffer overscan).

Note that your code should not make assumptions about the actual pitch of the source memory. If the memory was allocated using an **MbufAlloc...**() function, use **MbufInquire()** with **M_PITCH** or **M_PITCH_BYTE** to establish the required pitch.

For specifying the size of the pitch	
Value	Description
M_DEFAULT	Specifies that when dealing with a 1-bit buffer, the pitch is a multiple of 4 bytes; otherwise the pitch equals the SizeX parameter.
Value	Specifies the pitch in pixels or bytes (as determined by the ControlFlag parameter). For an M_DIB buffer, if the pitch is in pixels (M_PITCH), the pitch must equal the SizeX parameter. If the pitch is in bytes (M_PITCH_BYTE), the pitch must be the next multiple of 4 that is larger or equal to (SizeX * <i>bytes per pixel</i>). (summarize)

DataPtr

Specifies the MIL identifier of a buffer, or a pointer to the data array, to which to map the created MIL buffer.

When the **ControlFlag** parameter is set to **M_MIL_ID**, **DataPtr** specifies the MIL identifier of the source buffer.

When the **ControlFlag** parameter is set to **M_HOST_ADDRESS**, **DataPtr** specifies a logical address that points to the data.

When the **ControlFlag** parameter is set to **M_HOST_ADDRESS_REMOTE**, **DataPtr** specifies the remote Host's logical address that points to the data.

When the **ControlFlag** parameter is set to **M_PHYSICAL_ADDRESS**, **DataPtr** specifies a physical address that points to the data.

BufIdPtr

Specifies the address of the variable in which the buffer identifier is to be written. Since the **MbufCreate2d()** function also returns the buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the buffer identifier. If allocation fails, an identifier of 0 is returned.

Remarks

- [MIL-Lite]
Note that MIL-Lite compression support, particularly for an **M_COMPRESS** buffer type, is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.
- [MIL-Lite]
You can allocate an image buffer with an **M_KERNEL** or an **M_STRUCT_ELEMENT** attribute with MIL-Lite. However, these attributes are not required to work under MIL-Lite.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufCreateColor

Synopsis

Create a color data buffer.

Syntax

```
MIL_ID MbufCreateColor(  
    MIL_ID SystemId,  
    MIL_INT SizeBand,  
    MIL_INT SizeX,  
    MIL_INT SizeY,  
    MIL_INT Type,  
    MIL_INT64 Attribute,  
    MIL_INT64 ControlFlag,  
    MIL_INT Pitch,  
    void **ArrayOfDataPtr,  
    MIL_ID *BufIdPtr  
)
```

Description

This function creates a color data buffer using memory at a specified location, and associates it with a specific MIL system.

This function should be used with caution because, when using physical addresses, it provides direct access to any of your computer's memory mapped devices; when using logical addresses, memory protection errors could result.

It is generally better to leave buffer allocation, data loading, and memory control to MIL ([MbufAllocColor\(\)](#), [MbufGetColor\(\)](#), [MbufPutColor\(\)](#)), since MIL might require special memory attributes (such as non-paged memory) or alignment to associate the buffer with a particular target system.

You must allocate the appropriate memory before calling **MbufCreateColor()** and you must free the created buffer when no longer required, using [MbufFree\(\)](#) before freeing the memory. Using [MbufFree\(\)](#) on a buffer created with **MbufCreateColor()** will free the internal structure required for a mapped buffer, but it will not free the memory used.

If creating a 3-band buffer, be as specific as possible when setting the buffer attributes (using the **Attribute** parameter) and make sure to specify the color format of the buffer. The more information known about the associated buffer, the better the results.

[Matrox Helios eA/XA; Matrox Helios eCL/XCL; Matrox Helios eD/XD; Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD; Matrox Solios GigE; Matrox Solios eA/XA; Matrox Solios eCL/XCL]

Note that to map a buffer to on-board memory on a different system, the on-board memory must be shared memory, unless the created buffer is created on the same system. See your Board Specific Notes for more information regarding shared memory.

Parameters

SystemId

Specifies the MIL system on which to create the buffer. This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

SizeBand

Sets the number of (x,y) surfaces (also called color bands) that the buffer should have to contain each color component of the data. When acquiring or processing monochrome images, the buffer requires only one color band. For RGB

color images, it requires three color bands.

● For specifying the number of surfaces	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the same number color bands as the source buffer when the ControlFlag parameter is set to M_MIL_ID .
<input type="checkbox"/> 1 to n	Specifies the number of bands. The possible range is 1 to n . However, there are generally either 1 or 3 bands. (summarize)

SizeX

Sets the buffer width.

[Matrox GPU processing driver]
When dealing with monochrome (1-band) buffers, the buffer width must be a multiple of 4.

● For the buffer width	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the buffer width is identical to that of the source buffer when the ControlFlag parameter is set to M_MIL_ID .
<input type="checkbox"/> Value	Specifies the buffer width. The units are determined by the selected buffer attribute. For example, if the buffer has an image buffer attribute, width is specified in pixels. (summarize)

SizeY

Sets the buffer height.

● For the buffer height	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the buffer height is identical to that of the source buffer when the ControlFlag parameter is set to M_MIL_ID .
<input type="checkbox"/> Value	Specifies the buffer height. The units are determined by the selected buffer attribute. For example, if the buffer has an image buffer attribute, height is specified in pixels. (summarize)

Type

Sets a combination of two values: data depth and data type, per band. Specify the depth of the buffer per band as one of the following:

● For the depth of the buffer	
Value	Description
<input type="checkbox"/> M_DEFAULT +	Specifies that the data depth and type are identical to that of the source buffer when the ControlFlag parameter is set to M_MIL_ID .
<input type="checkbox"/> 1 +	Specifies a 1-bit (packed binary) buffer. Note that you cannot create a 1-bit LUT buffer. (summarize)
<input type="checkbox"/> 8 +	Specifies an 8-bit buffer.
<input type="checkbox"/> 16 +	Specifies a 16-bit buffer.
<input type="checkbox"/> 32 +	Specifies a 32-bit buffer.

Combination constants for the values listed in [For the depth of the buffer](#)

You can add one of the following values to the above-mentioned values to set the data type.

● For specifying the data type	
☐ Value	Description
☐ M_FLOAT	Specifies that the type of data is floating-point. The valid data depth is 32 bits. (summarize)
☐ M_SIGNED	Specifies that the type of data is signed. The valid data depths are 8, 16 or 32 bits. (summarize)
☐ M_UNSIGNED	Specifies that the type of data is unsigned. The valid data depths are 1, 8, 16 or 32 bits. This is the default value. (summarize)

Attribute

Specifies the buffer usage. This allows you to specify the type of buffer, intended purpose, compression type, storage format, data direction, location, internal storage format, memory type, and overscan region of the created buffer; all of which provides additional information for other MIL functions.

This function should be used with caution because, when using physical addresses, it provides direct access to any of your computer's memory mapped devices; when using logical addresses, memory protection errors could result.

Set this parameter to one of the following values:

● For specifying the buffer usage		corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ecl/xcl (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cd (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ecl/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecl/xcl (u)	solios gige (v)	vio (w)
☐ Value	Description	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
☐ M_IMAGE +	Specifies a buffer to store image data. You must specify a combination value from the table below. (summarize)																							
	<i>Board specific</i>																							
	Single-band (monochrome) buffers with a depth of 8 or 16 bits can have an M_IMAGE attribute.				d																			
	Three-band (BGR) unsigned packed buffers with a depth of 8 bits (M_BGR32) can have an M_IMAGE attribute.																							
☐ M_LUT +	Specifies a buffer to store lookup table information. The depth must be 8, 16, or 32-bit integer or floating-point and the internal storage format must be planar (M_PLANAR). (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constants for M_IMAGE;

You must add one or more of the following values to the above-mentioned value to set the intended purpose of this buffer.

Note that this value must not conflict with the attribute of the specified physical memory to which it is mapped. Such a conflict can result in errors that MIL cannot catch.

● For image buffers		corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ecl/xcl (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cd (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ecl/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecl/xcl (u)	solios gige (v)	vio (w)
☐ Value	Description	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

<input type="checkbox"/> M_COMPRESS +	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Specifies an image buffer that can hold compressed data. Note that a buffer with this attribute cannot have the M_SIGNED data type.</p> <p>When mapping buffers for operations that require a source and destination buffer, and one of the buffers has an M_COMPRESS attribute, the data will be compressed or decompressed depending on the attributes of the destination buffer. If both the source and destination buffers have M_COMPRESS specifiers but different compression types, the data will be re-compressed according to the settings of the destination buffer.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISP +	Specifies an image buffer that can be displayed.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_GRAB +	<p>Specifies an image buffer in which to grab data.</p> <p>For Host buffers (M_HOST_MEMORY), the maximum (total) number of grab (M_GRAB) buffers is restricted by the total amount of MIL-reserved, non-paged memory (specified at installation time or using MilConfig).</p> <p>For on-board buffers (M_ON_BOARD), maximum (total) number of grab (M_GRAB) buffers is restricted by the total amount of on-board memory.</p> <p>The physical nature of the buffers (using the ControlFlag) must be set to M_PHYSICAL_ADDRESS.</p> <p>(summarize)</p>	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	Only single-band (monochrome) buffers with a depth of 8 or 16 bits and 3-band (color) buffers with a depth of 8 bits can have an M_GRAB attribute.	a		c		e	f	g	h		j	k						q	r	s	t	u	v	
	Only single-band or 3-band buffers, with a depth of 8 bits, can have an M_GRAB attribute.		b							i			l	m	n	o	p							w
	Note that 3-band buffers are only available when using a Matrox Iris P300C.									i														
	Note that 3-band buffers are only available when using a Matrox Nexus S300CT.																p							
<input type="checkbox"/> M_PROC +	Specifies an image buffer that can be processed.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constants for [M_COMPRESS](#);

You can add one of the following values to the above-mentioned value to set the compression type.

The image buffer's data depth dictates which compression type can be selected.

Note that this value must not conflict with the attribute of the specified physical memory to which it is mapped. Such a conflict can result in errors that MIL cannot catch.

● For compression buffers																										
<input type="checkbox"/> Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ec/Xd (f)	helios ec/Xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /d1g (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ec/Xd (r)	odyssey ec/Xd (s)	solios ea/Xa (t)	solios ec/Xd (u)	solios gige (v)	vio (w)		
<input type="checkbox"/> M_JPEG2000_LOSSLESS	Specifies that the buffer will be used to hold JPEG2000 lossless data. The supported data formats are: 1-band, 8- or 16-bit data, and 3-band, 8- or 16-bit data in M_RGB24 + M_PLANAR , M_RGB48 + M_PLANAR , or M_YUV24 + M_PLANAR format. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<input type="checkbox"/> M_JPEG2000_LOSSY	Specifies that the buffer will be used to hold JPEG2000 lossy data. The supported data formats are: 1-band, 8- or 16-bit data, and 3-band, 8- or 16-bit data in M_RGB24 , M_RGB48 , M_YUV9 , M_YUV12 , M_YUV16 + M_PLANAR , or M_YUV24 format. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<input type="checkbox"/> M_JPEG_LOSSLESS	Specifies that the buffer will be used to hold JPEG lossless data. The supported data formats are: 1-band, 8- or 16-bit data, and 3-band data in M_RGB24 or M_RGB48 format.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

	This value can be used with grab (M_GRAB) buffers.	a	b	c		e	f	g	h				l	m	n	o		q	r	s	t	u	v	w
	This value can be used with grab (M_GRAB) buffers only if a color DCF is used during the grab.									i	j	k												
<input type="checkbox"/> M_RGB15	Specifies 16-bit color depth packed pixels (XRGB 1:5:5:5). Note that when accessing an M_RGB15 + M_PACKED buffer as a 3-band 8-bit buffer, the least significant bits are set to 0. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_RGB16	Specifies 16-bit color depth packed pixels (RGB 5:6:5). Note that when accessing an M_RGB16 + M_PACKED buffer as a 3-band 8-bit buffer, the least significant bits are set to 0. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	Only available on the PC/104-Plus form factor.													l	m									
<input type="checkbox"/> M_YUV16_UYVY	Specifies YUV16 packed (4:2:2) pixels, whereby the components of each pixel are stored in the UYVY order. For more information, see the YUV buffers section in Chapter 18: Specifying and managing your data buffers . (summarize)	a		c					h		j	k	l	m										
	<i>Board specific</i>																							
	This value can be used with grab (M_GRAB) buffers.			c																				
<input type="checkbox"/> M_YUV16_YUYV	Specifies YUV16 packed (4:2:2) pixels, whereby the components of each pixel are stored in the YUYV order. For more information, see the YUV buffers section in Chapter 18: Specifying and managing your data buffers . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	This value can be used with grab (M_GRAB) buffers.	a	b	c		e	f	g	h	i					n	o	p	q	r	s	t	u	v	w

Combination constants for the values listed in [For image buffers](#) or [For specifying a storage format for color buffers](#)

You must add one of the following values to the above-mentioned values to set a color buffer format.

Note that the values below cannot be used with grab buffers ([M_GRAB](#)), unless otherwise stated.

The specified format can differ from that of the specified physical memory.

[Matrox Iris]

Note that the following color buffer formats are only available when using a Matrox Iris P300C.

[Matrox Nexis]

Note that the following color buffer formats are only available when using a Matrox Nexis S300CT.

● For specifying a packed or planar color buffer format																									
<input type="checkbox"/> Value	Description	corona-II (a)	glge vision (c)	crnosplus (b)	gpu processing (d)	helios ea/xa (e)	helios ea/xd (f)	helios ed/xd (g)	ieee 1394 ifdc (h)	iris (i)	met-II /d (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ea/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ea/xd (u)	solios glge (v)	v/o (w)	
<input type="checkbox"/> M_RGB24	Specifies 24-bit color depth (RGB 8:8:8) packed or planar pixels. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
	Board specific																								
	This value can be used with planar (M_PLANAR) grab (M_GRAB) buffers.	a		c		e	f	g			j	k	l	m				q	r	s	t	u			

Combination constants for the values listed in [For image and LUT buffers](#); and for the following value: [M_ON_BOARD](#);

You can add one of the following values to the above-mentioned values to set a specific location in memory.

● For on-board memory buffers, accessible by the FPGA																						
Value	Description	corona-II (a)	cronosplus (b)	gpu processing (d)	gige vision (c)	helios ea/xa (e)	helios ed/xcl (f)	helios ed/xd (g)	leee 1394 ilic (h)	iris (i)	met-II /cl (j)	met-II /mc (l)	met-II /std (m)	morphis (n)	nexis (p)	odyssey ea/xa (q)	odyssey ecl/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecl/xd (u)	solios gige (v)	vio (w)
<input type="checkbox"/> M_FAST_MEMORY	<p>Specifies that the buffer is created in the fastest memory available.</p> <p>Use MbufInquire() with M_EXTENDED_FORMAT to establish if the buffer was created in fast processing memory.</p> <p>(summarize)</p> <p><i>Board specific</i></p> <p>For Matrox Odyssey Xpro, the fastest memory available for on-board processing is the L3 external cache. It must be enabled using the OdcMon utility before it can be used. L3 external cache is only accessible to the G4 PowerPC microprocessor and not to the Host, frame grabber module, PA or Processing FPGA. Therefore, depending on the operation, allocating a buffer created in fast processing memory might not accelerate processing. Specifying M_ON_BOARD + M_FAST_MEMORY will allocate the buffer in L3 external cache.</p> <p>For Matrox Odyssey Xpro+, the fastest memory is the SRAM directly connected to the Processing FPGA. Specifying M_FPGA_ACCESSIBLE + M_FAST_MEMORY will create the buffer in SRAM directly connected to the Processing FPGA.</p> <p>Note that M_GRAB and M_DISP cannot be combined with this attribute.</p> <p>This attribute can only be combined with M_FPGA_ACCESSIBLE. M_FAST_MEMORY is typically the same as using M_MEMORY_BANK_2.</p> <p>For Matrox Solios eA/XA and eCL/XCL, the fastest memory available is the SRAM. If SRAM is not present on your Matrox Solios, SDRAM will be used instead.</p>															q	r	s	t	u	v	
<input type="checkbox"/> M_SHARED	<p>Specifies that the buffer is in shared processing memory. This memory is mapped on the PCI bus.</p> <p>Note that this combination value can only be added to M_ON_BOARD.</p> <p>(summarize)</p>						e	f	g							q	r	s	t	u	v	

Combination constants for the values listed in [For specifying the buffer usage](#)

You can add one of the following values to the above-mentioned values to set a type of memory.

Note that this value must not conflict with the attribute of the specified physical memory to which it is mapped. Such a conflict can result in errors that MIL cannot catch.

● For specifying a type of memory	
Value	Description
<input type="checkbox"/> M_NON_PAGED	Specifies that the buffer is in MIL-reserved, non-pageable memory.
<input type="checkbox"/> M_PAGED	Specifies that the buffer is in pageable memory.

Combination constant for [M_PROC](#);

You can add the following value to the above-mentioned value to set the allocation of an overscan region.

Note that this value must not conflict with the attribute of the specified physical memory to which it is mapped. Such a conflict can result in errors that MIL cannot catch.

● For specifying overscan	
Value	Description
<input type="checkbox"/> M_ALLOCATION_OVERSCAN	Specifies that the buffer has an overscan region.

ControlFlag

Specifies the physical nature of the buffer. This parameter can be set to one of the values below.

For specifying the physical nature of the buffer

Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	hellios ea/xa (e)	hellios ecd/xd (f)	hellios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cd (j)	met-II /dlig (k)	met-II /fmc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ecd/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecd/xd (u)	solios gige (v)	vio (w)
<div>M_HOST_ADDRESS +</div>	<p>Specifies that ArrayOfDataPtr is an array of logical addresses that point to the data.</p> <p>Note that when using logical addresses, memory protection errors could result.</p> <p>Note that you can use MbufInquire() with M_HOST_ADDRESS to get the logical address of a MIL allocated buffer.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div>M_HOST_ADDRESS_REMOTE +</div>	<p>Specifies that ArrayOfDataPtr is an array of the remote Host's logical addresses that point to the data.</p> <p>Note that when using logical addresses, memory protection errors could result.</p> <p>Note that you can use MbufInquire() with M_HOST_ADDRESS_REMOTE to get the logical address of a MIL allocated buffer.</p> <p>(summarize)</p>																	q	r	s				
<div>M_MIL_ID +</div>	<p>Specifies that the new buffer maps to an existing source buffer. ArrayOfDataPtr points to the MIL identifier of the source buffer.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div>M_PHYSICAL_ADDRESS +</div>	<p>Specifies that ArrayOfDataPtr is an array of physical addresses that point to the data.</p> <p>Note that using physical addresses provides direct access to any of your computer's memory mapped devices.</p> <p>Note that you can use MbufInquire() with M_PHYSICAL_ADDRESS to get the physical address of a MIL allocated buffer.</p> <p>This setting must be used for all buffers with the M_GRAB attribute.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constants for any of the possible values of the **ControlFlag** parameter

You can add one of the following values to the above-mentioned values to set how the pitch is measured.

● For specifying how the pitch is measured	
Value	Description
<input type="checkbox"/> M_PITCH	Specifies that the pitch is in pixels. This is the default value. (summarize)
<input type="checkbox"/> M_PITCH_BYTE	Specifies that the pitch is in bytes. Note that when creating an M_BGR24 + M_PACKED buffer, you should use M_PITCH_BYTE instead of M_PITCH because the latter might not be able to take into account internal padding. (summarize)

Pitch

Sets the size of the pitch. The pitch is the number of pixels or bytes (as specified by the **ControlFlag** parameter) between the start of any two adjacent rows of the buffer. The actual length of a row in the buffer, in physical memory, might be different from the value of the **SizeX** parameter (for example, due to use of buffer overscan).

Note that your code should not make assumptions about the actual pitch of the source memory. If the memory was allocated using an **MbufAlloc...**() function, use **MbufInquire()** with **M_PITCH** or **M_PITCH_BYTE** to establish the required pitch.

● For specifying the size of the pitch	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that when dealing with a 1-bit buffer, the pitch is a multiple of 4 bytes; otherwise the pitch equals the SizeX parameter.
<input type="checkbox"/> Value	Specifies the pitch in pixels or bytes (as determined by the ControlFlag parameter). For an M_DIB buffer, if the pitch is in pixels (M_PITCH), the pitch must equal the SizeX parameter. If the pitch is in bytes (M_PITCH_BYTE), the pitch must be the next multiple of 4 that is larger or equal to (SizeX * <i>bytes per pixel</i>). (summarize)

ArrayOfDataPtr

Specifies one or more data pointers that address the memory to which to map the created MIL buffer.

When pointing to a planar buffer, one pointer per band must be provided. Pointers to a 3-band planar buffer must be ordered R-G-B or Y-U-V in the array. When pointing to a single-band buffer or a packed buffer, a pointer to the packed data must be provided.

When the **ControlFlag** parameter is set to **M_MIL_ID**, **ArrayOfDataPtr** specifies a pointer to the MIL identifier of the source buffer.

When the **ControlFlag** parameter is set to **M_HOST_ADDRESS**, **ArrayOfDataPtr** specifies an array of logical addresses that point to the data.

When the **ControlFlag** parameter is set to **M_HOST_ADDRESS_REMOTE**, **ArrayOfDataPtr** specifies an array of the remote Host's logical addresses that point to the data.

When the **ControlFlag** parameter is set to **M_PHYSICAL_ADDRESS**, **ArrayOfDataPtr** specifies an array of physical addresses that point to the data.

BufIdPtr

Specifies the address of the variable in which the buffer identifier is to be written. Since the **MbufCreateColor()** function also returns the buffer identifier, you can set this parameter to **M_NULL**. If mapping fails, **M_NULL** is written as the identifier.

Return value

The returned value is the buffer identifier. If allocation fails, **M_NULL** is returned.

Remark

- *[MIL-Lite]*
Note that MIL-Lite compression support, particularly for an **M_COMPRESS** buffer type, is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufDiskInquire

Synopsis

Inquire about the buffer data in a file.

Syntax

```
MIL_INT MFTYPE MbufDiskInquire(
    MIL_CONST_TEXT_PTR FileName,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires about the buffer data in the specified file on disk.

Parameters

FileName

Specifies the file that contains the buffer data.

● For specifying the file name or memory stream		
☐ Value	Description	
☐ MIL_TEXT(MIL_TEXT_PTR FileName)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". Note, an error occurs if the file does not have a known file format or the file format is not supported. The supported file types include all the formats supported by the MbufExport() and MbufExportSequence() functions. Since a raw data file does not have any information regarding size or type, you can only use MbufDiskInquire() to determine the file format of this type of file. This function cannot access a file on the remote computer; the file must be located on the master computer. (summarize)	
		Parameters
	FileName	Specifies the drive, directory, and name of the file.

InquireType

Specifies the type of buffer setting about which to inquire. This parameter can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

● For specifying the type of buffer setting		
☐ Value	Description	
☐ M_ATTRIBUTE	Returns the file attribute. Note that AVI files return M_IMAGE as file attribute. (summarize)	
	UserVarPtr info	Return values: M_ARRAY; M_IMAGE; M_KERNEL; M_LUT; M_STRUCT_ELEMENT; (details)
☐ M_FILE_FORMAT	Returns the file format. See MbufExport() and MbufExportSequence() for all supported file formats.	

	(summarize)
<input type="checkbox"/> M_SIGN	<p>Returns the file data range.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: M_SIGNED Data range is signed. M_UNSIGNED Data range is unsigned. </div>
<input type="checkbox"/> M_SIZE_BAND +	Returns the number of color bands in the file.
<input type="checkbox"/> M_SIZE_BIT	Returns the file data depth, in bits.
<input type="checkbox"/> M_SIZE_X +	Returns the width of the data in the file, in pixels.
<input type="checkbox"/> M_SIZE_Y	Returns the height of the data in the file, in pixels.
<input type="checkbox"/> M_TYPE	<p>Returns the file data type and depth. The depth is returned in bits.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: depth value + M_FLOAT Data range is floating point. depth value + M_SIGNED Data range is signed. depth value + M_UNSIGNED Data range is unsigned. </div>

Combination constant for [M_SIZE_BAND](#); [M_SIZE_X](#);

You can add the following value to the above-mentioned values to get the width or the number of bands of the LUT associated with the image.

This option is available only if the file contains an image data buffer. This option is not available for AVI files.

● For M_SIZE_X and M_SIZE_BAND	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_LUT	<p>Returns either the width of the LUT associated with the image in the file (for M_SIZE_X) or the number of bands of the LUT associated with the image in the file (for M_SIZE_BAND). When there is no LUT associated with the image, M_INVALID is returned.</p> <p>(summarize)</p>

You can inquire the following values for [M_IMAGE](#) files (images or AVI sequences).

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_INT*.

● For M_IMAGE	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ASPECT_RATIO	<p>Returns the aspect ratio of the images in the file.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Data type: MIL_DOUBLE </div>
<input type="checkbox"/> M_COMPRESSION_TYPE	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Returns the compression type of the images in the file. Returns M_NULL if the images are not compressed (for example, in a BMP file format). See MbufAllocColor() for all possible compression formats.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> </div>

	Return values: M_JPEG_LOSSLESS; M_JPEG2000_LOSSLESS; M_JPEG_LOSSLESS_INTERLACED; M_JPEG_LOSSY; M_JPEG2000_LOSSY; M_JPEG_LOSSY_INTERLACED; M_MPEG4; (details)
<input type="checkbox"/> M_FRAME_RATE	<p>Returns the frame rate (number of images/second) of an AVI file.</p> <p>This value is only valid for sequences.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Data type: MIL_DOUBLE</p>
<input type="checkbox"/> M_LUT_PRESENT	<p>Returns the presence of LUT data in the file.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Return values:</p> <p>M_NO LUT data is not available.</p> <p>M_YES LUT data is available.</p>
<input type="checkbox"/> M_NUMBER_OF_IMAGES	<p>Returns the number of images in an AVI file.</p> <p>This value is only valid for sequences.</p> <p>(summarize)</p>

You can inquire the following values for [M_KERNEL](#) and [M_STRUCT_ELEMENT](#) files.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_INT](#).

• For M_KERNEL and M_STRUCT_ELEMENT	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_OFFSET_CENTER_X	Returns the X-coordinate of the center of the kernel or structuring element.
<input type="checkbox"/> M_OFFSET_CENTER_Y	Returns the Y-coordinate of the center of the kernel or structuring element.
<input type="checkbox"/> M_OVERSCAN	<p>Returns the overscan type.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Return values: M_DEFAULT; M_DISABLE; M_FAST; M_MIRROR; M_REPLACE; M_TRANSPARENT; (details)</p>
<input type="checkbox"/> M_OVERSCAN_REPLACE_VALUE	Returns the overscan replace value.

You can inquire the following values for [M_KERNEL](#) files.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_INT](#).

• For M_KERNEL	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ABSOLUTE_VALUE	<p>Returns the absolute value setting.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_FILTER_MODE	<p>Returns the mode in which the neighborhood operations are performed.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Return values: M_KERNEL; M_RECURSIVE; (details)</p>
<input type="checkbox"/> M_FILTER_OPERATION	Returns the type of neighborhood operation performed using the selected filter.

	<p>If <code>MbufControlNeighborhood()</code> <code>M_FILTER_TYPE</code> is set to <code>M_USER_DEFINED</code>, then <code>M_FILTER_OPERATION</code> is not supported. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>M_EDGE_DETECT</code>; <code>M_EDGE_DETECT_SQR</code>; <code>M_FIRST_DERIVATIVE_X</code>; <code>M_FIRST_DERIVATIVE_Y</code>; <code>M_HORIZ_EDGE</code>; <code>M_LAPLACIAN_EDGE</code>; <code>M_SECOND_DERIVATIVE_X</code>; <code>M_SECOND_DERIVATIVE_XY</code>; <code>M_SECOND_DERIVATIVE_Y</code>; <code>M_SHARPEN</code>; <code>M_SMOOTH</code>; <code>M_VERT_EDGE</code>; (details)</p>
<input type="checkbox"/> <code>M_FILTER_SMOOTHNESS</code>	<p>Returns the degree of smoothness (strength of the denoising) applied by the filter during the neighborhood operation. If <code>MbufControlNeighborhood()</code> <code>M_FILTER_TYPE</code> is set to <code>M_USER_DEFINED</code>, then <code>M_FILTER_SMOOTHNESS</code> is not supported. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 to 100; (details)</p>
<input type="checkbox"/> <code>M_FILTER_TYPE</code>	<p>Returns the type of filter used to perform the neighborhood operation. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>M_DERICHE</code>; <code>M_SHEN</code>; <code>M_USER_DEFINED</code>; (details)</p>
<input type="checkbox"/> <code>M_NORMALIZATION_FACTOR</code>	<p>Returns the normalization factor. (summarize)</p> <p><i>UserVarPtr info</i> Return values: Value > 0; (details)</p>
<input type="checkbox"/> <code>M_SATURATION</code>	<p>Returns the saturation flag. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>M_DISABLE</code>; <code>M_ENABLE</code>; (details)</p>

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- `MIL_DOUBLE`
- `MIL_INT`

Specifies the address in which to write the requested information.

Since the `MbufDiskInquire()` function also returns the requested information, you can set this parameter to `M_NULL`.

Return value

The returned value is the requested information about the buffer data in the specified file, cast to `MIL_INT`. If the requested information is not available, `M_INVALID` is returned.

Remarks

- *[MIL-Lite]*
Note that MIL-Lite does not support the inquiries available for `M_KERNEL` and `M_STRUCT_ELEMENT` data buffers.
- *[MIL-Lite]*
Note that MIL-Lite compression support is reliant upon the presence of Matrox compression acceleration hardware; the compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufExport

Synopsis

Export a data buffer to a file.

Syntax

```
void MbufExport(
    MIL_CONST_TEXT_PTR Filename,
    MIL_INT FileFormat,
    MIL_ID SrcBufId
)
```

Description

This function exports a data buffer to a file, in the specified output file format.

Note that you can also save a buffer in an [M_MIL](#) file format, using [MbufSave\(\)](#). The [M_MIL](#) file format is TIFF compatible.

To export an image with a LUT (color palette), associate the LUT to the image, using [MbufControl\(\)](#). Upon export, the image is saved with its associated color palette (MIM, TIFF, and BMP file formats).

[MIL-Lite with restriction]
When exporting uncompressed data to a file in an [M_JPEG_...](#) or [M_JPEG2000_...](#) file format, this function will automatically compress the data, according to the specified file format. The buffer does not need an [M_COMPRESS](#) attribute. If you are exporting compressed data to a file in an uncompressed file format, this function will automatically decompress the data.

Parameters

Filename

Specifies the name and path of the file in which to save the data buffer. It is recommended that you use the MIM file extension for easier use with other Matrox Imaging software products. The function handles (internally) the opening and closing of the file. If the file already exists, it will be overwritten.

This parameter can be set to one of the following:

● For specifying the file name and path				
<input type="checkbox"/> Value	Description			
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p> <table><tr><td>Parameters</td></tr><tr><td>FileName</td></tr><tr><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters	FileName	Specifies the drive, directory, and name of the file.
Parameters				
FileName				
Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<p>[This is only applicable to Windows]</p> <p>Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</p>			

FileFormat

Specifies the file format. This parameter can be set to one of the following values. Note that, except for the [M_MIL](#) and [M_RAW](#) file formats, the source buffer should have an [M_IMAGE](#) attribute; otherwise, the data is saved as if it were image data.

When [M_INTERACTIVE](#) is specified, these values can also be OR'ed together, using the OR operator, to use combinations of various file formats. This allows you to put the combined files' extensions, each on a separate line, in the file

type field of the **File Save As** dialog box. **M_DEFAULT** will list all the file extensions and **M_COMPRESS** is equivalent to combining **M_JPEG_...** and **M_JPEG2000_...**

Note **M_BMP**, **M_MIL**, **M_RAW**, **M_TIFF**, **M_COMPRESS** or **M_DEFAULT** cannot be combined with **M_MIL+M_PLANAR**, **M_TIFF+M_PLANAR**, **M_JPEG_...** or **M_JPEG2000_...**

● For specifying the file format	
☐ Value	Description
☐ M_BMP	<p>Saves the image buffer contents in a BMP file format. The standard Windows BMP format is used.</p> <p>This file format supports images that are grayscale or RGB. Images in a different format are internally converted before being saved.</p> <p>This file format supports only 8-bit buffers. If you are saving a non 8-bit buffer in this file format, only the 8 least-significant bits are saved.</p> <p>(summarize)</p>
☐ M_JPEG2000_LOSSLESS	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Saves the image buffer contents in a JPEG2000 lossless file format. If the buffer is 3-band and does not have an M_COMPRESS + M_JPEG2000_LOSSLESS attribute, the data will be stored in RGB format; otherwise, it will be stored in the same color format as the buffer.</p> <p>This file format supports only 8- and 16-bit buffers. 1-bit buffers are unpacked and saved as 8-bit buffers. 32-bit buffers are saved as 16-bit buffers; only the 16-least significant bits are saved. In the case of a 32-bit buffer, it is best to shift the buffer by 16 bits to the right before exporting the image to keep the most significant data.</p> <p>(summarize)</p>
☐ M_JPEG2000_LOSSLESS_JP2	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Saves the image buffer contents in a JPEG2000 lossless file format using the standard JP2 format. Only the color specification of the image buffer can be specified from the additional JP2 information. If the buffer is 3-band and does not have an M_COMPRESS + M_JPEG2000_LOSSLESS attribute, the data will be stored in RGB format; otherwise, it will be stored in the same color format as the buffer.</p> <p>This file format supports only 8- and 16-bit buffers. 1-bit buffers are unpacked and saved as 8-bit buffers. 32-bit buffers are saved as 16-bit buffers; only the 16-least significant bits are saved. In the case of a 32-bit buffer, it is best to shift the buffer by 16 bits to the right before exporting the image to keep the most significant data.</p> <p>(summarize)</p>
☐ M_JPEG2000_LOSSY	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Saves the image buffer contents in a JPEG2000 lossy file format. If the buffer is 3-band and does not have an M_COMPRESS + M_JPEG2000_LOSSY attribute, the data will be stored in YUV16 planar format if the buffer is 8-bit or in RGB format if the buffer is 16-bit; otherwise, it will be stored in the same color format as the buffer.</p> <p>This file format supports only 8- and 16-bit buffers. 1-bit buffers are unpacked and saved as 8-bit buffers. 32-bit buffers are saved as 16-bit buffers; only the 16-least significant bits are saved. In the case of a 32-bit buffer, it is best to shift the buffer by 16 bits to the right before exporting the image to keep the most significant data.</p> <p>(summarize)</p>
☐ M_JPEG2000_LOSSY_JP2	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Saves the image buffer contents in a JP2000 lossy file format using the standard JP2 file format. Only the color specification of the image buffer can be specified from the additional JP2 information. If the buffer is 3-band and does not have an M_COMPRESS + M_JPEG2000_LOSSY attribute, the data will be stored in YUV16 planar format if the buffer is 8-bit or in RGB format if the buffer is 16-bit; otherwise, it will be stored in the same color format as the buffer.</p> <p>This file format supports only 8- and 16-bit buffers. 1-bit buffers are unpacked and saved as 8-bit buffers. 32-bit buffers are saved as 16-bit buffers; only the 16-least significant bits are saved. In the case of a 32-bit buffer, it is best to shift the buffer by 16 bits to the right before exporting the image to keep the most significant data.</p> <p>(summarize)</p>
☐ M_JPEG_LOSSLESS	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Saves the image buffer contents in a JPEG lossless file format. If the buffer is 3-band, the data will be stored in RGB format.</p> <p>This file format supports only 8- and 16-bit buffers. 1-bit buffers are unpacked and saved as 8-bit buffers. 32-bit buffers are saved as 16-bit buffers; only the 16-least significant bits are saved. In the case of a 32-bit buffer, it is best to shift the buffer by 16 bits to the right before exporting the image to keep the most significant data.</p> <p>(summarize)</p>
☐ M_JPEG_LOSSLESS_INTERLACED	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Saves the image buffer contents in an interlaced JPEG lossless file format. This file format is only applicable to 1-band image buffers.</p> <p>This file format supports only 8- and 16-bit buffers. 1-bit buffers are unpacked and saved as 8-bit buffers. 32-bit buffers are saved as 16-bit buffers; only the 16-least significant bits are saved. In the case of a 32-bit buffer, it is best to shift the buffer by 16 bits to the right before exporting the image to keep the most significant data.</p> <p>(summarize)</p>
☐ M_JPEG_LOSSY	<p>[For essential MIL-Lite information, see remarks.]</p> <p>Saves the image buffer contents in a JPEG lossy file format. If the buffer is 3-band and does not have an M_COMPRESS + M_JPEG_LOSSY attribute, the data will be stored in YUV16 packed format; otherwise, it will be stored in the same color format as the buffer.</p>

	This file format supports only 8-bit buffers. If you are saving a non 8-bit buffer in this file format, only the 8 least-significant bits are saved. (summarize)
<input type="checkbox"/> M_JPEG_LOSSY_INTERLACED	[For essential MIL-Lite information, see remarks .] Saves the image buffer contents in an interlaced JPEG lossy file format. If the buffer is 3-band, the data will be stored in YUV16 packed format. This file format supports only 8-bit buffers. If you are saving a non 8-bit buffer in this file format, only the 8 least-significant bits are saved. (summarize)
<input type="checkbox"/> M_JPEG_LOSSY_RGB	[For essential MIL-Lite information, see remarks .] Saves the image buffer contents in a JPEG lossy file format. If the buffer is 3-band, the data will be stored in RGB format. This file format supports only 8-bit buffers. If you are saving a non 8-bit buffer in this file format, only the 8 least-significant bits are saved. (summarize)
<input type="checkbox"/> M_MIL +	Saves the buffer contents in a MIL file format. It uses a baseline TIFF 6.0 file format with extra information included in the comment field. The buffer attributes and data type are also saved in the file. This file format supports images that are binary, grayscale, and RGB, as well as palette-color images. The MIL file format is a single-page, uncompressed TIFF file format. By default, most color image buffers are saved in a packed (chunky) format (in accordance with baseline TIFF 6.0 specifications). Color binary buffers are saved in a 1-bit per pixel format (data is stored in a 3-band, packed binary format). (summarize)
<input type="checkbox"/> M_RAW	Saves the buffer contents as raw data. The contents are dumped directly (byte stream) into the file and no header is added. If the buffer is multi-band, the buffer is saved in planar format (one band after another). (summarize)
<input type="checkbox"/> M_TIFF +	Saves the image buffer contents in a TIFF file format. The TIFF file format that is used respects the baseline TIFF 6.0 specification. The TIFF file format that is used supports images that are binary, grayscale, and RGB, as well as palette-color images. Also, this TIFF format supports any data type and depth that MIL supports for buffers. An image buffer cannot be saved using a multi-page TIFF file format; it is always saved as a single-page TIFF file. In addition, MIL only supports uncompressed TIFF files. By default, most color image buffers are saved in a packed (chunky) format (in accordance with baseline TIFF 6.0 specifications). Color binary buffers are saved in a 1-bit per pixel format (data is stored in a 3-band, packed binary format). (summarize)

Combination constant for [M_MIL](#); [M_TIFF](#);

You can add the following value to the above-mentioned values to set whether to save the color image buffer in a planar format.

● For M_MIL or M_TIFF	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_PLANAR	Saves the color image buffer in a planar format.

SrcBufId

Specifies the identifier of the data buffer to save.

Remarks

- This function is optimized for packed binary buffers.
- [MIL-Lite]
Note that MIL-Lite compression support, particularly for [M_IMAGE](#) + [M_COMPRESS](#) buffer type, is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufExportSequence

Synopsis

Export a sequence of image buffers to an AVI file.

Syntax

```
void MbufExportSequence(  
    MIL_CONST_TEXT_PTR Filename,  
    MIL_INT FileFormat,  
    const MIL_ID *BufArrayPtr,  
    MIL_INT NumberOfImages,  
    MIL_DOUBLE FrameRate,  
    MIL_INT ControlFlag  
)
```

Description

This function exports a sequence of image buffers to an audio video interleave (AVI) file.

Parameters

Filename

Specifies the name and path of the AVI file. The function handles (internally) the opening and closing of the file. If the file already exists, it will be overwritten.

This parameter can be set to one of the following:

For the name and path of the AVI file					
Value	Description				
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p> <table><tr><th colspan="2">Parameters</th></tr><tr><th>FileName</th><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters		FileName	Specifies the drive, directory, and name of the file.
Parameters					
FileName	Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<p>[This is only applicable to Windows]</p> <p>Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</p>				

FileFormat

Specifies the format of the file. This parameter can be set to one of the values below.

When **M_INTERACTIVE** is specified, these values can also be OR'ed together, using the OR operator, to use combinations of various file formats. This allows you to put the combined files' extensions, each on a separate line, in the file type field of the **File Save As** dialog box. **M_DEFAULT** will list all the file extensions.

For specifying the file format	
Value	Description
<input type="checkbox"/> M_DEFAULT	MIL automatically decides the appropriate format.

<input type="checkbox"/> M_AVI_DIB	Specifies an AVI format used to hold non-compressed DIB image buffers. If necessary, the image buffers will be converted to a non-compressed DIB format before exporting. This type of sequence is supported by Windows Media Player. (summarize)
<input type="checkbox"/> M_AVI_MIL	Specifies an AVI format used to hold image buffers in their MIL format. Since this function saves images in the format in which they are sent, it can be used with any format. Also, since the images are saved "as is", no additional loss is introduced in the images. To be supported by Windows Media Player, this type of sequence requires a codec. (summarize)
<input type="checkbox"/> M_AVI_MJPEG	[For essential MIL-Lite information, see remarks .] Specifies an AVI format used to hold JPEG compressed sequences. When this format is specified, the image buffers will be in YUV16 packed format. In addition, image buffers must have a width that is a multiple of 16 pixels. For image buffers that have the M_JPEG_LOSSY compression type, the height must be a multiple of 8, and less than or equal to 240 pixels. For image buffers that have the M_JPEG_LOSSY_INTERLACED compression type, the height must be a multiple of 16 pixels, greater than 240 pixels. If the image buffers are not already in these dimensions and format, MIL will automatically convert them appropriately. To be supported by Windows Media Player, this type of sequence requires a codec or DirectShow 8.0 (or better); DirectShow 8.1 is included with Windows XP. (summarize)

BufArrayPtr

Specifies the address of the array containing the MIL identifiers of the image buffers to export.

NumberOfImages

Specifies the number of image buffers to export. If the supplied array is larger than this number, the remaining buffer identifiers are ignored.

FrameRate

Specifies the frame rate (number of image buffers/second) of the sequence.

ControlFlag

Specifies whether to append the image buffers to the AVI file, if the file already exists, or overwrite the file. This parameter can be set to one of the following:

● For specifying whether to append the image or overwrite the file	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Overwrites the file. The file will be opened, written into, and then the file will be closed. (summarize)
<input type="checkbox"/> M_APPEND	Appends the image buffers to the file. The file will be opened, the specified images will be appended, and then the file will be closed. (summarize)
<input type="checkbox"/> M_CLOSE	Closes the AVI file. BufArrayPtr , NumberOfImages , and FrameRate should be set to M_NULL . (summarize)
<input type="checkbox"/> M_OPEN	Opens the AVI file for writing, and set the pointer to the beginning of the file. Note that if M_APPEND is added to M_OPEN , the file is opened and the file pointer is set to the end of the file. BufArrayPtr , NumberOfImages , and FrameRate should be set to M_NULL . (summarize)
<input type="checkbox"/> M_WRITE	Writes the specified number of images in the files starting from the current file pointer position. After the write operation, the file pointer is left at the end of the file, ready for the next M_WRITE operation. BufArrayPtr , NumberOfImages , and FrameRate should be set to the appropriate values. (summarize)

Remarks

- [MIL-Lite]
Note that MIL-Lite compression support, particularly for [M_AVI_MJPEG](#) format, is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.
- When a "remote" path is specified (the [Filename](#) parameter begins with "remote:///"), the first element of the array that contains the MIL identifiers must point to the remote system, when the AVI file is being opened or closed ([M_OPEN](#) or [M_CLOSE](#)). The buffers are allocated on this same remote system.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufFree

Synopsis

Free a data buffer.

Syntax

```
void MbufFree(
    MIL_ID BufId
)
```

Description

This function deallocates a previously allocated data buffer. The memory reserved for the specified buffer is released.

Child buffers associated to a parent buffer must be deallocated, using **MbufFree()**, prior to deallocating the parent buffer.

Note that LUT buffers, once associated with another buffer, a digitizer, or a display, should either be disassociated before being freed, or freed after the associated buffer, digitizer, or display is freed.

To disassociate a display or digitizer LUTs, use [MdigLut\(\)](#) or [MdispLut\(\)](#) with [M_DEFAULT](#). In the case of buffers, use [MbufControl\(\)](#) with [M_ASSOCIATED_LUT](#) set to [M_DEFAULT](#).

Using **MbufFree()** on a buffer created with either [MbufCreateColor\(\)](#) or [MbufCreate2d\(\)](#) will free the internal structure required to map the buffer to existing memory, but **MbufFree()** will not free the allocated memory used by the created buffer.

Parameter

BufId

Specifies the identifier of the data buffer to deallocate.

Remark

- If you are creating a DLL that includes a call to **MbufFree()**, ensure that the call is not made from the **DllMain()** function, because **MbufFree()** might unload any DLL loaded with [MbufAlloc1d\(\)](#), [MbufAlloc2d\(\)](#), or [MbufAllocColor\(\)](#) and you cannot unload a DLL from **DllMain()**. If necessary, call **MbufFree()** from a clean-up function in your DLL instead.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufGet1d

Synopsis

Get data from a 1D area of a buffer and place it in a user-supplied array.

Syntax

```
void MbufGet1d(
    MIL_ID SrcBufId,
    MIL_INT OffX,
    MIL_INT SizeX,
    void *UserArrayPtr
)
```

Description

This function copies data from a specified one-dimensional area of a MIL source buffer to a user-supplied array.

Note, if the source buffer is compressed, the data is decompressed before it is copied to the destination array.

Note, for multi-band buffers, this function linearly copies the data from the one-dimensional region of each band (RRR...GGG...BBB...).

Parameters

SrcBufId

Specifies the identifier of the source buffer.

OffX

Specifies the horizontal offset of the required area, relative to the top-left coordinate of the source buffer.

SizeX

Specifies the width of the required area of the source buffer.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer

Specifies the address of the user array in which to copy the data from the source buffer. Ensure that the user array is large enough to accommodate the data to be copied from the source buffer. **MbufGet1d()** assumes that the array is of the same data type as the source buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufGet2d

Synopsis

Get data from a 2d area of a buffer and place it in a user-supplied array.

Syntax

```
void MbufGet2d(  
    MIL_ID SrcBufId,  
    MIL_INT OffX,  
    MIL_INT OffY,  
    MIL_INT SizeX,  
    MIL_INT SizeY,  
    void *UserArrayPtr  
)
```

Description

This function copies data from a specified two-dimensional region of a MIL source buffer to a user-supplied array.

Note, if the source buffer is compressed, the data is decompressed before it is copied to the destination array.

Note, for multi-band buffers, this function linearly copies the data from the specified two-dimensional region of each band (RRR...GGG...BBB...).

Parameters

SrcBufId

Specifies the identifier of the source buffer.

OffX

Specifies the horizontal offset of the required area, relative to the top-left coordinate of the source buffer.

OffY

Specifies the vertical offset of the required area, relative to the top-left coordinate of the source buffer.

SizeX

Specifies the width of the required area of the source buffer.

SizeY

Specifies the height of the required area of the source buffer.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer

Specifies the address of the user array in which to copy the data from the source buffer. Ensure that the user array is large enough to accommodate the data to be copied. **MbufGet2d()** assumes that the array is of the same data type and depth as the source buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufGet

Synopsis

Get data from a buffer and place it in a user-supplied array.

Syntax

```
void MbufGet(  
    MIL_ID SrcBufId,  
    void *UserArrayPtr  
)
```

Description

This function copies data from a specified MIL source buffer to a user-supplied array.

Note, if the source buffer is compressed, the data is decompressed before it is copied to the destination array.

Note, for multi-band buffers, **MbufGet()** behaves like **MbufGetColor()** with its **DataFormat** parameter set to **M_PLANAR**, and its **Band** parameter set to **M_ALL_BANDS**. Refer to **MbufGetColor()** for more details.

Parameters

SrcBufId

Specifies the identifier of the source buffer.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer

Specifies the address of the user array in which to copy source buffer data. Ensure that the user array is large enough to accommodate the data from the source buffer. **MbufGet()** assumes that the array is of the same data type and depth as the source buffer's bands.

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufGetArc

Synopsis

Read the pixels along a specified arc and store their values in a user-defined array.

Syntax

```
void MbufGetArc(  
    MIL_ID ImageBufId,  
    MIL_INT XCenter,  
    MIL_INT YCenter,  
    MIL_INT XRad,  
    MIL_INT YRad,  
    double StartAngle,  
    double EndAngle,  
    void *UserArrayPtr,  
    MIL_INT *NbPixelsPtr  
)
```

Description

This function reads the series of pixels along an elliptic arc from an image and stores their values in a user-defined array. The ellipse is centered at (XCenter, YCenter) with radii XRad and YRad and is defined by the start angle StartAngle and the end angle EndAngle.

Note, if the source buffer is compressed, the data is decompressed before it is copied to the destination array.

Parameters

ImageBufId

Specifies the identifier of the source image buffer.

XCenter

Specifies the X-coordinate of the ellipse center, relative to the top-left coordinate of the source buffer.

YCenter

Specifies the Y-coordinate of the ellipse center, relative to the top-left coordinate of the source buffer.

XRad

Specifies the radius along the X-axis. The radius should be given in pixels and must be greater than 0.

YRad

Specifies the radius along the Y-axis. The radius should be given in pixels and must be greater than 0.

StartAngle

Specifies the angle at which to start reading the arc, moving in a counter-clockwise direction. Express the angle in degrees relative to the positive X-axis.

EndAngle

Specifies the angle at which to stop reading the arc, moving in a counter-clockwise direction. Express the angle in degrees relative to the positive X-axis.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer

Specifies the address of the user array in which to store the pixels from the image buffer. Ensure that the user array is large enough to accommodate the data to be stored. **MbufGetArc()** assumes that the array is of the same data type and depth as the source buffer. To determine the required size of the array, you can set this parameter to **M_NULL** and pass a non-null address to [NbPixelsPtr](#), which will store the value of the required size of the array. In this case, nothing is read from the image buffer.

NbPixelsPtr

Specifies the address of the variable in which to write the number of pixels found along the arc.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufGetColor2d

Synopsis

Get data from a region of one or all bands of a buffer and place it in a user-supplied array.

Syntax

```
void MbufGetColor2d(
    MIL_ID SrcBufId,
    MIL_INT DataFormat,
    MIL_INT Band,
    MIL_INT OffX,
    MIL_INT OffY,
    MIL_INT SizeX,
    MIL_INT SizeY,
    void *UserArrayPtr
)
```

Description

This function copies data from a specific region of one or all color bands of a specified MIL source buffer to a user-supplied array.

Note, if the source buffer is compressed, the data is decompressed before it is copied to the destination array.

Parameters

SrcBufId

Specifies the identifier of the source buffer.

DataFormat

Specifies the data format to use to save the data in the user array. The internal data format of the source buffer need not match the specified data format of the user-supplied array; an internal conversion will be performed if necessary. Note, however, if the formats do match, the operation will be much faster.

This parameter must be set to one of the following values. Note that Sx and Sy denote the source width and height, respectively.

Unless otherwise specified, the following values require that you pass the *UserArrayPtr* parameter the address of a MIL_INT.

For specifying the data format	
Value	Description
<input type="checkbox"/> M_PACKED +	Copies the bands in an interleaved manner (for example, RGB RGB RGB...). You must specify a combination value from the table below. (summarize)
<input type="checkbox"/> M_PLANAR	Copies the bands one after the other (for example, RRR...GGG...BBB...). This format is to be used when copying from all color bands of the source buffer. (summarize) <div><i>UserArrayPtr info</i> Data type: array of the same type as the buffer Array size: Sx x Sy x number of color bands of the source buffer</div>
<input type="checkbox"/> M_SINGLE_BAND	Copies a single color band. (summarize) <div><i>UserArrayPtr info</i></div>

Data type: array of the same type as the buffer
 Array size: $S_x \times S_y$
 Note: In the case of a [M_BGR32](#) buffer, point to an array of type *char*, since only one band will be copied.

Combination constants for [M_PACKED](#);

You must add one of the following values to the above-mentioned value to set the color buffer format.

See the [RGB buffers](#) section in [Chapter 18: Specifying and managing your data buffers](#).

Note that the source buffer must be a three-band, 8-bit bit buffer.

● For M_PACKED	
☐ Value	Description
☐ M_BGR24	<p>Stores the data in a BGR24 packed format. In this format, each pixel is internally stored as three consecutive bytes in little-endian order. (summarize)</p> <p><i>UserArrayPtr info</i> Data type: array of type <i>char</i> Array size: $S_x \times S_y \times 3$ bytes (<i>char</i>)</p>
☐ M_BGR32	<p>Stores the data in a BGR32 packed format. In this format, each pixel is internally stored as four consecutive bytes, in little-endian order. The most-significant byte is a "don't care" byte. (summarize)</p> <p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>MIL_INT32</i> Array size: $S_x \times S_y \times 4$ bytes (size of BGR32 pixel) • Data type: array of type <i>MIL_UINT32</i> Array size: $S_x \times S_y \times 4$ bytes (size of BGR32 pixel)
☐ M_RGB15	<p>Stores the data in a RGB15 packed format. In this format, each pixel is internally stored as a 16-bit word with a 5-bit blue value (least significant), a 5-bit green value, a 5-bit red value, and a "don't care" bit (most significant), in little-endian order (RGB 5:5:5). (summarize)</p> <p><i>UserArrayPtr info</i> Data type: array of type <i>MIL_INT16</i> Array size: $S_x \times S_y \times 2$ bytes (unsigned <i>char</i>)</p>
☐ M_RGB16	<p>Stores the data in a RGB16 packed format. In this format, each pixel is internally stored as a 16-bit word with a 5-bit blue value (least significant), a 6-bit green value, and a 5-bit red value (most significant), in little-endian order (RGB 5:6:5). (summarize)</p> <p><i>UserArrayPtr info</i> Data type: array of type <i>MIL_INT16</i> Array size: $S_x \times S_y \times 2$ bytes (unsigned <i>char</i>)</p>

Band

Specifies the index of the color band to copy. This parameter can be set to one of the values below.

● For specifying the index of the color band to copy							
☐ Value	Description						
☐ $0 \leq \text{Value} \leq \text{\#bands} - 1$	<p>Specifies the index of the band to copy. The relationship between index value and band for RGB and HSL buffers is the following:</p> <table> <tr> <td>0</td><td>Corresponds to the red band for RGB buffers or the hue band for HSL buffers.</td></tr> <tr> <td>1</td><td>Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.</td></tr> <tr> <td></td><td></td></tr> </table>	0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.	1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.		
0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.						
1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.						

	2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.
		(summarize)
<input type="checkbox"/> M_ALL_BANDS		Copies from all color bands.
<input type="checkbox"/> M_BLUE		Copies from the blue color band (for RGB buffers).
<input type="checkbox"/> M_GREEN		Copies from the green color band (for RGB buffers).
<input type="checkbox"/> M_HUE		Copies from the hue color band (for HSL buffers).
<input type="checkbox"/> M_LUMINANCE		Copies from the luminance color band (for HSL buffers).
<input type="checkbox"/> M_RED		Copies from the red color band (for RGB buffers).
<input type="checkbox"/> M_SATURATION		Copies from the saturation color band (for HSL buffers).

OffX

Specifies the horizontal offset (relative to the top-left source buffer coordinate) of the source buffer region from which to get the data.

OffY

Specifies the vertical pixel offset (relative to the top-left source buffer coordinate) of the source buffer region in which to get the data.

SizeX

Specifies the width of the source buffer region from which to get the data.

SizeY

Specifies the height of the source buffer region from which to get the data.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer
- array of type char
- array of type MIL_INT16
- array of type MIL_INT32
- array of type MIL_UINT32
- MIL_INT

Specifies the address of the user array from which to copy the data. Ensure that there are enough entries in the user array to receive the data of the specified source buffer region.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufGetColor

Synopsis

Get data from one or all bands of a buffer and place it in a user-supplied array.

Syntax

```
void MbufGetColor(
    MIL_ID SrcBufId,
    MIL_INT DataFormat,
    MIL_INT Band,
    void *UserArrayPtr
)
```

Description

This function copies data from one or all color bands of a specified MIL source buffer to a user-supplied array.

Note, if the source buffer is compressed, the data is decompressed before it is copied to the destination array.

Parameters

SrcBufId
Specifies the identifier of the source buffer.

DataFormat
Specifies the data format to use to save the data in the user array. The internal data format of the source buffer need not match the specified data format; an internal conversion will be performed if necessary. Note, however, if the formats do match, the operation will be much faster.

This parameter must be set to one of the following values. Note that Sx and Sy denote the source width and height, respectively.

Unless otherwise specified, the following values require that you pass the *UserArrayPtr* parameter the address of a MIL_INT.

● For specifying the data format	
☐ Value	Description
☐ M_PACKED +	Copies buffer bands in an interleaved manner and stores them consecutively (for example, RGB RGB RGB...). (summarize)
☐ M_PLANAR	Copies the bands one after the other (for example, RRR...GGG...BBB...). This format is to be used when copying all color bands of the source buffer. (summarize)
	<i>UserArrayPtr info</i> Data type: array of the same type as the buffer Array size: Sx x Sy x number of color bands of the source buffer
☐ M_SINGLE_BAND	Copies a single color band. (summarize)
	<i>UserArrayPtr info</i> Data type: array of the same type as the buffer Array size: Sx x Sy Note: In the case of a M_BGR32 buffer, point to an array of type <i>char</i> , since only one band will be copied.

Combination constants for **M_PACKED**;

You can add one of the following values to the above-mentioned value to set the data's color format.

For more information on these formats, see the [RGB buffers](#) section in [Chapter 18: Specifying and managing your data buffers](#).

Note that the source buffer must be a three-band, 8-bit bit buffer.

● For M_PACKED	
☐ Value	Description
☐ M_BGR24	<p>Stores the data in a BGR24 packed format. In this format, each pixel is internally stored as three consecutive bytes in little-endian order.</p> <p>(summarize)</p> <p><i>UserArrayPtr info</i> Data type: array of type char Array size: Sx x Sy x 3 bytes (char)</p>
☐ M_BGR32	<p>Stores the data in a BGR32 packed format. In this format, each pixel is internally stored as four consecutive bytes, in little-endian order. The most-significant byte is a "don't care" byte.</p> <p>(summarize)</p> <p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: Sx x Sy x 4 bytes (size of BGR32 pixel) • Data type: array of type MIL_UINT32 Array size: Sx x Sy x 4 bytes (size of BGR32 pixel)
☐ M_RGB15	<p>Stores the data in a RGB15 packed format. In this format, each pixel is internally stored as a 16-bit word with a 5-bit blue value (least significant), a 5-bit green value, a 5-bit red value, and a "don't care" bit (most significant), in little-endian order (RGB 5:5:5).</p> <p>(summarize)</p> <p><i>UserArrayPtr info</i> Data type: array of type MIL_INT16 Array size: Sx x Sy x 2 bytes (unsigned char)</p>
☐ M_RGB16	<p>Stores the data in a RGB16 packed format. In this format, each pixel is internally stored as a 16-bit word with a 5-bit blue value (least significant), a 6-bit green value, and a 5-bit red value (most significant), in little-endian order (RGB 5:6:5).</p> <p>(summarize)</p> <p><i>UserArrayPtr info</i> Data type: array of type MIL_INT16 Array size: Sx x Sy x 2 bytes (unsigned char)</p>

Band

Specifies the index of the color band to copy. This parameter can be set to one of the values below.

● For specifying the index of the color band to copy							
☐ Value	Description						
☐ 0 <= Value <= #bands - 1	<p>Specifies the index of the band to copy.</p> <p>The relationship between index value and band for RGB and HSL buffers is the following:</p> <table border="1"> <tr> <td>0</td><td>Corresponds to the red band for RGB buffers or the hue band for HSL buffers.</td></tr> <tr> <td>1</td><td>Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.</td></tr> <tr> <td>2</td><td>Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.</td></tr> </table> <p>(summarize)</p>	0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.	1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.	2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.
0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.						
1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.						
2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.						
☐ M_ALL_BANDS	Copies from all color bands.						
☐ M_BLUE	Copies from the blue color band (for RGB buffers).						

<input type="checkbox"/> M_GREEN	Copies from the green color band (for RGB buffers).
<input type="checkbox"/> M_HUE	Copies from the hue color band (for HSL buffers).
<input type="checkbox"/> M_LUMINANCE	Copies from the luminance color band (for HSL buffers).
<input type="checkbox"/> M_RED	Copies from the red color band (for RGB buffers).
<input type="checkbox"/> M_SATURATION	Copies from the saturation color band (for HSL buffers).

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer
- array of type char
- array of type MIL_INT16
- array of type MIL_INT32
- array of type MIL_UINT32
- MIL_INT

Specifies the address of the user array in which to copy data from the source buffer. Ensure that the user array is large enough to accommodate the data from the source buffer in the data type and depth specified.

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufGetHookInfo

Synopsis

Get information about a hook event.

Syntax

```
MIL_INT MbufGetHookInfo(
    MIL_ID EventId,
    MIL_INT InfoType,
    void *UserVarPtr
)
```

Description

This function allows you to get information about the event that caused the hook handler function to be called. **MbufGetHookInfo()** should only be called within the scope of a buffer hook-handler function (see [MbufHookFunction\(\)](#)).

Parameters

EventId

Specifies the buffer event identifier received by the hook-handler function (see [MbufHookFunction\(\)](#)).

InfoType

Specifies the event type and the type of information about the event to return.

Set this parameter to the following event type:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

For the event type	
Value	Description
M_MODIFIED_BUFFER +	Identifies a modified buffer type of event. This event occurs if the hook-handler function was hooked using MbufHookFunction() with M_MODIFIED_BUFFER . You must specify a combination value from the table below . (summarize)

Combination constants for **M_MODIFIED_BUFFER**;

You must add one of the following values to the above-mentioned value to set the type of information to return.

For M_MODIFIED_BUFFER	
Value	Description
	solos gige (v) solos ed/xcl (u) solos ea/xa (t) odyssey ed/xd (s) odyssey ed/xcl (r) odyssey ea/xa (n) nexis (p) morphis qxt (o) morphis (n) met-II/std (m) met-II/mc (l) met-II/dig (k) met-II/cl (j) iris (i) itee I394 hlc (h) helios ed/xd (g) helios ed/xcl (f) helios ea/xa (e) gpu processing (d) gige vision (c) cronosplus (b) corona-II (a)
M_BUFFER_ID	Returns the MIL identifier of the modified buffer. (summarize)
	UserVarPtr info

[illegible]

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- MIL_DOUBLE
- MIL_ID
- MIL_INT

Specifies the address in which to write the requested information.

Return value

Returns null (**M_NULL**) on success, and returns a non-null (!**M_NULL**) value on failure, without logging any errors in the application.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufGetLine

Synopsis

Read the pixels along a specified theoretical line, count the pixels, and store their values in a user-defined array.

Syntax

```
void MbufGetLine(  
    MIL_ID ImageBufId,  
    MIL_INT StartX,  
    MIL_INT StartY,  
    MIL_INT EndX,  
    MIL_INT EndY,  
    MIL_INT Mode,  
    MIL_INT *NbPixelsPtr,  
    void *UserArrayPtr  
)
```

Description

This function reads the series of pixels along a theoretical line (defined by coordinates) from an image and stores their values in a user-defined array. The Bresenham algorithm is used to determine the theoretical line.

Note, if the source buffer is compressed, the data is decompressed before it is copied to the destination array.

Parameters

ImageBufId

Specifies the identifier of the source image buffer. This must be a single-band (monochrome) buffer.

StartX

Specifies the horizontal pixel offset of the starting position of the line, relative to the top-left pixel of the source buffer.

StartY

Specifies the vertical pixel offset of the starting position of the line, relative to the top-left pixel of the source buffer.

EndX

Specifies the horizontal pixel offset of the finishing position of the line, relative to the top-left pixel of the source buffer.

EndY

Specifies the vertical pixel offset of the finishing position of the line, relative to the top-left pixel of the source buffer.

Mode

Specifies the operation mode. This parameter must be set to **M_DEFAULT**.

NbPixelsPtr

Specifies the address of the variable in which to write the number of pixels found along the theoretical line. You can set this parameter to **M_NULL** if you don't want this value to be evaluated.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer

Specifies the address of the user array in which to store the pixels from the image buffer. Ensure that the user array is large enough to accommodate the data to be stored. To determine the required size of the array, you can set this parameter to **M_NULL** and pass a non-null address to [NbPixelsPtr](#). In this case, nothing is read from the image buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufHookFunction

Synopsis

Hook a function to a buffer event.

Syntax

```
void MbufHookFunction(
    MIL_ID BufferId,
    MIL_INT HookType,
    MIL_BUF_HOOK_FUNCTION_PTR HookHandlerPtr,
    void *UserDataPtr
)
```

Description

This function allows you to attach or detach a user-defined function to a specified buffer event (for example, a modification to the buffer's contents). Once a hook-handler function is defined and hooked to an event, it is automatically called when the event occurs.

You can hook more than one function to an event by making separate calls to **MbufHookFunction()** for each function that you want to hook. MIL automatically chains and keeps an internal list of all these hooked functions. When a function is hooked, this new function is added to the end of the list. When the event happens, all user-defined functions in the list will be executed in the same order that they were hooked to the event. You can also remove any function from the list; in this case, MIL preserves the order of the remaining functions in the list.

Parameters

BufferId

Specifies the identifier of the buffer.

HookType

Specifies the buffer event to which to hook the function. This parameter can be set to following value:

● For specifying the buffer event	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MODIFIED_BUFFER +	Calls the hook-handler function each time the specified buffer is modified by a MIL function.

Combination constant for the values listed in [For specifying the buffer event](#)

You can add the following value to the above-mentioned value to set whether to detach the hook-handler function.

● For M_MODIFIED_BUFFER	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_UNHOOK	Unhooks the specified function if hooked to an M_MODIFIED_BUFFER event.

HookHandlerPtr

Specifies the address of the function that should be called when the specified event occurs. The hook-handler function must be declared as follows:

Upon successful completion, the hook-handler function should return **M_NULL**. Note MFTYPE and MIL_BUF_HOOK_FUNCTION_PTR are reserved MIL predefined types for functions and data pointers.

```
MIL_INT MFTYPE HookHandler(  
    MIL_INT HookType,  
    MIL_ID EventId,  
    void *UserDataPtr  
)  
  
Parameters:  
  
    HookType  
        Type of buffer event that generated the call.  
  
    EventId  
        Event identifier to pass to MbufGetHookInfo\(\) when inquiring about the hooked event.  
  
    UserDataPtr  
        User data pointer that was passed (as UserDataPtr) to MbufHookFunction\(\).
```

UserDataPtr

Specifies the address of the user data that you want to make available to the hook-handler function. This address is passed to the hook-handler function, through its [UserDataPtr](#) parameter, when the specified event occurs. Set this parameter to **M_NULL** if not used.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufImport

Synopsis

Import data from a file into a data buffer.

Syntax

```
MIL_ID MbufImport (
    MIL_TEXT_PTR Filename,
    MIL_INT FileFormat,
    MIL_INT Operation,
    MIL_ID SystemId,
    MIL_ID *BufIdPtr
)
```

Description

This function imports data from a file into an existing or automatically allocated MIL data buffer. The function assumes that the data in the file is in the specified format.

Note, you can also import data using [MbufLoad\(\)](#) or [MbufRestore\(\)](#); however, these functions try to determine the format from the data rather than allowing you to specify the format.

For an [M_RESTORE](#) operation, this function will allocate the destination buffer with the same attributes as the original buffer, with the exception of an [M_IMAGE](#) buffer. In the case of an [M_IMAGE](#) buffer, the [MbufImport\(\)](#) function tries to allocate the buffer so that it can be used for acquisition ([M_GRAB](#)), display ([M_DISP](#)), and processing ([M_PROC](#)) operations. If there is insufficient appropriate memory to allocate such a buffer, it allocates one that can be used in all of the above operations except for acquisition ([M_GRAB](#)). Note that the maximum (total) number of grab ([M_GRAB](#)) buffers that can be allocated is restricted by the total amount of MIL non-paged (DMA) memory (specified at installation time or using MilConfig). For systems with on-board processors, the total number of [M_GRAB](#) buffers and [M_PROC](#) buffers is limited by the amount of on-board memory.

[MIL-Lite with restriction]

When performing an [M_RESTORE](#) operation on a file containing compressed data, the buffer will have an [M_COMPRESS](#) attribute.

[MIL-Lite with restriction]

When importing compressed data, the following applies. When performing an [M_LOAD](#) operation and the destination buffer has an [M_IMAGE](#) attribute (but not an [M_COMPRESS](#) attribute), this function will automatically decompress it. If necessary, the data in the file will be transformed to the format of the buffer. If unsure of the compression type of the data, use [M_DEFAULT](#) as the file format rather than [M_JPEG_...](#) or [M_JPEG2000_...](#); the data will be read correctly.

[MIL-Lite with restriction]

When importing uncompressed data into a buffer with an [M_COMPRESS](#) attribute, this function will automatically compress it, according to the compression settings found in the buffer.

When importing an image file that has been saved with an associated LUT (color palette) into a 3-band 8-bit image buffer, the LUT is automatically applied to the data to generate 3-band image data. In this case, a LUT buffer is not created and, therefore, is not associated to the 3-band 8-bit buffer. When importing an image file that has been saved with an associated LUT (color palette) into any other type of image buffer, the LUT is also imported and associated with the resulting image buffer. You can obtain the identifier of the associated LUT, using [MbufInquire\(\)](#).

Note that the associated LUT will be automatically selected on the display ([MdispLut\(\)](#)) if the image buffer is selected on a display and the default LUT has not been overridden by a former call to [MdispLut\(\)](#).

Using [MbufDiskInquire\(\)](#), you can inquire about the dimensions of the data saved in a file (except for raw files) without importing it.

After restoring a buffer, we recommend that you check if the operation was successful, using [MappGetError\(\)](#), or by verifying that the returned buffer identifier is not [M_NULL](#).

Parameters

Filename

Specifies the name and path of the file from which to import the data. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following values.

- For specifying the file name and path

<div><div></div>Value</div>	Description
<div><div><div></div><div>MIL_TEXT(<div>MIL_TEXT_PTR <i>FileName</i></div>)</div></div></div>	<div><div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</div><div>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile".</div><div><div>(summarize)</div></div><div><div><div></div><div>Parameters</div></div><div><div></div><div>FileName</div><div>Specifies the drive, directory, and name of the file.</div></div></div></div>
<div><div><div></div><div>M_INTERACTIVE</div></div></div>	<div><div>[This is only applicable to Windows]</div><div>Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.</div></div>

FileFormat

Specifies the file format. This parameter can be set to one of the following values. Note that except for the [M_MIL](#) and [M_RAW](#) file formats, data is treated as image data. When performing an [M_LOAD](#) operation, the data is internally converted appropriately.

When [M_INTERACTIVE](#) is specified, these values can also be OR'ed together, using the OR operator, to use combinations of various file formats. This allows you to put the combined files' extensions, each on a separate line, in the file type field of the **File Open** dialog box. [M_DEFAULT](#) will list all the file extensions and [M_COMPRESS](#) is equivalent to combining [M_JPEG_...](#) and [M_JPEG2000_...](#)

Note [M_BMP](#), [M_MIL](#), [M_RAW](#), [M_TIFF](#), [M_COMPRESS](#) or [M_DEFAULT](#) cannot be combined with [M_JPEG_...](#) or [M_JPEG2000_...](#)

● For specifying the file format	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	<p>Specifies that MIL automatically determines the file format. If the file format is not supported, its data will be treated as raw data.</p> <p>(summarize)</p>
<input type="checkbox"/> M_BMP	<p>Imports image data that is in BMP file format. The standard Windows BMP format is used.</p> <p>(summarize)</p>
<input type="checkbox"/> M_JPEG2000_LOSSLESS	<p>[<i>For essential MIL-Lite information, see remarks.</i>]</p> <p>Imports a JPEG2000 lossless image.</p>
<input type="checkbox"/> M_JPEG2000_LOSSLESS_JP2	<p>[<i>For essential MIL-Lite information, see remarks.</i>]</p> <p>Imports a JPEG2000 lossless image, as well as the additional JP2 information.</p>
<input type="checkbox"/> M_JPEG2000_LOSSY	<p>[<i>For essential MIL-Lite information, see remarks.</i>]</p> <p>Imports a JPEG2000 lossy image.</p>
<input type="checkbox"/> M_JPEG2000_LOSSY_JP2	<p>[<i>For essential MIL-Lite information, see remarks.</i>]</p> <p>Imports a JPEG2000 lossy image, as well as the additional JP2 information.</p>
<input type="checkbox"/> M_JPEG_LOSSLESS	<p>[<i>For essential MIL-Lite information, see remarks.</i>]</p> <p>Imports a JPEG lossless image.</p>
<input type="checkbox"/> M_JPEG_LOSSLESS_INTERLACED	<p>[<i>For essential MIL-Lite information, see remarks.</i>]</p> <p>Imports a JPEG lossless image stored in two separate fields.</p>
<input type="checkbox"/> M_JPEG_LOSSY	<p>[<i>For essential MIL-Lite information, see remarks.</i>]</p> <p>Imports a JPEG lossy image.</p>
<input type="checkbox"/> M_JPEG_LOSSY_INTERLACED	<p>[<i>For essential MIL-Lite information, see remarks.</i>]</p> <p>Imports a JPEG lossy image stored in two separate fields.</p>
<input type="checkbox"/> M_JPEG_LOSSY_RGB	<p>[<i>For essential MIL-Lite information, see remarks.</i>]</p> <p>Imports a 3-band JPEG lossy image that is in an RGB format.</p>
<input type="checkbox"/> M_MIL	<p>Imports data that is in a MIL file format. The baseline TIFF 6.0 specification is used.</p> <p>(summarize)</p>

<input type="checkbox"/> M_RAW	Imports raw data.
<input type="checkbox"/> M_TIFF	Imports image data that is in a TIFF file format. The baseline TIFF 6.0 specification is used. Only image formats, data types, and depths that are supported by MIL buffers will be accepted. Multi-page TIFF files can be imported into MIL buffers but only the first page is saved in the buffer. In addition, MIL only supports uncompressed TIFF files. (summarize)

Operation

Specifies the import operation. This parameter can be set to one of the following values.

● For specifying the import operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_LOAD	Imports data from the specified file into a previously allocated MIL data buffer.
<input type="checkbox"/> M_RESTORE	Imports data from the specified file into an automatically allocated MIL data buffer. Note, you cannot restore (M_RESTORE) a RAW data file (M_RAW) because its dimensions are unknown. (summarize)

SystemId

Specifies the system on which to allocate the buffer for an [M_RESTORE](#) operation. For an [M_LOAD](#) operation, this parameter should be set to **M_NULL**.

For an [M_RESTORE](#) operation, this parameter should be set to one of the following values:

● For specifying the system on which to allocate the buffer (with M_RESTORE)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

BufIdPtr

Specifies the address of the variable containing the buffer identifier, for an [M_LOAD](#) operation, or the address of the variable in which to store the new buffer identifier, for an [M_RESTORE](#) operation.

For an [M_LOAD](#) operation, the destination buffer must be large enough in depth and dimensions to hold the data. If the data is deeper than the buffer, the most-significant bits of the data are truncated when loaded into the buffer. If the buffer depth is greater than that of the data, the data is zero or sign-extended (depending on the data type) when loaded into the buffer. If the buffer is larger in size than the data, exceeding areas of the buffer are unaffected.

For an [M_RESTORE](#) operation, the destination buffer will be allocated with an appropriate size and type to hold the data. Since **MbufImport()** also returns the buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the buffer identifier (for an [M_RESTORE](#) operation only). If allocation fails, **M_NULL** is returned.

Remark

- *[MIL-Lite]*
Note that MIL-Lite compression support is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufImportSequence

Synopsis

Import a sequence of images from an AVI file into separate image buffers.

Syntax

```
void MbufImportSequence(
    MIL_CONST_TEXT_PTR Filename,
    MIL_INT FileFormat,
    MIL_INT Operation,
    MIL_ID SystemId,
    MIL_ID *BufArrayPtr,
    MIL_INT StartImage,
    MIL_INT NumberOfImages,
    MIL_INT ControlFlag
)
```

Description

This function imports a sequence of images from an AVI file into separate image buffers. **MbufImportSequence()** can automatically allocate the necessary buffers or you can use previously allocated buffers. In the latter case, the **BufArrayPtr** parameter should point to an array containing the buffer identifiers. In the former case, **MbufImportSequence()** will write the identifiers of the new buffers into the array pointed to by **BufArrayPtr**.

Rather than importing all the required images at once, you can also use this function to only open the AVI file for reading. Then, call this function as many times as required to import different sets of images from the file. Once you have finished importing images from the file, you must call this function again to close the file.

Parameters

Filename

Specifies the name and path of the AVI file. This function handles the opening and/or closing of the file depending on the **ControlFlag** parameter setting.

This parameter can be set to one of the following:

For specifying the name and path of the AVI file	
Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile". (summarize)</div> <div><div>Parameters</div><div>FileName</div><div>Specifies the drive, directory, and name of the file.</div></div>
<input type="checkbox"/> M_INTERACTIVE	<div>[This is only applicable to Windows] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.</div>

FileFormat

Specifies the format of the file. This parameter can be set to one of the values listed below.

When **M_INTERACTIVE** is specified, these values can also be OR'ed together, using the OR operator, to use combinations of various file formats. This allows you to put the combined files' extensions, each on a separate line, in the file type field of the **File Open** dialog box. **M_DEFAULT** will list all the file extensions.

● For specifying the file format	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that MIL automatically determines the file format.
<input type="checkbox"/> M_AVI_DIB	Specifies an AVI format containing non-compressed images.
<input type="checkbox"/> M_AVI_MIL	Specifies an AVI format containing images in their MIL format.
<input type="checkbox"/> M_AVI_MJPG	[For essential MIL-Lite information, see remarks .] Specifies an AVI format containing compressed images.

Operation

Specifies whether to import the specified sequence of images into automatically allocated buffers or previously allocated buffers. If not importing images, this parameter should be set to **M_NULL**.

When importing images, this parameter can be set to one of the following:

● For importing images	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_LOAD	Imports the sequence into previously allocated buffers.
<input type="checkbox"/> M_RESTORE	Imports the sequence into automatically allocated buffers.

SystemId

Specifies the system on which to allocate the buffers for an **M_RESTORE** operation. In other cases, this parameter should be set to **M_NULL**.

For an **M_RESTORE** operation, this parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

BufArrayPtr

Specifies the address of the array containing the buffer identifiers for an **M_LOAD** operation, or the address of the array in which to store the new buffer identifiers for an **M_RESTORE** operation. If not importing images, this parameter should be set to **M_NULL**.

For an **M_LOAD** operation, the destination buffers should be large enough to hold the imported images.

For an **M_RESTORE** operation, the destination buffers will be allocated with an appropriate size and type to hold the images. If an **M_RESTORE** operation fails, zero will be written for the buffer identifiers.

[MIL-Lite with restriction]

When importing compressed images with the **M_LOAD** operation, if the destination buffers have only an **M_IMAGE** attribute, the images will automatically be decompressed. When importing uncompressed images with the **M_LOAD** operation, if the destination buffers have an **M_IMAGE** + **M_COMPRESS** attribute, the images will automatically be compressed.

When importing compressed images with the **M_RESTORE** operation, the **M_COMPRESS** attribute will automatically be added to the destination buffer (**M_IMAGE** + **M_COMPRESS**) and the images will remain compressed in the destination buffer.

StartImage

Specifies the frame number of the image from which to begin importing. If not importing images, this parameter should be set to **M_NULL**.

When importing images, note that the frame number of the first image in the sequence is zero. To import the image at the current read position, set **StartImage** to **M_DEFAULT**.

NumberOfImages

Specifies the number of images, starting at [StartImage](#), to import. If not importing images, this parameter should be set to **M_NULL**.

When importing images, this number cannot be larger than the size of the array pointed to by [BufArrayPtr](#). Note that you can inquire about the number of frames (images) in the AVI file using [MbufDiskInquire\(\)](#).

ControlFlag

Specifies the function's control flag. This parameter must be set to one of the following:

● For specifying the function's control flag	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Opens the AVI file, imports the specified images, and then closes the file.
<input type="checkbox"/> M_CLOSE	Closes the AVI file, and (re)sets the pointer position to the first image. No images are imported. (summarize)
<input type="checkbox"/> M_OPEN	Opens the AVI file for reading, and sets the pointer to the first image. No images are imported. (summarize)
<input type="checkbox"/> M_READ	Imports the specified images from the AVI file, starting at the specified StartImage position. After the importing the images, the file pointer is left at the position of the next image, ready for the next M_READ operation. (summarize)

Remarks

- *[MIL-Lite]*
Note that MIL-Lite compression support, particularly for [M_AVI_MJPG](#) format and [M_IMAGE](#) + [M_COMPRESS](#) buffer type, is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.
- When a "remote" path is specified (the [Filename](#) parameter begins with "remote:///"), the [SystemId](#) parameter must point to the remote system when the AVI file is being opened or closed ([M_OPEN](#) or [M_CLOSE](#)). The buffers that are receiving the images are allocated on this same remote system.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufInquire

Synopsis

Inquire about a MIL data buffer setting.

Syntax

```
MIL_INT MFTYPE MbufInquire(
    MIL_ID BufId,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires about a specified setting of a MIL data buffer.

Parameters

BufId

Specifies the identifier of the source data buffer. The buffer must have been previously allocated on the required system using a function of the Buffer module (for example, [MbufAllocColor\(\)](#) or [MbufImport\(\)](#)).

InquireType

Specifies the type of buffer setting about which to inquire. This parameter can be set to one of the following values.

The following values require that you pass the value listed in the data-type area of the specified parameter.

For specifying the type of buffer setting																									
Value	Description	corona-II (a)	glige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ilee 1394 iIac (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	vio (w)		
M_ALLOCATION_OVERSCAN_SIZE	Returns the size of the overscan region of the image buffer, in pixels.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_ANCESTOR_ID	Returns the MIL identifier of the ancestor buffer. Only child buffers have an ancestor buffer. The ancestor buffer is the buffer from which the specified buffer (BufId) ultimately originated. It is the root buffer; it does not have a parent buffer (it is not a child buffer of another buffer). Note that the identifier of the specified buffer is returned if it does not have an ancestor buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
	To establish the parent buffer of the specified buffer, use M_PARENT_ID instead. (summarize)																								
	UserVarPtr info Data type: MIL_ID																								
M_ANCESTOR_OFFSET_BAND	Returns the band offset relative to the ancestor buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_ANCESTOR_OFFSET_BIT	Returns the bit offset relative to the ancestor buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_ANCESTOR_OFFSET_X	Returns the X-offset relative to the ancestor buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_ANCESTOR_OFFSET_Y	Returns the Y-offset relative to the ancestor buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
M_ASSOCIATED_LUT	Returns the identifier of the LUT buffer associated with the image buffer (returns	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

	M_DEFAULT if no LUT). (summarize)		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	UserVarPtr info Data type: MIL_ID Return values: MIL LUT buffer identifier; M_DEFAULT; (details)																								
M_BITMAPINFO	[This is only applicable to Windows] Returns a pointer (LPBITMAPINFO) to the header of the DIB associated with the MIL buffer (if any) or M_NULL .		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_DATA_FORMAT	Returns the color or monochrome storage format of the buffer. This includes whether it is packed or planar format. Note that to retrieve the entire list of attributes, use MbufInquire() with M_EXTENDED_ATTRIBUTE or M_EXTENDED_FORMAT . (summarize)		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_DC_HANDLE	[This is only applicable to Windows] Returns the device context handle (HDC) of the buffer. The buffer device context must have been successfully allocated using MbufControl() with M_DC_ALLOC . (summarize)		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	UserVarPtr info Return values: M_NULL Value No device context is associated with the buffer. Handle of the device context associated with the buffer.																								
M_DDRAWSURFACE	[This is only applicable to Windows] Returns a pointer (LPDIRECTDRAWSURFACE) to the DirectDraw surface associated with the MIL buffer (if any) or M_NULL .		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_DIB_HANDLE	[This is only applicable to Windows] Returns the handle (HBITMAP) of the DIB associated with the MIL buffer. To ensure that the buffer has a DIB handle, the buffer must have been successfully allocated using with M_DIB + M_GDI . (summarize)		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	UserVarPtr info Return values: M_NULL Value No DIB handle is associated with the buffer. Handle of the DIB associated with the buffer.																								
M_EXTENDED_ATTRIBUTE	Returns the requested format of the specified buffer as set using MbufAlloc...Q with Attribute . Note that any setting left to its default is not returned by this inquire type. To retrieve only the color or monochrome storage format of the buffer, use MbufInquire() with M_DATA_FORMAT . To retrieve the actual format of the specified buffer, use MbufInquire() with M_EXTENDED_FORMAT . Note that M_NULL cannot be passed in the UserVarPtr parameter when InquireType is set to M_EXTENDED_ATTRIBUTE ; a valid MIL_INT64 pointer must be passed to the function or an error will occur. (summarize)		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	UserVarPtr info Data type: MIL_INT64 Return values: Value A bit-encoded value that specifies the buffer's attributes, set at allocation.																								
M_EXTENDED_FORMAT	Returns the actual format of the specified buffer. Note that any setting left to its default is returned by this inquire type. To retrieve only the color or monochrome storage format of the buffer, use MbufInquire() with M_DATA_FORMAT . To retrieve the requested formatting		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

[illegible]

	<p>If the image buffer is accessed externally, for example, when using <code>MbufCreateColor()</code> or <code>MbufCreate2d()</code>, <code>MbufControl()</code> with <code>M_MODIFIED</code> must be called to indicate that the image buffer's contents have been modified. Calling this function will increment the counter.</p> <p>This feature is useful for optimization. For example, you can avoid repeating certain computations (for example, analysis computations) if you know that the image buffer has not been modified. In this case, inquire the count before the first computation in the sequence of computations, and then inquire it again before repeating the same sequence. If no modifications have been made to the image buffer, you can avoid repeating the sequence unnecessarily.</p> <p>(summarize)</p>										j	k						q	r	s				
<div><div></div><div>M_NATIVE_ID</div></div>	Returns the native identifier (handle) of the buffer. This identifier can be used when mixing board-specific code (from the native library function set) with MIL code.																							
<div><div></div><div>M_OWNER_SYSTEM</div></div>	Returns the identifier of the system on which the buffer has been allocated.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<div><div>UserVarPtr info</div><div>Data type: MIL_ID</div><div>Return values: MIL system identifier; M_DEFAULT_HOST; (details)</div></div>																							
<div><div></div><div>M_OWNER_SYSTEM_TYPE</div></div>	Returns the type of system on which the buffer was allocated. For possible return values, see the MsysInquire() <code>M_SYSTEM_TYPE</code> inquire type.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<div><div>UserVarPtr info</div><div>Data type: MIL_TEXT_CHAR</div></div>																							
<div><div></div><div>M_PARENT_ID</div></div>	Returns the MIL identifier of the parent buffer. Only child buffers have a parent buffer. The parent buffer is the buffer from which the specified buffer (BufId) was defined. The parent buffer can itself have a parent buffer. If the specified buffer has no parent buffer, the identifier of the specified buffer is returned.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<p>To establish the ancestor buffer (root buffer) of the specified buffer, use M_ANCESTOR_ID instead.</p> <p>(summarize)</p>																							
	<div><div>UserVarPtr info</div><div>Data type: MIL_ID</div></div>																							
<div><div></div><div>M_PARENT_OFFSET_BAND</div></div>	Returns the band offset relative to the parent buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_PARENT_OFFSET_X</div></div>	Returns the X-offset relative to the parent buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_PARENT_OFFSET_Y</div></div>	Returns the Y-offset relative to the parent buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_PHYSICAL_ADDRESS</div></div>	Returns the physical address of the buffer if it is not a planar 3-band buffer. For a planar 3-band buffer, you can determine its physical address by allocating a child buffer for the required band and then using M_PHYSICAL_ADDRESS to determine its physical address. This type of address is available only for a non-paged buffer mapped to the Host or from a buffer allocated in a frame grabber's on-board memory. This type of address is used mostly for access by bus masters other than the Host CPU.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<div><div>UserVarPtr info</div><div>Return values:</div><div><div>NULL</div><div>The buffer is not visible from the physical address space.</div></div><div><div>Value</div><div>Address of the buffer.</div></div></div>																							
<div><div></div><div>M_PHYSICAL_ADDRESS_REMOTE</div></div>	Returns the physical on-board address of the buffer, if allocated on-board and is not a planar 3-band buffer. For a planar 3-band buffer, you can determine its physical on-board address by allocating a child buffer for the required band and then using M_PHYSICAL_ADDRESS_REMOTE to determine its physical on-board address. This address is only meaningful on-board the same system.																	q	r	s				
	<div><div>UserVarPtr info</div></div>																							

[illegible]

<input type="checkbox"/> M_OVERSCAN	Returns the overscan type. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_FAST; M_MIRROR; M_REPLACE; M_TRANSPARENT; (details)
<input type="checkbox"/> M_OVERSCAN_REPLACE_VALUE	Returns the overscan replace value. (summarize)
	<i>UserVarPtr info</i> Return values: M_REPLACE_MAX; M_REPLACE_MIN; Value; (details)
<input type="checkbox"/> M_SATURATION	Returns whether to saturate the result. (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)

For [M_KERNEL](#) data buffers only (see [MbufControlNeighborhood\(\)](#) for possible values), you can set this parameter to one of the following values.

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_INT*.

● For M_KERNEL data buffers only	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ABSOLUTE_VALUE	Returns whether the absolute value should be taken. (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FILTER_MODE	Returns the mode in which to apply the filter. (summarize)
	<i>UserVarPtr info</i> Return values: M_KERNEL; M_RECURSIVE; (details)
<input type="checkbox"/> M_FILTER_OPERATION	Returns the type of neighborhood operation to perform using the selected filter. If MbufControlNeighborhood() M_FILTER_TYPE is set to M_USER_DEFINED , then M_FILTER_OPERATION is not supported. (summarize)
	<i>UserVarPtr info</i> Return values: M_EDGE_DETECT; M_EDGE_DETECT_SQR; M_FIRST_DERIVATIVE_X; M_FIRST_DERIVATIVE_Y; M_HORIZ_EDGE; M_LAPLACIAN_EDGE; M_SECOND_DERIVATIVE_X; M_SECOND_DERIVATIVE_XY; M_SECOND_DERIVATIVE_Y; M_SHARPEN; M_SMOOTH; M_VERT_EDGE; (details)
<input type="checkbox"/> M_FILTER_SMOOTHNESS	Returns the degree of smoothness (strength of the denoising) applied by the filter during the neighborhood operation. If MbufControlNeighborhood() M_FILTER_TYPE is set to M_USER_DEFINED , then M_FILTER_SMOOTHNESS is not supported. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 100; (details)
<input type="checkbox"/> M_FILTER_TYPE	Returns the type of filter used to perform the neighborhood operation. (summarize)
	<i>UserVarPtr info</i> Return values: M_DERICHE; M_SHEN; M_USER_DEFINED; (details)
<input type="checkbox"/> M_NORMALIZATION_FACTOR	Returns the normalization factor. (summarize)
	<i>UserVarPtr info</i> Return values: Value > 0; (details)

For [M_IMAGE](#) + [M_COMPRESS](#) image buffers (see [MbufAlloc...\(\)](#) for possible values), you can set this parameter to one of the following values.

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a MIL_INT.

For M_IMAGE + M_COMPRESS image buffers	
Value	Description
M_COMPRESSION_TYPE	<p>Returns the type of compression. See MbufAlloc...() for possible values. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_JPEG_LOSSLESS; M_JPEG2000_LOSSLESS; M_JPEG_LOSSLESS_INTERLACED; M_JPEG_LOSSY; M_JPEG2000_LOSSY; M_JPEG_LOSSY_INTERLACED; M_MPEG4; (details)</p>

For M_IMAGE + M_COMPRESS image buffers with a M_JPEG_LOSSY, M_JPEG_LOSSY_INTERLACED, or M_JPEG2000_LOSSY compression type, you can set this parameter to one of the following values.

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a MIL_INT.

For M_IMAGE + M_COMPRESS (with M_JPEG_LOSSY, M_JPEG_LOSSY_INTERLACED, or M_JPEG2000_LOSSY)	
Value	Description
M_Q_FACTOR +	<p>Returns the quantization factor for both JPEG2000 lossy and JPEG lossy buffers. Note that for 3-band buffers, only the quantization factor associated with the first band is returned. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 1 to 99; (details)</p>
M_Q_FACTOR_CHROMINANCE	<p>Returns the quantization factor of the U and V bands for both JPEG2000 lossy and JPEG lossy buffers in YUV format. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 1 to 99; (details)</p>
M_Q_FACTOR_LUMINANCE	<p>Returns the quantization factor of the Y band for both JPEG2000 lossy and JPEG lossy buffers in YUV format. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 1 to 99; (details)</p>
M_QUANTIZATION +	<p>Returns the identifier of the array buffer containing the quantization table (for a JPEG lossy or JPEG2000 lossy buffer), which is associated with the image buffer. Note that for 3-band buffers, only the identifier of the array buffer associated with the first band is returned. (summarize)</p> <p><i>UserVarPtr info</i> Return values: MIL_M_ARRAY buffer identifier; (details)</p>
M_QUANTIZATION_CHROMINANCE	<p>Returns the identifier of the array buffer containing the quantization table associated with the U and V bands, for both JPEG2000 lossy and JPEG lossy buffers in YUV format. (summarize)</p> <p><i>UserVarPtr info</i> Return values: MIL_M_ARRAY buffer identifier; (details)</p>
M_QUANTIZATION_LUMINANCE	<p>Returns the identifier of the array buffer containing the quantization table associated with the Y band, for both JPEG2000 lossy and JPEG lossy buffers in YUV format. (summarize)</p> <p><i>UserVarPtr info</i> Return values: MIL_M_ARRAY buffer identifier; (details)</p>

Combination constants for M_Q_FACTOR; M_QUANTIZATION; M_DECOMPOSITION_LEVEL; M_NUMBER_SUBBAND;

You can add one of the following values to the above-mentioned values to set the band about which to inquire.

Note that when dealing with [M_Q_FACTOR](#) and [M_QUANTIZATION](#), the following combination constants are available only for use with JPEG2000 lossy buffers.

● For M_DECOMPOSITION_LEVEL , M_NUMBER_SUBBAND , M_Q_FACTOR (for JPEG2000 lossy buffers), and M_QUANTIZATION (for JPEG2000 lossy buffers)	
☐ Value	Description
☐ M_BLUE	Returns the inquired value for the blue band (for RGB buffers).
☐ M_GREEN	Returns the inquired value for the green band (for RGB buffers).
☐ M_RED	Returns the inquired value for the red band (for RGB buffers).
☐ M_U	Returns the inquired value for the U band (for YUV buffers).
☐ M_V	Returns the inquired value for the V band (for YUV buffers).
☐ M_Y	Returns the inquired value for the Y band (for YUV buffers).

For [M_IMAGE](#) + [M_COMPRESS](#) image buffers with a [M_JPEG2000_LOSSY](#) or [M_JPEG2000_LOSSLESS](#) compression type, you can set this parameter to one of the following values.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_INT`.

● For M_IMAGE + M_COMPRESS (with M_JPEG2000_LOSSY or M_JPEG2000_LOSSLESS)	
☐ Value	Description
☐ M_DECOMPOSITION_LEVEL +	Returns the number of iterations the discrete wavelet transform is applied to the image (for single-band images) or on the first band of the image (for 3-band images). (summarize)
	<i>UserVarPtr info</i> Return values: Value >= 0; (details)
☐ M_NUMBER_SUBBAND +	Returns the number of sub-bands rendered from the discrete wavelet transform passed on an image (for 1-band images) or on the first band of a 3-band image.

For [M_IMAGE](#) + [M_COMPRESS](#) image buffers with a [M_JPEG2000_LOSSY](#) compression type, you can set this parameter to the following value.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_INT`.

● For M_IMAGE + M_COMPRESS (with M_JPEG2000_LOSSY)	
☐ Value	Description
☐ M_TARGET_SIZE	Returns the requested size of the compressed buffer in bytes. Returns M_DEFAULT if no size has been requested. Use M_SIZE_BYTE to establish the real size of the buffer. (summarize)
	<i>UserVarPtr info</i> Return values: Value > 0; (details)

For [M_IMAGE](#) + [M_COMPRESS](#) image buffers with a [M_JPEG_LOSSY](#), [M_JPEG_LOSSY_INTERLACED](#), [M_JPEG_LOSSLESS](#), or [M_JPEG_LOSSLESS_INTERLACED](#) compression type, you can set this parameter to one of the following values.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_INT`.

● For M_IMAGE + M_COMPRESS (with M_JPEG_LOSSY , M_JPEG_LOSSY_INTERLACED , M_JPEG_LOSSLESS , or M_JPEG_LOSSLESS_INTERLACED)	
☐ Value	Description
☐ M_HUFFMAN_DC	Returns the identifier of the array buffer containing the DC Huffman table which is associated with the image buffer. For 3-band buffers, only the identifier of the array buffer associated with the first band is returned. (summarize)
	<i>UserVarPtr info</i> Data type: <code>MIL_ID</code>

	Return values: MIL_M_ARRAY buffer identifier; (details)
<input type="checkbox"/> M_HUFFMAN_DC_CHROMINANCE	<p>Returns the identifier of the array buffer containing the DC Huffman table that is associated with the U and V bands of a JPEG lossy buffer in YUV format. (summarize)</p> <p><i>UserVarPtr info</i> Data type: MIL_ID Return values: MIL_M_ARRAY buffer identifier; (details)</p>
<input type="checkbox"/> M_HUFFMAN_DC_LUMINANCE	<p>Returns the identifier of the array buffer containing the DC Huffman table that is associated with the Y band of a JPEG lossy buffer in YUV format. (summarize)</p> <p><i>UserVarPtr info</i> Data type: MIL_ID Return values: MIL_M_ARRAY buffer identifier; (details)</p>

For [M_IMAGE](#) + [M_COMPRESS](#) image buffers with a [M_JPEG_LOSSLESS](#) or [M_JPEG_LOSSLESS_INTERLACED](#) compression type, you can set this parameter to one of the following values.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_INT`.

● For M_IMAGE + M_COMPRESS (with M_JPEG_LOSSLESS , or M_JPEG_LOSSLESS_INTERLACED)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_PREDICTOR	<p>Returns the type of predictor. This inquire type is supported for JPEG lossless buffers only. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0; 1; 2; (details)</p>
<input type="checkbox"/> M_RESTART_INTERVAL	<p>Returns the number of lines between restart markers (for JPEG lossless buffers) or number of 8x8 blocks of data between restart markers (for JPEG lossy buffers). (summarize)</p> <p><i>UserVarPtr info</i> Return values: Value > 0; (details)</p>

For [M_IMAGE](#) + [M_COMPRESS](#) image buffers with a [M_JPEG_LOSSY](#) or [M_JPEG_LOSSY_INTERLACED](#) compression type, you can set this parameter to one of the following values.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_INT`.

● For M_IMAGE + M_COMPRESS (with M_JPEG_LOSSY , or M_JPEG_LOSSY_INTERLACED)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_HUFFMAN_AC	<p>Returns the identifier of the array buffer containing the AC Huffman table that is associated with the image buffer. For 3-band buffers, only the identifier of the array buffer associated with the first band is returned. (summarize)</p> <p><i>UserVarPtr info</i> Data type: MIL_ID Return values: MIL_M_ARRAY buffer identifier; (details)</p>
<input type="checkbox"/> M_HUFFMAN_AC_CHROMINANCE	<p>Returns the identifier of the array buffer containing the AC Huffman table that is associated with the U and V bands of a JPEG buffer in YUV format. (summarize)</p> <p><i>UserVarPtr info</i> Data type: MIL_ID Return values: MIL_M_ARRAY buffer identifier; (details)</p>
<input type="checkbox"/> M_HUFFMAN_AC_LUMINANCE	<p>Returns the identifier of the array buffer containing the AC Huffman table that is associated with the Y band of a JPEG buffer in YUV format. (summarize)</p> <p><i>UserVarPtr info</i></p>

		Data type: MIL_ID Return values: MIL M_ARRAY buffer identifier; (details)
--	--	------------------------------------------------------------------------------------------------

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT64
- MIL_TEXT_CHAR

Specifies the address in which to write the requested information.

Since the **MbufInquire()** function also returns the requested information, you can set this parameter to **M_NULL**, except when the specified **InquireType** requires the **UserVarPtr** parameter to be set to the address of a *MIL_INT64*. In this case, you must set this parameter to the address of a *MIL_INT64*.

Return value

The return value is the requested information, cast to a MIL_INT; otherwise **M_ERROR** is returned. For inquire types which specify that the **UserVarPtr** parameter must be set to the address of a *MIL_INT64*, the return value is *M_NULL*.

Remark

- *[MIL-Lite]*
Note that MIL-Lite compression support, particularly for **M_IMAGE** + **M_COMPRESS** buffer type, is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufLoad

Synopsis

Load data from a file into a data buffer.

Syntax

```
void MbufLoad(  
    MIL_CONST_TEXT_PTR Filename,  
    MIL_ID BufId  
)
```

Description

This function loads data from a file into a previously allocated data buffer. The function detects the file format from the data.

Note, you can perform the same operation as **MbufLoad()** using **MbufImport()**, which uses the specified file format to open the file instead of trying to determine the format from the data.

[MIL-Lite with restriction]
When loading compressed data and the destination buffer has an **M_IMAGE** attribute (but not an **M_COMPRESS** attribute), this function will automatically decompress it. If necessary, the data in the file will be transformed to the format of the buffer. When loading uncompressed data into a buffer with an **M_COMPRESS** attribute, this function will automatically compress it, according to the compression settings found in the buffer.

When loading an image file that has been saved with an associated LUT (color palette) into a 3-band 8-bit image buffer, the LUT is automatically applied to the data to generate 3-band image data. In this case, a LUT buffer is not created and, therefore, is not associated with the 3-band 8-bit buffer. When loading an image file that has been saved with an associated LUT (color palette) into any other type of image buffer, the LUT is also imported and associated with the resulting image buffer. You can obtain the identifier of the associated LUT, using **MbufInquire()**.

Note that the associated LUT will be automatically selected on the display (**MdispLut()**) if the image buffer is selected on a display and the default LUT has not been overridden by a former call to **MdispLut()**.

Using **MbufDiskInquire()**, you can inquire about the dimensions of the data saved in a file (except for raw files) without loading it.

Parameters

Filename

Specifies the name and path of the file from which to load the data. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

For specifying the file name and path					
Value	Description				
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)</div> <table><tr><th colspan="2">Parameters</th></tr><tr><td>FileName</td><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters		FileName	Specifies the drive, directory, and name of the file.
Parameters					
FileName	Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<div>[This is only applicable to Windows] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.</div>				

BufId

Specifies the identifier of the destination buffer. This buffer must be large enough in depth and dimensions to hold the data. If the data is deeper than the buffer, the most-significant bits of the data are truncated when loaded into the buffer. If the buffer depth is greater than that of the data, the data is zero or sign-extended (depending on the data type) when loaded into the buffer. If the buffer is larger in size than the data, exceeding areas of the buffer are unaffected.

Remark

- *[MIL-Lite]*
Note that MIL-Lite compression support, particularly for `M_IMAGE` + `M_COMPRESS` buffer type, is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufPut1d

Synopsis

Put data from a user-supplied array into a one-dimensional area of a buffer.

Syntax

```
void MbufPut1d(
    MIL_ID DestBufId,
    MIL_INT OffX,
    MIL_INT SizeX,
    const void *UserArrayPtr
)
```

Description

This function copies data from a user-supplied array to a one-dimensional area of the specified MIL destination buffer.

Note, for multi-band buffers, **MbufPut1d()** behaves like **MbufPutColor()** with its **DataFormat** parameter set to **M_PLANAR** and its **Band** parameter set to **M_ALL_BANDS**. See **MbufPutColor()** for more details.

Parameters

- DestBufId
- Specifies the identifier of the destination buffer.
- OffX
- Specifies the horizontal offset of the destination buffer area in which to put data, relative to the destination buffer's top-left pixel.
- SizeX
- Specifies the width of the destination buffer area in which to copy the data (starting from the specified offset **OffX**).
- UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer

Specifies the address of the user array from which to copy data into the destination buffer. Ensure that there are enough entries in the user array to fill the specified destination buffer area. **MbufPut1d()** assumes that the array is of the same data type and depth as the destination buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufPut2d

Synopsis

Put data from a user-supplied array into a two-dimensional area of a buffer.

Syntax

```
void MbufPut2d(
    MIL_ID DestBufId,
    MIL_INT OffX,
    MIL_INT OffY,
    MIL_INT SizeX,
    MIL_INT SizeY,
    const void *UserArrayPtr
)
```

Description

This function copies data from a user-supplied array to a two-dimensional area of the specified MIL destination buffer.

Note, for multi-band buffers, **MbufPut2d()** behaves like **MbufPutColor2d()** with its **DataFormat** parameter set to **M_PLANAR** and its **Band** parameter set to **M_ALL_BANDS**. See **MbufPutColor()** for more details.

Parameters

- DestBufId
- Specifies the identifier of the destination buffer.
- OffX
- Specify the horizontal offset of the destination buffer area in which to put the data, relative to the destination buffer's top-left pixel.
- OffY
- Specify the vertical offset of the destination buffer area in which to put the data, relative to the destination buffer's top-left pixel.
- SizeX
- Specify the width of the destination buffer area in which to copy the data (starting from the specified offsets **OffX** and **OffY**).
- SizeY
- Specify the height of the destination buffer area in which to copy the data (starting from the specified offsets **OffX** and **OffY**).
- UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer

Specifies the address of the user array from which to copy data into the destination buffer. Ensure that there are enough entries in the user array to fill the specified destination buffer area. **MbufPut2d()** assumes that the array is of the same data type and depth as the destination buffer.

Compilation information

Header	Include mil.h.

Library	Use mil.lib.
DLL	Requires mil.dll.

MbufPut

Synopsis

Put data from a user-supplied array into a data buffer.

Syntax

```
void MbufPut(
    MIL_ID DestBufId,
    const void *UserArrayPtr
)
```

Description

This function copies data from a user-supplied array to a specified MIL destination buffer.

Note, for multi-band buffers, **MbufPut()** behaves like **MbufPutColor()** with its **DataFormat** parameter set to **M_PLANAR** and its **Band** parameter set to **M_ALL_BANDS**. See **MbufPutColor()** for more details.

Parameters

DestBufId

Specifies the identifier of the destination buffer.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer

Specifies the address of the user array from which to copy data into the destination buffer. Ensure that there are enough entries in the user array to fill the destination buffer. **MbufPut()** assumes that the array is of the same data type and depth as the destination buffer's bands.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufPutColor2d

Synopsis

Put data from a user-supplied array into a region of one or all bands of a data buffer.

Syntax

```
void MbufPutColor2d(
    MIL_ID DestBufId,
    MIL_INT DataFormat,
    MIL_INT Band,
    MIL_INT OffX,
    MIL_INT OffY,
    MIL_INT SizeX,
    MIL_INT SizeY,
    const void *UserArrayPtr
)
```

Description

This function copies data from a user-supplied array to a specified region in one or all bands of a specified MIL destination buffer.

Parameters

DestBufId

Specifies the identifier of the destination buffer.

DataFormat

Specifies the data format of the user-supplied array. This information is required to properly copy the data. The internal data format of the destination buffer need not match the specified data format of the user-supplied array; an internal conversion will be performed if necessary. Note, however, if the formats do match, the operation will be much faster.

This parameter must be set to one of the following values. Note that Dx and Dy denote the destination width and height, respectively.

Unless otherwise specified, the following values require that you pass the *UserArrayPtr* parameter the address of a MIL_INT.

For specifying the data format of the user-supplied array	
Value	Description
<input checked="" type="checkbox"/> M_PACKED +	Copies the bands in an interleaved manner (for example, RGB RGB RGB...). You must specify a combination value from the table below. (summarize)
<input checked="" type="checkbox"/> M_PLANAR	Copies the bands one after the other (RRR...GGG...BBB...). This format is to be used when copying to all color bands (M_ALL_BANDS) of the destination buffer. (summarize) <div><i>UserArrayPtr info</i> Data type: array of the same type as the buffer Array size: Dx x Dy x number of color band of the destination buffer</div>
<input checked="" type="checkbox"/> M_SINGLE_BAND	Copies to a single color band. (summarize) <div><i>UserArrayPtr info</i> Data type: array of the same type as the buffer Array size: Dx x Dy</div>

Note: In the case of a [M_BGR32](#) buffer, point to an array of type *char*, since only one band will be copied.

Combination constants for [M_PACKED](#);

You must add one of the following values to the above-mentioned value to set the color buffer format.

See the [RGB buffers](#) section in [Chapter 18: Specifying and managing your data buffers](#).

Note that the destination buffer must be a three-band, 8-bit buffer.

● For M_PACKED	
☐ Value	Description
☐ M_BGR24	<p>Stores the data in a BGR24 packed format. In this format, each pixel is internally stored as three consecutive bytes in little-endian order.</p> <p>(summarize)</p> <p><i>UserArrayPtr info</i> Data type: array of type <i>char</i> Array size: Dx x Dy x 3 bytes (<i>char</i>)</p>
☐ M_BGR32	<p>Stores the data in a BGR32 packed format. In this format, each pixel is internally stored as four consecutive bytes, in little-endian order. The most-significant byte is a "don't care" byte.</p> <p>(summarize)</p> <p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>MIL_INT32</i> Array size: Dx x Dy x 4 bytes (size of BGR32 pixel) • Data type: array of type <i>MIL_UINT32</i> Array size: Dx x Dy x 4 bytes (size of BGR32 pixel)
☐ M_RGB15	<p>Stores the data in a RGB15 packed format. In this format, each pixel is internally stored as a 16-bit word with a 5-bit blue value (least significant), a 5-bit green value, a 5-bit red value, and a "don't care" bit (most significant), in little-endian order (RGB 5:5:5).</p> <p>(summarize)</p> <p><i>UserArrayPtr info</i> Data type: array of type <i>MIL_INT16</i> Array size: Dx x Dy x 2 bytes (unsigned <i>char</i>)</p>
☐ M_RGB16	<p>Stores the data in a RGB16 packed format. In this format, each pixel is internally stored as a 16-bit word with a 5-bit blue value (least significant), a 6-bit green value, and a 5-bit red value (most significant), in little-endian order (RGB 5:6:5).</p> <p>(summarize)</p> <p><i>UserArrayPtr info</i> Data type: array of type <i>MIL_INT16</i> Array size: Dx x Dy x 2 bytes (unsigned <i>char</i>)</p>

Band

Specifies the index of the color band in which to copy. This parameter can be set to one of the values below.

● For specifying the index of the color band							
☐ Value	Description						
☐ 0 <= Value <= #bands - 1	<p>Specifies the index of the band to copy.</p> <p>The relationship between index value and band for RGB and HSL buffers is the following:</p> <table> <tr> <td>0</td><td>Corresponds to the red band for RGB buffers or the hue band for HSL buffers.</td></tr> <tr> <td>1</td><td>Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.</td></tr> <tr> <td>2</td><td>Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.</td></tr> </table>	0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.	1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.	2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.
0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.						
1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.						
2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.						

	(summarize)
<input type="checkbox"/> M_ALL_BANDS	Copies to all color bands.
<input type="checkbox"/> M_BLUE	Copies to the blue color band (for RGB buffers).
<input type="checkbox"/> M_GREEN	Copies to the green color band (for RGB buffers).
<input type="checkbox"/> M_HUE	Copies to the hue color band (for HSL buffers).
<input type="checkbox"/> M_LUMINANCE	Copies to the luminance color band (for HSL buffers).
<input type="checkbox"/> M_RED	Copies to the red color band (for RGB buffers).
<input type="checkbox"/> M_SATURATION	Copies to the saturation color band (for HSL buffers).

OffX

Specifies the horizontal offset of the destination buffer region in which to put the data, relative to the destination buffer's top-left coordinate.

OffY

Specifies the vertical offset of the destination buffer region in which to put the data, relative to the destination buffer's top-left coordinate.

SizeX

Specifies the width of the destination buffer region in which to put the data.

SizeY

Specifies the height of the destination buffer region in which to put the data.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer
- array of type char
- array of type MIL_INT16
- array of type MIL_INT32
- array of type MIL_UINT32
- MIL_INT

Specifies the address of the user array from which to copy data into the destination buffer. Ensure that there are enough entries in the user array to fill the color band of the destination buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufPutColor

Synopsis

Put data from a user-supplied array into one or all bands of a data buffer.

Syntax

```
void MbufPutColor(
    MIL_ID DestBufId,
    MIL_INT DataFormat,
    MIL_INT Band,
    const void *UserArrayPtr
)
```

Description

This function copies data from a user-supplied array to one or all bands of a specified MIL destination buffer.

Parameters

DestBufId
Specifies the identifier of the destination buffer.

DataFormat
Specifies the data format of the user-supplied array. This information is required to properly copy the data. The internal data format of the destination buffer need not match the specified data format; an internal conversion will be performed if necessary. Note, however, if the formats do match, the operation will be much faster.

This parameter must be set to one of the following values. Note that Dx and Dy denote the destination width and height, respectively.

Unless otherwise specified, the following values require that you pass the *UserArrayPtr* parameter the address of a MIL_INT.

For specifying the data format	
Value	Description
M_PACKED +	Copies the bands in an interleaved manner (for example, RGB RGB RGB...). You must specify a combination value from the table below. (summarize)
M_PLANAR	Copies the bands one after the other (RRR...GGG...BBB...). This format is to be used when copying to all color bands (M_ALL_BANDS) of the destination buffer. (summarize) <div><i>UserArrayPtr info</i> Data type: array of the same type as the buffer Array size: Dx x Dy x number of color band of the destination buffer</div>
M_SINGLE_BAND	Copies to a single color band. (summarize) <div><i>UserArrayPtr info</i> Data type: array of the same type as the buffer Array size: Dx x Dy Note: In the case of a M_BGR32 buffer, point to an array of type char, since only one band will be copied</div>

Combination constants for M_PACKED;

You must add one of the following values to the above-mentioned value to set the color buffer format.

See the [RGB buffers](#) section in [Chapter 18: Specifying and managing your data buffers](#).

Note that the destination buffer must be a three-band, 8-bit buffer.

● For M_PACKED	
☐ Value	Description
☐ M_BGR24	Stores the data in a BGR24 packed format. In this format, each pixel is internally stored as three consecutive bytes in little-endian order. (summarize)
	<i>UserArrayPtr info</i> Data type: array of type char Array size: Dx x Dy x 3 bytes (char)
☐ M_BGR32	Stores the data in a BGR32 packed format. In this format, each pixel is internally stored as four consecutive bytes, in little-endian order. The most-significant byte is a "don't care" byte. (summarize)
	<i>UserArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: Dx x Dy x 4 bytes (size of BGR32 pixel) • Data type: array of type MIL_UINT32 Array size: Sx x Sy x 4 bytes (size of BGR32 pixel)
☐ M_RGB15	Stores the data in a RGB15 packed format. In this format, each pixel is internally stored as a 16-bit word with a 5-bit blue value (least significant), a 5-bit green value, a 5-bit red value, and a "don't care" bit (most significant), in little-endian order (RGB 5:5:5). (summarize)
	<i>UserArrayPtr info</i> Data type: array of type MIL_INT16 Array size: Dx x Dy x 2 bytes (unsigned char)
☐ M_RGB16	Stores the data in a RGB16 packed format. In this format, each pixel is internally stored as a 16-bit word with a 5-bit blue value (least significant), a 6-bit green value, and a 5-bit red value (most significant), in little-endian order (RGB 5:6:5). (summarize)
	<i>UserArrayPtr info</i> Data type: array of type MIL_INT16 Array size: Dx x Dy x 2 bytes (unsigned char)

Band

Specifies the index of the color band in which to copy. This parameter can be set to one of the values below.

● For specifying the index of the color band							
☐ Value	Description						
☐ 0 <= Value <= #bands - 1	Specifies the index of the band to copy. The relationship between index value and band for RGB and HSL buffers is the following: <table border="1"> <tr> <td>0</td><td>Corresponds to the red band for RGB buffers or the hue band for HSL buffers.</td></tr> <tr> <td>1</td><td>Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.</td></tr> <tr> <td>2</td><td>Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.</td></tr> </table> (summarize)	0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.	1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.	2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.
0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.						
1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.						
2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.						
☐ M_ALL_BANDS	Copies to all color bands.						
☐ M_BLUE	Copies to the blue color band (for RGB buffers).						
☐ M_GREEN	Copies to the green color band (for RGB buffers).						

<input type="checkbox"/> M_HUE	Copies to the hue color band (for HSL buffers).
<input type="checkbox"/> M_LUMINANCE	Copies to the luminance color band (for HSL buffers).
<input type="checkbox"/> M_RED	Copies to the red color band (for RGB buffers).
<input type="checkbox"/> M_SATURATION	Copies to the saturation color band (for HSL buffers).

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer
- array of type char
- array of type MIL_INT16
- array of type MIL_INT32
- array of type MIL_UINT32
- MIL_INT

Specifies the address of the user array from which to copy data into the destination buffer. Ensure that there are enough entries in the user array to fill the color band of the destination buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufPutLine

Synopsis

Write a specified series of pixels along a specified theoretical line.

Syntax

```
void MbufPutLine(  
    MIL_ID ImageBufId,  
    MIL_INT StartX,  
    MIL_INT StartY,  
    MIL_INT EndX,  
    MIL_INT EndY,  
    MIL_INT Mode,  
    MIL_INT *NbPixelsPtr,  
    const void *UserArrayPtr  
)
```

Description

This function reads pixel values from a user-defined array and writes them to the series of pixels, in the specified image, along the theoretical line defined by specified coordinates. The Bresenham algorithm is used to determine the theoretical line.

Parameters

ImageBufId

Specifies the identifier of the destination image buffer. This must be a single-band (monochrome) buffer.

StartX

Specifies the horizontal pixel offset of the starting position of the line, relative to the top-left pixel of the source buffer.

StartY

Specifies the vertical pixel offset of the starting position of the line, relative to the top-left pixel of the source buffer.

EndX

Specifies the horizontal pixel offset of the finishing position on the line, relative to the top-left pixel of the source buffer.

EndY

Specifies the vertical pixel offset of the finishing position on the line, relative to the top-left pixel of the source buffer.

Mode

Specifies the operation mode. This parameter must be set to **M_DEFAULT**.

NbPixelsPtr

Specifies the address of the variable in which to write the number of pixels found along the theoretical line. You can set this parameter to **M_NULL** if you don't want this value to be evaluated.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of the same type as the buffer

Specifies the address of the user array containing the pixels to insert in the image buffer. Ensure that the user array contains all the pixel values to be written. To determine the number of pixel values required, you can set this parameter to **M_NULL** and pass a non-null address to [NbPixelsPtr](#). In this case, nothing is written to the image buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufRestore

Synopsis

Restore data from a file into an automatically allocated data buffer.

Syntax

```
MIL_ID MbufRestore(
    MIL_CONST_TEXT_PTR Filename,
    MIL_ID SystemId,
    MIL_ID *BufIdPtr
)
```

Description

This function restores the data from the specified file and loads it into an automatically allocated buffer. It tries to detect the file format from the data.

If the file is in an [M_MIL](#) file format, this function will allocate the destination buffer with the same attributes as the original buffer, with the exception of an [M_IMAGE](#) buffer. In the case of an [M_IMAGE](#) buffer, the **MbufRestore()** function tries to allocate the buffer so that it can be used for acquisition ([M_GRAB](#)), display ([M_DISP](#)), and processing ([M_PROC](#)) operations. If there is insufficient appropriate memory to allocate such a buffer, it allocates one that can be used in all of the above operations except for acquisition ([M_GRAB](#)). Note that the maximum (total) number of grab ([M_GRAB](#)) buffers that can be allocated is restricted by the total amount of MIL non-paged (DMA) memory (specified at installation time or using MilConfig). For systems with on-board processors, the total number of [M_GRAB](#) buffers and [M_PROC](#) buffers is limited by the amount of on-board memory.

[MIL-Lite with restriction]
When restoring compressed data, the buffer will have an [M_COMPRESS](#) attribute.

When restoring an image file that was saved with an associated LUT (color palette), the LUT is also restored and associated with the restored image buffer. You can obtain the identifier of the associated LUT, using [MbufInquire\(\)](#).

After restoring a buffer, we recommend that you check that the operation was successful by using [MappGetError\(\)](#) or by checking that the buffer identifier returned is not **M_NULL**.

Note, you can perform the same operation as **MbufRestore()** by using [MbufImport\(\)](#), which uses the specified file format to restore the data instead of trying to determine the format from the data.

Parameters

Filename

Specifies the name and path of the file from which to restore the data buffer. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

● For specifying the file name and path					
<input checked="" type="checkbox"/> Value	Description				
<input checked="" type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p> <table><tr><th colspan="2">Parameters</th></tr><tr><th>FileName</th><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters		FileName	Specifies the drive, directory, and name of the file.
Parameters					
FileName	Specifies the drive, directory, and name of the file.				
<input checked="" type="checkbox"/> M_INTERACTIVE	<p>[This is only applicable to Windows]</p> <p>Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.</p>				

SystemId

Specifies the system on which to allocate the buffer.

This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

BufIdPtr

Specifies the address of the variable in which the buffer identifier is to be written. Since the **MbufRestore()** function also returns the buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the buffer identifier. If allocation fails, M_NULL is returned.

Remark

- *[MIL-Lite]*
Note that MIL-Lite compression support, particularly for **M_IMAGE** + **M_COMPRESS** buffer type, is reliant upon the presence of Matrox compression acceleration hardware; the compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufSave

Synopsis

Save a data buffer in a file, using the MIL output file format.

Syntax

```
void MbufSave(  
    MIL_CONST_TEXT_PTR FileName,  
    MIL_ID BufId  
)
```

Description

This function saves a previously allocated data buffer to a file, in the MIL file format. This is a baseline TIFF 6.0 file format with extra information included in the comment field. The extra information includes the buffer attributes and data type.

The MIL file format supports images that are binary, grayscale, and RGB, as well as palette-color images. Images in a different format are internally converted before being saved. In addition, by default, most color image buffers are saved in a packed (chunky) format (in accordance with baseline TIFF 6.0 specifications). Color binary buffers are saved in a 1-bit per pixel format (data is stored in a 3-band, packed binary format).

When saving an image buffer (**M_IMAGE**) that has an associated LUT buffer (color palette), the content of the LUT is also saved with the image.

Note, you can perform the same operation as **MbufSave()** by using **MbufExport()** with its **FileFormat** parameter set to **M_MIL**.

Parameters

FileName

Specifies the name and path of the file in which to save the model. It is recommended that you use the MIM file extension for easier use with other Matrox Imaging software products. The function handles (internally) the opening and closing of the file. If the file already exists, it will be overwritten.

This parameter can be set to one of the following values:

For specifying the file name and path				
Value	Description			
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)</div> <table><tr><td>Parameters</td></tr><tr><td>FileName</td></tr><tr><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters	FileName	Specifies the drive, directory, and name of the file.
Parameters				
FileName				
Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<div>[This is only applicable to Windows] Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</div>			

BufId

Specifies the identifier of the data buffer to save.

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MbufTransfer

Synopsis

Copy a two-dimensional region of one or all bands from the source buffer into a two-dimensional region of one or all bands in the destination buffer, using a specified transfer function and transfer type.

Syntax

```
void MbufTransfer(
    MIL_ID SrcBufId,
    MIL_ID DestBufId,
    MIL_INT SrcOffX,
    MIL_INT SrcOffY,
    MIL_INT SrcSizeX,
    MIL_INT SrcSizeY,
    MIL_INT SrcBand,
    MIL_INT DestOffX,
    MIL_INT DestOffY,
    MIL_INT DestSizeX,
    MIL_INT DestSizeY,
    MIL_INT DestBand,
    MIL_INT TransferFunction,
    MIL_INT TransferType,
    MIL_INT OperationFlag,
    void *ExtraParameter
)
```

Description

This function copies a two-dimensional region of one or all color bands of the source buffer to a two-dimensional region of one or all bands of the destination buffer, using a specified transfer function and transfer type.

If the size of the regions in the source and destination buffers differ, the size of the largest region is reduced so that the size of the source and destination regions become equal.

Parameters

SrcBufId

Specifies the identifier of the source buffer. When the **TransferFunction** parameter is set to [M_CLEAR](#), you must set this parameter to **M_NULL**; when the **TransferFunction** parameter is set to [M_BYTE_SWAP](#), you can set this parameter to **M_NULL** to swap bytes within the same buffer.

[All except Matrox GPU processing driver]
Floating-point buffers are only supported when the **TransferFunction** parameter is set to [M_COPY](#).

DestBufId

Specifies the identifier of the destination buffer.

[All except Matrox GPU processing driver]
Floating-point buffers are only supported when the **TransferFunction** parameter is set to [M_COPY](#).

SrcOffX

Specifies the horizontal offset of the source region, relative to the top-left coordinate of the source buffer.

● For the horizontal offset of the source region	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0 pixels. (summarize)
<input type="checkbox"/> M_NULL	Specifies that the horizontal offset of the source region is ignored when the TransferFunction parameter is set to M_CLEAR .
<input type="checkbox"/> Value	Specifies the horizontal offset of the source region, in pixels.

SrcOffY

Specifies the vertical offset of the source region, relative to the top-left coordinate of the source buffer.

● For the vertical offset of the source region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0 pixels. (summarize)
<input type="checkbox"/> M_NULL	Specifies that the vertical offset of the source region is ignored when the TransferFunction parameter is set to M_CLEAR .
<input type="checkbox"/> Value	Specifies the vertical offset of the source region, in pixels.

SrcSizeX

Specifies the width of the source region.

● For the width of the source region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default width of the source region. The default is equal to the difference between the width of the source buffer and the horizontal offset of the source region. (summarize)
<input type="checkbox"/> M_NULL	Specifies that the width of the source region is ignored when the TransferFunction parameter is set to M_CLEAR .
<input type="checkbox"/> Value	Specifies the width of the source region, in pixels.

SrcSizeY

Specifies the height of the source region.

● For the height of the source region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default height of the source region. The default is equal to the difference between the height of the source buffer and the vertical offset of the source region. (summarize)
<input type="checkbox"/> M_NULL	Specifies that the height of the source region is ignored when the TransferFunction parameter is set to M_CLEAR .
<input type="checkbox"/> Value	Specifies the height of the source region, in pixels.

SrcBand

Specifies the color band of the region from which to copy. This parameter can be set to one of the following values:

● For the color band of the region from which to copy	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ALL_BANDS.

<input type="checkbox"/> $0 \leq \text{Value} \leq \text{\#bands} - 1$	<p>Specifies the index of the band to copy.</p> <p>The relationship between the index value and band for RGB and HSL buffers is the following:</p> <table> <tr> <td>0</td><td>Corresponds to the red band for RGB buffers or the hue band for HSL buffers.</td></tr> <tr> <td>1</td><td>Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.</td></tr> <tr> <td>2</td><td>Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.</td></tr> </table> <p>(summarize)</p>	0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.	1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.	2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.
0	Corresponds to the red band for RGB buffers or the hue band for HSL buffers.						
1	Corresponds to the green band for RGB buffers or the saturation band for HSL buffers.						
2	Corresponds to the blue band for RGB buffers or the luminance band for HSL buffers.						
<input type="checkbox"/> M_ALL_BANDS	Specifies all color bands (for RGB, HSL, and YUV buffers).						
<input type="checkbox"/> M_BLUE	Specifies the blue band (for RGB buffers).						
<input type="checkbox"/> M_GREEN	Specifies the green band (for RGB buffers).						
<input type="checkbox"/> M_HUE	Specifies the hue band (for HSL buffers).						
<input type="checkbox"/> M_LUMINANCE	Specifies the luminance band (for HSL buffers).						
<input type="checkbox"/> M_NULL	Specifies that the band of the source region is ignored when the TransferFunction parameter is set to M_CLEAR .						
<input type="checkbox"/> M_RED	Specifies the red band (for RGB buffers).						
<input type="checkbox"/> M_SATURATION	Specifies the saturation band (for HSL buffers).						
<input type="checkbox"/> M_U	Specifies the U band (for YUV buffers).						
<input type="checkbox"/> M_V	Specifies the V band (for YUV buffers).						
<input type="checkbox"/> M_Y	Specifies the Y band (for YUV buffers).						

DestOffX

Specifies the horizontal offset of the destination region, relative to the top-left coordinate of the destination buffer.

● For the horizontal offset of the destination region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default horizontal offset of the destination region. The default is equal to the horizontal offset of the source region.</p> <p>(summarize)</p>
<input type="checkbox"/> Value	Specifies the horizontal offset of the destination region, in pixels.

DestOffY

Specifies the vertical offset of the destination region, relative to the top-left coordinate of the destination buffer.

● For the vertical offset of the destination region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default vertical offset of the destination region. The default is equal to the vertical offset of the source region.</p> <p>(summarize)</p>
<input type="checkbox"/> Value	Specifies the value of the vertical offset of the destination region, in pixels.

DestSizeX

Specifies the width of the destination region.

● For the width of the destination region	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_DEFAULT	Specifies the default width of the destination region. The default is equal to the difference between the destination buffer width and the horizontal offset of the destination region. (summarize)
<input type="checkbox"/> Value	Specifies the width of the destination region, in pixels.

DestSizeY

Specifies the height of the destination region.

● For the height of the destination region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default height of the destination region. The default is equal to the difference between the destination buffer height and the vertical offset of the destination region. (summarize)
<input type="checkbox"/> Value	Specifies the height of the destination region, in pixels.

DestBand

Specifies the color band of the destination region in which to copy. This parameter can be set to the same values as the [SrcBand](#) parameter, except for **M_NULL**.

TransferFunction

Specifies the function to use in the transfer. This parameter can be set to one of the following:

● For specifying the function to use in the transfer																								
☐ Value	Description	corona-II (a)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ed/xd (g)	lee 1394 i1dc (h)	iris (i)	met-II /cd (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	vio (w)		
☐ M_ALPHA_BLENDING	<p>Specifies a transfer function that combines an image with an overlay to create the appearance of partial transparency.</p> <p>The source and destination buffers must both be allocated on-board with a format of M_YUV16 + M_PACKED. When using MbufCreateColor(), the supported overlay buffer formats are M_YUV16 + M_PACKED and M_RGB16 + M_PACKED.</p> <p>Note that overlay buffers are only available when the TransferType parameter is set to M_DRIVER_METHOD.</p> (summarize)													o										
☐ M_BYTE_SWAP	<p>Specifies a transfer function that extracts the bytes in each pixel of the source region, swaps them, and copies the new pixels into the destination region.</p> <p>Note that M_BYTE_SWAP is only supported when the source and destination buffers are both either 1-band 16-bit buffers or 1-band 32-bit buffers and have the same internal storage specifiers.</p> <p>When using 16-bit buffers, the first byte is swapped with the last byte. When using 32-bit buffers, the first byte is swapped with the last byte and the two middle bytes are swapped.</p> <p>You can swap bytes within the same buffer. To do so, set the SrcBufId parameter to M_NULL.</p> (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
☐ M_CLEAR	<p>Specifies a transfer function that clears the destination region. Set all source buffer parameters to M_NULL.</p> (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
☐ M_COMPOSITION	<p>Specifies a transfer function that copies all pixels, except for pixels in the source region that are equal to the composition color specified by the OperationFlag parameter.</p> (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

<div><div></div><div>M_DEFAULT</div><div>+</div></div>	Transfers data using the most efficient transfer mode among the transfer modes listed.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_DIB_MODE</div><div>+</div></div>	<div>[This is only applicable to Windows]</div> <div>Transfers data using the DIB interface. Both the source and destination buffers must have the M_DIB attribute.</div> <div>This transfer mode is only possible when the TransferFunction parameter is set to M_COPY.</div> <div>You must specify a combination value from the table below.</div> <div>(summarize)</div>	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_DIRECTX_MODE</div><div>+</div></div>	<div>[This is only applicable to Windows]</div> <div>Transfers data using the DirectX interface. Both the source and destination buffers must have the M_DIRECTX attribute.</div> <div>This transfer mode is only possible when the TransferFunction parameter is set to M_CLEAR, M_COMPOSITION, or M_COPY.</div> <div>You must specify a combination value from the table below.</div> <div>(summarize)</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_DRIVER_METHOD</div><div>+</div></div>	<div>Transfers data using the driver.</div> <div>Note that overlay buffers are only available when the TransferFunction parameter is set to M_ALPHA_BLENDING and the ExtraParameter parameter is set to a MIL buffer identifier.</div> <div>You must specify a combination value from the table below.</div> <div>(summarize)</div>															o								
<div><div></div><div>M_MIL_MODE</div><div>+</div></div>	<div>Transfers data using the standard MIL mode.</div> <div>This transfer mode is always possible.</div> <div>You must specify a combination value from the table below.</div> <div>(summarize)</div>	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constants for the values listed in [For specifying the type of transfer](#)

You must add one of the following values to the above-mentioned values to set whether to transfer the data synchronously.

By default, the **MbufTransfer()** function chooses the value that is the most efficient for data transfer.

● For specifying synchronous data transfer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ASYNCHRONOUS	Specifies that the transfer of data is not synchronous. This setting is only possible when the TransferType parameter is equal to M_DIRECTX_MODE . (summarize)
<input type="checkbox"/> M_SYNCHRONOUS	Specifies that the transfer of data is synchronous.

OperationFlag

Specifies the value that is passed to the transfer function. If the **TransferFunction** parameter is set to **M_COPY** or **M_BYTE_SWAP**, this parameter must be set to **M_NULL**. If the **TransferFunction** parameter is set to **M_CLEAR** or **M_COMPOSITION**, set this parameter to one of the following values:

[Matrox Morphis QxT]

When dealing with an overlay buffer, this value specifies the overlay transparency value. Supported values are from 0 to 255.

● For specifying the value	
<input type="checkbox"/> Value	Description
<input type="checkbox"/>	<div> <div>vio (w)</div> <div>solios glge (v)</div> <div>solios ecd/xd (u)</div> <div>solios ea/xa (t)</div> <div>odysssey ed/xd (s)</div> <div>odysssey ecd/xd (r)</div> <div>odysssey ea/xa (q)</div> <div>nexis (p)</div> <div>morphis qxt (o)</div> <div>morphis (n)</div> <div>met-II/std (m)</div> <div>met-II /mc (l)</div> <div>met-II /dig (k)</div> <div>met-II /cl (j)</div> <div>ifs (i)</div> <div>leee I 394 ilic (h)</div> <div>helios ed/xd (g)</div> <div>helios ecd/xd (f)</div> <div>helios ea/xa (e)</div> <div>gpu processing (d)</div> <div>glge vision (c)</div> <div>cronoplus (b)</div> <div>corona-II (a)</div> </div>

M_RGB888(MIL_INT <i>Red</i> , MIL_INT <i>Green</i> , MIL_INT <i>Blue</i>)	Specifies an RGB value. Specify an RGB value when the source buffer is a 3-band buffer. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Parameters</i>																							
	<i>Red</i> Specifies the red component, as a value between 0 and 255.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Green</i> Specifies the green component, as a value between 0 and 255.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Blue</i> Specifies the blue component, as a value between 0 and 255.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_BLACK	Specifies the color black.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_BLUE	Specifies the color blue.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_BRIGHT_GRAY	Specifies the color bright gray.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_CYAN	Specifies the color cyan.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_BLUE	Specifies the color dark blue.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_CYAN	Specifies the color dark cyan.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_GREEN	Specifies the color dark green.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_MAGENTA	Specifies the color dark magenta.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_RED	Specifies the color dark red.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_YELLOW	Specifies the color dark yellow.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_GRAY	Specifies the color gray.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_GREEN	Specifies the color green.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_LIGHT_BLUE	Specifies the color light blue.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_LIGHT_GRAY	Specifies the color light gray.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_LIGHT_GREEN	Specifies the color light green.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_LIGHT_WHITE	Specifies the color light white.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_MAGENTA	Specifies the color magenta.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_RED	Specifies the color red.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_WHITE	Specifies the color white.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_YELLOW	Specifies the color yellow.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> Value	Specifies a grayscale value. Specify a grayscale value when the source buffer is a 1-band buffer. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	Specifies the overlay transparency value. A value of 0 represents 100% of the source buffer and 0% of the overlay buffer. A value of 255 represents 0% of the source buffer and 100% of the overlay buffer. A value of 128 represents 50% of the source buffer and 50% of the overlay buffer. Note that overlay buffers are only available when the TransferFunction parameter is set to M_ALPHA_BLENDING and the TransferType parameter is set to M_DRIVER_METHOD .																o							

ExtraParameter

Specifies a pointer to the MIL identifier of the overlay buffer, if applicable. Set this parameter to **M_NULL** in all other cases.

[Matrox Morphis QxT]
Note that overlay buffers are only available when the **TransferFunction** parameter is set to [M_ALPHA_BLENDING](#) and the **TransferType** parameter is set to [M_DRIVER_METHOD](#).

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

Mcal functions

Synopsis

The functions prefixed with Mcal make up the Calibration module. The Calibration module allows you to create a pixel coordinate to real-world coordinate mapping for your images, despite the presence of optical (for example, pincushion and barrel), aspect ratio, rotation, and perspective distortions in the images. Other MIL modules can use this mapping to compensate for distortions in the images, and return results in real-world units. The mapping can also be used to physically correct the distortions in the images. The calibration mapping can be created using a simple physical grid image or a list of points.

Functions

- [McalAlloc](#)
- [McalAssociate](#)
- [McalControl](#)
- [McalDraw](#)
- [McalFree](#)
- [McalGetCoordinateSystem](#)
- [McalGrid](#)
- [McalInquire](#)
- [McalInquireSingle](#)
- [McalList](#)
- [McalRelativeOrigin](#)
- [McalRestore](#)
- [McalSave](#)
- [McalSetCoordinateSystem](#)
- [McalStream](#)
- [McalTransformCoordinate](#)
- [McalTransformCoordinate3dList](#)
- [McalTransformCoordinateList](#)
- [McalTransformImage](#)
- [McalTransformResult](#)

McalAlloc

Synopsis

Allocate a calibration object.

Syntax

```
MIL_ID McalAlloc(  
    MIL_ID SystemId,  
    MIL_INT Mode,  
    MIL_INT ModeFlag,  
    MIL_ID *CalibrationIdPtr  
)
```

Description

This function allocates a calibration object. Use [McalGrid\(\)](#) or [McalList\(\)](#) to define the pixel-to-world mapping for the calibration object.

Parameters

SystemId

Specifies the system on which to allocate the calibration object. This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Mode

Specifies the calibration mode. This parameter must be set to one of the following values:

● For specifying the default calibration mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_LINEAR_INTERPOLATION .
<input type="checkbox"/> M_3D_ROBOTICS	Specifies to use a 3D calibration mode for a camera setup with the camera mounted on a robot arm at an unknown position.
<input type="checkbox"/> M_LINEAR_INTERPOLATION	Specifies to use piecewise linear interpolation.
<input type="checkbox"/> M_PERSPECTIVE_TRANSFORMATION	Specifies to use perspective transformation.
<input type="checkbox"/> M_TSAI_BASED	Specifies to use a 3D calibration mode based on the technique developed by Roger Y. Tsai. (summarize)

ModeFlag

Specifies the function's operation flag. This parameter must be set to **M_DEFAULT**.

CalibrationIdPtr

Specifies the address in which to return the identifier of the calibration object. Since **McalAlloc()** also returns the identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the calibration object identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalAssociate

Synopsis

Associate/disassociate a calibration object to/from an image or digitizer

Syntax

```
void McalAssociate(
    MIL_ID CalibrationId,
    MIL_ID ImageOrDigitizerId,
    MIL_INT ControlFlag
)
```

Description

This function associates a calibration object to an image or digitizer. It can also be used to disassociate a calibration object from an image or digitizer. Note that you do not have to first disassociate a calibration object from an image or digitizer to associate it with a different calibration object.

The calibration object gets associated to the image, not to the buffer containing the image. This implies that if you copy or process the image, the operation will copy the image's calibration settings to the destination image.

When you grab an image with a calibrated digitizer, the calibration object currently associated to the digitizer gets associated to the grabbed image.

When you associate a calibration object to an image (either with a call to **McalAssociate()** or a grab with a calibrated digitizer), the image receives a copy of the calibration object's current relative coordinate system and current camera position and a reference to the calibration object for all other settings. When you associate a calibration object to a digitizer, the digitizer only receives a reference to the calibration object.

When a calibrated image is used within a MIL module, results from this module can be returned in pixel units or in real-world units; use the calibration object's **M_OUTPUT_UNITS** control to specify your preference. This control is set by **McalControl()**. By default, this control is set to return results in real-world units.

Note that a few results are always returned in pixel units. If a result can be returned in either real-world or pixel units, it will be stated in the function description.

When associating a calibration object with an image, its child images, if any, will also be associated with the same calibration object; their offset is automatically taken into account.

Parameters

CalibrationId

Specifies the identifier of the calibration object you want to associate. To disassociate a calibration object from an image or digitizer, set this parameter to **M_NULL**.

ImageOrDigitizerId

Specifies the identifier of the image or digitizer.

ControlFlag

Specifies the function's control flag. This parameter must be set to the following value:

For specifying the function's control flag	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Sets the function's control to the default.

Compilation information

Header	Include mil.h.

Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalControl

Synopsis

Control a calibration object parameter setting.

Syntax

```
void McalControl(  
    MIL_ID CalibrationOrMilId,  
    MIL_INT ControlType,  
    MIL_DOUBLE ControlValue  
)
```

Description

This function changes a setting of a calibration object.

Parameters

CalibrationOrMilId

Specifies the identifier of the calibration object or calibrated image.

ControlType

Specifies the setting to change.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the setting's new value.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For a calibration object](#)
- [For a calibrated image](#)

The following [ControlType](#) and [ControlValue](#) parameter settings can be specified for a calibration object:

For a calibration object	
ControlType	Description
ControlValue	
M_CALIBRATION_PLANE	Sets the plane in which the calibration points are defined. (summarize)
M_DEFAULT	Same as M_ABSOLUTE_COORDINATE_SYSTEM .
M_ABSOLUTE_COORDINATE_SYSTEM	Specifies that the calibration points are defined in the absolute coordinate system.
M_RELATIVE_COORDINATE_SYSTEM	Specifies that the calibration points are defined in the relative coordinate system.

<input type="checkbox"/> M_CCD_ASPECT_RATIO	Sets the width-to-height ratio of the individual elements of the CCD. If this value is not set and is required by the calibration mode, the ratio will default to 1.0. (summarize)
<input type="checkbox"/> 1.0	Specifies that the width and height of the CCD element are equal. This is the default value. (summarize)
<input type="checkbox"/> Value > 0	Specifies the value of the width of a CCD element divided by its height.
<input type="checkbox"/> M_FOREGROUND_VALUE	Sets whether the circles in a circle grid, used with McalGrid() , are lighter or darker than the background. (summarize)
<input type="checkbox"/> M_DEFAULT	Determines the appropriate setting automatically.
<input type="checkbox"/> M_FOREGROUND_BLACK	Specifies that the grid's circles are darker than the background.
<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that the grid's circles are lighter than the background.
<input type="checkbox"/> M_GRID_CORNER_HINT_X	Specifies the approximate X-coordinate, in the image, corresponding to the top-left corner of the calibration grid in the real world. If the image of the calibration grid is rotated, such that the real-world top-left corner of the grid is not located in the top-left corner of the image, M_GRID_CORNER_HINT_X specifies the X-coordinate of the real-world top-left corner of the calibration grid in the image coordinate system. The corner grid point nearest to the specified coordinates (M_GRID_CORNER_HINT_X and M_GRID_CORNER_HINT_Y) will be used. This control type is only used for calibrations performed using McalGrid() . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_NONE .
<input type="checkbox"/> M_NONE	Specifies to use the top-left grid point in the image.
<input type="checkbox"/> Value	Specifies the X-coordinate of the hint point, in the image.
<input type="checkbox"/> M_GRID_CORNER_HINT_Y	Specifies the approximate Y-coordinate, in the image, corresponding to the top-left corner of the calibration grid in the real world. If the image of the calibration grid is rotated, such that the real-world top-left corner of the grid is not located in the top-left corner of the image, M_GRID_CORNER_HINT_X specifies the Y-coordinate of the real-world top-left corner of the calibration grid in the image coordinate system. The corner grid point nearest to the specified coordinates (M_GRID_CORNER_HINT_X and M_GRID_CORNER_HINT_Y) will be used. This control type is only used for calibrations performed using McalGrid() . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_NONE .
<input type="checkbox"/> M_NONE	Specifies to use the top-left grid point in the image.
<input type="checkbox"/> Value	Specifies the Y-coordinate of the hint point, in the image.
<input type="checkbox"/> M_LINK_TOOL_AND_CAMERA	Creates a rigid link between the camera coordinate system and the tool coordinate system. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ENABLE .
<input type="checkbox"/> M_DISABLE	Disables the link between the two coordinate systems, allowing both to be moved independently.
<input type="checkbox"/> M_ENABLE	Enables the link between the two coordinate systems, allowing both to be moved together.
<input type="checkbox"/> M_OUTPUT_UNITS	Sets whether MIL modules will use pixels or world units to return results. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_WORLD .
<input type="checkbox"/> M_PIXEL	Specifies that results are returned in pixel units.
<input type="checkbox"/> M_WORLD	Specifies that results are returned in world units.
<input type="checkbox"/> M_PRINCIPAL_POINT_X	Sets the X-coordinate, in pixels, of the intersection of the camera's optical axis and the image plane. (summarize)

<input type="checkbox"/> Value	Specifies the X-coordinate in pixels. If this value is not set and is required by the calibration method, it will default to half of the image's width. (summarize)
<input type="checkbox"/> M_PRINCIPAL_POINT_Y	Sets the Y-coordinate, in pixels, of the intersection of the camera's optical axis and the image plane. (summarize)
<input type="checkbox"/> Value	Specifies the Y-coordinate in pixels. The default value is half the image's height. If this value is not set and is required by the calibration method, it will default to half of the image's height. (summarize)
<input type="checkbox"/> M_TOOL_POSITION_X	Sets the X-position of the tool coordinate system in the absolute coordinate system. (summarize)
<input type="checkbox"/> 0.0	This is the default value.
<input type="checkbox"/> Value	Specifies the X-coordinate in world units.
<input type="checkbox"/> M_TOOL_POSITION_Y	Sets the Y-position of the tool coordinate system in the absolute coordinate system. (summarize)
<input type="checkbox"/> 0.0	This is the default value.
<input type="checkbox"/> Value	Specifies the Y-coordinate in world units.
<input type="checkbox"/> M_TOOL_POSITION_Z	Sets the Z-position of the tool coordinate system in the absolute coordinate system. When using the linear interpolation or perspective transformation calibration mode, this must be set to M_DEFAULT . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies that 0.0 is the Z-position of the tool coordinate system in the absolute coordinate system. (summarize)
<input type="checkbox"/> Value	Specifies the Z-coordinate in world units.
<input type="checkbox"/> M_TRANSFORM_CACHE	Sets whether to use a cache to accelerate the McalTransformImage() function. (summarize)
<input type="checkbox"/> M_DISABLE	Disables the cache. Disabling the cache saves memory but slows down subsequent calls to McalTransformImage() . (summarize)
<input type="checkbox"/> M_ENABLE	Enables the cache. This is the default value. (summarize)
<input type="checkbox"/> M_TRANSFORM_IMAGE_FILL_MODE	Sets the mode in which the source image is scaled and positioned when filling the destination image using McalTransformImage() . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_FIT .
<input type="checkbox"/> M_CLIP	Specifies that the source image is scaled and positioned such that every destination pixel maps to a valid source pixel. This fill mode does not produce any invalid pixels in the destination image, but some information might be lost. (summarize)
<input type="checkbox"/> M_FIT	Specifies that the source image is scaled and positioned such that every source pixel maps to a valid destination pixel. This fill mode might produce invalid (undefined) pixels in the destination image, but no information is lost. (summarize)
<input type="checkbox"/> M_USER_DEFINED	Specifies that the source image is scaled and positioned according to the M_TRANSFORM_IMAGE_PIXEL_SIZE_X , M_TRANSFORM_IMAGE_PIXEL_SIZE_Y , M_TRANSFORM_IMAGE_WORLD_POS_X , and M_TRANSFORM_IMAGE_WORLD_POS_Y control types.
<input type="checkbox"/> M_TRANSFORM_IMAGE_PIXEL_SIZE_X	Sets the width of the pixels, in real-world units, in the destination image of McalTransformImage() . The width is specified in real-world units and is only used when M_TRANSFORM_IMAGE_FILL_MODE is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)

<input type="checkbox"/> Value > 0	Specifies the real-world width of the pixels in the destination image.
<input type="checkbox"/> M_TRANSFORM_IMAGE_PIXEL_SIZE_Y	Sets the height of the pixels, in real-world units, in the destination image of McalTransformImage() . The height is specified in real-world units and is only used when M_TRANSFORM_IMAGE_FILL_MODE is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the real-world height of the pixels in the destination image.
<input type="checkbox"/> M_TRANSFORM_IMAGE_WORLD_POS_X	Sets the X-offset, in real-world units, of the top-left pixel in the destination image of McalTransformImage() . The offset is specified in real-world units and is only used when M_TRANSFORM_IMAGE_FILL_MODE is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the real-world X-offset of the top-left pixel of the destination image.
<input type="checkbox"/> M_TRANSFORM_IMAGE_WORLD_POS_Y	Sets the Y-offset, in real-world units, of the top-left pixel in the destination image of McalTransformImage() . The offset is specified in real-world units and is only used when M_TRANSFORM_IMAGE_FILL_MODE is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the real-world Y-offset of the top-left pixel of the destination image.

The following [ControlType](#) and [ControlValue](#) parameter settings can be specified for a calibrated image.

For a calibrated image	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_CALIBRATION_CHILD_OFFSET_X	Sets the X-offset of a child buffer, relative to its highest order calibrated parent buffer. (summarize)
<input type="checkbox"/> Value	Specifies the X-offset, relative to the child buffer's highest order calibrated parent buffer.
<input type="checkbox"/> M_CALIBRATION_CHILD_OFFSET_Y	Sets the Y-offset of a child buffer, relative to its highest order calibrated parent buffer. (summarize)
<input type="checkbox"/> Value	Specifies the Y-offset, relative to the child buffer's highest order calibrated parent buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalDraw

Synopsis

Draw specific features of results obtained from a calibration operation.

Syntax

```
void McalDraw(  
    MIL_ID GraphContId,  
    MIL_ID CalibrationId,  
    MIL_ID DestImageBufId,  
    MIL_INT Operation,  
    MIL_INT Index,  
    MIL_INT ControlFlag  
)
```

Description

This function draws specific features of results, obtained from a calibration operation, in the destination image buffer.

If the destination image is calibrated and [M_DRAW_WORLD_POINTS](#) is used, the calibration object associated with the image is used to transform the calibration points. However, if the destination image is not calibrated and [M_DRAW_WORLD_POINTS](#) is used, the calibration object will be used to transform the calibration points.

Parameters

GraphContId

Specifies the graphics context to use when drawing.

For specifying the graphics context	
Value	Description
M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
MIL_graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

CalibrationId

Specifies the calibration object from which to extract the calibration information to draw. The calibration object must have been previously allocated on the required system using [McalAlloc\(\)](#). The calibration object should have been successfully calibrated using [McalList\(\)](#) or [McalGrid\(\)](#). When working in [M_3D_ROBOTICS](#) calibration mode, it is possible to use [McalDraw\(\)](#) with [M_DRAW_IMAGE_POINTS](#) between two successful calls to [McalGrid\(\)](#) / [McalList\(\)](#), to draw the calibration points of one calibration pose before being fully calibrated.

DestImageBufId

Specifies the identifier of the destination image buffer in which to draw. This buffer must be an unsigned 8-bit image buffer. By drawing into the display's overlay buffer, you can also annotate an image.

Operation

Specifies the type of operation to perform.

For specifying the operation type	
Value	Description
M_DRAW_IMAGE_POINTS	Draws the calibration points used to calibrate.

<input type="checkbox"/> M_DRAW_VALID_REGION	Draws a contour of the rectangle covered by the grid in M_LINEAR_INTERPOLATION mode. The points transformed in this region are never extrapolated, only interpolated. (summarize)
<input type="checkbox"/> M_DRAW_VALID_REGION_FILLED	Draws a filled rectangle covered by the grid in M_LINEAR_INTERPOLATION mode. The points transformed in this region are never extrapolated, only interpolated. (summarize)
<input type="checkbox"/> M_DRAW_WORLD_POINTS	Draws real-world positions of calibration points used with McalGrid() or McalList() . These positions are transformed to pixel units in the destination image using the calibration object associated with it, or using CalibrationId if the destination image is not calibrated. Note that, when transforming calibration points in the world, the position of the camera at calibration time is used, not the current position. (summarize)

Index

Specifies the index of the pose of a [M_3D_ROBOTICS](#) calibration object from which the calibration points should be drawn. This index should be between 0 and the return value of [McalInquire\(\)](#) with [M_NUMBER_OF_CALIBRATION_POSES](#) minus one. When using a calibration mode different than [M_3D_ROBOTICS](#) calibration mode or when not drawing calibration points, this parameter should be set to **M_DEFAULT**.

If the destination image is calibrated, [Index](#) must be 0 or **M_DEFAULT**.

● For specifying the index of the pose	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies to draw the calibration points of the last pose for M_3D_ROBOTICS calibration objects. This setting must be used when the calibration object is not in M_3D_ROBOTICS calibration mode, or when not drawing calibration points. (summarize)
<input type="checkbox"/> 0 <= Value <= (Number of poses - 1)	Specifies the index of a calibration pose. Use McalInquire() with M_NUMBER_OF_CALIBRATION_POSES to determine the number of poses. (summarize)

ControlFlag

Specifies the function's control flag and should be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalFree

Synopsis

Free a calibration object.

Syntax

```
void McalFree(  
    MIL_ID CalibrationId  
)
```

Description

This function frees a calibration object.

Parameter

CalibrationId

Specifies the identifier of the calibration object.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalGetCoordinateSystem

Synopsis

Return the position and orientation of a coordinate system

Syntax

```
void McalGetCoordinateSystem(
    MIL_ID CalibrationOrImageId,
    MIL_INT TargetCoordinateSystem,
    MIL_INT ReferenceCoordinateSystem,
    MIL_INT InquireType,
    MIL_ID MatrixId,
    MIL_DOUBLE *Param1,
    MIL_DOUBLE *Param2,
    MIL_DOUBLE *Param3,
    MIL_DOUBLE *Param4
)
```

Description

This function returns the position and orientation of one coordinate system as a transformation of another coordinate system. For example, you can use this function to inquire about the position of your camera in the absolute coordinate system; essentially, you can inquire about the origin orientation of the camera coordinate system with respect to the absolute coordinate system.

The specified target and reference coordinate systems must be defined prior to calling this function.

Parameters

CalibrationOrImageId

Specifies the identifier of the calibration object or calibrated image.

If you use a calibration object to inquire about the position and orientation of a coordinate system, the results returned are relevant to the current position and orientation of that coordinate system. However, if you use a calibrated image, results are relevant to the position and orientation of the coordinate system only at the time the image was calibrated or associated.

TargetCoordinateSystem

Specifies which coordinate system to inquire. This parameter must be set to one of the following values:

● For specifying the target coordinate system:	
Value	Description
M_ABSOLUTE_COORDINATE_SYSTEM	Specifies an implicit and unmovable coordinate system from which all other coordinate systems are defined. By default, it corresponds to the center of the top-left calibration point when using McalGrid() . (summarize)
M_CAMERA_COORDINATE_SYSTEM	Specifies the coordinate system whose origin corresponds to the effective pinhole of the camera and whose Z-axis points in the direction that the camera is facing. This coordinate system is only defined after a successful calibration in M_TSAI_BASED or M_3D_ROBOTICS mode, or by using McalSetCoordinateSystem() with M_ASSIGN . (summarize)
M_RELATIVE_COORDINATE_SYSTEM	Specifies the coordinate system determining the world plane used to return results which can be recentered and/or reoriented using McalSetCoordinateSystem() or McalRelativeOrigin() . By default, it corresponds to the absolute coordinate system. (summarize)
M_ROBOT_BASE_COORDINATE_SYSTEM	Specifies a coordinate system whose origin is the base of the robot encoder holding the robotic arm. This coordinate system is only supported for M_3D_ROBOTICS calibration mode. It is defined using McalSetCoordinateSystem() with M_ASSIGN , or after a successful calibration with McalGrid() or McalList() .

	(summarize)
<input type="checkbox"/> M_TOOL_COORDINATE_SYSTEM	Specifies the coordinate system used to position the camera. Although you can use this coordinate system to move the camera, it needs not be associated with the real camera position. By default, its axes are parallel to the absolute coordinate system, and its origin is the same as that of the absolute coordinate system. (summarize)

ReferenceCoordinateSystem

Specifies the coordinate system that will be used as a reference for calculations. All returned coordinates or angles will be calculated relative to this coordinate system. This parameter must be set to one of the following values:

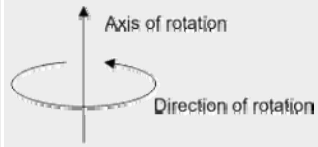
● For specifying the reference coordinate system:	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ABSOLUTE_COORDINATE_SYSTEM .
<input type="checkbox"/> M_ABSOLUTE_COORDINATE_SYSTEM	Specifies an implicit and unmovable coordinate system from which all other coordinate systems are defined. By default, it corresponds to the center of the top-left calibration point when using McalGrid() . (summarize)
<input type="checkbox"/> M_CAMERA_COORDINATE_SYSTEM	Specifies the coordinate system whose origin corresponds to the effective pinhole of the camera and whose Z-axis points in the direction that the camera is facing. This coordinate system can only be used as a reference coordinate system after it has been defined, either by a successful calibration in M_TSAI_BASED mode or using McalSetCoordinateSystem() with M_ASSIGN . (summarize)
<input type="checkbox"/> M_RELATIVE_COORDINATE_SYSTEM	Specifies the coordinate system determining the world plane used to return results which can be recentered and/or reoriented using McalSetCoordinateSystem() or McalRelativeOrigin() . By default, it corresponds to the absolute coordinate system. (summarize)
<input type="checkbox"/> M_ROBOT_BASE_COORDINATE_SYSTEM	Specifies the coordinate system whose origin is the base of the robot encoder holding the robotic arm. This coordinate system is only supported for M_3D_ROBOTICS calibration mode. It is defined using McalSetCoordinateSystem() with M_ASSIGN , or after a successful calibration with McalGrid() or McalList() . (summarize)
<input type="checkbox"/> M_TOOL_COORDINATE_SYSTEM	Specifies the coordinate system used to position the camera. Though this is the coordinate system used to move the camera it needs not be associated to the real camera position. By default, its axes are parallel to the absolute coordinate system, and its origin is the same as that of the absolute coordinate system. (summarize)

InquireType

Specifies the type of transformation to use to return the location of the target coordinate system with respect to the reference coordinate system.

Any parameters for which a value is not returned ([MatrixId](#), and [Param1](#) to [Param4](#)) must be set to **M_NULL**.

● For inquiring about the transformation parameters:			
<input type="checkbox"/> Value	Description		
<input type="checkbox"/> M_HOMOGENEOUS_MATRIX	<p>Returns the transformation which makes the axes of the reference coordinate system parallel to those of the target coordinate system, and makes the origin of reference coordinate system coincide with that of target coordinate system. The transformation is returned as a 4x4 homogenous matrix. (summarize)</p> <div> <p><i>MatrixId info</i></p> <p>Return values:</p> <table> <tr> <td>Value</td><td>Matrix representation of the transformation which causes the reference coordinate system to overlap with the target coordinate system.</td></tr> </table> </div>	Value	Matrix representation of the transformation which causes the reference coordinate system to overlap with the target coordinate system.
Value	Matrix representation of the transformation which causes the reference coordinate system to overlap with the target coordinate system.		
<input type="checkbox"/> M_ROTATION_AXIS_ANGLE	Returns the axis and angle of rotation which makes the axes of the reference coordinate system parallel to those of the target coordinate system. The axis of rotation is defined by the origin of reference coordinate system and one other point. The rotation angle is measured in the counter-clockwise direction around the axis of rotation, as per the right-hand rule.		



The axis of rotation is always normalized.

When the axes of the coordinate systems are already parallel, [Param1](#) to [Param4](#) will be set to (0, 0, 0, 0).

[\(summarize\)](#)

Param1 Info

Return values:

Value X-coordinate of one of the points which defines the axis of rotation.

Param2 Info

Return values:

Value Y-coordinate of one of the points which defines the axis of rotation.

Param3 Info

Return values:

Value Z-coordinate of one of the points which defines the axis of rotation.

Param4 Info

Return values:

0 <= Value < 360 Angle of the rotation in degrees around the axis of rotation.

☐ **M_ROTATION_MATRIX**

Returns the rotation matrix that makes the axes of the reference coordinate system parallel to those of the target coordinate system. The transformation is returned as a 3x3 matrix.

[\(summarize\)](#)

MatrixId Info

Return values:

Value Matrix representation of the rotation which brings the axes of [ReferenceCoordinateSystem](#) parallel to those of [TargetCoordinateSystem](#).

☐ **M_ROTATION_QUATERNION**

Returns the components of the quaternion that defines the rotation which makes the axes of [ReferenceCoordinateSystem](#) parallel to those of [TargetCoordinateSystem](#).

The quaternion is always normalized.

When the axes of the coordinate systems are already parallel, [Param1](#) to [Param4](#) will be set to (1, 0, 0, 0).

[\(summarize\)](#)

Param1 Info

Return values:

Value Scalar component of the quaternion.

Param2 Info

Return values:

Value X-component of the quaternion.

Param3 Info

Return values:

Value Y-component of the quaternion.

Param4 Info

Return values:

Value Z-component of the quaternion.

☐ **M_ROTATION_XYZ**

Returns the angles of the three rotations that make the axes of the reference coordinate system parallel to those of the target coordinate system. The rotation is described by three distinct rotations about the axes of the reference coordinate system in the following order: the X-axis, Y-axis, and Z-axis rotation.

[\(summarize\)](#)

	<p><i>Param1 Info</i> Return values: 0.0 <= Value < 360.0 Returns the X-axis rotation, in degrees.</p> <p><i>Param2 Info</i> Return values: 270.0 <= Value < 360.0 0.0 <= Value <= 90.0 Returns the Y-axis rotation, in degrees.</p> <p><i>Param3 Info</i> Return values: 0.0 <= Value < 360.0 Returns the Z-axis rotation, in degrees.</p>
<input type="checkbox"/> M_ROTATION_XZY	<p>Returns the angles of the three rotations that make the axes of the reference coordinate system parallel to those of the target coordinate system. The rotation is described by three distinct rotations about the axes of the reference coordinate system in the following order: the X-axis, Z-axis, and Y-axis rotation. (summarize)</p> <p><i>Param1 Info</i> Return values: 0.0 <= Value < 360.0 Returns the X-axis rotation, in degrees.</p> <p><i>Param2 Info</i> Return values: 270.0 <= Value < 360.0 0.0 <= Value <= 90.0 Returns the Z-axis rotation, in degrees.</p> <p><i>Param3 Info</i> Return values: 0.0 <= Value < 360.0 Returns the Y-axis rotation, in degrees.</p>
<input type="checkbox"/> M_ROTATION_YXZ	<p>Returns the angles of the three rotations that make the axes of the reference coordinate system parallel to those of the target coordinate system. The rotation is described by three distinct rotations about the axes of the reference coordinate system in the following order: the Y-axis, X-axis, and Z-axis rotation. (summarize)</p> <p><i>Param1 Info</i> Return values: 0.0 <= Value < 360.0 Returns the Y-axis rotation, in degrees.</p> <p><i>Param2 Info</i> Return values: 270.0 <= Value < 360.0 0.0 <= Value <= 90.0 Returns the X-axis rotation, in degrees.</p> <p><i>Param3 Info</i> Return values: 0.0 <= Value < 360.0 Returns the Z-axis rotation, in degrees.</p>
<input type="checkbox"/> M_ROTATION_YZX	<p>Returns the angles of the three rotations that make the axes of the reference coordinate system parallel to those of the target coordinate system. The rotation is described by three distinct rotations about the axes of the reference coordinate system in the following order: the Y-axis, Z-axis, and X-axis rotation. (summarize)</p> <p><i>Param1 Info</i> Return values: 0.0 <= Value < 360.0 Returns the Y-axis rotation, in degrees.</p> <p><i>Param2 Info</i> Return values: 270.0 <= Value < 360.0 0.0 <= Value <= 90.0 Returns the Z-axis rotation, in degrees.</p>

	<i>Param3 Info</i> Return values: 0.0 <= Value < 360.0 Returns the X-axis rotation, in degrees.
<input type="checkbox"/> M_ROTATION_ZXY	Returns the angles of the three rotations that make the axes of the reference coordinate system parallel to those of the target coordinate system. The rotation is described by three distinct rotations about the axes of the reference coordinate system in the following order: the Z-axis, X-axis, and Y-axis rotation. (summarize)
	<i>Param1 Info</i> Return values: 0.0 <= Value < 360.0 Returns the Z-axis rotation, in degrees.
	<i>Param2 Info</i> Return values: 270.0 <= Value < 360.0 0.0 <= Value <= 90.0 Returns the X-axis rotation, in degrees.
	<i>Param3 Info</i> Return values: 0.0 <= Value < 360.0 Returns the Y-axis rotation, in degrees.
<input type="checkbox"/> M_ROTATION_ZYX	Returns the angles of the three rotations that make the axes of the reference coordinate system parallel to those of the target coordinate system. The rotation is described by three distinct rotations about the axes of the reference coordinate system in the following order: Z-axis, Y-axis, and X-axis rotation. (summarize)
	<i>Param1 Info</i> Return values: 0.0 <= Value < 360.0 Returns the Z-axis rotation, in degrees.
	<i>Param2 Info</i> Return values: 270.0 <= Value < 360.0 0.0 <= Value <= 90.0 Returns the Y-axis rotation, in degrees.
	<i>Param3 Info</i> Return values: 0.0 <= Value < 360.0 Returns the X-axis rotation, in degrees.
<input type="checkbox"/> M_TRANSLATION	Returns the components of the translation vector that makes the origin of the reference coordinate system coincide with the origin of the target coordinate system. (summarize)
	<i>Param1 Info</i> Return values: Value Displacement along the X-axis of the reference coordinate system.
	<i>Param2 Info</i> Return values: Value Displacement along the Y-axis of the reference coordinate system.
	<i>Param3 Info</i> Return values: Value Displacement along the Z-axis of the reference coordinate system.

MatrixId

See [InquireType](#) table above for possible return values. If a particular inquire type does not return a value for this parameter it must be set to **M_NULL**.

[MatrixId](#) must be a 32-bit float buffer, allocated using [MbufAlloc2d\(\)](#) with [M_ARRAY](#). The required buffer size will depend on the results being returned.

Param1

See [InquireType](#) table above for possible return values. If a particular inquire type does not return a value for this parameter it must be set to **M_NULL**.

Param2

See [InquireType](#) table above for possible return values. If a particular inquire type does not return a value for this parameter it must be set to **M_NULL**.

Param3

See [InquireType](#) table above for possible return values. If a particular inquire type does not return a value for this parameter it must be set to **M_NULL**.

Param4

See [InquireType](#) table above for possible return values. If a particular inquire type does not return a value for this parameter it must be set to **M_NULL**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalGrid

Synopsis

Calibrate your camera setup using a grid.

Syntax

```
void McalGrid(
    MIL_ID CalibrationId,
    MIL_ID SrcImageBufId,
    MIL_DOUBLE GridOffsetX,
    MIL_DOUBLE GridOffsetY,
    MIL_DOUBLE GridOffsetZ,
    MIL_INT RowNumber,
    MIL_INT ColumnNumber,
    MIL_DOUBLE RowSpacing,
    MIL_DOUBLE ColumnSpacing,
    MIL_INT Operation,
    MIL_INT GridType
)
```

Description

This function uses an image of a user-defined grid of circles or squares and the world description of this grid to establish calibration points and calibrate your camera setup. The mapping is stored with the specified calibration object. The specified calibration object determines the calibration mode used.

Typically, you specify the world description of the grid in the absolute coordinate system. However, for 3D-based calibration modes ([M_TSAI_BASED](#) or [M_3D_ROBOTICS](#)), it is possible to use a grid that is not in the Z=0 plane of the absolute coordinate system. In this case, it is possible to describe the grid in the relative coordinate system instead, using [McalControl\(\)](#) with [M_CALIBRATION_PLANE](#) set to [M_RELATIVE_COORDINATE_SYSTEM](#). You can then use [McalSetCoordinateSystem\(\)](#) to move the relative coordinate system. Both of these calls must be made before calling **McalGrid()**.

When working in [M_3D_ROBOTICS](#) calibration, you must perform at least three calls to **McalGrid()** with [M_ACCUMULATE](#) before performing a full calibration. Before each call, you must change the position and orientation of the tool holding the camera with respect to the robot base. More specifically, you must rotate the tool along at least two non-parallel axes. You must also use [McalSetCoordinateSystem\(\)](#) to set the position of the tool coordinate system with respect to the robot base coordinate system before each of these calls. Using **McalGrid()** with [M_ACCUMULATE](#) more than three times before performing a full calibration improves the accuracy of your calibration.

You can inquire about the pixel coordinates and associated real-world coordinates of each calibration point of one particular call using [McalInquireSingle\(\)](#).

For 3D-based calibration modes, **McalGrid()** performs two types of calculations. It calculates the camera's internal attributes, and it calculates the orientation and distance between the camera and calibration plane. Initially, the latter is used to set the camera coordinate system. If you move the camera or the grid (not both), you can have **McalGrid()** recalculate only the orientation and distance between them. In this case, depending on what you moved, you can specify that **McalGrid()** displace either the camera coordinate system or the relative coordinate system, using [M_DISPLACE_CAMERA_COORD](#) or [M_DISPLACE_RELATIVE_COORD](#), respectively. When using [M_DISPLACE_RELATIVE_COORD](#), the calibration plane is considered to be the relative coordinate system regardless of the [M_CALIBRATION_PLANE](#) setting.

Also, for 3D-based calibration modes, a successful call to **McalGrid()** with [M_FULL_CALIBRATION](#) creates a rigid link between the camera coordinate system and the tool coordinate system. This link ensures that moving either the tool or the camera coordinate systems will affect both. This link can be broken using [McalControl\(\)](#) with [M_LINK_TOOL_AND_CAMERA](#).

Note that the internal attributes calculated for the camera are not those of the physical camera, but those of the ideal pinhole-camera used to model the physical camera.

After calling **McalGrid()**, you can inquire about the success of the calibration using [McalInquire\(\)](#) with [M_CALIBRATION_STATUS](#). Also, you can inquire about the pixel coordinates and associated real-world coordinates of each calibration point using [McalInquire\(\)](#) with [M_CALIBRATION_IMAGE_POINTS_X](#), [M_CALIBRATION_IMAGE_POINTS_Y](#), [M_CALIBRATION_WORLD_POINTS_X](#) and [M_CALIBRATION_WORLD_POINTS_Y](#). These coordinates correspond to the center of circles in a circle grid or to points connecting four corners in a chessboard grid.

For more information on performing calibration, see the [Steps to performing a calibration](#) section in [Chapter 5: Camera calibration](#).

Parameters

CalibrationId

Specifies the identifier of the calibration object.

SrcImageBufId

Specifies the identifier of the image buffer containing the grid. The buffer must be 8- or 16-bit unsigned, with 1 or 3 bands.

Note that the source image used for calibration is automatically calibrated after a successful call to `McalGrid()` with `M_FULL_CALIBRATION`.

When performing a `M_FULL_CALIBRATION` operation in `M_3D_ROBOTICS` calibration mode, set this parameter to `M_NULL`.

GridOffsetX

Specifies the X-position of the top-left circle of the calibration grid of circles or the top-left point connecting four squares in a chessboard grid. The offset is set in the calibration-plane coordinate system in world-units.

When performing a `M_FULL_CALIBRATION` operation in `M_3D_ROBOTICS` calibration mode, set this parameter to `M_NULL`.

GridOffsetY

Specifies the Y-position of the top-left circle of the calibration grid of circles or the top-left point connecting four squares in a chessboard grid. The offset is set in the calibration-plane coordinate system in world-units.

When performing a `M_FULL_CALIBRATION` operation in `M_3D_ROBOTICS` calibration mode, set this parameter to `M_NULL`.

GridOffsetZ

Specifies the Z-position of the top-left circle of the calibration grid of circles or the top-left point connecting four squares in a chessboard grid. The offset is set in the calibration-plane coordinate system in world-units. This parameter must be set to `M_NULL`.

RowNumber

Specifies the number of rows in the calibration grid. The minimum number of rows is 2.

For `M_LINEAR_INTERPOLATION` and `M_PERSPECTIVE_TRANSFORMATION` calibration modes, the minimum number of grid points required is 4. For `M_TSAI_BASED` and `M_3D_ROBOTICS` calibrations, you must provide a grid that contains at least 6 points, even though the minimum number of rows and the minimum number of columns are both 2. Accuracy increases with the number of points.

When performing a `M_FULL_CALIBRATION` operation in `M_3D_ROBOTICS` calibration mode, set this parameter to `M_NULL`.

ColumnNumber

Specifies the number of columns in the calibration grid. The minimum number of columns is 2.

When performing a `M_FULL_CALIBRATION` operation in `M_3D_ROBOTICS` calibration mode, set this parameter to `M_NULL`.

RowSpacing

Specifies the number of world units between rows.

When performing a `M_FULL_CALIBRATION` operation in `M_3D_ROBOTICS` calibration mode, set this parameter to `M_NULL`.

ColumnSpacing

Specifies the number of world units between columns.

When performing a `M_FULL_CALIBRATION` operation in `M_3D_ROBOTICS` calibration mode, set this parameter to `M_NULL`.

Operation

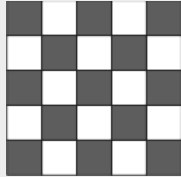
Specifies the calibration operation to perform. This parameter must be set to one of the following values:

• For specifying the calibration operation

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_FULL_CALIBRATION .
<input type="checkbox"/> M_ACCUMULATE	Calculates only the position of the calibration points and stores them in the calibration object. At least three calls to <code>McalGrid()</code> should be made with this operation before performing full calibration with M_FULL_CALIBRATION . This operation is only supported for M_3D_ROBOTICS calibration. (summarize)
<input type="checkbox"/> M_DISPLACE_CAMERA_COORD	Calculates only the position and orientation between the camera and the calibration plane, and displaces the camera coordinate system accordingly. This calibration operation is only supported for 3D-based calibration objects that are fully calibrated with M_FULL_CALIBRATION . If the camera is moved to a new position but its internal attributes are already known by a previous full calibration, you can use this operation to allow for a faster calibration. (summarize)
<input type="checkbox"/> M_DISPLACE_RELATIVE_COORD	Calculates only the position and orientation between the camera and the calibration plane, and displaces the relative coordinate system accordingly. This calibration operation is only supported for 3D-based calibration objects that are fully calibrated with M_FULL_CALIBRATION . You can use this operation to move the relative coordinate system only and remain calibrated. This operation keeps the camera fixed with respect to the absolute coordinate system, and its intrinsic attributes unmodified. You can then obtain the position and orientation of the relative coordinate system with respect to any other coordinate system using <code>McalGetCoordinateSystem()</code> , and calculate the unknown position of an object in space. When using this operation, the calibration plane is considered to be the relative coordinate system regardless of the M_CALIBRATION_PLANE setting. (summarize)
<input type="checkbox"/> M_FULL_CALIBRATION	Performs a full calibration. For 3D-based calibration objects, this operation calculates the camera's internal attributes and the difference in position and orientation between the camera and the calibration plane. It then sets the camera coordinate system accordingly. When working in M_TSAI_BASED calibration mode, the camera's optical axis should be at least 30 degrees away from the axis perpendicular to the calibration plane (also known as, angle of incidence). Otherwise, the calibration might fail. (summarize)

GridType

Specifies the type of grid used. This parameter must be set to one of the following values:

● For specifying the type of grid	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_CIRCLE_GRID .
<input type="checkbox"/> M_CHESSBOARD_GRID +	Specifies a chessboard grid. The number of rows and columns represents the number of corners horizontally and vertically. A corner is the intersection of 4 cells, meaning that the corners on the perimeter do not count. Note that, for a chessboard grid type, the number of rows and the number of columns must be at least 3. The following is an example of a valid 4x4 chessboard grid.  (summarize)
<input type="checkbox"/> M_CIRCLE_GRID +	Specifies a grid of circles. To create an accurate (subpixel) mapping, the image of the grid should meet the following guidelines (at the working resolution): <ul style="list-style-type: none"> • The radius of the grid's circles should range between 6 and 10 pixels. • The center-to-center distance between the grid's circles should range from 18 to 32 pixels (22 pixels recommended). • The minimum distance between the edges of the circles should be 6 pixels. • The grid should be large enough to cover the area of the image from which you want real-world results. • The grid must be on a single plane although it can be at a slant. • The image of the grid should have high contrast.

Combination constants for [M_CIRCLE_GRID](#); [M_CHESSBOARD_GRID](#);

You can add one of the following values to the above-mentioned values to set the orientation of the Y-axis.

● For specifying the orientation of the Y-axis	
▢ Value	Description
▢ M_Y_AXIS_DOWN	Orients the positive Y-axis along the first column of circles in a grid of circles, or along square corners in a chessboard grid. The axis is oriented from the top-left to the bottom-left. When no offset is specified, the origin is located at the center of the top-left circle. This is the default value. (summarize)
▢ M_Y_AXIS_UP	Orients the positive Y-axis along the first column of circles in a grid of circles, or along square corners in a chessboard grid. The axis is oriented from the bottom-left to the top-left. When no offset is specified, the origin is located in the center of the top-left circle. (summarize)

Combination constant for [M_CIRCLE_GRID](#); [M_CHESSBOARD_GRID](#);

You can add the following value to the above-mentioned values to set the speed of calibration.

● For increasing the speed of calibration	
▢ Value	Description
▢ M_FAST	Speeds up the time it takes to perform the calibration by using a faster calibration algorithm. Note that using a faster calibration algorithm can decrease the robustness and accuracy of the calibration. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalInquire

Synopsis

Inquire about a calibration object or calibrated image setting, or about the calibration state of an image or digitizer.

Syntax

```
MIL_INT McalInquire(
    MIL_ID CalibrationOrMilId,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires about a setting of a calibration object or a calibration image. It can also be used to determine if a calibration object is associated with an image or digitizer and whether or not an image has been corrected.

When working in [M_3D_ROBOTICS](#) calibration mode, the function returns information about the last calibration performed on that object. To inquire about previous calibration poses within the [M_3D_ROBOTICS](#) calibration object, you can use [McalInquireSingle\(\)](#) with the [Index](#) parameter set to the required calibration pose.

Parameters

CalibrationOrMilId

Specifies the identifier of the calibration object, image buffer, or digitizer.

InquireType

Specifies the setting about which to inquire. The setting for [InquireType](#) depends on whether you are inquiring about a calibration object, image, or digitizer.

For a calibration object, [InquireType](#) can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For a calibration object	
☐ Value	Description
☐ M_CALIBRATION_MODE +	Returns the calibration mode. (summarize) <i>UserVarPtr info</i> Return values: M_3D_ROBOTICS; M_LINEAR_INTERPOLATION; M_PERSPECTIVE_TRANSFORMATION; M_TSAI_BASED; (details)
☐ M_CALIBRATION_PLANE +	Returns the plane in which the calibration points are defined. (summarize) <i>UserVarPtr info</i> Return values: M_ABSOLUTE_COORDINATE_SYSTEM; M_DEFAULT; M_RELATIVE_COORDINATE_SYSTEM; (details)
☐ M_LINK_TOOL_AND_CAMERA +	Returns whether a rigid link exists between the camera coordinate system and the tool coordinate system. (summarize) <i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
☐ M_OUTPUT_UNITS +	Returns whether results are measured in pixel or world units. (summarize)

	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_PIXEL; M_WORLD; (details)
<input type="checkbox"/> M_RELATIVE_ORIGIN_ANGLE +	Returns the counter-clockwise angle of rotation (about the Z-axis) of the relative coordinate system, as measured in the absolute coordinate system. For 3D calibrations (M_TSAI_BASED or M_3D_ROBOTICS) the axis of rotation is not necessarily the Z-axis. The returned value is equivalent to Param4 of M_ROTATION_AXIS_ANGLE in McalGetCoordinateSystem() . (summarize)
<input type="checkbox"/> M_RELATIVE_ORIGIN_X +	Returns the X-coordinate of the origin of the relative coordinate system.
<input type="checkbox"/> M_RELATIVE_ORIGIN_Y +	Returns the Y-coordinate of the origin of the relative coordinate system.
<input type="checkbox"/> M_RELATIVE_ORIGIN_Z +	Returns the Z-coordinate of the origin of the relative coordinate system.
<input type="checkbox"/> M_TOOL_POSITION_X +	Returns the X-position of the origin of the tool coordinate system. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0; Value; (details)
<input type="checkbox"/> M_TOOL_POSITION_Y +	Returns the Y-position of the origin of the tool coordinate system. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0; Value; (details)
<input type="checkbox"/> M_TOOL_POSITION_Z +	Returns the Z-position of the origin of the tool coordinate system. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_TRANSFORM_CACHE +	Returns whether a cache is used to accelerate McalTransformImage() . (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_TRANSFORM_IMAGE_FILL_MODE +	Returns the manner in which the source image used with McalTransformImage() will be scaled and positioned when filling the destination image. (summarize)
	<i>UserVarPtr info</i> Return values: M_CLIP; M_DEFAULT; M_FIT; M_USER_DEFINED; (details)
<input type="checkbox"/> M_TRANSFORM_IMAGE_PIXEL_SIZE_X +	Returns the real-world width of the pixels in the destination image of McalTransformImage() . The width is specified in real-world units and is only used when M_TRANSFORM_IMAGE_FILL_MODE is set to M_USER_DEFINED . (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)
<input type="checkbox"/> M_TRANSFORM_IMAGE_PIXEL_SIZE_Y +	Returns the real-world height of the pixels in the destination image of McalTransformImage() . The height is specified in real-world units and is only used when M_TRANSFORM_IMAGE_FILL_MODE is set to M_USER_DEFINED . (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)
<input type="checkbox"/> M_TRANSFORM_IMAGE_WORLD_POS_X +	Returns the real-world X-offset of the top-left pixel in the destination image of McalTransformImage() . The offset is specified in real-world units and is only used when M_TRANSFORM_IMAGE_FILL_MODE is set to M_USER_DEFINED . (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_TRANSFORM_IMAGE_WORLD_POS_Y +	Returns the real-world Y-offset of the top-left pixel in the destination image of McalTransformImage() . The offset is specified in real-world units and is only used when M_TRANSFORM_IMAGE_FILL_MODE is set to M_USER_DEFINED . (summarize)

	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
--	----------------------------------------------------------------------------------------

The [InquireType](#) parameter can be set to one of the following values only after calibration with [McalGrid\(\)](#):

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a calibration object used with McalGrid()	
☐ Value	Description
☐ M_COLUMN_NUMBER +	Returns the number of columns in the calibration grid.
☐ M_COLUMN_SPACING +	Returns the number of world units between columns.
☐ M_FOREGROUND_VALUE +	Returns whether the grid's circles are lighter or darker than the background. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_FOREGROUND_BLACK; M_FOREGROUND_WHITE; (details)
☐ M_GRID_CORNER_HINT_X +	Returns the approximate image X-coordinate of the circle that is actually at the top-left corner of the calibration grid in the real-world. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_NONE; Value; (details)
☐ M_GRID_CORNER_HINT_Y +	Returns the approximate image Y-coordinate of the circle that is actually at the top-left corner of the calibration grid in the real-world. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_NONE; Value; (details)
☐ M_GRID_ORIGIN_X +	Returns the X-position of the top-left circle of the calibration grid, in the calibration plane coordinate system.
☐ M_GRID_ORIGIN_Y +	Specifies the Y-position of the top-left circle of the calibration grid, in the calibration plane coordinate system.
☐ M_GRID_ORIGIN_Z +	Specifies the Z-position of the top-left circle of the calibration grid, in the calibration plane coordinate system.
☐ M_GRID_TYPE +	Returns the type of grid used to perform the calibration. (summarize)
	<i>UserVarPtr info</i> Return values: M_CHESSBOARD_GRID; M_CIRCLE_GRID; (details)
☐ M_ROW_NUMBER +	Returns the number of rows in the calibration grid.
☐ M_ROW_SPACING +	Returns the number of world units between rows.
☐ M_Y_AXIS_UP +	Returns whether the positive Y-axis is oriented from the bottom-left circle to the top-left circle, along the first column. Please see GridType parameter of McalGrid() . (summarize)
	<i>UserVarPtr info</i> Return values: <div> <div>M_FALSE</div> <div>The positive Y-axis is oriented from the top-left circle to the bottom-left circle, along the first column.</div> </div> <div> <div>M_TRUE</div> <div>The positive Y-axis is oriented from the bottom-left circle to the top-left circle, along the first column.</div> </div>

The [InquireType](#) parameter can be set to one of the following values after a call to either [McalGrid\(\)](#) or [McalList\(\)](#):

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a calibrated calibration object used with McalGrid() or McalList()	
☐ Value	Description

M_ASPECT_RATIO +	<p>Returns the average aspect ratio.</p> <p>If this value is inquired before successful calibration, or if the camera is positioned and oriented in such a way that renders all points in the image plane invalid, the return value is M_INVALID_SCALE.</p> <p>(summarize)</p>												
M_AVERAGE_PIXEL_ERROR +	<p>Returns the average calibration error, in pixels. You can only inquire this value for a successfully calibrated calibration object.</p> <p>When working in M_3D_ROBOTICS calibration mode, this only returns the average calibration error of the last calibration call. To return an average calibration error for all calibration grids used, it is recommended to use M_GLOBAL_AVERAGE_PIXEL_ERROR instead.</p> <p>(summarize)</p>												
M_AVERAGE_WORLD_ERROR +	<p>Returns the average calibration error, in world units. You can only inquire this value for a successfully calibrated calibration object.</p> <p>When working in M_3D_ROBOTICS calibration mode, this only returns the average calibration error of the last calibration call. To return an average calibration error for all calibration grids used, it is recommended to use M_GLOBAL_AVERAGE_WORLD_ERROR instead.</p> <p>(summarize)</p>												
M_CALIBRATION_IMAGE_POINTS_X +	<p>Returns the X-coordinate of the calibration points in pixel units.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS. </div>												
M_CALIBRATION_IMAGE_POINTS_Y +	<p>Returns the Y-coordinate of the calibration points in pixel units.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS. </div>												
M_CALIBRATION_STATUS +	<p>Returns the status of a calibration.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: <table> <tr> <td>M_CALIBRATED</td><td>The calibration was successful.</td></tr> <tr> <td>M_GRID_NOT_FOUND</td><td>McalGrid() was unable to find an appropriate calibration grid in the provided image.</td></tr> <tr> <td>M_INVALID_CALIBRATION_POINTS</td><td>The provided calibration points do not contain sufficient spatial information to perform a calibration.</td></tr> <tr> <td>M_MATHEMATICAL_EXCEPTION</td><td>The calculation of the camera's parameters has failed.</td></tr> <tr> <td>M_NOT_INITIALIZED</td><td>No calibration has been performed yet.</td></tr> <tr> <td>M_PLANE_ANGLE_TOO_SMALL</td><td>The camera's optical axis is not sufficiently inclined. For full M_TSAI_BASED calibration, the camera's optical axis should be placed at least at a 30-degrees angle away from the axis perpendicular to the calibration plane.</td></tr> </table> </div>	M_CALIBRATED	The calibration was successful.	M_GRID_NOT_FOUND	McalGrid() was unable to find an appropriate calibration grid in the provided image.	M_INVALID_CALIBRATION_POINTS	The provided calibration points do not contain sufficient spatial information to perform a calibration.	M_MATHEMATICAL_EXCEPTION	The calculation of the camera's parameters has failed.	M_NOT_INITIALIZED	No calibration has been performed yet.	M_PLANE_ANGLE_TOO_SMALL	The camera's optical axis is not sufficiently inclined. For full M_TSAI_BASED calibration, the camera's optical axis should be placed at least at a 30-degrees angle away from the axis perpendicular to the calibration plane.
M_CALIBRATED	The calibration was successful.												
M_GRID_NOT_FOUND	McalGrid() was unable to find an appropriate calibration grid in the provided image.												
M_INVALID_CALIBRATION_POINTS	The provided calibration points do not contain sufficient spatial information to perform a calibration.												
M_MATHEMATICAL_EXCEPTION	The calculation of the camera's parameters has failed.												
M_NOT_INITIALIZED	No calibration has been performed yet.												
M_PLANE_ANGLE_TOO_SMALL	The camera's optical axis is not sufficiently inclined. For full M_TSAI_BASED calibration, the camera's optical axis should be placed at least at a 30-degrees angle away from the axis perpendicular to the calibration plane.												
M_CALIBRATION_WORLD_POINTS_X +	<p>Returns the X-coordinate of the calibration points in real-world units of the calibration plane (M_CALIBRATION_PLANE).</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS. </div>												
M_CALIBRATION_WORLD_POINTS_Y +	<p>Returns the Y-coordinate of the calibration points in real-world units of the calibration plane (M_CALIBRATION_PLANE).</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS. </div>												
M_CALIBRATION_WORLD_POINTS_Z +	<p>Returns the Z-coordinate of the calibration points in real-world units of the calibration plane (M_CALIBRATION_PLANE). If this value is inquired for 2D calibration mode objects, the array returned will be filled with 0.0 values.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS. </div>												

M_GLOBAL_AVERAGE_PIXEL_ERROR +	Returns the average calibration error, in pixels, for all the points used in all successive calls to McalGrid() and McalList() . You can only inquire this value for a successfully calibrated M_3D_ROBOTICS calibration object. (summarize)
M_GLOBAL_AVERAGE_WORLD_ERROR +	Returns the average calibration error, in world units, for all the points used in all successive calls to McalGrid() and McalList() . You can only inquire this value for a successfully calibrated M_3D_ROBOTICS calibration object. (summarize)
M_GLOBAL_MAXIMUM_PIXEL_ERROR +	Returns the maximum calibration error, in pixel units, for all the points used in all successive calls to McalGrid() and McalList() . You can only inquire this value for a successfully calibrated M_3D_ROBOTICS calibration object. (summarize)
M_GLOBAL_MAXIMUM_WORLD_ERROR +	Returns the maximum calibration error, in world units, for all the points used in all successive calls to McalGrid() and McalList() . You can only inquire this value for a successfully calibrated M_3D_ROBOTICS calibration object. (summarize)
M_MAXIMUM_PIXEL_ERROR +	Returns the maximum calibration error, in pixels. You can only inquire this value for a successfully calibrated calibration object. For an M_3D_ROBOTICS calibration objects, this only returns the average calibration error of the last calibration call. To return an average calibration error for all calibration grids used, it is recommended to use M_GLOBAL_MAXIMUM_PIXEL_ERROR instead. (summarize)
M_MAXIMUM_WORLD_ERROR +	Returns the maximum calibration error, in world units. You can only inquire this value for a successfully calibrated calibration object. For an M_3D_ROBOTICS calibration objects, this only returns the average calibration error of the last calibration call. To return an average calibration error for all calibration grids used, it is recommended to use M_GLOBAL_MAXIMUM_WORLD_ERROR instead. (summarize)
M_NUMBER_OF_CALIBRATION_POINTS +	Returns the number of calibration points found by McalGrid() or passed to McalList() .
M_NUMBER_OF_CALIBRATION_POSES +	Returns the number of calls made to McalGrid() or McalList() with the same CalibrationOrMild parameter passed. You can only inquire this value for a successfully calibrated M_3D_ROBOTICS calibration object. (summarize)

The [InquireType](#) parameter can be set to one of the following values only for an [M_TSAI_BASED](#) calibration object:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For M_TSAI_BASED calibration object	
Value	Description
M_CCD_ASPECT_RATIO +	Returns the width to height ratio of the individual elements of the CCD. (summarize)
	<i>UserVarPtr info</i> Return values: 1.0; Value > 0; (details)
M_DISTORTION_RADIAL_1 +	Returns the value of the first radial distortion parameter used in the calibration algorithm.
M_FOCAL_LENGTH +	Returns the effective focal length of the pinhole camera model used in the calibration, expressed in horizontal pixels.
M_PRINCIPAL_POINT_X +	Returns the X-coordinate of the intersection of the camera's optical axis and the image plane. (summarize)
	<i>UserVarPtr info</i> Return values: Please see McalControl() with M_PRINCIPAL_POINT_X . (details) M_DEFAULT This value is returned if the user has not set the corresponding control type using McalControl() .
M_PRINCIPAL_POINT_Y +	Returns the Y-coordinate of the intersection of the camera's optical axis and the image plane. (summarize)
	<i>UserVarPtr info</i> Return values: Please see McalControl() with M_PRINCIPAL_POINT_Y . (details) M_DEFAULT This value is returned if the user has not set the corresponding control type using McalControl() .

Combination constant for the values listed in [For M_TSAI_BASED calibration object](#); and for the following values: [M_CALIBRATION_PLANE](#); [M_LINK_TOOL_AND_CAMERA](#); [M_OUTPUT_UNITS](#); [M_RELATIVE_ORIGIN_ANGLE](#); [M_RELATIVE_ORIGIN_X](#); [M_RELATIVE_ORIGIN_Y](#); [M_RELATIVE_ORIGIN_Z](#); [M_TOOL_POSITION_X](#); [M_TOOL_POSITION_Y](#); [M_TOOL_POSITION_Z](#); [M_TRANSFORM_CACHE](#); [M_COLUMN_NUMBER](#); [M_COLUMN_SPACING](#); [M_FOREGROUND_VALUE](#); [M_GRID_CORNER_HINT_X](#); [M_GRID_CORNER_HINT_Y](#); [M_GRID_ORIGIN_X](#); [M_GRID_ORIGIN_Y](#); [M_GRID_ORIGIN_Z](#); [M_GRID_TYPE](#); [M_ROW_NUMBER](#); [M_ROW_SPACING](#);

You can add the following value to the above-mentioned values to determine the default value of an inquire type.

● For inquiring the default value	
☐ Value	Description
☐ M_DEFAULT	Returns the default value of the specified inquire type. (summarize)
	<div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div>

For any image or digitizer, the [InquireType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For an image or digitizer	
☐ Value	Description
☐ M_ASSOCIATED_CALIBRATION +	Returns the identifier of its associated calibration object or M_NULL if it is not calibrated.
☐ M_CORRECTION_STATE +	Returns whether the image has been physically corrected using McalTransformImage() . For a digitizer, it returns M_FALSE . (summarize)
	<div>UserVarPtr info</div> <div>Return values:</div> <div> <div>M_FALSE</div> <div>Image has not been corrected or not calibrated.</div> </div> <div> <div>M_TRUE</div> <div>Image has been corrected.</div> </div>

If an image or digitizer is calibrated, the [InquireType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a calibrated image	
☐ Value	Description
☐ M_CALIBRATION_CHILD_OFFSET_X +	Returns the X-offset of a child buffer relative to the highest calibrated parent image that was originally associated with the calibration object. (summarize)
	<div>UserVarPtr info</div> <div>Return values: Value; (details)</div>
☐ M_CALIBRATION_CHILD_OFFSET_Y +	Returns the Y-offset of a child buffer relative to the highest calibrated parent image that was originally associated with the calibration object. (summarize)
	<div>UserVarPtr info</div> <div>Return values: Value; (details)</div>

If an image is calibrated and corrected, the [InquireType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a calibrated and corrected image	
▢ Value	Description
▢ M_CORRECTED_PIXEL_SIZE_X +	Returns the real-world width of the pixels in the corrected image.
▢ M_CORRECTED_PIXEL_SIZE_Y +	Returns the real-world height of the pixels in the corrected image.
▢ M_CORRECTED_WORLD_POS_X +	Returns the real-world X-offset of the top-left pixel in the corrected image.
▢ M_CORRECTED_WORLD_POS_Y +	Returns the real-world Y-offset of the top-left pixel in the corrected image.

Combination constants for any of the possible values of the [InquireType](#) parameter

You can add one of the following values to the above-mentioned values to cast the requested information to the required data type.

● For specifying the data type	
▢ Value	Description
▢ M_TYPE_CHAR	Casts the requested information to a <i>char</i> . (summarize)
	<i>UserVarPtr info</i> Data type: char
▢ M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . This is the default. (summarize)
	<i>UserVarPtr info</i> Data type: double
▢ M_TYPE_FLOAT	Casts the requested information to a <i>float</i> . (summarize)
	<i>UserVarPtr info</i> Data type: float
▢ M_TYPE_LONG	Casts the requested information to a <i>long</i> . (summarize)
	<i>UserVarPtr info</i> Data type: long
▢ M_TYPE_MIL_DOUBLE	Casts the requested information to a <i>MIL_DOUBLE</i> . (summarize)
	<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type <i>MIL_DOUBLE</i> Array size: Depends on the result type. Note: When multiple results. • Data type: <i>MIL_DOUBLE</i> Note: When a single result.
▢ M_TYPE_MIL_ID	Casts the requested information to a <i>MIL_ID</i> . (summarize)
	<i>UserVarPtr info</i> Data type: <i>MIL_ID</i>
▢ M_TYPE_MIL_INT	Casts the requested information to a <i>MIL_INT</i> . (summarize)
	<i>UserVarPtr info</i> Data type: <i>MIL_INT</i>
▢ M_TYPE_MIL_INT32	Casts the requested information to a <i>MIL_INT32</i> .

	(summarize)
	<i>UserVarPtr info</i> Data type: MIL_INT32
<input type="checkbox"/> M_TYPE_MIL_INT64	Casts the requested information to a <i>MIL_INT64</i> . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_INT64
<input type="checkbox"/> M_TYPE_SHORT	Casts the requested information to a <i>short</i> . (summarize)
	<i>UserVarPtr info</i> Data type: short

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• array of type MIL_DOUBLE• char• double• float• long• MIL_DOUBLE• MIL_ID• MIL_INT• MIL_INT32• MIL_INT64• short

Specifies the address in which to return the value of the inquired setting.

Since `McallInquire()` also returns the value of the inquired setting, you can set `UserVarPtr` to `M_NULL`.

Return value

The returned value is the value of the inquired setting, cast to `MIL_INT`.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalInquireSingle

Synopsis

Inquire about a single pose in a sequence of poses taken during calibration.

Syntax

```
MIL_INT McalInquireSingle(
    MIL_ID CalibrationOrMilId,
    MIL_ID Index,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function allows you to inquire about the settings of a single pose in the sequence of poses performed to calibrate [M_3D_ROBOTICS](#) calibration objects. Every call to [McalGrid\(\)](#) or [McalList\(\)](#) with [M_ACCUMULATE](#) corresponds to a pose used to obtain information about the camera setup during the calibration process.

Parameters

CalibrationOrMilId

Specifies the identifier of the calibration object or image.

Index

Specifies the index of the pose of a [M_3D_ROBOTICS](#) calibration object from which settings should be inquired. The first call corresponds to the index value 0 and the last one is the return value of [McalInquire\(\)](#) with [M_NUMBER_OF_CALIBRATION_POSES](#) minus one.

InquireType

Specifies the setting about which to inquire.

The [InquireType](#) parameter can be set to one of the following values only after calibrating with [McalGrid\(\)](#):

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For a calibration object used with McalGrid()	
☐ Value	Description
☐ M_COLUMN_NUMBER +	Returns the number of columns in the calibration grid.
☐ M_COLUMN_SPACING +	Returns the number of world units between columns.
☐ M_GRID_ORIGIN_X +	Returns the X-position of the top-left circle of the calibration grid, in the calibration plane coordinate system.
☐ M_GRID_ORIGIN_Y +	Returns the Y-position of the top-left circle of the calibration grid, in the calibration plane coordinate system.
☐ M_GRID_ORIGIN_Z +	Returns the Z-position of the top-left circle of the calibration grid, in the calibration plane coordinate system.
☐ M_GRID_TYPE +	Returns the type of grid used to perform the calibration. (summarize)
	<i>UserVarPtr info</i> Return values: M_CHESSBOARD_GRID ; M_CIRCLE_GRID ; (details)
☐ M_ROW_NUMBER +	Returns the number of rows in the calibration grid.

<input type="checkbox"/> M_ROW_SPACING +	Returns the number of world units between rows.
------------------------------------------	-------------------------------------------------

Combination constant for [M_COLUMN_NUMBER](#); [M_COLUMN_SPACING](#); [M_GRID_ORIGIN_X](#); [M_GRID_ORIGIN_Y](#); [M_GRID_ORIGIN_Z](#); [M_GRID_TYPE](#); [M_ROW_NUMBER](#); [M_ROW_SPACING](#);

You can add the following value to the above-mentioned values to determine the default value of an inquire type.

● For inquiring the default value	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Returns the default value of the specified inquire type. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE

The [InquireType](#) parameter can be set to one of the following values after a call to either [McalGrid\(\)](#) or [McalList\(\)](#):

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a calibrated calibration object used with McalGrid() or McalList()	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_AVERAGE_PIXEL_ERROR +	Returns the average calibration error, in pixels.
<input type="checkbox"/> M_AVERAGE_WORLD_ERROR +	Returns the average calibration error, in world units.
<input type="checkbox"/> M_CALIBRATION_IMAGE_POINTS_X +	Returns the X-coordinate of the calibration points in pixel units. (summarize)
	<i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS .
<input type="checkbox"/> M_CALIBRATION_IMAGE_POINTS_Y +	Returns the Y-coordinate of the calibration points in pixel units. (summarize)
	<i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS .
<input type="checkbox"/> M_CALIBRATION_WORLD_POINTS_X +	Returns the X-coordinate of the calibration points in real-world units of the calibration plane (M_CALIBRATION_PLANE). (summarize)
	<i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS .
<input type="checkbox"/> M_CALIBRATION_WORLD_POINTS_Y +	Returns the Y-coordinate of the calibration points in real-world units of the calibration plane (M_CALIBRATION_PLANE). (summarize)
	<i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS .
<input type="checkbox"/> M_CALIBRATION_WORLD_POINTS_Z +	Returns the Z-coordinate of the calibration points in real-world units of the calibration plane (M_CALIBRATION_PLANE). If this value is inquired for 2D calibration mode objects, the array returned will be filled with 0.0 values. (summarize)
	<i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The return value of McalInquire() with M_NUMBER_OF_CALIBRATION_POINTS .

M_MAXIMUM_PIXEL_ERROR +	Returns the maximum calibration error, in pixels.
M_MAXIMUM_WORLD_ERROR +	Returns the maximum calibration error, in world units.
M_NUMBER_OF_CALIBRATION_POINTS +	Returns the number of calibration points found by McalGrid() or passed to McalList() .

Combination constants for any of the possible values of the [InquireType](#) parameter

You can add one of the following values to the above-mentioned values to cast the requested information to the required data type.

● For specifying the data type	
Value	Description
M_TYPE_CHAR	Casts the requested information to a <i>char</i> . (summarize)
	<i>UserVarPtr info</i> Data type: char
M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . (summarize)
	<i>UserVarPtr info</i> Data type: double
M_TYPE_FLOAT	Casts the requested information to a <i>float</i> . (summarize)
	<i>UserVarPtr info</i> Data type: float
M_TYPE_LONG	Casts the requested information to a <i>long</i> . (summarize)
	<i>UserVarPtr info</i> Data type: long
M_TYPE_MIL_DOUBLE	Casts the requested information to a <i>MIL_DOUBLE</i> . This is the default. (summarize)
	<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type <i>MIL_DOUBLE</i> Array size: Depends on the result type. Note: When multiple results. • Data type: <i>MIL_DOUBLE</i> Note: When a single result.
M_TYPE_MIL_ID	Casts the requested information to a <i>MIL_ID</i> . (summarize)
	<i>UserVarPtr info</i> Data type: <i>MIL_ID</i>
M_TYPE_MIL_INT	Casts the requested information to a <i>MIL_INT</i> . (summarize)
	<i>UserVarPtr info</i> Data type: <i>MIL_INT</i>
M_TYPE_MIL_INT32	Casts the requested information to a <i>MIL_INT32</i> . (summarize)
	<i>UserVarPtr info</i> Data type: <i>MIL_INT32</i>
M_TYPE_MIL_INT64	Casts the requested information to a <i>MIL_INT64</i> .

	(summarize)
	<i>UserVarPtr info</i> Data type: MIL_INT64
☐ M_TYPE_SHORT	Casts the requested information to a <i>short</i> . (summarize)
	<i>UserVarPtr info</i> Data type: short

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• array of type MIL_DOUBLE• char• double• float• long• MIL_DOUBLE• MIL_ID• MIL_INT• MIL_INT32• MIL_INT64• short

Specifies the address in which to return the value of the inquired setting.

Since **McallInquireSingle()** also returns the value of the inquired setting, you can set [UserVarPtr](#) to **M_NULL**.

Return value

The returned value is the value of the inquired setting, cast to *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalList

Synopsis

Calibrate your camera setup using a list of coordinates.

Syntax

```
void McalList(
    MIL_ID CalibrationId,
    const MIL_DOUBLE *XPixArray,
    const MIL_DOUBLE *YPixArray,
    const MIL_DOUBLE *XWorldArray,
    const MIL_DOUBLE *YWorldArray,
    const MIL_DOUBLE *ZWorldArray,
    MIL_INT NumPoint,
    MIL_INT Operation,
    MIL_INT ControlFlag
)
```

Description

This function uses a specified list of calibration points to calibrate your camera setup. Calibration points are pixel coordinates and their associated real-world coordinates. The mapping is stored with the specified calibration object. The specified calibration object determines the calibration mode used.

Typically, you specify the list of calibration points in the absolute coordinate system. However, for 3D-based calibration modes ([M_TSAL_BASED](#) or [M_3D_ROBOTICS](#)), it is possible to use a list of calibration points that are not in the Z=0 plane of the absolute coordinate system. In this case, it is possible to describe the points in the relative coordinate system instead, using [McalControl\(\)](#) with [M_CALIBRATION_PLANE](#) set to [M_RELATIVE_COORDINATE_SYSTEM](#). You can then use [McalSetCoordinateSystem\(\)](#) to move the relative coordinate system. Both of these calls must be made before calling [McalList\(\)](#).

When working in [M_3D_ROBOTICS](#) calibration mode, you must perform at least three calls to [McalList\(\)](#) with [M_ACCUMULATE](#) before performing a full calibration. Before each call, you must change the position and orientation of the tool holding the camera with respect to the robot base. More specifically, you must rotate the tool along at least two non-parallel axes. You must also use [McalSetCoordinateSystem\(\)](#) to set the position of the tool coordinate system with respect to the robot base coordinate system before each of these calls. Using [McalList\(\)](#) with [M_ACCUMULATE](#) more than three times before performing a full calibration improves the accuracy of your calibration.

You can inquire about the pixel coordinates and associated real-world coordinates of each calibration point of one particular call using [McalInquireSingle\(\)](#).

For 3D-based calibration modes, [McalList\(\)](#) performs two types of calculations. It calculates the camera's internal attributes, and it calculates the orientation and distance between the camera and calibration plane. Initially, the latter is used to set the camera coordinate system. If you move the camera or the grid (not both), you can have [McalList\(\)](#) recalculate only the orientation and distance between them. In this case, depending on what you moved, you can specify that [McalList\(\)](#) displace either the camera coordinate system or the relative coordinate system, using [M_DISPLACE_CAMERA_COORD](#) or [M_DISPLACE_RELATIVE_COORD](#), respectively. When using [M_DISPLACE_RELATIVE_COORD](#), the calibration plane is considered to be the relative coordinate system regardless of the [M_CALIBRATION_PLANE](#) setting.

For 3D-based calibrations, it is also mandatory to set the image coordinates of the principal point prior to calibrating using [McalControl\(\)](#) with [M_PRINCIPAL_POINT_X](#), and [M_PRINCIPAL_POINT_Y](#). If the aspect ratio of the CCD element is different than 1, specify it using [McalControl\(\)](#) with [M_CCD_ASPECT_RATIO](#) before calling [McalList\(\)](#).

Also, for 3D-based calibration modes, a successful call to [McalList\(\)](#) with [M_FULL_CALIBRATION](#) creates a rigid link between the camera coordinate system and the tool coordinate system. This link ensures that moving either the tool or the camera coordinate systems will affect both. This link can be broken using [McalControl\(\)](#) with [M_LINK_TOOL_AND_CAMERA](#).

Note that the internal attributes calculated for the camera are not those of the physical camera, but those of the ideal pinhole-camera used to model the physical camera.

After calling [McalList\(\)](#), you can inquire about the success of the calibration using [McalInquire\(\)](#) with [M_CALIBRATION_STATUS](#). Also, you can inquire about the pixel coordinates and associated real-world coordinates of each calibration point using [McalInquire\(\)](#) with [M_CALIBRATION_IMAGE_POINTS_X](#), [M_CALIBRATION_IMAGE_POINTS_Y](#), [M_CALIBRATION_WORLD_POINTS_X](#) and [M_CALIBRATION_WORLD_POINTS_Y](#). These coordinates correspond to the center of circles in a circle grid or to points connecting four corners in a chessboard grid.

For more information on performing calibration, see the [Steps to performing a calibration](#) section in [Chapter 5: Camera calibration](#).

Parameters

CalibrationId

Specifies the identifier of the calibration object.

XPixArray

Specifies the address of the array containing the X-pixel coordinates.

When performing a [M_FULL_CALIBRATION](#) operation in [M_3D_ROBOTICS](#) calibration mode, set this parameter to **M_NULL**.

YPixArray

Specifies the address of the array containing the Y-pixel coordinates.

When performing a [M_FULL_CALIBRATION](#) operation in [M_3D_ROBOTICS](#) calibration mode, set this parameter to **M_NULL**.

XWorldArray

Specifies the address of the array containing the X-world coordinates in the calibration-plane coordinate system.

When performing a [M_FULL_CALIBRATION](#) operation in [M_3D_ROBOTICS](#) calibration mode, set this parameter to **M_NULL**.

YWorldArray

Specifies the address of the array containing the Y-world coordinates in the calibration-plane coordinate system.

When performing a [M_FULL_CALIBRATION](#) operation in [M_3D_ROBOTICS](#) calibration mode, set this parameter to **M_NULL**.

ZWorldArray

Specifies the address of the array containing the Z-world coordinates.

For 2D-based calibraion objects ([M_LINEAR_INTERPOLATION](#) or [M_PERSPECTIVE_TRANSFORMATION](#)), this parameter must be set to *M_NULL*.

For [M_TSAI_BASED](#) calibration, if the object you are imaging is planar (has all its points on a single plane), this parameter should be set to *M_NULL* for better performance. However, if the object is non-planar, it is possible to provide the Z coordinates. In that case, the minimum camera angle requirement is lifted.

For [M_3D_ROBOTICS](#) calibration objects, the parameter must be set to *M_NULL* if the system is not yet calibrated. However, you can provide Z-world coordinates when using **McalList()** with [M_DISPLACE_RELATIVE_COORD](#) or [M_DISPLACE_CAMERA_COORD](#).

NumPoint

Specifies the number of coordinates in the supplied arrays. For a piecewise linear interpolation calibration mode, the minimum number of coordinates is 3. For a perspective transformation calibration mode, the minimum number of coordinates is 4. For a 3D-based calibration modes, the minimum number of coordinates is 6.

Note, the specified pixel coordinates should cover the area of the image from which you want real-world coordinates (the working area). When performing a [M_FULL_CALIBRATION](#) operation in [M_3D_ROBOTICS](#) calibration mode, set this parameter to **M_NULL**.

Operation

Specifies the calibration operation to perform. This parameter must be set to one of the following values:

● For specifying the calibration operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_FULL_CALIBRATION .
<input type="checkbox"/> M_ACCUMULATE	Specifies to receive only the position of the calibration points and stores them in the calibration object. At least three calls to McalList() should be made with this operation before performing full calibration with M_FULL_CALIBRATION . This operation is only

	supported for M_3D_ROBOTICS calibration mode. (summarize)
☐ M_DISPLACE_CAMERA_COORD	Calculates only the position and orientation between the camera and the calibration plane, and displaces the camera coordinate system accordingly. This calibration operation is only supported for 3D-based calibration objects that are fully calibrated with M_FULL_CALIBRATION . If the camera is moved to a new position but its internal attributes are already known by a previous full calibration, you can use this operation to allow for a faster calibration. (summarize)
☐ M_DISPLACE_RELATIVE_COORD	Calculates only the relative coordinate system position in space. This calibration operation is only supported for 3D-based calibration objects that are fully calibrated with M_FULL_CALIBRATION . You can use this operation to move the relative coordinate system only and remain calibrated. This operation keeps the camera fixed with respect to the absolute coordinate system, and its intrinsic attributes unmodified. You can then obtain the position and orientation of the relative coordinate system with respect to any other coordinate system using McalGetCoordinateSystem() , and calculate the unknown position of an object in space. When using this operation, the calibration plane is considered to be the relative coordinate system regardless of the M_CALIBRATION_PLANE setting. (summarize)
☐ M_FULL_CALIBRATION	Performs a full calibration. For 3D-based calibration objects, this operation calculates the camera's internal attributes and the difference in position and orientation between the camera and the calibration plane. It then sets the camera coordinate system accordingly. When working in M_TSAI_BASED calibration mode, the camera's optical axis should be at least 30 degrees away from the axis perpendicular to the calibration plane (also known as, angle of incidence). Otherwise, the calibration might fail. (summarize)

ControlFlag

Specifies the function's operation flag. This parameter must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalRelativeOrigin

Synopsis

Change the origin and/or orientation of a relative coordinate system.

Syntax

```
void McalRelativeOrigin(
    MIL_ID CalibrationId,
    MIL_DOUBLE XOffset,
    MIL_DOUBLE YOffset,
    MIL_DOUBLE ZOffset,
    MIL_DOUBLE AngularOffset,
    MIL_INT ControlFlag
)
```

Description

This function changes the origin and/or orientation of a calibration object's relative coordinate system. This function can only be called after a successful calibration using either [McalGrid\(\)](#) or [McalList\(\)](#).

If you move the relative coordinate system, results returned in world-units from other modules are returned relative to the relative coordinate system's new position.

Note that setting the [XOffset](#), [YOffset](#), [ZOffset](#), and [AngularOffset](#) parameters to 0 has the effect of resetting the relative coordinate system to that of the absolute coordinate system.

To manipulate the relative coordinate system in 3D-based calibration modes ([M_TSAI_BASED](#) or [M_3D_ROBOTICS](#)), it is recommended to use [McalSetCoordinateSystem\(\)](#) over [McalRelativeOrigin\(\)](#) since the former supports more transformation types.

Parameters

CalibrationId

Specifies the identifier of the calibration object.

XOffset

Specifies the X-offset from the absolute coordinate system origin to the relative coordinate system origin in world-units.

YOffset

Specifies the Y-offset from the absolute coordinate system origin to the relative coordinate system origin in world-units.

ZOffset

Specifies the Z-offset from the absolute coordinate system origin to the relative coordinate system origin in world-units.

When using the linear interpolation and perspective transformation calibration modes, this parameter must be set to **M_NULL**.

AngularOffset

Specifies the angle, in degrees, at which to orient the relative coordinate system. This angle is applied in the counter-clockwise direction relative to the X-axis of the absolute coordinate system.

ControlFlag

Specifies the function's control flag. This parameter must be set to the following value:

• For specifying the function's control flag

<div><input type="checkbox"/> Value</div>	Description
<div><input type="checkbox"/> M_DEFAULT</div>	Sets the function's control flag to the default.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalRestore

Synopsis

Restore a calibration object from a file.

Syntax

```
MIL_ID McalRestore(  
    MIL_CONST_TEXT_PTR Filename,  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *CalibrationIdPtr  
)
```

Description

This function restores a calibration object that was previously saved to a file, using [McalSave\(\)](#) or [McalStream\(\)](#), and assigns it an identifier.

Parameters

Filename

Specifies the name and path of the file from which to restore the calibration object. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

● For specifying the file name and path		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile". (summarize)	
		<i>Parameters</i>
		<i>FileName</i> Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.	

SystemId

Specifies the system on which to restore the calibration object. This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Specifies the function's control flag. This parameter must be set to the following value:

● For specifying the function's control flag	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Sets the function's control flag to the default.

CalibrationIdPtr

Specifies the address in which to return the identifier of the calibration object. Since **McalRestore()** also returns the identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the identifier of the calibration object if the operation was successful; **M_NULL** if the operation failed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalSave

Synopsis

Save a calibration object to a file.

Syntax

```
void McalSave(  
    MIL_CONST_TEXT_PTR FileName,  
    MIL_ID CalibrationId,  
    MIL_INT ControlFlag  
)
```

Description

This function saves a calibration object to a file. This information can be reloaded, using [McalRestore\(\)](#) or [McalStream\(\)](#).

Parameters

FileName

Specifies the name and path of the file in which to save the calibration object. It is recommended that you use the MCA file extension for easier use with other Matrox Imaging software products. The function internally handles the opening and closing of this file. If this file already exists, it will be overwritten.

This parameter can be set to one of the following values:

● For specifying the file name and path					
<input type="checkbox"/> Value	Description				
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p>				
	<table><tr><th colspan="2">Parameters</th></tr><tr><td><i>FileName</i></td><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters		<i>FileName</i>	Specifies the drive, directory, and name of the file.
Parameters					
<i>FileName</i>	Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</p>				

CalibrationId

Specifies the calibration object to save.

ControlFlag

Specifies the function's control flag. This parameter must be set to the following value:

● For specifying the function's control flag	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Sets the function's control flag to the default.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalSetCoordinateSystem

Synopsis

Change the position and orientation of a coordinate system.

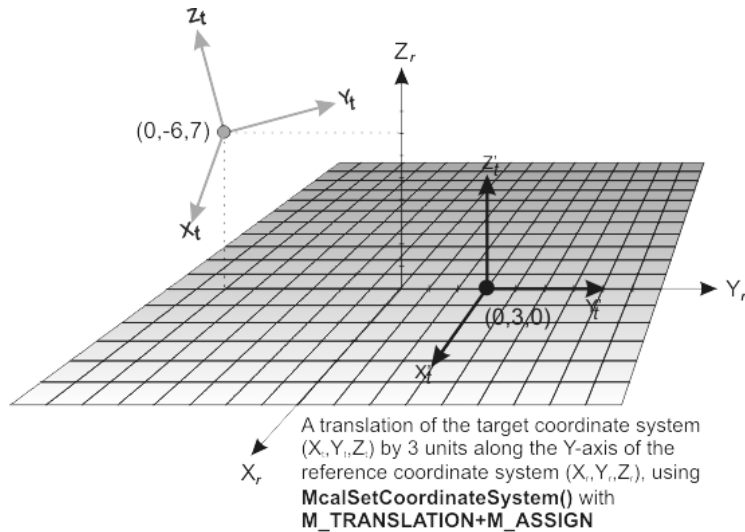
Syntax

```
void McalSetCoordinateSystem(
    MIL_ID CalibrationId,
    MIL_INT TargetCoordinateSystem,
    MIL_INT ReferenceCoordinateSystem,
    MIL_INT TransformType,
    MIL_ID MatrixId,
    MIL_DOUBLE Param1,
    MIL_DOUBLE Param2,
    MIL_DOUBLE Param3,
    MIL_DOUBLE Param4
)
```

Description

This function moves a specified (target) coordinate system, relative to a reference coordinate system. For example, you can move the relative coordinate system by $(0\vec{x}, 6\vec{y}, 0\vec{z})$ in the absolute coordinate system. This will displace the origin of the relative coordinate system by 6 units along the Y-axis of the absolute coordinate system.

You can apply the transformation of the target coordinate system in a reference coordinate system as a transformation from the current position (**M_COMPOSE_WITH_CURRENT**) of the target coordinate system or from the origin of the reference coordinate system (**M_ASSIGN**).



If you move the relative coordinate system, results returned in world units from other modules are returned relative to the reference coordinate system's new position.

If you transform the camera or tool coordinate system, MIL assumes that the camera have been moved in your camera setup, so it adjusts the calibration mapping between the world coordinate system and the image coordinate system.

Note that you cannot move the image coordinate system nor the absolute coordinate system.

This function can only be used with 3D-based calibration objects ([M_TSAI_BASED](#) or [M_3D_ROBOTICS](#)).

Parameters

CalibrationId

Specifies the identifier of the calibration object.

TargetCoordinateSystem

Specifies the coordinate system on which to apply the transformation. This parameter can be set to one of the following.

By default, a rigid link exists between the tool and camera coordinate systems such that moving one automatically moves the other accordingly. However, this can be broken using [McalControl\(\)](#) with [M_LINK_TOOL_AND_CAMERA](#) set to **M_DISABLE**. So that you can set the tool coordinate system to a known location without affecting the camera coordinate system (and the pixel-to-world mapping). After moving the tool coordinate system, you can then reestablish the link so that you move the camera coordinate system by moving the tool coordinate system.

● For specifying the target coordinate system	
☐ Value	Description
☐ M_CAMERA_COORDINATE_SYSTEM	Specifies to apply the transformation to the camera coordinate system. The origin of the camera coordinate system corresponds to the effective pinhole of the modeled camera and the Z-axis of the camera coordinate system points in the direction that the camera is facing. This coordinate system is only defined for 3D-based calibration modes, after a successful calibration or after using McalSetCoordinateSystem() with M_ASSIGN . (summarize)
☐ M_RELATIVE_COORDINATE_SYSTEM	Specifies to apply the transformation to the relative coordinate system. The relative coordinate system defines the world plane in which results are measured. By default, it corresponds to the absolute coordinate system. (summarize)
☐ M_ROBOT_BASE_COORDINATE_SYSTEM	Specifies to apply the transformation to the robot base coordinate system. This coordinate system is defined as the origin of the robot encoders. If the encoders of the robot all indicate 0, it means that the tool coordinate system is at the same position and orientation as the robot base coordinate system. This is only available for M_3D_ROBOTICS calibration mode. The position and orientation of this coordinate system are only found in M_3D_ROBOTICS mode, after a successful calibration or after using McalSetCoordinateSystem() with M_ASSIGN . The position and orientation of this coordinate system are only changed when resetting the encoders on a mobile robot. (summarize)
☐ M_TOOL_COORDINATE_SYSTEM	Specifies to apply the transformation to the tool coordinate system. The tool coordinate system is used to position the camera. Though this coordinate system can be used to move the camera it needs not be associated to the real camera position. By default, its axes are parallel to the absolute coordinate system, and its origin is the same as that of the absolute coordinate system. (summarize)

ReferenceCoordinateSystem

Specifies the reference coordinate system. The reference coordinate system must be defined before calling this function. This parameter can be set to one of the following values:

● For specifying the reference coordinate system	
☐ Value	Description
☐ M_DEFAULT	Same as M_ABSOLUTE_COORDINATE_SYSTEM .
☐ M_ABSOLUTE_COORDINATE_SYSTEM	Specifies to use the absolute coordinate system as a reference coordinate system for the transformation. The absolute coordinate system is unmovable and is the coordinate system from which all others are defined. By default, it corresponds to the center of the top-left calibration point when using McalGrid() . (summarize)
☐ M_CAMERA_COORDINATE_SYSTEM	Specifies to use the camera coordinate system as a reference coordinate system for the transformation. The camera coordinate system's origin corresponds to the effective pinhole of the modeled camera and its Z-axis points in the direction that the camera is facing. This coordinate system can only be used as a reference coordinate system for 3D-based calibration mode, after a successful calibration or after using McalSetCoordinateSystem() with M_ASSIGN . (summarize)
☐ M_RELATIVE_COORDINATE_SYSTEM	Specifies to use the relative coordinate system as a reference coordinate system for the transformation. The relative coordinate system defines the world plane in which results are returned. By default, it corresponds to the absolute coordinate system. The relative coordinate system can be recentered and/or reoriented using either McalSetCoordinateSystem() or McalRelativeOrigin() .

	(summarize)
<input type="checkbox"/> M_ROBOT_BASE_COORDINATE_SYSTEM	<p>Specifies to apply the transformation to the robot base coordinate system. This coordinate system is defined as the origin of the robot encoders. If the encoders of the robot all indicate 0, it means that the tool coordinate system is at the same position and orientation as the robot base coordinate system.</p> <p>This is only available for M_3D_ROBOTICS calibration mode. The position and orientation of this coordinate system are only found in M_3D_ROBOTICS mode, after a successful calibration or after using <code>McalSetCoordinateSystem()</code> with M_ASSIGN.</p> <p>The position and orientation of this coordinate system are only changed when resetting the encoders on a mobile robot.</p> <p>(summarize)</p>
<input type="checkbox"/> M_TOOL_COORDINATE_SYSTEM	<p>Specifies that the transformation will be defined in the tool coordinate system. The tool coordinate system is used to position the camera. By default, its axes are parallel to the absolute coordinate system, and its origin is the same as that of the absolute coordinate system.</p> <p>(summarize)</p>

TransformType

Specifies the type of transformation to apply to the target coordinate system.

See the [Parameter associations](#) section for possible values.

MatrixId

Specifies the identifier of a MIL array buffer which contains the transformation matrix to use. The expected transformation matrix is dependent on the type of transformation selected.

The MIL array buffer must be a 32-bit floating-point buffer, allocated using `MbufAlloc2d()` with **M_ARRAY**. The required buffer size is dependent on the type of transformation selected.

See the [Parameter associations](#) section for possible values.

Param1

Specifies an attribute of the transformation. Its definition is dependent on the type of transformation selected.

See the [Parameter associations](#) section for possible values.

Param2

Specifies an attribute of the transformation. Its definition is dependent on the type of transformation selected.

See the [Parameter associations](#) section for possible values.

Param3

Specifies an attribute of the transformation. Its definition is dependent on the type of transformation selected.

See the [Parameter associations](#) section for possible values.

Param4

Specifies an attribute of the transformation. Its definition is dependent on the type of transformation selected.

See the [Parameter associations](#) section for possible values.

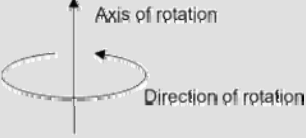
Parameter associations

Possible values for the **TransformType**, **MatrixId**, **Param1**, **Param2**, **Param3**, and **Param4** parameters are described in the table below.

- [For specifying the type of transformation](#)

Note that if **Param1**, **Param2**, **Param3**, or **Param4** are not required by the transformation, they should be set to **M_DEFAULT**.

• For specifying the type of transformation

TransformType		Description
	MatrixId	
	Param1	
	Param2	
	Param3	
	Param4	
M_HOMOGENEOUS_MATRIX +		Specifies to apply a translation, a rotation, or both to the target coordinate system. Specify the transformation using a 4x4 homogenous matrix. (summarize)
	MatrixId	Specifies the matrix representation of the transformation. (summarize)
M_IDENTITY +		Specifies to apply the identity transformation to the target coordinate system. When used with the M_ASSIGN combination constant, this operation transforms the target coordinate system so that it has the same position and orientation as the reference coordinate system. (summarize)
	MatrixId	This parameter must be set to M_NULL . (summarize)
M_ROTATION_AXIS_ANGLE +		<p>Specifies to apply a rotation operation described by an axis and angle of rotation. The axis of rotation is defined by the origin of the reference coordinate system and one other point. The angle of rotation is measured in the counter-clockwise direction around the axis of rotation.</p>  <p>(summarize)</p>
	MatrixId	This parameter must be set to M_NULL . (summarize)
	Param1	Specifies the X-coordinate of the point which defines the axis of rotation. (summarize)
	Param2	Specifies the Y-coordinate of the point which defines the axis of rotation. (summarize)
	Param3	Specifies the Z-coordinate of the point which defines the axis of rotation. (summarize)
	Param4	Specifies the angle of the rotation, in degrees, around the axis of rotation. (summarize)
M_ROTATION_MATRIX +		Specifies to apply a rotation operation that is described by a 3x3 rotation matrix. (summarize)
	MatrixId	Specifies the matrix representation of the rotation. (summarize)
M_ROTATION_QUATERNION +		Specifies to apply a rotation operation that is described by a rotation quaternion. (summarize)
	MatrixId	This parameter must be set to M_NULL . (summarize)
	Param1	Specifies the scalar component of the quaternion. (summarize)
	Param2	Specifies the X-component of the quaternion. (summarize)
	Param3	Specifies the Y-component of the quaternion. (summarize)

	<input type="checkbox"/> Param4	Specifies the Z-component of the quaternion. (summarize)
<input type="checkbox"/> M_ROTATION_X +		Specifies to apply a rotation operation that is described by a rotation around the X-axis. (summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the X-axis rotation, in degrees. (summarize)
<input type="checkbox"/> M_ROTATION_XYZ +		Specifies to apply a rotation operation that is described by three distinct rotations about the axes of the reference coordinate system in the following order: a rotation about the X-axis, a rotation about the Y-axis, and a rotation about Z-axis rotation. (summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the X-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param2	Specifies the Y-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param3	Specifies the Z-axis rotation, in degrees. (summarize)
<input type="checkbox"/> M_ROTATION_XZY +		Specifies to apply a rotation operation that is described by three distinct rotations about the axes of the reference coordinate system in the following order: a rotation about the X-axis, a rotation about the Z-axis, and a rotation about the Y-axis rotation. (summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the X-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param2	Specifies the Z-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param3	Specifies the Y-axis rotation, in degrees. (summarize)
<input type="checkbox"/> M_ROTATION_Y +		Specifies to apply a rotation operation that is described by a rotation about the Y-axis. (summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the Y-axis rotation, in degrees. (summarize)
<input type="checkbox"/> M_ROTATION_YXZ +		Specifies to apply a rotation operation that is described by three distinct rotations about the axes of the reference coordinate system in the following order: a rotation about the Y-axis, a rotation about the X-axis, and a rotation about Z-axis rotation. This is also known as roll-pitch-yaw rotation. (summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the Y-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param2	Specifies the X-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param3	Specifies the Z-axis rotation, in degrees. (summarize)
<input type="checkbox"/> M_ROTATION_YZX +		Specifies to apply a rotation operation that is described by three distinct rotations about the axes of the reference coordinate system in the following order: a rotation about the Y-axis, a rotation about the Z-axis, and a rotation about the X-axis rotation.

		(summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the Y-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param2	Specifies the Z-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param3	Specifies the X-axis rotation, in degrees. (summarize)
<input type="checkbox"/> M_ROTATION_Z +		Specifies to apply a rotation operation that is described by a rotation about the Z-axis. (summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the Z-axis rotation, in degrees. (summarize)
<input type="checkbox"/> M_ROTATION_ZXY +		Specifies to apply a rotation operation that is described by three distinct rotations about the axes of the reference coordinate system in the following order: a rotation about the Z-axis, a rotation about the X-axis, and a rotation about the Y-axis rotation. (summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the Z-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param2	Specifies the X-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param3	Specifies the Y-axis rotation, in degrees. (summarize)
<input type="checkbox"/> M_ROTATION_ZYX +		Specifies to apply a rotation operation that is described by three distinct rotations about the axes of the reference coordinate system in the following order: a rotation about the Z-axis, a rotation about the Y-axis, and a rotation about the X-axis rotation. (summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the Z-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param2	Specifies the Y-axis rotation, in degrees. (summarize)
	<input type="checkbox"/> Param3	Specifies the X-axis rotation, in degrees. (summarize)
<input type="checkbox"/> M_TRANSLATION +		Specifies to apply a translation operation along each of the reference coordinate system's axes. (summarize)
	<input type="checkbox"/> MatrixId	This parameter must be set to M_NULL . (summarize)
	<input type="checkbox"/> Param1	Specifies the displacement along the X-axis of the reference coordinate system. (summarize)
	<input type="checkbox"/> Param2	Specifies the displacement along the Y-axis of the reference coordinate system. (summarize)
	<input type="checkbox"/> Param3	Specifies the displacement along the Z-axis of the reference coordinate system. (summarize)

Combination constants for the values listed in [For specifying the type of transformation](#)

You must add one of the following values to the above-mentioned value to specify how to apply the transformation.

● For specifying how to apply the transformation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ASSIGN	<p>Specifies to assign the specified transformation to the target coordinate system from the origin and orientation of the reference coordinate system.</p> <p>This is the default value.</p> <p>(summarize)</p>
<input type="checkbox"/> M_COMPOSE_WITH_CURRENT	<p>Specifies to compose the specified transformation with the current position and orientation in the reference coordinate system. This effectively applies the transformation to the current position and orientation of the target coordinate system. Note that this constant cannot be used with an undefined coordinate system.</p> <p>(summarize)</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

	Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE .
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. The parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (McalStream()) first with M_INQUIRE_SIZE_BYTE . (summarize)

SystemId

Specifies the system on which to restore the calibration object. For [M_INQUIRE_SIZE_BYTE](#), [M_LOAD](#), and [M_SAVE](#), **SystemId** is not used and must be set to **M_NULL**. For an [M_RESTORE](#) operation, this parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform on the calibration object. This parameter must be set to one of the following values:

● For specifying the operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save a calibration object to memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated calibration object.
<input type="checkbox"/> M_RESTORE	Restores a calibration object from a file or memory stream and assigns it an identifier.
<input type="checkbox"/> M_SAVE	Saves a calibration object to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the calibration object. This parameter must be set to one of the following values:

● For specifying the stream type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream.
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)

Version

Specifies the MIL version of the calibration object. This parameter must be set to one of the following values.

● For specifying the MIL version	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL.

	For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Specifies the function's control flag and should be set to **M_DEFAULT**.

CalibrationIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the calibration object.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, **CalibrationIdPtr** specifies the address of the variable from which to read the calibration object identifier.

For an [M_LOAD](#) operation, the **CalibrationIdPtr** specifies the address of the variable from which to read the identifier of the calibration object where the file or memory stream content will be loaded.

For an [M_RESTORE](#) operation, the **CalibrationIdPtr** specifies the address in which to return the identifier of the restored calibration object. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the calibration object, in bytes. If the size is not required, you can set this parameter to **M_NULL**.

Note that the size of a calibration object will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalTransformCoordinate

Synopsis

Convert coordinates between their world and pixel values.

Syntax

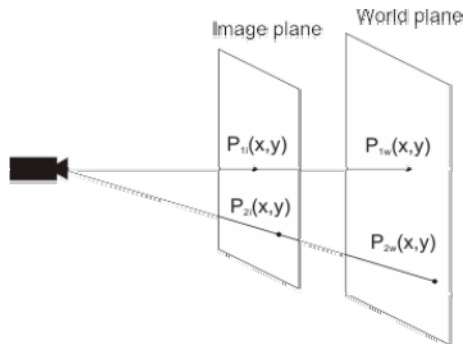
```
void McalTransformCoordinate(
    MIL_ID CalibrationOrImageId,
    MIL_INT TransformType,
    MIL_DOUBLE X,
    MIL_DOUBLE Y,
    MIL_DOUBLE *ResXPtr,
    MIL_DOUBLE *ResYPtr
)
```

Description

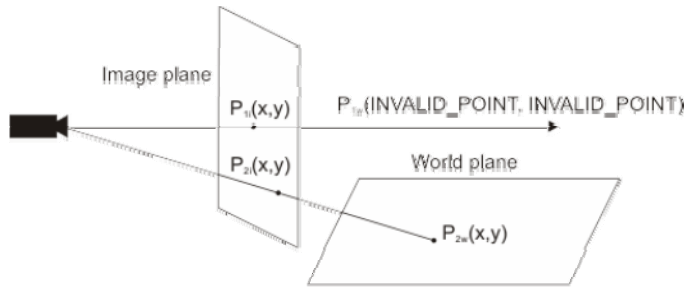
This function converts a pair of coordinates from their pixel value to their world value (or vice versa). The conversion can be performed according to a calibration object, calibrated image, or corrected image.

Note that, if you changed the origin and/or orientation of the relative coordinate system (using [McalRelativeOrigin\(\)](#) or [McalSetCoordinateSystem\(\)](#)), world coordinates will be returned, or assumed to be given, with respect to this relative coordinate system.

This function converts the coordinates of a point by making a line connecting the center of the camera's lens with the point provided, and then finding the intersection of that line with the required plane. To convert an image pixel to a world point, this function defines a line connecting the center of the camera's lens with the image plane, and then returns the intersection of this line with the world plane.



If the plane in which you requested results has been transformed using [McalSetCoordinateSystem\(\)](#) such that the projected line does not intersect with the requested plane, the contents of parameters [ResXPtr](#) and [ResYPtr](#) are undefined and a MIL error is reported. However, if `M_ALLOW_INVALID_POINT_OUTPUT` is used, (`M_INVALID_POINT`, `M_INVALID_POINT`) are the returned coordinates.



Parameters

CalibrationOrImageId

Specifies the identifier of the calibration object, calibrated image, or corrected image.

If you are transforming coordinates obtained from a calibrated image, you can either pass the identifier of the image or that of its calibration object. However, if the coordinates are obtained from a child buffer, you must pass the identifier of the child buffer.

TransformType

Specifies whether to perform a pixel-to-world or world-to-pixel conversion. This parameter must be set to one of the following values:

● For specifying pixel-to-world or world-to-pixel	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_PIXEL_TO_WORLD +	Converts from pixel to world.
<input type="checkbox"/> M_WORLD_TO_PIXEL +	Converts from world to pixel.

Combination constant for any of the possible values of the [TransformType](#) parameter

You can add the following value to the above-mentioned values to set the return values of invalid points.

● For specifying to return invalid points	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALLOW_INVALID_POINT_OUTPUT	Returns (M_INVALID_POINT, M_INVALID_POINT) instead of reporting a MIL error.

X

Specifies the X-coordinate of the input.

Y

Specifies the Y-coordinate of the input.

ResXPtr

Specifies the address in which to place the returned value of the X-coordinate. If the projected line does not intersect the requested output plane, this parameter will be set to **M_INVALID_POINT**.

ResYPtr

Specifies the address in which to place the returned value of the Y-coordinate. If the projected line does not intersect the requested output plane, this parameter will be set to **M_INVALID_POINT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalTransformCoordinate3dList

Synopsis

Convert three-dimensional coordinates between two coordinate systems.

Syntax

```
void McalTransformCoordinate3dList(
    MIL_ID CalibrationOrImageId,
    MIL_INT SrcCoordinateSystem,
    MIL_INT DstCoordinateSystem,
    MIL_INT NumPoints,
    const MIL_DOUBLE *XSrcArray,
    const MIL_DOUBLE *YSrcArray,
    const MIL_DOUBLE *ZSrcArray,
    MIL_DOUBLE *XDstArray,
    MIL_DOUBLE *YDstArray,
    MIL_DOUBLE *ZDstArray,
    MIL_INT ModeFlag
)
```

Description

This function converts a list of coordinates from a source coordinate system to a destination coordinate system. It supports conversion only for 3D-based calibration objects ([M_TSAL_BASED](#) or [M_3D_ROBOTICS](#)) that have been successfully calibrated, or for images calibrated with 3D-based calibration objects, whether corrected or not.

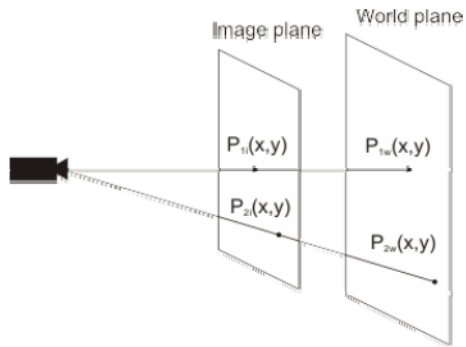
Using this function, you can convert from:

- Uncorrected image pixels to world points in any 3D coordinate system.
- Corrected image pixels to world points in any 3D coordinate system.
- World points in any 3D coordinate system to any other 3D coordinate system.
- World points in any 3D coordinate system to uncorrected image pixels.

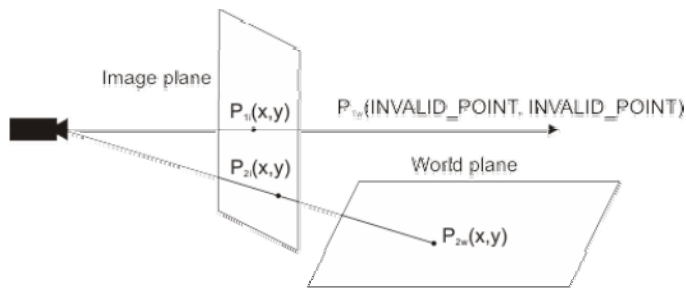
Trying to convert from world points to corrected image pixels is not supported and will return an error. This is the case if [CalibrationOrImageId](#) is a calibrated and corrected image, and the destination coordinate system is [M_IMAGE_COORDINATE_SYSTEM](#).

This function converts the coordinates of a point by making a line (an optical ray) connecting the center of the camera's lens with the point provided, and then finding the intersection of that line with the required plane specified in [DstCoordinateSystem](#) parameter. To convert an image pixel to world point, this function defines a line connecting the center of the camera's lens with the image plane, and then returns the intersection of this line with the world plane.

If the source coordinate system is set to [M_IMAGE_COORDINATE_SYSTEM](#), every point of the optical ray is a possible solution since all of these points project on the same pixel in the image coordinate system. In this case, you can specify to return the point of intersection with the world plane (Z=0) or return a unit direction vector of the optical ray using [M_PLANE_INTERSECTION](#) or [M_UNIT_DIRECTION_VECTOR](#).



If the destination coordinate system is set up such that the optical ray does not intersect with it, a MIL error is returned and all the values of the three destination arrays will be undefined. However, if [M_ALLOW_INVALID_POINT_OUTPUT](#) is used, ([M_INVALID_POINT](#), [M_INVALID_POINT](#), [M_INVALID_POINT](#)) are the returned coordinates and no MIL error is returned.



Parameters

CalibrationOrImageId

Specifies the identifier of the calibration object, calibrated image, or corrected image. When an identifier of an image buffer is specified, the transformation uses the calibration object associated with this image.

If you are transforming coordinates obtained from a calibrated image, you can pass either its identifier or the identifier of its calibration object. However, if the coordinates are obtained from a child buffer of a calibrated image, you must only pass the identifier of the child buffer.

SrcCoordinateSystem

Specifies the coordinate system of the source coordinates. This parameter must be set to one of the following values:

● For specifying the type of the source coordinate system	
Value	Description
<input type="checkbox"/> M_ABSOLUTE_COORDINATE_SYSTEM	Specifies an implicit and fixed coordinate system from which all other coordinate systems are defined.
<input type="checkbox"/> M_CAMERA_COORDINATE_SYSTEM	Specifies the coordinate system whose origin corresponds to the effective pinhole of the camera and whose Z-axis points in the direction that the camera is facing.
<input type="checkbox"/> M_IMAGE_COORDINATE_SYSTEM	Specifies the coordinate system defined in pixels in the image. It is centered on the top-left pixel of the image. It is a 2D coordinate system. (summarize)
<input type="checkbox"/> M_RELATIVE_COORDINATE_SYSTEM	Specifies the coordinate system determining the world plane used to return results.
<input type="checkbox"/> M_ROBOT_BASE_COORDINATE_SYSTEM	Specifies the coordinate system whose origin corresponds to the base of the robot. This coordinate system is only available for calibration in M_3D_ROBOTICS mode. (summarize)

<input type="checkbox"/> M_TOOL_COORDINATE_SYSTEM	Specifies the coordinate system used to position the camera. Although you can use this coordinate system to move the camera, it needs not be associated with the real camera position. (summarize)
---------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

DstCoordinateSystem

Specifies the coordinate system of the transformed points. This parameter must be set to one of the following values:

● For specifying the type of the destination coordinate system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ABSOLUTE_COORDINATE_SYSTEM	Specifies an implicit and unmovable coordinate system from which all other coordinate systems are defined.
<input type="checkbox"/> M_CAMERA_COORDINATE_SYSTEM	Specifies the coordinate system whose origin corresponds to the effective pinhole of the camera and whose Z-axis points in the direction that the camera is facing.
<input type="checkbox"/> M_IMAGE_COORDINATE_SYSTEM	Specifies the coordinate system defined in pixels in the image. It is centered on the top-left pixel of the image. It is a 2D coordinate system. (summarize)
<input type="checkbox"/> M_RELATIVE_COORDINATE_SYSTEM	Specifies the coordinate system determining the world plane used to return results.
<input type="checkbox"/> M_ROBOT_BASE_COORDINATE_SYSTEM	Specifies the coordinate system whose origin corresponds to the base of the robot. This coordinate system is only available for calibration in M_3D_ROBOTICS mode. (summarize)
<input type="checkbox"/> M_TOOL_COORDINATE_SYSTEM	Specifies the coordinate system used to position the camera. Although you can use this coordinate system to move the camera, it needs not be associated with the real camera position. (summarize)

NumPoints

Specifies the number of points in the coordinate list to transform.

XSrcArray

Specifies the address of the array containing the X-coordinates of the source points/pixels. These coordinates must be expressed in the source coordinate system.

YSrcArray

Specifies the address of the array containing the Y-coordinates of the source points/pixels. These coordinates must be expressed in the source coordinate system.

ZSrcArray

Specifies the address of the array containing the Z-coordinates of the source points/pixels. These coordinates must be expressed in the source coordinate system.

This parameter must be set to **M_NULL** if the source coordinate system is set to [M_IMAGE_COORDINATE_SYSTEM](#).

XDstArray

Specifies the address of the array that receives the transformed X-coordinates. These coordinates are expressed in the destination coordinate system.

If [M_ALLOW_INVALID_POINT_OUTPUT](#) is used and one of the source points cannot be converted, the entry in the destination array corresponding to such point is set to **M_INVALID_POINT**.

YDstArray

Specifies the address of the array that receives the transformed Y-coordinates. These coordinates are expressed in the destination coordinate system.

If [M_ALLOW_INVALID_POINT_OUTPUT](#) is used and one of the source points cannot be converted, the entry in the the destination array corresponding to such point is set to **M_INVALID_POINT**.

ZDstArray

Specifies the address of the array that receives the transformed Z-coordinates. These coordinates are expressed in the destination coordinate system.

If `M_ALLOW_INVALID_POINT_OUTPUT` is used and one of the source points cannot be converted, the entry in the the destination array corresponding to such point is set to `M_INVALID_POINT`.

This parameter must be set to `M_NULL` if the destination coordinate system is set to `M_IMAGE_COORDINATE_SYSTEM`.

ModeFlag

Specifies the mode of transformation when transforming an image pixel to a world point; that is, the source coordinate system is set to `M_IMAGE_COORDINATE_SYSTEM`. For all other cases, it should be set to `M_DEFAULT`. If the source coordinate system is set to `M_IMAGE_COORDINATE_SYSTEM`, every 3D point positioned at the optical ray is a possible solution since all points project on that same pixel. Therefore, this parameter can take one of the following values to specify which of the 3D points on the optical ray must be returned.

● For specifying the mode of transformation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT +	Same as <code>M_PLANE_INTERSECTION</code> .
<input type="checkbox"/> M_PLANE_INTERSECTION +	Returns the intersection between the optical ray and the world plane (Z=0) in the relative coordinate system, which is then converted to the coordinates of the destination coordinate system. If the destination coordinate system is <code>M_RELATIVE_COORDINATE_SYSTEM</code> , then all output Z-coordinates will be set to 0. If the optical ray does not intersect with the destination plane, this transformation mode returns a MIL error or <code>M_INVALID_POINT</code> . (summarize)
<input type="checkbox"/> M_UNIT_DIRECTION_VECTOR +	Returns the unit direction vector of the optical ray, expressed in the destination coordinate system. It is the vector going from the pinhole to the pixel. (summarize)

Combination constant for the values listed in [For specifying the mode of transformation](#)

You can add the following value to the above-mentioned values to set the return values of invalid points.

● For specifying to return invalid points	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALLOW_INVALID_POINT_OUTPUT	Returns <code>M_INVALID_POINT</code> for the X, Y, and Z coordinates of an invalid point.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalTransformCoordinateList

Synopsis

Convert a list of coordinates between their world and pixel values.

Syntax

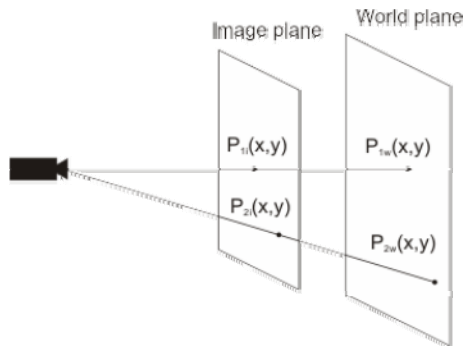
```
void McalTransformCoordinateList(
    MIL_ID CalibrationOrImageId,
    MIL_INT TransformType,
    MIL_INT NumPoints,
    const MIL_DOUBLE *SrcArrayXPtr,
    const MIL_DOUBLE *SrcArrayYPtr,
    MIL_DOUBLE *DesArrayXPtr,
    MIL_DOUBLE *DesArrayYPtr
)
```

Description

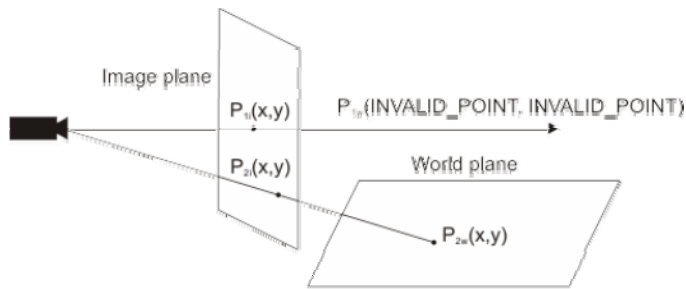
This function converts a list of coordinates from their pixel value to their world value (or vice versa). The conversion can be performed according to a calibration object, calibrated image, or corrected image.

Note that, if you changed the origin and/or orientation of the relative coordinate system (using [McalRelativeOrigin\(\)](#) or [McalSetCoordinateSystem\(\)](#)), world coordinates will be returned, or assumed to be given, with respect to this relative coordinate system.

This function converts the coordinates of a point by making a line connecting the center of the camera's lens with the point provided, and then finding the intersection of that line with the required plane. To convert an image pixel to a world point, this function defines a line connecting the center of the camera's lens with the image plane, and then returns the intersection of this line with the world plane.



If the plane in which you requested results has been transformed using [McalSetCoordinateSystem\(\)](#) such that the projected lines of some points in the input arrays do not intersect with the requested plane, the entries corresponding to these points in parameters [DesArrayXPtr](#) and [DesArrayYPtr](#) are undefined, and a MIL error is reported. However, if [M_ALLOW_INVALID_POINT_OUTPUT](#) is used, ([M_INVALID_POINT](#), [M_INVALID_POINT](#)) are the returned coordinates and no MIL error is reported.



Parameters

CalibrationOrImageId

Specifies the identifier of the calibration object, calibrated image, or corrected image. When an identifier of an image buffer is specified, the transformation uses the calibration object associated with this image.

If you are transforming the coordinates of a calibrated image, you can pass either its identifier or the identifier of its calibration object. However, if the coordinates being passed were obtained from a child buffer you must only pass the identifier of the child buffer.

TransformType

Specifies whether to perform a pixel-to-world or world-to-pixel conversion. This parameter must be set to one of the following values:

• For specifying pixel-to-world or world-to-pixel

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_PIXEL_TO_WORLD +	Converts from pixel to world.
<input type="checkbox"/> M_WORLD_TO_PIXEL +	Converts from world to pixel.

Combination constant for the values listed in [For specifying pixel-to-world or world-to-pixel](#)

You can add the following value to the above-mentioned values to set the input and/or output as interleaved.

• For optimizing :

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_PACKED	Specifies that the source coordinates are passed in a packed format and/or that the result coordinates should be returned in a packed format depending on what you pass to the SrcArrayYPtr and DesArrayYPtr parameters. If SrcArrayYPtr is passed M_NULL , SrcArrayXPtr is assumed to be a packed set of X- and Y-coordinates. If DesArrayYPtr is passed M_NULL , DesArrayXPtr is filled with a packed set of X- and Y-coordinates. If both SrcArrayYPtr and DesArrayYPtr are passed M_NULL , SrcArrayXPtr is assumed to be a packed set of X- and Y-coordinates, and DesArrayXPtr is filled with a packed set of X- and Y-coordinates. (summarize)

Combination constant for any of the possible values of the [TransformType](#) parameter

You can add the following value to the above-mentioned values to set the return values of invalid points.

• For specifying to return invalid points

<input type="checkbox"/> Value	Description
--------------------------------	-------------

<input type="checkbox"/> <code>M_ALLOW_INVALID_POINT_OUTPUT</code>	Returns (<code>M_INVALID_POINT</code> , <code>M_INVALID_POINT</code>) instead of reporting a MIL error.
--------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------

NumPoints

Specifies the number of points in the coordinate list.

SrcArrayXPtr

Specifies the address of the array of the input X-coordinates.

SrcArrayYPtr

Specifies the address of the array of the input Y-coordinates.

DesArrayXPtr

Specifies the address of the array in which to place the output X-coordinates.

If `M_ALLOW_INVALID_POINT_OUTPUT` is used and one of the source points cannot be converted, the entry in the destination array corresponding to such point is set to `M_INVALID_POINT`.

DesArrayYPtr

Specifies the address of the array in which to place the output Y-coordinates.

If `M_ALLOW_INVALID_POINT_OUTPUT` is used and one of the source points cannot be converted, the entry in the destination array corresponding to such point is set to `M_INVALID_POINT`.

Compilation information

Header	Include <code>mil.h</code> .
Library	Use <code>mil.lib</code> ; <code>milcal.lib</code> .
DLL	Requires <code>mil.dll</code> ; <code>milcal.dll</code> .

McalTransformImage

Synopsis

Physically transform an image to a corrected one by removing distortions.

Syntax

```
void McalTransformImage (
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_ID CalibrationId,
    MIL_INT InterpolationMode,
    MIL_INT OperationType,
    MIL_INT ControlFlag
)
```

Description

This function removes distortions in an image by physically transforming it according to a specified calibration object. The image is transformed such that:

- Its image coordinate system is aligned with its relative coordinate system.
- All the pixels in the destination image represents the same size in world units.
- It is scaled and positioned in the destination image according to the fill mode selected using [McalControl\(\)](#) with [M_TRANSFORM_IMAGE_FILL_MODE](#).

Instead of generating a transformed image, you can create look up tables (LUTs) that you can use to transform images using [MimWarp\(\)](#).

Note that you can transform an image once. If you pass a transformed image to [McalTransformImage\(\)](#), it will just be copied to the destination image.

Parameters

SrcImageBufId

Specifies the identifier of the source image buffer.

DestImageBufId

Specifies the identifier of the destination image buffer.

If you set the parameter [ControlFlag](#) to [M_EXTRACT_LUT_X](#) or [M_EXTRACT_LUT_Y](#), then this specifies a look up table (LUT) buffer that receives transformation information, which can be used with [MimWarp\(\)](#) to transform an image. In this case, the LUT destination buffer should be the same size as the destination image in [MimWarp\(\)](#), and the depth must be signed 32-bit.

CalibrationId

Specifies the calibration object.

● For specifying the calibration object	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the calibration object associated with the source image will be used. If the source image is not associated with a calibration object, a MIL error is returned. (summarize)
<input type="checkbox"/> MIL calibration object identifier	Specifies the identifier of a valid calibration object, which you have calibrated using either McalGrid() or McalList() .

InterpolationMode

Specifies the interpolation mode to use when associating destination pixels with source points. This parameter must be set to one of the following values:

● For specifying the type of interpolation to perform	
☐ Value	Description
<input type="checkbox"/> M_DEFAULT +	Same as M_NEAREST_NEIGHBOR + M_OVERSCAN_ENABLE .
<input type="checkbox"/> M_BICUBIC +	Specifies bicubic interpolation. When using bicubic interpolation, saturation is performed according to the type of the destination buffer. (summarize)
<input type="checkbox"/> M_BILINEAR +	Specifies bilinear interpolation. No saturation is performed. (summarize)
<input type="checkbox"/> M_NEAREST_NEIGHBOR +	Specifies nearest-neighbor interpolation. No saturation is performed. (summarize)

Combination constants for any of the possible values of the [InterpolationMode](#) parameter

You can add one of the following values to the above-mentioned values to specify how to determine the value of a destination pixel when its associated point falls outside the source buffer.

● For specifying overscan	
☐ Value	Description
<input type="checkbox"/> M_OVERSCAN_CLEAR	Sets the destination pixel to 0, if the associated point falls outside the source buffer.
<input type="checkbox"/> M_OVERSCAN_DISABLE	Leaves the destination pixel as is, if the associated point falls outside the source buffer.
<input type="checkbox"/> M_OVERSCAN_ENABLE	Uses pixels from the source buffer's ancestor buffer, if the associated point falls outside the source buffer. If the source buffer is not a child buffer or if the associated point falls outside the ancestor buffer, leave the destination pixel as is. This is the default value. (summarize)

OperationType

Specifies the function's operation type. This parameter must be set to the following value:

● For specifying the function's operation type	
☐ Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_FULL_CORRECTION .
<input type="checkbox"/> M_CORRECT_LENS_DISTORTION_ONLY	Specifies a partial correction of the source image by only removing lens distortion, without modifying the perspective effect. In this case, the destination image is not calibrated. This operation type is only supported for 3D-based calibration modes (M_TSAI_BASED or M_3D_ROBOTICS). (summarize)
<input type="checkbox"/> M_FULL_CORRECTION	Specifies a full correction of the source image. This corrects all distortions of the source image based on the provided calibration object. (summarize)

ControlFlag

Specifies the function's control flag. This parameter must be set to the following value:

● For specifying the function's control flag	
☐ Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_WARP_IMAGE .
<input type="checkbox"/> M_EXTRACT_LUT_X	Extracts the X warping look up table for the image transformation. You can use the extracted LUT to transform an image using MimWarp() with the OperationMode

	parameter to M_WARP_LUT + M_FIXED_POINT + 10. (summarize)
M_EXTRACT_LUT_Y	Extracts the Y warping look up table for the image transformation. You can use the extracted LUT to transform an image using MimWarp() with the OperationMode parameter to M_WARP_LUT + M_FIXED_POINT + 10. (summarize)
M_WARP_IMAGE	Transforms the source image into a corrected destination image.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

McalTransformResult

Synopsis

Convert a result between its world and pixel value.

Syntax

```
void McalTransformResult(
    MIL_ID CalibrationOrImageId,
    MIL_INT TransformType,
    MIL_INT ResultType,
    MIL_DOUBLE Result,
    MIL_DOUBLE *ResResult
)
```

Description

This function converts a specific result (a length, area, or angle) from its pixel value to its world value or vice-versa. The conversion can be performed according to a calibration object, calibrated image, or corrected image. However, since this function uses the average pixel size to perform the conversion, results will be more accurate if you use a corrected image.

Note that, if the relative plane used to return results is set behind the camera, the return is undefined and a MIL error is reported, unless [M_ALLOW_INVALID_POINT_OUTPUT](#) is used.

Parameters

CalibrationOrImageId

Specifies the identifier of the calibration object, calibrated image, or corrected image.

TransformType

Specifies whether to perform a pixel-to-world or world-to-pixel conversion. This parameter must be set to one of the following values:

● For specifying pixel-to-world or world-to-pixel	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_PIXEL_TO_WORLD +	Converts from pixel to world.
<input type="checkbox"/> M_WORLD_TO_PIXEL +	Converts from world to pixel.

Combination constant for any of the possible values of the [TransformType](#) parameter
You can add the following value to the above-mentioned values to set the return values of invalid points.

● For specifying to return invalid points	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALLOW_INVALID_POINT_OUTPUT	Returns M_INVALID_POINT instead of reporting a MIL error.

ResultType

Specifies the type of result the given input value represents. This parameter must be set to one of the following values:

● For specifying the type of result

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ANGLE	Represents an angle.
<input type="checkbox"/> M_AREA	Represents an area.
<input type="checkbox"/> M_LENGTH	Represents a length (for example, the perimeter of an object).
<input type="checkbox"/> M_LENGTH_X	Represents a length in the X-direction only.
<input type="checkbox"/> M_LENGTH_Y	Represents a length in the Y-direction only.

Result

Specifies the input value.

ResResult

Specifies the address in which to place the output value.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcal.lib.
DLL	Requires mil.dll; milcal.dll.

Mcode functions

Synopsis

The functions prefixed with Mcode make up the Code module. The Code module allows you to read, write, and verify symbols (codes) from most popular one-dimensional (1D) and two-dimensional (2D) symbologies (code types). The supported 1D code types include Code39, Code93, Code128, Codabar, BC412, UPC-A, UPC-E, EAN-8, EAN-13, Interleaved 2 of 5, Pharmacode, Planet, and Postnet, and RSS. The supported 2D code types are Data Matrix, Maxicode, PDF417, Micro PDF417, Truncated PDF417, and QR Code. Composite codes, which are combinations of 1D and 2D code types, are also supported. The module easily handles rotated, scaled, and degraded codes, even in complex scenes.

Functions

- [McodeAlloc](#)
- [McodeAllocResult](#)
- [McodeControl](#)
- [McodeDraw](#)
- [McodeFree](#)
- [McodeGetResult](#)
- [McodeGetResultSingle](#)
- [McodeInquire](#)
- [McodeModel](#)
- [McodeRead](#)
- [McodeRestore](#)
- [McodeSave](#)
- [McodeStream](#)
- [McodeVerify](#)
- [McodeWrite](#)

McodeAlloc

Synopsis

Allocate a code context.

Syntax

```
MIL_ID McodeAlloc(  
    MIL_ID SystemId,  
    MIL_INT ContextType,  
    MIL_INT ControlFlag,  
    MIL_ID *CodeContextIdPtr  
)
```

Description

This function allocates a code context. The code context contains the models of the codes to read, verify, or write.

To add code models to the context, use [McodeModel\(\)](#). A code context can contain multiple 1D code models of various types (excluding RSS, Planet, and Postnet), but for 2D code types, a context can contain at most one model. To adjust code model settings, use [McodeControl\(\)](#).

When the code context is no longer needed, you should free it, using [McodeFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the context. This parameter should be set to one of the following values:

For specifying the system	
Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ContextType

Specifies the context type. This parameter must be set to **M_DEFAULT**.

ControlFlag

Specifies the function's control flag. This parameter must be set to **M_DEFAULT**.

CodeContextIdPtr

Specifies the address in which to return the identifier of the code context. Since this function also returns the identifier, this parameter can be set to **M_NULL**.

Return value

The returned value is the code context identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.

DLL	Requires mil.dll; milcode.dll.
-----	--------------------------------

McodeAllocResult

Synopsis

Allocate a code result buffer.

Syntax

```
MIL_ID McodeAllocResult(  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *CodeResultIdPtr  
)
```

Description

This function allocates a code result buffer on the specified system to store results obtained from the [McodeRead\(\)](#) or [McodeVerify\(\)](#) operations. When the result buffer is no longer required, release it, using [McodeFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the buffer. This parameter should be set to one of the following values:

For specifying the system	
Value	Description
M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

CodeResultIdPtr

Specifies the address of the variable in which to write the code result buffer identifier. Since the **McodeAllocResult()** function also returns the model result buffer identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeControl

Synopsis

Control a code context or code model setting.

Syntax

```
void McodeControl(  
    MIL_ID CodeId,  
    MIL_INT ControlType,  
    MIL_DOUBLE ControlValue  
)
```

Description

This function changes the setting of a specified code context or code model.

Parameters

- CodeId
- Specifies the identifier of the code context or code model.
- ControlType
- Specifies the setting to change.
- See the [Parameter associations](#) section for possible values.
- ControlValue
- Specifies the setting's new value.
- See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For the context and models](#)
- [Code context settings for McodeRead operations](#)
- [Code context settings for McodeVerify operations](#)
- [Code context settings for McodeRead and McodeVerify operations](#)
- [Code model settings for McodeRead operations](#)
- [Code model settings for McodeRead and McodeVerify operations](#)
- [Code model settings for McodeRead, McodeVerify, and McodeWrite operations](#)

The following **ControlType** and corresponding **ControlValue** parameter settings can be specified for both the code context and each individual model in the code context:

For the context and models		
ControlType		Description
	ControlValue	
☐ M_INTERACTIVE		[This is only applicable to Windows]

	Opens a control dialog which allows you to edit the control types interactively. (summarize)
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.

The following **ControlType** and corresponding **ControlValue** parameter settings can be specified for a code context to control an **McodeRead()** operation.

Code context settings for McodeRead operations	
ControlType	Description
ControlValue	
<input type="checkbox"/> M_STOP_READ	Stops the current read operation. You must call McodeControl() with M_STOP_READ from another thread (typically of higher priority). The results returned after making this call are invalid and discarded. (summarize)
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.
<input type="checkbox"/> M_TOTAL_NUMBER	Sets the maximum number of codes to read in one image. Note that this number is limited by the maximum number of occurrences to read in each code model (M_NUMBER). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> M_ALL	Specifies to read all code occurrences in the image, up to the maximum number of occurrences to read in each code model (M_NUMBER).
<input type="checkbox"/> Value >= 0	Specifies the maximum number of codes to read in the image.

The following **ControlType** and corresponding **ControlValue** parameter settings for a code context, to control **McodeVerify()** operations.

Code context settings for McodeVerify operations	
ControlType	Description
ControlValue	
<input type="checkbox"/> M_ABSOLUTE_APERTURE_SIZE	Sets the absolute size (diameter) of the aperture. Use this control type when using an absolute aperture mode (using M_APERTURE_MODE set to M_ABSOLUTE). This control type is available for all 1D, 2D cross-row, and composite code types. (summarize)
<input type="checkbox"/> Value >= 2	Specifies the absolute aperture size, in pixels.
<input type="checkbox"/> M_APERTURE_MODE	Sets the way in which the aperture size is determined. This control type is available for all 1D, 2D cross-row, and composite code types. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_RELATIVE .
<input type="checkbox"/> M_ABSOLUTE	Uses the absolute aperture size, set using M_ABSOLUTE_APERTURE_SIZE .
<input type="checkbox"/> M_DISABLE	Disables the aperture. Note that this is provided for backwards compatibility with older code. In this case, the verify operation's returned results will more closely resemble the results of a read operation. (summarize)
<input type="checkbox"/> M_RELATIVE	Uses a relative aperture size, based on the cell size (using M_CELL_SIZE...) and the relative aperture factor (using M_RELATIVE_APERTURE_FACTOR). (summarize)
<input type="checkbox"/> M_MAXIMUM_CALIBRATED_REFLECTANCE	Sets the maximum grayscale value in the target image. This control type is used in the scan reflectance profile analysis. If this control type does not accurately represent the actual maximum grayscale value from the target image, the resulting values from the scan reflectance profile are invalid. Note that this value must be higher than the value of M_MINIMUM_CALIBRATED_REFLECTANCE . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 255. (summarize)

<input type="checkbox"/> 0 to 255	Specifies the maximum calibrated reflectance.
<input type="checkbox"/> M_MINIMUM_CALIBRATED_REFLECTANCE	Sets the minimum grayscale value in the target image. This control type is used in the scan reflectance profile analysis. If this control type does not accurately represent the actual minimum grayscale value from the target image, the resulting values from the scan reflectance profile are invalid. Note that this value must be lower than the value of M_MAXIMUM_CALIBRATED_REFLECTANCE . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to 255	Specifies the minimum calibrated reflectance.
<input type="checkbox"/> M_PIXEL_SIZE_IN_MM	Sets the scale between a pixel and its physical measurement in the world. To simplify the computation of this value, measure the smallest bar or gap in a code and divide it by the cell size; this yields a good approximation. Alternately, calculate the scale using the following formula: $(size\ of\ the\ object) / (number\ of\ pixels\ in\ object)$. It is important that the measure should be in millimeters, since all measures related to bar code symbologies are generally made in millimeters. Note that this control type impacts the aperture size selection, and the M_SCAN_INTERCHARACTER_GAP_GRADE result (retrieved using McodeGetResult()). This control type is available for all 1D, 2D cross-row, and composite code types. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_UNKNOWN .
<input type="checkbox"/> M_UNKNOWN	Specifies that the scale between a pixel and its physical measurement is not known.
<input type="checkbox"/> Value > 0	Specifies the scale between a pixel and its physical measurement, in millimeters per pixel units.
<input type="checkbox"/> M_RELATIVE_APERTURE_FACTOR	Sets the aperture factor to use when M_APERTURE_MODE is set to M_RELATIVE . This control type is available for all 1D, 2D cross-row, and composite code types. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_AUTO .
<input type="checkbox"/> 0 <= Value <= 2	Specifies the aperture factor. The aperture factor is multiplied by the M_CELL_SIZE_MIN to determine the aperture's diameter. (summarize)
<input type="checkbox"/> M_AUTO	Specifies that the aperture factor is chosen according to ISO 15416. Note that if the nominal bar width is less than 0.1 mm, the aperture factor is 0.5. When dealing with EAN/UPC code types the aperture factor is 0.15. (summarize)

The following **ControlType** and **ControlValue** parameter settings can be specified for a code context, to control [McodeRead\(\)](#) and [McodeVerify\(\)](#) operations.

Code context settings for McodeRead and McodeVerify operations	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_MINIMUM_CONTRAST	Sets the minimum contrast between the foreground and background in the target image for 1D codes (excluding Planet and Postnet) when using the M_ADAPTIVE threshold mode. Increasing the minimum contrast will typically improve read operations, particularly in the presence of noise and non-uniform lighting. However, if the minimum contrast is higher than the contrast of a code, the code will not be read correctly. For 2D codes, the minimum contrast is determined automatically, so the adaptive threshold mode does not use the M_MINIMUM_CONTRAST setting. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 50. (summarize)
<input type="checkbox"/> 1 to 255	Specifies the minimum contrast.
<input type="checkbox"/> M_SCANLINE_HEIGHT	Sets the scanline height. A scanline is the line along which the code is read. The scanline is usually first taken across the middle of the image, and then, if the code is not found, it is taken either higher or lower in the image. The scanline height is the thickness of the scanline.

	<p>This control type overrides the automatic selection based on the value of M_SPEED. In general, the higher the value for this control type, the slower the code is read. Higher values for this control increase the robustness of read and verify operations. Using the default value is recommended; the higher the search speed, the lower the default value.</p> <p>This control type only applies to 1D code types, Micro PDF417, and code types with 1D components. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies that the scanline height is based on the value of M_SPEED .
<input type="checkbox"/> 0<Value<=256	Specifies the scanline height, in pixels. Note that only integer values are accepted. (summarize)
<input type="checkbox"/> M_SCANLINE_STEP	<p>Sets the scanline step.</p> <p>The scanline step is the gap between scanlines.</p> <p>This control type overrides the automatic selection based on the value of M_SPEED. In general, the higher the value for this control type, the faster the code is read. The bigger the step, the more chance of missing a readable line of code. Using the default value is recommended. The higher the search speed, the higher the default value.</p> <p>This control type only applies to 1D code types, Micro PDF417, and code types with 1D components. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies that the scanline frequency is based on the value of M_SPEED .
<input type="checkbox"/> Value > 1	Specifies the scanline frequency, in pixels. Note that only integer values are accepted. (summarize)
<input type="checkbox"/> M_SEARCH_ANGLE_MODE	<p>Sets whether to perform calculations specific to angular-range strategies for the code context.</p> <p>Typically, to search for codes within the specified angular range (M_SEARCH_ANGLE - M_SEARCH_ANGLE_DELTA_NEG to M_SEARCH_ANGLE + M_SEARCH_ANGLE_DELTA_POS) in the target, calculations specific to angular-range search strategies should be enabled for the context. These calculations are not required to search for models at their nominal angle (M_SEARCH_ANGLE). In addition, if you expect that the codes sought are close to their model's nominal angle, you can try disabling these calculations to see if the Code module can still find the required codes. Disabling these calculations might speed up the search depending on the model and the target. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_ENABLE .
<input type="checkbox"/> M_DISABLE	Disables calculations specific to angular-range search strategies.
<input type="checkbox"/> M_ENABLE	Enables calculations specific to angular-range search strategies.
<input type="checkbox"/> M_SPEED	<p>Sets the search speed. The faster the search speed, the less robust the operation. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_MEDIUM .
<input type="checkbox"/> M_HIGH	Specifies a high search speed.
<input type="checkbox"/> M_LOW	Specifies a low search speed.
<input type="checkbox"/> M_MEDIUM	Specifies a medium search speed.
<input type="checkbox"/> M_VERY_HIGH	Specifies a very high search speed.
<input type="checkbox"/> M_VERY_LOW	Specifies a very low search speed.
<input type="checkbox"/> M_STRING_SIZE_MAX	<p>Sets the maximum size of the string (number of characters) encoded to read in each code. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> 1 to 65, 535	<p>Specifies the maximum string size.</p> <p>For a M_POSTNET code, the value must be either 5, 9, or 11.</p> <p>For a M_PLANET code, the value must be 11 or 13.</p> <p>For a M_CODABAR code, the value must be at least 3.</p>

	(summarize)
<input type="checkbox"/> M_ANY	Specifies that there is no maximum string size.
<input type="checkbox"/> M_STRING_SIZE_MIN	Sets the minimum string size (number of characters) encoded to read in each code. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> 1 to 65, 535	Specifies the minimum string size. For a M_POSTNET code, the value must be either 5, 9, or 11. For a M_PLANET code, the value must be 11 or 13. For a M_CODABAR code, the value must be at least 3. (summarize)
<input type="checkbox"/> M_ANY	Specifies that there is no minimum string size.
<input type="checkbox"/> M_THRESHOLD	Sets the threshold value used to internally binarize the source image. Note that if the background is darker than the code(s) in some places but lighter in others, a simple binarization will not separate the code(s) from the background. In this case, you should use the M_ADAPTIVE mode or process the image before performing the read or verify operation so that the background is strictly darker or strictly lighter than the code. (summarize)
<input type="checkbox"/> M_DEFAULT	Selects the best threshold value automatically using the source image's histogram.
<input type="checkbox"/> $0 \leq \text{Value} \leq 255$	Specifies the threshold value.
<input type="checkbox"/> M_ADAPTIVE	Specifies to use a fast dynamic local threshold. This mode selects an individual threshold for each pixel based on the range of intensity values in its local neighborhood. M_ADAPTIVE works better than M_DEFAULT in cases where the image has non-uniform lighting. For linear (1D) code types (excluding Planet and Postnet), you must specify the minimum contrast (McodeControl() with M_MINIMUM_CONTRAST) between the foreground and background in the target image when using the M_ADAPTIVE mode. For 2D codes, the minimum contrast is determined automatically, so the adaptive threshold mode does not use the M_MINIMUM_CONTRAST setting. The adaptive threshold mode is not supported when requesting high accuracy position results (M_POSITION_ACCURACY set to M_HIGH) M_DATAMATRIX , M_MAXICODE , M_MICROPDF417 , M_QRCODE , and 1D code types (excluding Planet and Postnet) are the code types that support M_ADAPTIVE . (summarize)
<input type="checkbox"/> M_TIMEOUT	Specifies the maximum decoding time for a read or verify operation. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 2000 msec. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that there is no maximum decoding time.
<input type="checkbox"/> Value ≥ 0	Specifies the maximum decoding time, in msec.

The following **ControlType** and corresponding **ControlValue** parameter settings can be specified for a code model, to control [McodeRead\(\)](#) operations. If you pass a code context to the **CodeId** parameter, the specified control type setting is applied to all the code models in the context. If none of the code models support the specified control type, an error occurs. If only some code models do not support the specified control type, the control type setting is ignored for these code models.

Code model settings for McodeRead operations	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_ECC_CORRECTED_NUMBER	Forces McodeRead() to perform a more robust read operation to minimize the number of errors to correct. Enabling M_ECC_CORRECTED_NUMBER decreases performance, particularly in good quality images or in images where the error count is not important.

	M_PDF417 and M_TRUNCATED_PDF417 are the code types to which this control type applies. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies not to perform a more robust read operation.
<input type="checkbox"/> M_ENABLE	Specifies to perform a more robust read operation.
<input type="checkbox"/> M_NUMBER	Sets the maximum number of codes to read for the specified code model. Note that this number is limited by the maximum total number of codes to read in the image (McodeControl() with M_TOTAL_NUMBER) and does not apply to McodeVerify() operations. Only 1D code types (excluding RSS, Planet, and Postnet code types) support searching for multiple occurrences. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1. (summarize)
<input type="checkbox"/> M_ALL	Specifies that all code model occurrences are read up to the maximum number limited by M_TOTAL_NUMBER .
<input type="checkbox"/> Value >= 0	Specifies the maximum number of codes to read for the specified code model.
<input type="checkbox"/> M_POSITION_ACCURACY	Sets the accuracy of positional results. Accuracy depends on the settings of the code context and its models. Note that setting this control type to M_HIGH can lead to longer read times. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_LOW .
<input type="checkbox"/> M_HIGH	Specifies to report the positional results of code read operations with high accuracy.
<input type="checkbox"/> M_LOW	Specifies to report the positional results of code read operations with low accuracy.

The following **ControlType** and corresponding **ControlValue** parameter settings can be specified for a code model to control [McodeRead\(\)](#) and [McodeVerify\(\)](#) operations. If you pass a code context to the **CodeId** parameter, the specified control type setting is applied to all the code models in the context. If none of the code models support the specified control type, an error will occur. If only some code models do not support the specified control type, the control type setting is ignored for these code models.

Code model settings for McodeRead and McodeVerify operations	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_BEARER_BAR	Sets whether bearer bars surround the codes to read. Note that this control type is only taken into account when the search angle is not specified. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ABSENT .
<input type="checkbox"/> M_ABSENT	Specifies that no bearer bars surround the codes.
<input type="checkbox"/> M_PRESENT	Specifies that bearer bars surround the codes.
<input type="checkbox"/> M_CELL_NUMBER_X_MAX	Sets the maximum number of cells for which to search, in the X-direction of a 2D code. M_DATAMATRIX is the only code type to which this control type applies. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Specifies to search for codes with any number of cells.
<input type="checkbox"/> Value > 0	Specifies the maximum number of cells for which to search. If a value is specified, it must be larger than M_CELL_NUMBER_X_MIN , otherwise an error occurs. (summarize)
<input type="checkbox"/> M_CELL_NUMBER_X_MIN	Sets the minimum number of cells for which to search, in the X-direction of a 2D code. M_DATAMATRIX is the only code type to which this control type applies. (summarize)

<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Specifies to search for codes with any number of cells.
<input type="checkbox"/> Value > 0	Specifies the minimum number of cells for which to search. If a value is specified, it must be smaller than M_CELL_NUMBER_X_MAX , otherwise an error occurs. (summarize)
<input type="checkbox"/> M_CELL_NUMBER_Y_MAX	Sets the maximum number of cells for which to search, in the Y-direction of a 2D code. M_DATAMATRIX is the only code type to which this control type applies. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Specifies to search for codes with any number of cells.
<input type="checkbox"/> Value > 0	Specifies the maximum number of cells for which to search. If a value is specified, it must be larger than M_CELL_NUMBER_Y_MIN , otherwise an error occurs. (summarize)
<input type="checkbox"/> M_CELL_NUMBER_Y_MIN	Sets the minimum number of cells for which to search, in the Y-direction of a 2D code. M_DATAMATRIX is the only code type to which this control type applies. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Specifies to search for codes with any number of cells.
<input type="checkbox"/> Value > 0	Specifies the minimum number of cells for which to search. If a value is specified, it must be smaller than M_CELL_NUMBER_Y_MAX , otherwise an error occurs. (summarize)
<input type="checkbox"/> M_CHECK_FINDER_PATTERN	Sets whether checking for a false Data Matrix pattern is enabled. If this control type is enabled, read and verify operations are more robust, and false Data Matrix code types are not read. This control type should only be enabled if it is possible that parts of your image could be falsely interpreted as a Data Matrix code types. M_DATAMATRIX is the only code type to which this control type applies. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Disables checking for false Data Matrix code types.
<input type="checkbox"/> M_ENABLE	Enables checking for false Data Matrix code types.
<input type="checkbox"/> M_DATAMATRIX_SHAPE	Sets the shape of the Data Matrix code type. M_DATAMATRIX is the only code type to which this control type applies. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Specifies that the Data Matrix code type can be any shape.
<input type="checkbox"/> M_RECTANGLE	Specifies that the Data Matrix code has a rectangular shape. This shape can only use the M_ECC_200 error correction scheme. (summarize)
<input type="checkbox"/> M_SQUARE	Specifies that the Data Matrix has a square shape.
<input type="checkbox"/> M_DOT_SPACING	Sets the distance between 2 dots in a matrix code type composed of dots. M_DOT_SPACING is half the distance between the edges of adjacent dots in the code. If the dark cells in a printed code are oversized (due to printing artifacts) and overlap, M_DOT_SPACING is half the depth of the overlap.



Note that over-estimating this value severely impairs read operations.

[\(summarize\)](#)

<input type="checkbox"/> M_DEFAULT	Specifies no spacing.										
<input type="checkbox"/> -256 to 256	Specifies the distance, in pixels. Specifying a positive value means that you are specifying half the distance between the edges of the dots. Specifying a negative value means you are specifying the half depth of the overlap. (summarize)										
<input type="checkbox"/> M_SCAN_REVERSE	Sets whether the bar code reader scans the bar code both left-to-right and right-to-left, or only left-to-right. (summarize)										
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .										
<input type="checkbox"/> M_DISABLE	Specifies that the bar code reader does not scan the bar code right-to-left.										
<input type="checkbox"/> M_ENABLE	Specifies that the bar code reader scans in both directions.										
<input type="checkbox"/> M_SEARCH_ANGLE	<p>Sets the nominal search angle.</p> <p>The search is performed between the range of angles defined by: (M_SEARCH_ANGLE - M_SEARCH_ANGLE_DELTA_NEG) to (M_SEARCH_ANGLE + M_SEARCH_ANGLE_DELTA_POS), inclusively, starting with an angle closest to that of M_SEARCH_ANGLE.</p> <p>For the PDF417, Truncated PDF417, QR Code, Maxicode, and Data Matrix code types, the angular range does not affect the speed of the operation.</p> <p>Note that M_SEARCH_ANGLE_DELTA_NEG and M_SEARCH_ANGLE_DELTA_POS must be set to M_DEFAULT when dealing with the following code types: M_PHARMACODE, M_RSSCODE (M_RSS14_STACKED sub-type only), M_COMPOSITECODE, and M_MICROPDF417.</p> <p>A few code types have maximum search angles. This limit must also be respected when specifying an angular range.</p> <table border="1"> <thead> <tr> <th>Code type</th><th>Maximum search angle</th></tr> </thead> <tbody> <tr> <td>M_PHARMACODE</td><td>±10°</td></tr> <tr> <td>M_RSSCODE (M_RSS14_STACKED sub-type only)</td><td>±5°</td></tr> <tr> <td>M_COMPOSITECODE</td><td>±5°</td></tr> <tr> <td>M_MICROPDF417</td><td>±2°</td></tr> </tbody> </table> <p>(summarize)</p>	Code type	Maximum search angle	M_PHARMACODE	±10°	M_RSSCODE (M_RSS14_STACKED sub-type only)	±5°	M_COMPOSITECODE	±5°	M_MICROPDF417	±2°
Code type	Maximum search angle										
M_PHARMACODE	±10°										
M_RSSCODE (M_RSS14_STACKED sub-type only)	±5°										
M_COMPOSITECODE	±5°										
M_MICROPDF417	±2°										
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)										
<input type="checkbox"/> 0.0 to 360.0	Specifies the nominal angle, in degrees.										
<input type="checkbox"/> M_SEARCH_ANGLE_DELTA_NEG	<p>Sets the negative angle range of the search.</p> <p>To search for code occurrences within the angular range, calculations specific to angular-range search strategies should be enabled (M_SEARCH_ANGLE_MODE) for the context. When enabled, the angular range should be used to cover an expected variance in angle. Note that the actual angle of the occurrence does not affect search speed. If you need to search for a model at discrete angles only (for example, at intervals of 90 degrees), it is typically more efficient to define several models with different expected angles, than to search through the full angular range.</p> <p>For the PDF417, Truncated PDF417, QR Code, Maxicode, and Data Matrix code types, the angular range does not affect the speed of the operation.</p> <p>For M_PHARMACODE, M_RSSCODE (M_RSS14_STACKED sub-type only), M_COMPOSITECODE, and M_MICROPDF417, the angular range should still be within the maximum search angle (as described for M_SEARCH_ANGLE). (summarize)</p>										

<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 5.0. (summarize)
<input type="checkbox"/> 0.0 to 180.0	Specifies a negative angle range, in degrees.
<input type="checkbox"/> M_SEARCH_ANGLE_DELTA_POS	<p>Sets the positive angle range of the search.</p> <p>To search for code occurrences within the angular range, calculations specific to angular-range search strategies should be enabled (M_SEARCH_ANGLE_MODE) for the context. When enabled, the angular range should be used to cover an expected variance in angle. Note that the actual angle of the occurrence does not affect search speed. If you need to search for a model at discrete angles only (for example, at intervals of 90 degrees), it is typically more efficient to define several models with different expected angles, than to search through the full angular range.</p> <p>For the PDF417, Truncated PDF417, QR Code, Maxicode, and Data Matrix code types, the angular range does not affect the speed of the operation.</p> <p>For M_PHARMACODE, M_RSSCODE (M_RSS14_STACKED sub-type only), M_COMPOSITECODE, and M_MICROPDF417, the angular range should still be within the maximum search angle (as described for M_SEARCH_ANGLE). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 5.0. (summarize)
<input type="checkbox"/> 0.0 to 180.0	Specifies a positive angle range, in degrees.
<input type="checkbox"/> M_SUB_TYPE	<p>Sets the particular code sub-types for which to search. Enabling fewer sub-types will help increase the speed of the operation.</p> <p>Some versions of RSS codes only differ by their bar height and not their structure. In these cases, the operation is successful even if only one of the sub-types is enabled. For example, RSS-14 and RSS-14 Truncated can be decoded if either M_RSS14 or M_RSS14_TRUNCATED is enabled.</p> <p>The sub-types are additive. If you set the sub-type to M_UPC_A + M_RSS14, you can search for UPC A and RSS 14 codes. If you enabled sub-types previously, they are no longer enabled. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	<p>Specifies to search for all of the code sub-types that can be specified for M_SUB_TYPE.</p> <p>M_COMPOSITECODE and M_RSSCODE are the only code types that support this setting. (summarize)</p>
<input type="checkbox"/> M_EAN8	<p>Specifies that the EAN-8 code sub-type is enabled.</p> <p>M_COMPOSITECODE is the only code type that supports this setting. (summarize)</p>
<input type="checkbox"/> M_EAN13	<p>Specifies that the EAN-13 code sub-type is enabled.</p> <p>M_COMPOSITECODE is the only code type that supports this setting. (summarize)</p>
<input type="checkbox"/> M_RSS14	<p>Specifies that the RSS-14 code sub-type is enabled.</p> <p>M_COMPOSITECODE and M_RSSCODE are the only code types that support this setting. (summarize)</p>
<input type="checkbox"/> M_RSS14_STACKED	<p>Specifies that the RSS-14 Stacked code sub-type is enabled.</p> <p>M_COMPOSITECODE and M_RSSCODE are the only code types that support this setting. (summarize)</p>
<input type="checkbox"/> M_RSS14_STACKED_OMNI	<p>Specifies that the RSS-14 Stacked Omni code sub-type is enabled.</p> <p>M_COMPOSITECODE and M_RSSCODE are the only code types that support this setting. (summarize)</p>
<input type="checkbox"/> M_RSS14_TRUNCATED	<p>Specifies that the RSS-14 Truncated code sub-type is enabled.</p> <p>M_COMPOSITECODE and M_RSSCODE are the only code types that support this setting. (summarize)</p>

<input type="checkbox"/> M_RSS_EXPANDED	Specifies that the RSS-14 Expanded code sub-type is enabled. M_COMPOSITECODE and M_RSSCODE are the only code types that support this setting. (summarize)
<input type="checkbox"/> M_RSS_EXPANDED_STACKED	Specifies that the RSS-14 Expanded Stacked code sub-type is enabled. M_COMPOSITECODE and M_RSSCODE are the only code types that support this setting. (summarize)
<input type="checkbox"/> M_RSS_LIMITED	Specifies that the RSS-14 Limited code sub-type is enabled. M_COMPOSITECODE and M_RSSCODE are the only code types that support this setting. (summarize)
<input type="checkbox"/> M_UCCEAN128	Specifies that the UPC/EAN-128 code sub-type is enabled. M_COMPOSITECODE is the only code type that supports this setting. (summarize)
<input type="checkbox"/> M_UPC_A	Specifies that the UPC-A code sub-type is enabled. M_COMPOSITECODE is the only code type that supports this setting. (summarize)
<input type="checkbox"/> M_UPC_E	Specifies that the UPC-E code sub-type is enabled. M_COMPOSITECODE is the only code type that supports this setting. (summarize)
<input type="checkbox"/> M_USE_PRESEARCH	Sets whether the localization operation is performed prior to the decoding step of an operation. Only 2D code types support this setting. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the operation is not performed.
<input type="checkbox"/> M_ENABLE	Specifies that the operation is performed.

The following **ControlType** and corresponding **ControlValue** parameter settings can be specified for a code model to control [McodeRead\(\)](#), [McodeVerify\(\)](#) and [McodeWrite\(\)](#) operations. If you pass a code context to the **CodeId** parameter, the specified control type setting is applied to all the code models in the context. If none of the code models support the specified control type, an error occurs. If only some code models do not support the specified control type, the control type setting is ignored for these code models.

Code model settings for McodeRead, McodeVerify, and McodeWrite operations	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_CELL_NUMBER_X	Sets the number of cells of a 2D code in the X-direction. This control type is only used for 2D code types (for example, M_DATAMATRIX , M_PDF417 , and M_TRUNCATED_PDF417 code types), not for bar codes. The number of cells is not required for Maxicode because it is a fixed size. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Searches for a code with any number of cells, when performing an McodeRead() or McodeVerify() operation. Fits the string to be encoded into the smallest possible number of cells, when performing an McodeWrite() operation. (summarize)
<input type="checkbox"/> Value > 0	Specifies the number of cells. This can be any integer value. Note that for the PDF417 code type, the value that you pass must be equal to $17c + 35$, where c is the required number of columns, and 35 represents the number of cells required for the start and stop characters.

	<p>For the Truncated PDF417 code type, the value that you pass must be equal to $17c + 18$, where c is the required number of columns, and 18 represents the number of cells required for the start and stop characters.</p> <p>For the MicroPDF417 code type, this setting should be set to 38 for a one-column code, 55 for a two-column code, 82 for a three-column code, and 99 for a four-column code.</p> <p>(summarize)</p>																
<input type="checkbox"/> M_CELL_NUMBER_Y	<p>Sets the number of cells of a 2D code in the Y-direction.</p> <p>This control type is only used for 2D code types (for example, M_DATAMATRIX, M_PDF417, and M_TRUNCATED_PDF417 code types), not for bar codes. The number of cells is not required for Maxicode because it is a fixed size.</p> <p>When used with M_PDF417, M_TRUNCATED_PDF417, and M_MICROPDF417, this represents the number of rows.</p> <p>(summarize)</p>																
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .																
<input type="checkbox"/> M_ANY	<p>Searches for a code with any number of cells, when performing an McodeRead() or McodeVerify() operation.</p> <p>Fits the string to be encoded into the smallest possible number of cells, when performing an McodeWrite() operation.</p> <p>(summarize)</p>																
<input type="checkbox"/> Value > 0	<p>Specifies the number of cells. For exact values, refer to the standard documentation of the code used.</p> <p>For the PDF417 and the Truncated PDF417 code types, the value of this setting is used when calculating the M_CELL_NUMBER_X.</p> <p>For the MicroPDF417 code type, this setting should be a value between 1 and 4.</p> <p>(summarize)</p>																
<input type="checkbox"/> M_CELL_SIZE_MAX	<p>Sets the maximum cell size.</p> <p>(summarize)</p>																
<input type="checkbox"/> M_DEFAULT	Selects an appropriate size, automatically.																
<input type="checkbox"/> Value	<p>Specifies the size, in pixels. Only integer values are accepted.</p> <p>(summarize)</p>																
<input type="checkbox"/> M_CELL_SIZE_MIN	<p>Sets the minimum cell size.</p> <p>(summarize)</p>																
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default cell size, in pixels. The default depends on the operation and the code type. For a read operation, M_DEFAULT is equivalent to 1.</p> <p>If you are using McodeWrite() with M_NULL to inquire about the minimum buffer size required, M_DEFAULT is equivalent to either 10 for a Maxicode code type, or 4 for all other codes types.</p> <p>If you are using McodeWrite() and specify a buffer, M_DEFAULT causes the code to be resized so as to just fit into the target image of the operation.</p> <p>(summarize)</p>																
<input type="checkbox"/> Value > 0	<p>Specifies the cell size, in pixels. Only integer values are accepted.</p> <p>(summarize)</p>																
<input type="checkbox"/> M_ENCODING	<p>Sets the type of encoding scheme.</p> <p>(summarize)</p>																
<input type="checkbox"/> M_DEFAULT	<p>Specifies to use the default encoding scheme for the code type. The following table lists the default encoding scheme:</p> <table> <tr> <td>BC412</td><td>M_ENC_STANDARD</td></tr> <tr> <td>Codabar</td><td>M_ENC_STANDARD</td></tr> <tr> <td>Code39</td><td>M_ENC_STANDARD</td></tr> <tr> <td>Code93</td><td>M_ENC_ASCII</td></tr> <tr> <td>Code128</td><td>M_ENC_ASCII</td></tr> <tr> <td>Composite</td><td>M_ENC_RSS14</td></tr> <tr> <td>Data Matrix</td><td>M_ANY</td></tr> <tr> <td></td><td></td></tr> </table>	BC412	M_ENC_STANDARD	Codabar	M_ENC_STANDARD	Code39	M_ENC_STANDARD	Code93	M_ENC_ASCII	Code128	M_ENC_ASCII	Composite	M_ENC_RSS14	Data Matrix	M_ANY		
BC412	M_ENC_STANDARD																
Codabar	M_ENC_STANDARD																
Code39	M_ENC_STANDARD																
Code93	M_ENC_ASCII																
Code128	M_ENC_ASCII																
Composite	M_ENC_RSS14																
Data Matrix	M_ANY																

	<table> <tr><td>Ean8</td><td>M_ENC_NUM</td></tr> <tr><td>Ean13</td><td>M_ENC_NUM</td></tr> <tr><td>Interleaved25</td><td>M_ENC_NUM</td></tr> <tr><td>Maxicode</td><td>M_ANY</td></tr> <tr><td>MicroPDF417</td><td>M_ENC_STANDARD</td></tr> <tr><td>PDF417</td><td>M_ENC_STANDARD</td></tr> <tr><td>Pharmacode</td><td>M_ENC_NUM</td></tr> <tr><td>Planet</td><td>M_ENC_NUM</td></tr> <tr><td>Postnet</td><td>M_ENC_NUM</td></tr> <tr><td>QR</td><td>M_ANY</td></tr> <tr><td>RSS</td><td>M_ENC_RSS14</td></tr> <tr><td>Truncated PDF</td><td>M_ENC_STANDARD</td></tr> <tr><td>UPC-A</td><td>M_ENC_NUM</td></tr> <tr><td>UPC-E</td><td>M_ENC_NUM</td></tr> </table> <p>(summarize)</p>	Ean8	M_ENC_NUM	Ean13	M_ENC_NUM	Interleaved25	M_ENC_NUM	Maxicode	M_ANY	MicroPDF417	M_ENC_STANDARD	PDF417	M_ENC_STANDARD	Pharmacode	M_ENC_NUM	Planet	M_ENC_NUM	Postnet	M_ENC_NUM	QR	M_ANY	RSS	M_ENC_RSS14	Truncated PDF	M_ENC_STANDARD	UPC-A	M_ENC_NUM	UPC-E	M_ENC_NUM
Ean8	M_ENC_NUM																												
Ean13	M_ENC_NUM																												
Interleaved25	M_ENC_NUM																												
Maxicode	M_ANY																												
MicroPDF417	M_ENC_STANDARD																												
PDF417	M_ENC_STANDARD																												
Pharmacode	M_ENC_NUM																												
Planet	M_ENC_NUM																												
Postnet	M_ENC_NUM																												
QR	M_ANY																												
RSS	M_ENC_RSS14																												
Truncated PDF	M_ENC_STANDARD																												
UPC-A	M_ENC_NUM																												
UPC-E	M_ENC_NUM																												
<input type="checkbox"/> M_ANY	<p>Specifies any type of encoding scheme.</p> <p>M_DATAMATRIX, M_MAXICODE, and M_QRCODE are the code types that can use this encoding scheme for read and verify operations.</p> <p>M_DATAMATRIX is the code type that can use this encoding scheme for write operations.</p> <p>(summarize)</p>																												
<input type="checkbox"/> M_ENC_ALPHA	<p>Specifies an encoding scheme that supports uppercase alphabetical characters, along with the space.</p> <p>M_DATAMATRIX is the code type that can use this encoding scheme.</p> <p>(summarize)</p>																												
<input type="checkbox"/> M_ENC_ALPHANUM	<p>Specifies an encoding scheme that supports alphanumeric characters, as well as the space.</p> <p>M_DATAMATRIX is the code type that can use this encoding scheme.</p> <p>(summarize)</p>																												
<input type="checkbox"/> M_ENC_ALPHANUM_PUNC	<p>Specifies a similar encoding scheme to M_ENC_ALPHANUM, except it also supports the (.), (,), (-) and (/) characters.</p> <p>M_DATAMATRIX is the code type that can use this encoding scheme.</p> <p>(summarize)</p>																												
<input type="checkbox"/> M_ENC_ASCII	<p>Specifies an encoding scheme that supports ASCII characters.</p> <p>M_DATAMATRIX, M_CODE39, M_CODE93, and M_CODE128 are the code types that can use this encoding scheme.</p> <p>(summarize)</p>																												
<input type="checkbox"/> M_ENC_EAN8	<p>Specifies an encoding scheme for a composite code whose 1D portion uses an EAN-8 format and whose 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE is the code type that can use this encoding scheme.</p> <p>(summarize)</p>																												
<input type="checkbox"/> M_ENC_EAN13	<p>Specifies an encoding scheme for a composite code whose 1D portion uses an EAN-13 format and whose 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE is the code type that can use this encoding scheme.</p> <p>(summarize)</p>																												
<input type="checkbox"/> M_ENC_ISO8	<p>Specifies a similar encoding scheme as M_ENC_ASCII, but supports the extended ASCII character set.</p> <p>M_DATAMATRIX is the code type that can use this encoding scheme.</p>																												

	(summarize)
<input type="checkbox"/> M_ENC_MODE2	<p>Specifies an encoding scheme that requires a Structured Carrier Message. This encoding scheme can be used for a numeric postal code.</p> <p>M_MAXICODE is the code type that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_MODE3	<p>Specifies an encoding scheme that requires a Structured Carrier Message. This encoding scheme can be used for an alphanumeric postal code.</p> <p>M_MAXICODE is the code type that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_MODE4	<p>Specifies an encoding scheme that requires a Free Format Message.</p> <p>M_MAXICODE is the code type that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_MODE5	<p>Specifies an encoding scheme that requires a Free Format Message. This data type has a higher level of error correction than M_ENC_MODE4.</p> <p>M_MAXICODE is the code type that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_MODE6	<p>Specifies an encoding scheme that requires a Free Format Message. This encoding scheme indicates that the resulting code is used to program a code reader.</p> <p>M_MAXICODE is the code type that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_NUM	<p>Specifies an encoding scheme that only supports numbers.</p> <p>M_EAN13, M_DATAMATRIX, M_POSTNET, M_PLANET, M_UPC_A, M_UPC_E, M_PHARMACODE, M_EAN8, and M_INTERLEAVED25 are the code types that can use this encoding scheme. If the code type is M_DATAMATRIX, spaces are also allowed.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_QRCODE_MODEL1	<p>Specifies an encoding scheme that uses an older version of the QR code format.</p> <p>M_QRCODE is the code type that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_QRCODE_MODEL2	<p>Specifies an encoding scheme that uses a newer version of the QR code format. This version can handle more data and is more robust than model 1.</p> <p>M_QRCODE is the code type that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_RSS14	<p>Specifies an encoding scheme that uses an RSS-14 format.</p> <p>For a composite code, this encoding scheme specifies that the 1D portion uses an RSS-14 format and the 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE and M_RSSCODE are the code types that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_RSS14_STACKED	<p>Specifies an encoding scheme that uses an RSS-14 Stacked format.</p> <p>For a composite code, this encoding scheme specifies that the 1D portion uses an RSS-14 Stacked format and the 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE and M_RSSCODE are the code types that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_RSS14_STACKED_OMNI	<p>Specifies an encoding scheme that uses an RSS-14 Stacked Omni format.</p> <p>For a composite code, this encoding scheme specifies that the 1D portion uses an RSS-14 Stacked Omni format and the 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE and M_RSSCODE are the code types that can use this encoding scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENC_RSS14_TRUNCATED	<p>Specifies an encoding scheme that uses an RSS-14 Truncated format.</p>

	<p>For a composite code, this encoding scheme specifies that the 1D portion uses an RSS-14 Truncated format and the 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE and M_RSSCODE are the code types that can use this encoding scheme. (summarize)</p>		
<input type="checkbox"/> M_ENC_RSS_EXPANDED	<p>Specifies an encoding scheme that uses an RSS-14 Expanded format.</p> <p>For a composite code, this encoding scheme specifies that the 1D portion uses an RSS-14 Expanded format and the 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE and M_RSSCODE are the code types that can use this encoding scheme. (summarize)</p>		
<input type="checkbox"/> M_ENC_RSS_EXPANDED_STACKED	<p>Specifies an encoding scheme that uses an RSS-14 Expanded Stacked format.</p> <p>For a composite code, this encoding scheme specifies that the 1D portion uses an RSS-14 Expanded Stacked format and the 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE and M_RSSCODE are the code types that can use this encoding scheme. (summarize)</p>		
<input type="checkbox"/> M_ENC_RSS_LIMITED	<p>Specifies an encoding scheme that uses an RSS-14 Limited format.</p> <p>For a composite code, this encoding scheme specifies that the 1D portion uses an RSS-14 Limited format and the 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE and M_RSSCODE are the code types that can use this encoding scheme. (summarize)</p>		
<input type="checkbox"/> M_ENC_STANDARD	<p>Specifies different types of encoding schemes, depending on what code type is used.</p> <p>If M_CODE39 or M_CODE93 is the code type, M_ENC_STANDARD supports the alphabet (capital letters), digits 0 - 9, and the (-), space, (\$), (/), (+), (.), (%) characters.</p> <p>If M_BC412 is the code type, M_ENC_STANDARD supports numbers and the alphabet (capital letters), except for the letter O.</p> <p>If M_PDF417, M_TRUNCATED_PDF417, or M_MICROPDF417 is the code type, M_ENC_STANDARD supports ASCII and extended ASCII characters.</p> <p>If M_CODABAR is the code type, M_ENC_STANDARD supports digits 0 - 9, the (-), (\$), (:), (/), (.), and (+) characters, and the a, b, c, and d characters. It uses the latter four characters as start and stop characters. (summarize)</p>		
<input type="checkbox"/> M_ENC_UCCEAN128_MICROPDF417	<p>Specifies an encoding scheme for a composite code whose 1D portion uses an UCC/EAN-128 format and whose 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE is the code type that can use this encoding scheme. (summarize)</p>		
<input type="checkbox"/> M_ENC_UCCEAN128_PDF417	<p>Specifies an encoding scheme for a composite code whose 1D portion uses an UCC/EAN-128 format and whose 2D portion uses a PDF417 format.</p> <p>M_COMPOSITECODE is the code type that can use this encoding scheme. (summarize)</p>		
<input type="checkbox"/> M_ENC_UPCA	<p>Specifies an encoding scheme for a composite code whose 1D portion uses an UPC-A format and whose 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE is the code type that can use this encoding scheme. (summarize)</p>		
<input type="checkbox"/> M_ENC_UPCE	<p>Specifies an encoding scheme for a composite code whose 1D portion uses an UPC-E format and whose 2D portion uses a MicroPDF417 format.</p> <p>M_COMPOSITECODE is the code type that can use this encoding scheme. (summarize)</p>		
<input type="checkbox"/> M_ERROR_CORRECTION	<p>Sets the type of error correction. (summarize)</p>		
<input type="checkbox"/> M_DEFAULT	<p>Specifies to use the default error correction scheme for the code type. The following table lists the default error correction scheme:</p> <table> <tr> <td>BC412</td><td>M_ECC_NONE</td></tr> </table>	BC412	M_ECC_NONE
BC412	M_ECC_NONE		

Codabar	M_ECC_NONE
Code39	M_ECC_NONE
Code93	M_ECC_CHECK_DIGIT
Code128	M_ECC_CHECK_DIGIT
Composite	M_ECC_COMPOSITE
Data Matrix	M_ANY
Ean8	M_ECC_CHECK_DIGIT
Ean13	M_ECC_CHECK_DIGIT
Interleaved25	M_ECC_NONE
Maxicode	M_ECC_REED_SOLOMON
MicroPDF417	M_ECC_REED_SOLOMON
PDF417	M_ANY
Pharmacode	M_ECC_NONE
Planet	M_ECC_CHECK_DIGIT
Postnet	M_ECC_CHECK_DIGIT
QR	M_ANY
RSS	M_ECC_CHECK_DIGIT
Truncated PDF417	M_ANY
UPC-A	M_ECC_CHECK_DIGIT
UPC-E	M_ECC_CHECK_DIGIT

[\(summarize\)](#)

<input type="checkbox"/> M_ANY	<p>Detects the error correction type for read and verify operations automatically. This error correction scheme is not supported for write operations (McodeWrite()).</p> <p>M_DATAMATRIX, M_PDF417, M_TRUNCATED_PDF417 and M_QRCODE are the code types that can use this error correction scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ECC_200	<p>Uses a Reed Solomon-based algorithm as an error correction scheme.</p> <p>M_DATAMATRIX is the code type that can use this error correction scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ECC_CHECK_DIGIT	<p>Uses an additional digit to check whether there is an error or not. There is error detection but not correction.</p> <p>M_BC412, M_EAN13, M_CODE39, M_CODE93, M_POSTNET, M_PLANET, M_UPC_A, M_UPC_E, M_INTERLEAVED25, M_RSSCODE, M_EAN8 and M_CODE128 are the code types that can use this error correction scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ECC_COMPOSITE	<p>Uses the default error correction scheme for the 1D and 2D portions of the composite code.</p> <p>M_COMPOSITECODE is the code type that can use this error correction scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ECC_H	<p>Uses the highest-level error correction scheme.</p> <p>M_QRCODE is the code type that can use this error correction scheme.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ECC_L	<p>Uses the lowest-level error correction scheme.</p> <p>M_QRCODE is the code type that can use this error correction scheme.</p> <p>(summarize)</p>

<input type="checkbox"/> M_ECC_M	<p>Uses a lower-level error correction scheme.</p> <p>M_QRCODE is the code type that can use this error correction scheme. (summarize)</p>
<input type="checkbox"/> M_ECC_n	<p>Specifies an error correction algorithm that uses convolution coding. The value of n must be a multiple of 10, between 010 and 140, excluding 20 and 30; n must be represented by three digits.</p> <p>M_DATAMATRIX is the code type that can use this error correction scheme. If M_DATAMATRIX_SHAPE is set to M_RECTANGLE, the code must use M_ECC_200 error correction scheme instead of this one. (summarize)</p>
<input type="checkbox"/> M_ECC_NONE	<p>Specifies no error correction.</p> <p>M_DATAMATRIX, M_CODE39, M_INTERLEAVED25, M_CODABAR, M_PHARMACODE and M_BC412 are the code types that can use this error correction scheme. (summarize)</p>
<input type="checkbox"/> M_ECC_Q	<p>Uses a higher-level error correction scheme.</p> <p>M_QRCODE is the code type that can use this error correction scheme. (summarize)</p>
<input type="checkbox"/> M_ECC_REED_SOLOMON	<p>Uses a Reed Solomon type of error correction.</p> <p>M_MAXICODE and M_MICROPDF417 are the code types that can use this error correction scheme. (summarize)</p>
<input type="checkbox"/> M_ECC_REED_SOLOMON_n	<p>Uses a Reed Solomon type of error correction. n is an integer from 0 to 8. The higher the number, the higher the level of error correction.</p> <p>M_PDF417 and M_TRUNCATED_PDF417 are the code types that can use this error correction scheme. (summarize)</p>
<input type="checkbox"/> M_FOREGROUND_VALUE	<p>Sets the foreground color of the code. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_FOREGROUND_BLACK .
<input type="checkbox"/> M_FOREGROUND_ANY	<p>Specifies the foreground color as black or white. This value is available only for 1D code types (excluding Planet, Postnet, and Pharmacode) and MicroPDF417. Note that using this value will impact the performance of read and verify operations and should be used only when the foreground can change or cannot be known beforehand. When used for write operations, the default color (M_FOREGROUND_BLACK) is used. (summarize)</p>
<input type="checkbox"/> M_FOREGROUND_BLACK	Specifies that the foreground color is black.
<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that the foreground color is white.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeDraw

Synopsis

Draw specific features of results obtained from a code read or verify operation.

Syntax

```
void McodeDraw(  
    MIL_ID GraphContId,  
    MIL_ID CodeResultId,  
    MIL_ID DestImageId,  
    MIL_INT Operation,  
    MIL_INT ResultIndex,  
    MIL_INT ControlFlag  
)
```

Description

This function draws specific features of results, obtained from a code read or verify operation, in the specified destination image buffer. Results are only available after calling [McodeRead\(\)](#) or [McodeVerify\(\)](#).

Parameters

GraphContId

Specifies the graphics context to use when drawing. This parameter must be set to one of the following values:

● For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies to use the default graphics context of the current MIL application.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

CodeResultId

Specifies the code result buffer from which to extract features to draw. The code result buffer must have been previously allocated on the required system using [McodeAllocResult\(\)](#).

DestImageId

Specifies the identifier of the destination image buffer in which to write specific features of results. The specified destination image buffer must be an 8-bit unsigned buffer, allocated using [McodeAlloc\(\)](#).

In general, the destination buffer should be the same size as the image that you just read. When dealing with reflectance profiles (using [M_DRAW_REFLECTANCE_PROFILE](#)), determine the required width (using [McodeGetResult\(\)](#) or [McodeGetResultSingle\(\)](#) with [M_SCAN_REFLECTANCE_PROFILE_LENGTH](#)). For best results, the height of a scan reflectance profile buffer must be 256.

By drawing into the display's overlay buffer, you can also annotate an image non-destructively.

Operation

Specifies the type of operation to perform. Operations can be added together to draw multiple features at a time. For example, to draw both the result occurrence's position and a bounding box around the code, set the [Operation](#) parameter to [M_DRAW_POSITION](#) + [M_DRAW_BOX](#).

The following types of operations are only available after a read operation, with [McodeRead\(\)](#).

● For specifying the type of operation after a read operation	

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_BOX	Draws a box around the code occurrence. This box is drawn according to the corner coordinates of the code that was read. You can retrieve these coordinates using McodeGetResult() or McodeGetResultSingle() with M_BOTTOM... and M_TOP... . The bounding box only contains the part of the code that was processed in the read operation. When dealing with composite codes, the box is drawn around the 1D part of the code. When drawing an occurrence, the bounding box is drawn maintaining the angle and scale of the occurrence. (summarize)
<input type="checkbox"/> M_DRAW_CODE	Draws the specified code as it was read by McodeRead() . This operation is not supported for RSS, MicroPDF417, and composite code types. (summarize)
<input type="checkbox"/> M_DRAW_POSITION	Draws a cross-like symbol at the mid-point of the code. Note that for Data matrix codes, the cross symbol is drawn in the middle of the top left cell. For composite codes, the cross is drawn at the midpoint of the 1D part of the code. For Maxicodes, the cross appears in the center of the bull's eye pattern. Regardless of the code type, the cross is drawn maintaining the angle of the occurrence. (summarize)

The following type of operations are only available after a verify operation, with [McodeVerify\(\)](#).

● For specifying the type of operation after a verify operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_REFLECTANCE_PROFILE +	Draws the scan reflectance profile of the code, as analyzed by the verify operation. The result is not scaled to the size of the destination image buffer. (summarize)
<input type="checkbox"/> M_DRAW_SCAN_PROFILES +	Draws the scan profile of the code, as analyzed by the verify operation. The drawn scan profile represents only the regions from which the results are successfully read and computed. (summarize)

Combination constants for the values listed in [For specifying the type of operation after a verify operation](#)

You can add one of the following values to the above-mentioned values to get the verification result of a composite code.

● Composite code verification results	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_2D_COMPONENT	Draws the verification result of the 2D component of the composite code.
<input type="checkbox"/> M_LINEAR_COMPONENT	Draws the verification result of the linear component of the composite code.

ResultIndex

Specifies from which occurrence to get and draw results. Note that, for read operations, the occurrence index is ignored.

This parameter should be set to one of the following values:

● For specifying the index of the code result	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> M_ALL	Performs the specified operation for all code occurrences in the code result buffer.
<input type="checkbox"/> Value > = 0	Specifies the index of the code occurrence in the code result buffer.

ControlFlag

Specifies the function's control flag and should be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeFree

Synopsis

Free a code context or a code result buffer.

Syntax

```
void McodeFree(
    MIL_ID ObjectId
)
```

Description

This function deletes the specified code context (and all its models) or code result buffer identifier, and releases any memory associated with it.

To only delete individual models in the context, use [McodeModel\(\)](#) with [M_DELETE](#).

All code contexts and all code result buffers allocated on a particular system must be freed before the system can be freed.

Parameter

ObjectId

Specifies the identifier of the code context or code result buffer to free. These must have been successfully allocated (with [McodeAlloc\(\)](#) or [McodeAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeGetResult

Synopsis

Get the specified type of result for all occurrences of all code models in a code context, obtained from a code read or verify operation.

Syntax

```
void McodeGetResult(
    MIL_ID CodeResultId,
    MIL_INT ResultType,
    void *ResultPtr
)
```

Description

This function retrieves the result(s) of the specified type from a code result buffer for all occurrences of all code models in a code context. Results are only available after calling [McodeRead\(\)](#) or [McodeVerify\(\)](#).

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with [M_OUTPUT_UNITS](#) set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [ResultPtr](#) to [M_NULL](#). When this parameter is set to [M_NULL](#), the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [McodeGetResult\(\)](#) again and you pass an array to the parameter [ResultPtr](#). You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared. Note that the requests for [McodeGetResult\(\)](#) and [McodeGetResultSingle\(\)](#) are put in the same queue.

By setting the [ResultType](#) parameter to [M_INTERACTIVE](#), you can view the results interactively.

Parameters

CodeResultId

Specifies the identifier of the code result buffer from which to retrieve results.

ResultType

Specifies the type of result(s) to retrieve or opens a dialog box that displays the results stored in the result buffer.

Not all result types are stored in the result buffer for all operations; the availability of results depends on which operation has been performed. In the tables below, the result types are grouped according to the operations after which they are available.

To display the results currently stored in the result buffer in an interactive dialog box, select the following value:

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter [M_NULL](#).

● For displaying results in an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] Opens a dialog box containing a tree structure that displays the results stored in the result buffer, organized by codeword, scan, and general results.

To retrieve a result that is returned for both code read and verify operations, the [ResultType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For retrieving results from a code read or verify operation	
Value	Description
<input type="checkbox"/> M_CODE_MODEL_ID +	Retrieves the identifier of the code model that was used to read or verify the code. Note that this result is not valid when the results buffer is on a remote system. To find out which model was read on a remote system, use M_CODE_MODEL_INDEX instead. (summarize)
<input type="checkbox"/> M_CODE_MODEL_INDEX +	Retrieves the index of the code model, that was used to read or verify the code. Note that the number of codes read can be obtained using McodeGetResult() with M_NUMBER . When dealing with successful verify operations, the number of codes verified is always 1. (summarize)
	<i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of codes read or verified.
<input type="checkbox"/> M_CODE_TYPE +	Retrieves the code type of the code that was read.
<input type="checkbox"/> M_ENCODING +	Retrieves the type of encoding. Note that since some versions of the RSS code types only differ by the bar height (and not the structure), the same result is returned for these similar code types. RSS-14 and RSS-14 Truncated will return M_ENC_RSS14 . RSS-14 Stacked and RSS-14 Stacked Omnidirectional will return M_ENC_RSS14_STACKED . It is possible to obtain a more accurate result to distinguish between these structurally similar code types, using McodeControl() with M_POSITION_ACCURACY set to M_HIGH . (summarize)
	<i>ResultPtr info</i> Return values: M_ANY; M_ENC_ALPHA; M_ENC_ALPHANUM; M_ENC_ALPHANUM_PUNC; M_ENC_ASCII; M_ENC_EAN8; M_ENC_EAN13; M_ENC_ISO8; M_ENC_MODE2; M_ENC_MODE3; M_ENC_MODE4; M_ENC_MODE5; M_ENC_MODE6; M_ENC_NUM; M_ENC_QRCODE_MODEL1; M_ENC_QRCODE_MODEL2; M_ENC_RSS14; M_ENC_RSS_EXPANDED; M_ENC_RSS_EXPANDED_STACKED; M_ENC_RSS_LIMITED; M_ENC_RSS14_STACKED; M_ENC_RSS14_STACKED_OMNI; M_ENC_RSS14_TRUNCATED; M_ENC_STANDARD; M_ENC_UCCEAN128_MICROPDF417; M_ENC_UCCEAN128_PDF417; M_ENC_UPCA; M_ENC_UPCE; (details)
<input type="checkbox"/> M_ERROR_CORRECTION +	Retrieves the type of error correction. (summarize)
	<i>ResultPtr info</i> Return values: M_ANY; M_ECC_200; M_ECC_CHECK_DIGIT; M_ECC_COMPOSITE; M_ECC_H; M_ECC_L; M_ECC_M; M_ECC_NONE; M_ECC_Q; M_ECC_REED_SOLOMON; M_ECC_REED_SOLOMON_n; M_ECC_n; (details) M_ECC_UNKNOWN (unknown error correction type)
<input type="checkbox"/> M_NUMBER +	Retrieves the number of codes read for all code models in the code context, during the last read or verify operation. Note that multiple occurrences of the same model are counted separately. (summarize)
<input type="checkbox"/> M_NUMBER_OF_CODEWORDS +	Retrieves the number of codewords in the code; this includes data, overhead, and error correction codewords. A codeword is a group of bars and spaces, representing one or more encoded characters. This result type is available for composite codes and 2D code types, with the exception of Data Matrix code types that do not use M_ERROR_CORRECTION with M_ECC_200 . (summarize)
<input type="checkbox"/> M_NUMBER_OF_ERASURES +	Retrieves the number of erasures in the code. Erasures are missing or unreadable codewords at known positions. This result type is available for composite codes and 2D code types, with the exception of Data Matrix code types that do not use M_ERROR_CORRECTION with M_ECC_200 . (summarize)
<input type="checkbox"/> M_NUMBER_OF_ERROR_CORRECTION_CODEWORDS +	Retrieves the number of error correction codewords in the code. Error correction can be used to compensate for defects found during the decoding process. This result type is available for composite codes and 2D code types, with the exception of Data Matrix code types that do not use M_ERROR_CORRECTION with M_ECC_200 . (summarize)
<input type="checkbox"/> M_NUMBER_OF_ERRORS +	Retrieves the number of errors found in the code. Note that the number of errors does not include the number of erasures. This result type is available for composite codes and 2D code types, with the exception of Data Matrix code types that do not use M_ERROR_CORRECTION with M_ECC_200 . (summarize)
<input type="checkbox"/> M_STATUS +	Retrieves the status of a read or verify operation. It can be helpful to obtain the number of codes read using McodeGetResult() with M_NUMBER , along with inquiring about M_STATUS . For example, if you set a code context to read two code occurrences and only one was found, the status returned is M_STATUS_NOT_FOUND . Calling McodeGetResult() with

	<p>M_NUMBER will return 1.</p> <p>To retrieve the confidence score of a read operation, use M_SCORE. To retrieve the confidence score of a verify operation as a grade, use M_OVERALL_SYMBOL_GRADE.</p> <p>Note that when reading multiple code occurrences and more than one read operation fails, the least-common error is returned. For example, if you read two codes and one fails because the code was not found (M_STATUS_NOT_FOUND) and the other fails because the encoding type was unknown (M_STATUS_ENC_UNKNOWN), the latter is returned.</p> <p>(summarize)</p>																		
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <table> <tr> <td>M_STATUS_CRC_FAILED</td><td>Code read operation failed when validating the CRC (only for read operations).</td></tr> <tr> <td>M_STATUS_ECC_UNKNOWN</td><td>Unknown error correction type (only for read operations).</td></tr> <tr> <td>M_STATUS_ENC_UNKNOWN</td><td>Unknown encoding type (only for read operations).</td></tr> <tr> <td>M_STATUS_NOT_FOUND</td><td>Code was not found (only for read operations).</td></tr> <tr> <td>M_STATUS_NO_RESULT_AVAILABLE</td><td>There are no results available (for read and verify operations).</td></tr> <tr> <td>M_STATUS_READ_OK</td><td>Code read operation was successful (only for read operations).</td></tr> <tr> <td>M_STATUS_TIMEOUT_END</td><td>Code operation timed out (only for read and verify operations).</td></tr> <tr> <td>M_STATUS_VERIFY_FAILED</td><td>Code verify operation failed (only for verify operations).</td></tr> <tr> <td>M_STATUS_VERIFY_OK</td><td>Code verify was successful (only for verify operations).</td></tr> </table>	M_STATUS_CRC_FAILED	Code read operation failed when validating the CRC (only for read operations).	M_STATUS_ECC_UNKNOWN	Unknown error correction type (only for read operations).	M_STATUS_ENC_UNKNOWN	Unknown encoding type (only for read operations).	M_STATUS_NOT_FOUND	Code was not found (only for read operations).	M_STATUS_NO_RESULT_AVAILABLE	There are no results available (for read and verify operations).	M_STATUS_READ_OK	Code read operation was successful (only for read operations).	M_STATUS_TIMEOUT_END	Code operation timed out (only for read and verify operations).	M_STATUS_VERIFY_FAILED	Code verify operation failed (only for verify operations).	M_STATUS_VERIFY_OK	Code verify was successful (only for verify operations).
M_STATUS_CRC_FAILED	Code read operation failed when validating the CRC (only for read operations).																		
M_STATUS_ECC_UNKNOWN	Unknown error correction type (only for read operations).																		
M_STATUS_ENC_UNKNOWN	Unknown encoding type (only for read operations).																		
M_STATUS_NOT_FOUND	Code was not found (only for read operations).																		
M_STATUS_NO_RESULT_AVAILABLE	There are no results available (for read and verify operations).																		
M_STATUS_READ_OK	Code read operation was successful (only for read operations).																		
M_STATUS_TIMEOUT_END	Code operation timed out (only for read and verify operations).																		
M_STATUS_VERIFY_FAILED	Code verify operation failed (only for verify operations).																		
M_STATUS_VERIFY_OK	Code verify was successful (only for verify operations).																		
<input type="checkbox"/> M_TIMEOUT_END +	<p>Retrieves whether the timeout was reached. You can set the timeout limit using McodeControl() with M_TIMEOUT.</p> <p>(summarize)</p>																		
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <table> <tr> <td>M_FALSE</td><td>Specifies that the timeout limit was not reached.</td></tr> <tr> <td>M_TRUE</td><td>Specifies that the timeout limit was reached.</td></tr> </table>	M_FALSE	Specifies that the timeout limit was not reached.	M_TRUE	Specifies that the timeout limit was reached.														
M_FALSE	Specifies that the timeout limit was not reached.																		
M_TRUE	Specifies that the timeout limit was reached.																		

To retrieve a result that is returned for a code read operation, the [ResultType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter the address of a `MIL_DOUBLE` or the address of an array of type `MIL_DOUBLE` with a size equal to the number of codes read. This number can be obtained using [M_NUMBER](#) (when more than one code is read).

● For retrieving results from a code read operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ANGLE +	<p>Retrieves the angle at which the code was read.</p> <p>For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY.</p> <p>(summarize)</p>
<input type="checkbox"/> M_BOTTOM_LEFT_X +	<p>Retrieves the X-coordinate of the bottom-left corner of the code that was read.</p> <p>For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY.</p> <p>(summarize)</p>
<input type="checkbox"/> M_BOTTOM_LEFT_Y +	<p>Retrieves the Y-coordinate of the bottom-left corner of the code that was read.</p> <p>For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY.</p> <p>(summarize)</p>
<input type="checkbox"/> M_BOTTOM_RIGHT_X +	<p>Retrieves the X-coordinate of the bottom-right corner of the code that was read.</p> <p>For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY.</p> <p>(summarize)</p>
<input type="checkbox"/> M_BOTTOM_RIGHT_Y +	<p>Retrieves the Y-coordinate of the bottom-right corner of the code that was read.</p> <p>For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY.</p> <p>(summarize)</p>
<input type="checkbox"/> M_CELL_NUMBER_X +	<p>Retrieves the number of cells in the X-direction.</p>

M_CELL_NUMBER_Y +	Retrieves the number of cells in the Y-direction.
M_CELL_SIZE +	Retrieves the size of the cell in X (module size (Z)) of the code that was read.
M_FOREGROUND_VALUE +	Retrieves the foreground color of the code that was read. (summarize)
	<i>ResultPtr info</i> Return values: Please see McodeControl() with M_FOREGROUND_BLACK . (details) Please see McodeControl() with M_FOREGROUND_WHITE . (details)
M_POSITION_X +	Retrieves the X-coordinate of the occurrence. For all types of codes, except the Data Matrix and composite code types, this result is the center of the code in the X-direction. For Data Matrix codes, this result is the center, in the X-direction, of the top-left cell. For composite codes, this result is the center, in the X-direction, of the 1D component of the code. Note that to retrieve the top-left and bottom-right coordinates, use M_BOTTOM... and M_TOP... For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
M_POSITION_Y +	Retrieves the Y-coordinate of the occurrence. For all types of codes, except the Data Matrix and composite code types, this result is the center of the code in the Y-direction. For Data Matrix codes, this result is the center, in the Y-direction, of the top-left cell. For composite codes, this result is the center, in the Y-direction, of the 1D component of the code. Note that to retrieve the top-left and bottom-right coordinates, use M_BOTTOM... and M_TOP... For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
M_RECOMMENDED_APERTURE_SIZE +	Retrieves the recommended aperture size for the verify operation, following the ISO 15416 specification. Note that this value depends on McodeControl() with M_PIXEL_SIZE_IN_MM . If M_PIXEL_SIZE_IN_MM is not set (or is set to M_UNKNOWN), the value returned is (0.5 x cell size) . If the M_PIXEL_SIZE_IN_MM is set to a value, the recommended aperture size is computed using the ISO 15416 specification. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)
M_SCORE +	Retrieves the confidence score of a read operation. The code type determines how this value is calculated. <ul style="list-style-type: none"> For linear 1D and RSS code types, the score is based on: <ul style="list-style-type: none"> The number of redundant scan lines. Redundant scan lines can be omitted from a code, while still yielding the same result. The difference between the actual code and theoretical code model. For 1D Planet and Postnet code types, the score is 1 if the code was read and 0 if it was not. For 2D code types, the score is calculated using: $((\text{Number of errors} + \text{Number of erasures}) / (\text{Number of codewords}))$. If the code could not be decoded successfully, the score is 0. For composite codes, the score is the score of either the 1D part or the 2D part; whichever was lower. To retrieve the error status of the verify operation, use M_STATUS . To retrieve the confidence score of a verify operation as a grade, use M_OVERALL_SYMBOL_GRADE . (summarize)
	<i>ResultPtr info</i> Return values: 0 <= Value <= 1 Specifies the confidence score of a read operation. 0 indicates 0% confidence and 1 indicates 100% confidence.
M_SIZE_X +	Retrieves the size of the code in the X-direction. For composite code types, this result is the width of the 1D component of the code. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
M_SIZE_Y +	Retrieves the size of the code in the Y-direction. For composite code types, this result is the height of the 1D component of the code. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
M_STRING +	Retrieves the decoded string. If the string is decoded from a composite code model, the string is returned in the following format: 1D string 2D string, where is used as the separator. (summarize)
	<i>ResultPtr info</i> Data type: array of type MIL_TEXT_CHAR Array size: Large enough to contain the decoded string (using either M_STRING_SIZE +1 (if reading one code), or M_TOTAL_STRING_SIZE (if reading multiple

	codes).
M_STRING_SIZE +	Retrieves the length of the decoded string.
M_THRESHOLD +	Retrieves the threshold used to internally binarize the source image of a read operation. (summarize)
	<i>ResultPtr info</i> Return values: 0<=Value<=255; M_ADAPTIVE; (details)
M_TOP_LEFT_X +	Retrieves the X-coordinate of the top-left corner of the code that was read. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
M_TOP_LEFT_Y +	Retrieves the Y-coordinate of the top-left corner of the code that was read. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
M_TOP_RIGHT_X +	Retrieves the X-coordinate of the top-right corner of the code that was read. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
M_TOP_RIGHT_Y +	Retrieves the Y-coordinate of the top-right corner of the code that was read. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
M_TOTAL_STRING_SIZE +	Retrieves the total length of all decoded strings. This is equal to the sum of the individual decoded string lengths (including the null terminator character of each string). (summarize)
	<i>ResultPtr info</i> Data type: MIL_DOUBLE

Combination constant for [M_STRING](#); [M_STRING_SIZE](#); [M_TOTAL_STRING_SIZE](#);

You can add the following value to the above-mentioned values to specify what type of symbols should be returned when dealing with unprintable characters.

● For use with string result types	
Value	Description
M_ESCAPE_SEQUENCE	Retrieves the string with its unprintable characters represented by their ASCII character codes, when combined with M_STRING . Any back slashes (\) in the string are doubled, while unprintable characters, such as carriage returns, are put in a \xNN format, where NN is the hexadecimal value of the ASCII character code for the unprintable character. When combined with M_STRING_SIZE , it returns the size of the string including its unprintable characters. (summarize)

To retrieve a result that is returned for a code verify operation, the [ResultType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter the address of a MIL_DOUBLE.

● For retrieving results from a code verify operation	
Value	Description
M_AXIAL_NONUNIFORMITY +	Retrieves the axial non-uniformity of the code. The axial non-uniformity is calculated using the following formula: $\frac{\text{abs}(X_{avg} - Y_{avg})}{0.5 \times (X_{q10} + Y_{q10})}$ Where X_{avg} is the average cell size in the X-dimension and Y_{avg} is the average measurement cell size in the Y-dimension. When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).

	<p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <p>0 <= Value <= 1 The axial nonuniformity measure.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_AXIAL_NONUNIFORMITY_GRADE +	<p>Retrieves the axial nonuniformity of the code, as a grade.</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_CODEWORD_DECODABILITY +	<p>Retrieves the measure of the print quality of each codeword relative to the decoding algorithm. Note that this excludes any start/stop patterns.</p> <p>To retrieve the decodability of the start/stop patterns of your code, use M_SCAN_DECODABILITY.</p> <p>This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS.</p>
<input type="checkbox"/> M_CODEWORD_DECODABILITY_GRADE +	<p>Retrieves the decodability of each codeword in the code, as a grade.</p> <p>To retrieve the decodability of the start/stop patterns of your code as a grade, use M_SCAN_DECODABILITY_GRADE.</p> <p>This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_CODEWORD_DEFECTS +	<p>Retrieves and ranks the deviations in the expected signal that denotes a codeword in your code. The larger the result, the greater the defect in the codeword and the less likely that the codeword can be decoded without error.</p> <p>To retrieve the defects for your code as a single grade use M_DEFECTS_GRADE. To retrieve the defects of the start/stop patterns use M_SCAN_DEFECTS.</p> <p>This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D cross-row and composite code types.</p>

	<p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS.</p>												
<p>☐ M_CODEWORD_DEFECTS_GRADE +</p>	<p>Retrieves the codeword defects, as a grade for each codeword. To retrieve the defects for your code as a single grade use M_DEFECTS_GRADE. To retrieve the defects of the start/stop patterns use M_SCAN_DEFECTS_GRADE. This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification. This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS. Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<p>☐ M_CODEWORD_MODULATION +</p>	<p>Retrieves the modulation (MOD) of each codeword. Modulation is the ratio of the minimum edge contrast to symbol contrast within the code. To retrieve the modulation from the start/stop patterns, use M_SCAN_MODULATION. This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification. When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE). This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS.</p>												
<p>☐ M_CODEWORD_MODULATION_GRADE +</p>	<p>Retrieves the modulation (MOD) of each codeword, as a grade. To retrieve the modulation for your code as a single grade use M_MODULATION_GRADE. To retrieve the modulation from the start/stop patterns, as a grade, use M_SCAN_MODULATION_GRADE. This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification. This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS. Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<p>☐ M_CODEWORD_YIELD +</p>	<p>Retrieves the codeword yield result. The codeword yield result determines how well your code can be read at an angle relative to the horizontal and vertical axis of code. When all other results are good, a poor codeword yield result can indicate a problem along the Y-axis of your code. For more information, refer to the ISO/IEC 15415 specification. When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE). This result type is available for 2D cross-row and composite code types.</p>												

	(summarize)
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <p>0 <= Value <= 1 The codeword yield.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_CODEWORD_YIELD_GRADE +	<p>Retrieves the codeword yield, as a grade.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_DECODABILITY_GRADE +	<p>Retrieves the measure of the print quality of the code relative to the decoding algorithm, as a grade.</p> <p>To determine the decodability for each codeword in the code, use M_CODEWORD_DECODABILITY. To retrieve the decodability of the start/stop patterns of your code as a grade, use M_SCAN_DECODABILITY_GRADE.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_DECODE_GRADE +	<p>Retrieves the code decodability, as a grade.</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A The code can be read.</p> <p>M_CODE_GRADE_F The code cannot be read.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_DEFECTS_GRADE +	<p>Retrieves the measure of defects in the code, as a grade.</p> <p>To retrieve the measure of defects for each codeword as a grade, use M_CODEWORD_DEFECTS_GRADE. To retrieve the measure of defects in the start/stop patterns as a grade, use M_SCAN_DEFECTS_GRADE.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p>

	<p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_MODULATION_GRADE +	<p>Retrieves the codeword print quality assessment of the modulation, as a grade. For more information, refer to the ISO/IEC 15415 specification.</p> <p>To retrieve the modulation of each codeword as a grade, use M_CODEWORD_MODULATION_GRADE. To retrieve the modulation for the start/stop patterns as a grade, use M_SCAN_MODULATION_GRADE.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p> </div>
M_NUMBER_OF_ROWS +	<p>Retrieves the number of rows verified in the code.</p> <p>For RSS code types, the number of rows is the number of stacks in the code. For cross-row code types, the number of rows is the number of rows verified in the start/stop pattern grading.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
M_NUMBER_OF_SCANS +	<p>Retrieves the total number of scan reflectance profiles analyzed in the code. This is the sum of the number of scans per row, for every row within the code. The number of scans determines the number of results within an array for all the M_SCAN ... values described in this function.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
M_NUMBER_OF_SCANS_PER_ROW +	<p>Retrieves the number of scan reflectance profiles per row. The number of scans determines the number of results within an array for all the M_SCAN ... values described in this function.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
M_OVERALL_SYMBOL_GRADE +	<p>Retrieves the overall symbol grade. The code type determines the calculation used to obtain the grade.</p> <p>For 1D code types, the overall symbol grade is the average grade of the scan reflectance profile (using M_SCAN_REFLECTANCE_PROFILE_GRADE).</p> <p>For composite code types, the overall symbol grade is the worst grade overall, obtained when analyzing either the 1D or the 2D portions.</p> <p>For 2D code types, the overall symbol grade is the worst of all the returned grades.</p> <p>For cross-row codes, the overall symbol grade is the worst of all possible grades (with the exception of M_SCAN...GRADE).</p> <p>To retrieve the overall symbol grade for each row in the code, use M_ROW_OVERALL_GRADE.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p> </div>
M_PRINT_GROWTH +	<p>Retrieves the print growth value. The print growth is a value determining whether the graphical features of the code have either grown or shrunk relative to theoretical model sufficiently to hinder readability.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not</p>

	<p>(M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p> <p>Value The print growth value.</p>
<input type="checkbox"/> M_PRINT_GROWTH_GRADE +	<p>Retrieves the print growth value, as a grade.</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_ROW_OVERALL_GRADE +	<p>Retrieves the overall grade for each row in the code.</p> <p>To retrieve the average of all the row overall grades, use M_OVERALL_SYMBOL_GRADE.</p> <p>For 1D code types, the overall row grade is the overall grade for the entire code.</p> <p>For 2D code types, the overall row grade is the overall grade per stack of the code.</p> <p>For cross-row code types, the overall row grade is the worst grade from the row.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of rows in the code. This number can be obtained using McodeGetResult() with M_NUMBER_OF_ROWS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_SCAN_DECODABILITY +	<p>Retrieves the decodability result of the scan reflectance profile.</p> <p>To retrieve the decodability for each codeword, use M_CODEWORD_DECODABILITY. To retrieve the decodability for your code as a grade, use M_DECODABILITY_GRADE.</p> <p>Note that certain code types have additional decodability measures that are taken into account when computing the decodability result. See the ISO 15417 (Code 128) and ISO 15420 (EAN and UPC code types) specifications.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>0 <= Value <= 1 The decodability parameter.</p>

<div> <div></div> <div>M_SCAN_DECODABILITY_GRADE +</div> </div>	<p>Retrieves the decodability result of the scan reflectance profile, as a grade.</p> <p>To retrieve the decodability for each codeword as a grade, use M_CODEWORD_DECODABILITY_GRADE. To retrieve the decodability for your cross-row code as a grade, use M_DECODABILITY_GRADE.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: <div> <div>M_CODE_GRADE_A</div> <div>Best grade for the result.</div> </div> <div> <div>M_CODE_GRADE_B</div> <div>Good grade for the result.</div> </div> <div> <div>M_CODE_GRADE_C</div> <div>Fair grade for the result.</div> </div> <div> <div>M_CODE_GRADE_D</div> <div>Poor grade for the result.</div> </div> <div> <div>M_CODE_GRADE_F</div> <div>Worst grade for the result.</div> </div> <div> <div>M_CODE_GRADE_NOT_AVAILABLE</div> <div>The result is not available for the code.</div> </div> </div> </div>
<div> <div></div> <div>M_SCAN_DECODE_GRADE +</div> </div>	<p>Retrieves whether the decoding algorithm succeeded or failed, as a grade.</p> <p>To retrieve the error status of the verify operation, use M_STATUS. To retrieve the confidence score of a read operation, use M_SCORE.</p> <p>Note that M_SCAN_EDGE_DETERMINATION_GRADE is used in calculating this value.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: <div> <div>M_CODE_GRADE_A</div> <div>The decoding algorithm succeeded.</div> </div> <div> <div>M_CODE_GRADE_F</div> <div>The decoding algorithm failed.</div> </div> <div> <div>M_CODE_GRADE_NOT_AVAILABLE</div> <div>The result is not available for the code.</div> </div> </div> </div>
<div> <div></div> <div>M_SCAN_DEFECTS +</div> </div>	<p>Retrieves and ranks the defects in the scan reflectance profile. The result is based on the ERN maximum (M_SCAN_ERN_MAXIMUM) and symbol contrast (M_SCAN_SYMBOL_CONTRAST).</p> <p>To retrieve the ranking measure for each codeword, use M_CODEWORD_DEFECTS. To determine the measure of defects in the code as a grade, use M_DEFECTS_GRADE.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: <div> <div>0 <= Value <= 1</div> <div>The ranking of the defects.</div> </div> </div> </div>
<div> <div></div> <div>M_SCAN_DEFECTS_GRADE +</div> </div>	<p>Retrieves the measure of defects of the scan reflectance profile, as a grade.</p> <p>To retrieve the ranking measure of defects in each codeword, as a grade, use M_CODEWORD_DEFECTS_GRADE. To determine the measure of defects in the code, use M_DEFECTS_GRADE.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: <div> <div>M_CODE_GRADE_A</div> <div>Best grade for the result.</div> </div> <div> <div>M_CODE_GRADE_B</div> <div>Good grade for the result.</div> </div> </div> </div>

	<p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SCAN_EDGE_CONTRAST_MINIMUM +	<p>Retrieves the minimum edge contrast of the scan reflectance profile.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>0 <= Value <= 1 The minimum edge contrast.</p>
M_SCAN_EDGE_CONTRAST_MINIMUM_GRADE +	<p>Retrieves the minimum edge contrast, as a grade.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SCAN_EDGE_DETERMINATION_GRADE +	<p>Retrieves the edge determination, as a grade. The edge determination examines the number of bars and spaces in your code. See the ANSI X3.182 - 1990 specification. Note that this grade is not considered when calculating the overall code grade (M_OVERALL_SYMBOL_GRADE), instead it is included in the decode grade (M_SCAN_DECODE_GRADE).</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SCAN_ERN_MAXIMUM +	<p>Retrieves the highest ERN (element reflectance non-uniformity) in the scan reflectance profile. The ERN is an unexpected peak in the scan reflectance profile. The result is expressed as the ratio of the maximum ERN relative to the symbol contrast (ERN^{max} / SC).</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p>

	<p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>0 <= Value <= 1 The element reflectance non-uniformity maximum.</p>												
<input type="checkbox"/> M_SCAN_GUARD_PATTERN +	<p>Retrieves the size of the interior guard pattern, expressed as a factor of the cell (module size (<i>Z</i>)) in the scan reflectance profile. Guard patterns consist of two one-cell wide groupings at the end of each code. RSS code types have guard patterns at the ends of each row of the code. See the ISO 24724 specification. Note that the value is negative if the left guard pattern is greater than the right guard pattern, to permit easy identification.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 1D RSS code types and composite code types containing a RSS code.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p>												
<input type="checkbox"/> M_SCAN_GUARD_PATTERN_GRADE +	<p>Retrieves the largest interior guard pattern, as a grade.</p> <p>This result type is available for 1D RSS code types and composite code types containing a RSS code.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<input type="checkbox"/> M_SCAN_INTERCHARACTER_GAP +	<p>Retrieves the size of the largest intercharacter gap in the scan reflectance profile, expressed as a factor of the cell size (module size (<i>Z</i>)) in the scan reflectance profile. See ISO 16388 and AIM BC3-1995 for more details.</p> <p>Note that this result type will always return the value of M_CODE_GRADE_NOT_AVAILABLE when McodeControl() with M_PIXEL_SIZE_IN_MM is not set, and the following condition is true:</p> <p>(5.3 <i>Z</i> <= Largest intercharacter gap <= 3 <i>Z</i>).</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for Code 39 and Codabar code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <table> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> <tr> <td>Value</td><td>The maximum intercharacter gap.</td></tr> </table>	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.	Value	The maximum intercharacter gap.								
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
Value	The maximum intercharacter gap.												
<input type="checkbox"/> M_SCAN_INTERCHARACTER_GAP_GRADE +	<p>Retrieves the largest intercharacter gap, as a grade.</p> <p>This result type is available for Code 39 and Codabar code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>The maximum intercharacter gap is under three times the module size (3Z).</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>The maximum intercharacter gap is over 5.3 times the module size (5.3Z).</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The maximum intercharacter gap cannot be calculated because the M_PIXEL_SIZE_IN_MM is not set.</td></tr> </table>	M_CODE_GRADE_A	The maximum intercharacter gap is under three times the module size (3Z).	M_CODE_GRADE_F	The maximum intercharacter gap is over 5.3 times the module size (5.3Z).	M_CODE_GRADE_NOT_AVAILABLE	The maximum intercharacter gap cannot be calculated because the M_PIXEL_SIZE_IN_MM is not set.						
M_CODE_GRADE_A	The maximum intercharacter gap is under three times the module size (3Z).												
M_CODE_GRADE_F	The maximum intercharacter gap is over 5.3 times the module size (5.3Z).												
M_CODE_GRADE_NOT_AVAILABLE	The maximum intercharacter gap cannot be calculated because the M_PIXEL_SIZE_IN_MM is not set.												

<input type="checkbox"/> M_SCAN_MODULATION +	<p>Retrieves the modulation (MOD) of the scan reflectance profile. The modulation is the ratio of the minimum edge contrast (M_SCAN_EDGE_CONTRAST_MINIMUM) to the symbol contrast (M_SCAN_SYMBOL_CONTRAST).</p> <p>To retrieve the modulation of each codeword, use M_CODEWORD_MODULATION.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <hr/> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>0 <= Value <= 1 The modulation parameter.</p>
<input type="checkbox"/> M_SCAN_MODULATION_GRADE +	<p>Retrieves the modulation (MOD) of the scan reflectance profile, as a grade.</p> <p>To retrieve the modulation of each codeword as a grade, use M_CODEWORD_MODULATION_GRADE. To retrieve the modulation for the code as a grade, use M_MODULATION_GRADE.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <hr/> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_SCAN_PRINT_CONTRAST_SIGNAL +	<p>Retrieves the print contrast signal (PCS) of the scan reflectance profile. Note that this is an approximation based on the maximum reflectance (M_SCAN_REFLECTANCE_MAXIMUM) and the code contrast (M_SCAN_SYMBOL_CONTRAST). For more details, see the ISO 15416 specification.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <hr/> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>0 <= Value <= 1 The print contrast signal.</p>
<input type="checkbox"/> M_SCAN_PROFILE_END_X +	<p>Retrieves the X-coordinate of the end of the scan reflectance profile.</p> <p>This result type relates to the position of the scan profiles, enabling you to visualize or draw the available analysis profiles using McodeDraw() with M_DRAW_SCAN_PROFILES. Note that the start and stop positions of any given scan are related to the steps of the analysis were performed.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
<input type="checkbox"/> M_SCAN_PROFILE_END_Y +	<p>Retrieves the Y-coordinate of the end of the scan reflectance profile.</p> <p>This result type relates to the position of the scan profiles, enabling you to visualize or draw the available analysis profiles using McodeDraw() with M_DRAW_SCAN_PROFILES. Note that the start and stop positions of any given scan are related to the steps of the analysis were performed.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
<input type="checkbox"/> M_SCAN_PROFILE_START_X +	<p>Retrieves the X-coordinate of the start of the scan reflectance profile.</p> <p>This result type relates to the position of the scan profiles, enabling you to visualize or draw the available analysis profiles using McodeDraw() with M_DRAW_SCAN_PROFILES. Note that the start and stop positions of any given scan are related to the steps of the analysis were performed.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
<input type="checkbox"/> M_SCAN_PROFILE_START_Y +	<p>Retrieves the Y-coordinate of the start of the scan reflectance profile.</p> <p>This result type relates to the position of the scan profiles, enabling you to visualize or draw the available analysis profiles using McodeDraw() with M_DRAW_SCAN_PROFILES. Note that the start and stop positions of any given scan are related to the steps of the analysis were performed.</p>

	<p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
<input type="checkbox"/> M_SCAN_QUIET_ZONE +	<p>Retrieves the ratio between the expected quiet zone from theoretical code model and the measured quiet zone of your code as a ratio of (<i>measured quiet zone size / expected quiet zone size</i>).</p> <p>The measurement is taken from both sides of the code. Note that if the returned value is negative then the left quiet zone is worse than the right. Refer to the appropriate specifications for more details (ISO 15417, ISO 15420, ISO 16388, and ISO 16390, respectively).</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 1D code types (except RSS) and composite code types encoded with a format of EAN/UPC or UCC/EAN120.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available.</p> <p>Value The scan quiet zone vale.</p> </div>
<input type="checkbox"/> M_SCAN_QUIET_ZONE_GRADE +	<p>Retrieves the quiet zone size, as a grade.</p> <p>This result type is available for 1D code types (except RSS) and composite code types encoded with a format of EAN/UPC or UCC/EAN120.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p> </div>
<input type="checkbox"/> M_SCAN_REFLECTANCE_MAXIMUM +	<p>Retrieves the highest reflectance (R_{max}) of the scan reflectance profile, as a percentage.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>0 <= Value <= 1 The maximum reflectance.</p> </div>
<input type="checkbox"/> M_SCAN_REFLECTANCE_MINIMUM +	<p>Retrieves the lowest reflectance (R_{min}) of the scan reflectance profile, as a percentage.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>0 <= Value <= 1 The minimum reflectance.</p> </div>
<input type="checkbox"/> M_SCAN_REFLECTANCE_MINIMUM_GRADE +	<p>Retrieves a pass or fail grade for the scan reflectance profile.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> </div>

	<p>Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: M_CODE_GRADE_A The scan reflectance profile passed. M_CODE_GRADE_F The scan reflectance profile failed. M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SCAN_REFLECTANCE_PROFILE_GRADE +	<p>Retrieves the lowest grade from all the scan reflectance profile validation tests. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i> Return values: M_CODE_GRADE_A Best grade for the result. M_CODE_GRADE_B Good grade for the result. M_CODE_GRADE_C Fair grade for the result. M_CODE_GRADE_D Poor grade for the result. M_CODE_GRADE_F Worst grade for the result. M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SCAN_REFLECTANCE_PROFILE_LENGTH +	<p>Retrieves the number of values in the scan reflectance profile. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p>
M_SCAN_REFLECTANCE_PROFILE_VALUES +	<p>Retrieves the reflectance values of the scan reflectance profile. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of values in the scan reflectance profile. This number can be obtained using McodeGetResult() with M_SCAN_REFLECTANCE_PROFILE_LENGTH.</p>
M_SCAN_SYMBOL_CONTRAST +	<p>Retrieves the code (symbol) contrast (<i>SC</i>) of the scan reflectance profile. Note that this value is used in conjunction with the maximum reflectance (M_SCAN_REFLECTANCE_MAXIMUM) to determine the print contrast signal (M_SCAN_PRINT_CONTRAST_SIGNAL). For more details, see ISO 15416. When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE). When dealing with 2D matrix codes, to retrieve the code contrast for the entire code, use M_SYMBOL_CONTRAST. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The sum of the length of the scan profiles. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: 0 <= Value <= 1 The symbol contrast.</p>
M_SCAN_SYMBOL_CONTRAST_GRADE +	<p>Retrieves the code contrast of the scan reflectance profile, as a grade. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: M_CODE_GRADE_A Best grade for the result.</p>

	<p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SCAN_WIDE_TO_NARROW_RATIO +	<p>Retrieves the ratio between the average of the widest and the average of the narrowest bar/space in the scan reflectance profile. See ISO 16388 and ISO 16390 for more details.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for Code 39, Codebar, and Interleaved 2/5.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p>
M_SCAN_WIDE_TO_NARROW_RATIO_GRADE +	<p>Retrieves wide to narrow ratio, as a grade.</p> <p>This result type is available for Code 39, Codebar, and Interleaved 2/5.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_START_STOP_PATTERN_GRADE +	<p>Retrieves the average of the worst scan reflectance profile grades of each row, as a grade. A start/stop pattern is a set pattern that marks the end points of the code. For each scan reflectance profile, the start/stop pattern grade is computed for each verified row, then the average start/stop pattern grade for all the scan reflectance profiles is computed.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SYMBOL_CONTRAST +	<p>Retrieves the code (symbol) contrast (SC) value for the code. In the case of matrix code types, the code contrast is calculated as the average 10% of the lighter pixels minus the average 10% of the darker pixels.</p> <p>To retrieve the code contrast from scan reflectance profiles, use M_SCAN_SYMBOL_CONTRAST.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Return values:</p>

	<p>0 <= Value <= 1 The SC value.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>												
M_SYMBOL_CONTRAST_GRADE +	<p>Retrieves the code (symbol) contrast (SC) value for the code as a grade. The higher the contrast, the better it is for scanning purposes and the better the grade. To retrieve the code contrast of the scan reflectance profile as a grade, use M_SCAN_SYMBOL_CONTRAST_GRADE. This result type is available for 2D matrix code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
M_UNUSED_ERROR_CORRECTION +	<p>Retrieves the ratio of unused error correction to the total amount of error correction available for the code. This tests the extent to which regional or spot damage in the code has eroded the reading safety margin that error correction provides. When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result is available for all 2D and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of cells scanned. This number can be obtained using McodeGetResult() with M_CELL_NUMBER_X. Return values:</p> <table> <tr> <td>0 <= Value <= 1</td><td>The ratio of unused error correction within the code.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	0 <= Value <= 1	The ratio of unused error correction within the code.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.								
0 <= Value <= 1	The ratio of unused error correction within the code.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
M_UNUSED_ERROR_CORRECTION_GRADE +	<p>Retrieves the ratio of unused error correction to the total amount of error correction available for the code, as a grade. This result is available for all 2D and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												

Combination constants for the values listed in [For retrieving results from a code read or verify operation](#); and for the following values: [M_CODEWORD_DECODABILITY](#); [M_CODEWORD_DECODABILITY_GRADE](#); [M_CODEWORD_DEFECTS](#); [M_CODEWORD_DEFECTS_GRADE](#); [M_CODEWORD_MODULATION](#); [M_CODEWORD_MODULATION_GRADE](#); [M_CODEWORD_YIELD](#); [M_CODEWORD_YIELD_GRADE](#); [M_DECODABILITY_GRADE](#); [M_DECODE_GRADE](#); [M_DEFECTS_GRADE](#); [M_MODULATION_GRADE](#); [M_NUMBER_OF_ROWS](#); [M_NUMBER_OF_SCANS](#); [M_NUMBER_OF_SCANS_PER_ROW](#); [M_OVERALL_SYMBOL_GRADE](#); [M_ROW_OVERALL_GRADE](#); [M_SCAN_DECODABILITY](#); [M_SCAN_DECODABILITY_GRADE](#); [M_SCAN_DECODE_GRADE](#); [M_SCAN_DEFECTS](#); [M_SCAN_DEFECTS_GRADE](#); [M_SCAN_EDGE_CONTRAST_MINIMUM](#); [M_SCAN_EDGE_CONTRAST_MINIMUM_GRADE](#); [M_SCAN_EDGE_DETERMINATION_GRADE](#); [M_SCAN_ERN_MAXIMUM](#); [M_SCAN_GUARD_PATTERN](#); [M_SCAN_GUARD_PATTERN_GRADE](#); [M_SCAN_MODULATION](#); [M_SCAN_MODULATION_GRADE](#); [M_SCAN_PRINT_CONTRAST_SIGNAL](#); [M_SCAN_PROFILE_START_X](#); [M_SCAN_PROFILE_START_Y](#); [M_SCAN_PROFILE_END_X](#); [M_SCAN_PROFILE_END_Y](#); [M_SCAN_QUIET_ZONE](#); [M_SCAN_QUIET_ZONE_GRADE](#); [M_SCAN_REFLECTANCE_MAXIMUM](#); [M_SCAN_REFLECTANCE_MINIMUM](#); [M_SCAN_REFLECTANCE_MINIMUM_GRADE](#); [M_SCAN_REFLECTANCE_PROFILE_GRADE](#); [M_SCAN_REFLECTANCE_PROFILE_LENGTH](#); [M_SCAN_REFLECTANCE_PROFILE_VALUES](#); [M_SCAN_SYMBOL_CONTRAST](#); [M_SCAN_SYMBOL_CONTRAST_GRADE](#); [M_UNUSED_ERROR_CORRECTION](#); [M_UNUSED_ERROR_CORRECTION_GRADE](#);

You can add one of the following values to the above-mentioned values to determine the verification result of a composite code.

If neither of these combination constants is added to a result type, the 1D results are returned followed by the 2D part. Note that this does not include general results (number of rows, number of scanlines) where results are compounded into a single value.

Composite code verification results	
Value	Description
M_2D_COMPONENT	Access the verification result of the 2D component of the composite code.
M_LINEAR_COMPONENT	Access the verification result of the linear component of the composite code.

Combination constants for [the values listed in](#) all tables **except For displaying results in an interactive dialog box**

You can add one of the following values to the above-mentioned values to cast the requested results to a required data type.

● For specifying the data type	
▢ Value	Description
▢ M_TYPE_DOUBLE	Casts the requested results to a <i>double</i> . (summarize)
	<i>ResultPtr info</i> Data type: double
▢ M_TYPE_LONG	Casts the requested results to a <i>long</i> . (summarize)
	<i>ResultPtr info</i> Data type: long
▢ M_TYPE_MIL_DOUBLE	Casts the requested results to a <i>MIL_DOUBLE</i> . (summarize)
	<i>ResultPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_DOUBLE Note: When a single result.
▢ M_TYPE_MIL_INT	Casts the requested results to a <i>MIL_INT</i> . (summarize)
	<i>ResultPtr info</i> Data type: MIL_INT
▢ M_TYPE_MIL_INT32	Casts the requested results to a <i>MIL_INT32</i> . (summarize)
	<i>ResultPtr info</i> Data type: MIL_INT32
▢ M_TYPE_MIL_INT64	Casts the requested results to a <i>MIL_INT64</i> . (summarize)
	<i>ResultPtr info</i> Data type: MIL_INT64
▢ M_TYPE_TEXT_CHAR	Cast the requested results to a <i>MIL_TEXT_CHAR</i> . (summarize)
	<i>ResultPtr info</i> <ul style="list-style-type: none"> • Data type: MIL_TEXT_CHAR • Data type: array of type MIL_TEXT_CHAR Array size: Size depends on the contents of the array.

ResultPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type MIL_DOUBLE
- array of type MIL_TEXT_CHAR
- double
- long
- M_NULL
- MIL_DOUBLE
- MIL_INT
- MIL_INT32
- MIL_INT64
- MIL_TEXT_CHAR

Specifies the address in which to place the specified result.

Set this parameter to **M_NULL** if you are setting the [ResultType](#) parameter to **M_INTERACTIVE** or if you are putting the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeGetResultSingle

Synopsis

Get the specified type of result for a single code model in a code context, obtained from a code read or verify operation.

Syntax

```
void McodeGetResultSingle(
    MIL_ID CodeResultId,
    MIL_INT ResultIndex,
    MIL_INT ResultType,
    void *ResultPtr
)
```

Description

This function retrieves the result of the specified type for a single code model occurrence from a result buffer. Results are only available after calling [McodeRead\(\)](#) or [McodeVerify\(\)](#). This function cannot be used to retrieve results which apply to the entire code context, to retrieve these results you must use [McodeGetResult\(\)](#).

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [ResultPtr](#) to [M_NULL](#). When this parameter is set to [M_NULL](#), the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [McodeGetResultSingle\(\)](#) again and you pass an array to the parameter [ResultPtr](#). You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared. Note that the requests for [McodeGetResult\(\)](#) and [McodeGetResultSingle\(\)](#) are put in the same queue.

By setting the [ResultType](#) parameter to [M_INTERACTIVE](#), you can view the results interactively.

Parameters

CodeResultId

Specifies the identifier of the code result buffer from which to retrieve results.

ResultIndex

Specifies the result index of the of the code model occurrence within the result buffer. This parameter must be between 0 and ([McodeGetResult\(\)](#) with [M_NUMBER](#)-1).

After a read or verify operation, the results stored in the result buffer are indexed starting at 0. See the [Retrieving results](#) section in [Chapter 13: Codes](#) for more information on result indices.

ResultType

Specifies the type of result(s) to retrieve or opens a dialog box that displays the results stored in the result buffer.

Not all result types are stored in the result buffer for all operations, the availability of results depends on which operation has been performed. In the tables below, the result types are grouped according to the operations after which they are available.

To display the results currently stored in the result buffer in an interactive dialog box, select the following value:

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter [M_NULL](#).

● For displaying results in an interactive dialog box	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens a dialog box containing a tree structure that displays the results stored in the result buffer. organized by codeword, scan and general results.</p> <p>(summarize)</p>
----------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

To retrieve a result that is returned for both code read and verify operations, the [ResultType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter the address of a MIL_DOUBLE.

● For retrieving results from a code read or verify operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CODE_MODEL_ID +	Retrieves the identifier of the code model that was used to read or verify the code. Note that this result is not valid when the results buffer is on a remote system. To find out which model was read on a remote system, use M_CODE_MODEL_INDEX instead. (summarize)
<input type="checkbox"/> M_CODE_MODEL_INDEX +	Retrieves the index of the code model, that was used to read or verify the code. Note that the number of codes read can be obtained using McodeGetResult() with M_NUMBER . When dealing with successful verify operations, the number of codes verified is always 1. (summarize)
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE Array size: The number of codes read or verified.</p>
<input type="checkbox"/> M_CODE_TYPE +	Retrieves the code type of the code that was read.
<input type="checkbox"/> M_ENCODING +	Retrieves the type of encoding. Note that since some versions of the RSS code types only differ by the bar height (and not the structure), the same result is returned for these similar code types. RSS-14 and RSS-14 Truncated will return M_ENC_RSS14 . RSS-14 Stacked and RSS-14 Stacked Omnidirectional will return M_ENC_RSS14_STACKED . It is possible to obtain a more accurate result to distinguish between these structurally similar code types, using McodeControl() with M_POSITION_ACCURACY set to M_HIGH . (summarize)
	<p><i>ResultPtr info</i></p> <p>Return values: M_ANY; M_ENC_ALPHA; M_ENC_ALPHANUM; M_ENC_ALPHANUM_PUNC; M_ENC_ASCII; M_ENC_EAN8; M_ENC_EAN13; M_ENC_ISO8; M_ENC_MODE2; M_ENC_MODE3; M_ENC_MODE4; M_ENC_MODE5; M_ENC_MODE6; M_ENC_NUM; M_ENC_QRCODE_MODEL1; M_ENC_QRCODE_MODEL2; M_ENC_RSS14; M_ENC_RSS_EXPANDED; M_ENC_RSS_EXPANDED_STACKED; M_ENC_RSS_LIMITED; M_ENC_RSS14_STACKED; M_ENC_RSS14_STACKED_OMNI; M_ENC_RSS14_TRUNCATED; M_ENC_STANDARD; M_ENC_UCCEAN128_MICROPDF417; M_ENC_UCCEAN128_PDF417; M_ENC_UPCA; M_ENC_UPCE; (details)</p>
<input type="checkbox"/> M_ERROR_CORRECTION +	Retrieves the type of error correction. (summarize)
	<p><i>ResultPtr info</i></p> <p>Return values: M_ANY; M_ECC_200; M_ECC_CHECK_DIGIT; M_ECC_COMPOSITE; M_ECC_H; M_ECC_L; M_ECC_M; M_ECC_NONE; M_ECC_Q; M_ECC_REED_SOLOMON; M_ECC_REED_SOLOMON_n; M_ECC_n; (details) M_ECC_UNKNOWN (unknown error correction type)</p>
<input type="checkbox"/> M_NUMBER +	Retrieves the number of codes read for all code models in the code context, during the last read or verify operation. Note that multiple occurrences of the same model are counted separately. (summarize)
<input type="checkbox"/> M_NUMBER_OF_CODEWORDS +	Retrieves the number of codewords in the code; this includes data, overhead, and error correction codewords. A codeword is a group of bars and spaces, representing one or more encoded characters. This result type is available for composite codes and 2D code types, with the exception of Data Matrix code types that do not use M_ERROR_CORRECTION with M_ECC_200 . (summarize)
<input type="checkbox"/> M_NUMBER_OF_ERASURES +	Retrieves the number of erasures in the code. Erasures are missing or unreadable codewords at known positions. This result type is available for composite codes and 2D code types, with the exception of Data Matrix code types that do not use M_ERROR_CORRECTION with M_ECC_200 . (summarize)
<input type="checkbox"/> M_NUMBER_OF_ERROR_CORRECTION_CODEWORDS +	Retrieves the number of error correction codewords in the code. Error correction can be used to compensate for defects found during the decoding process. This result type is available for composite codes and 2D code types, with the exception of Data Matrix code types that do not use M_ERROR_CORRECTION with M_ECC_200 .

	(summarize)																		
<input type="checkbox"/> M_NUMBER_OF_ERRORS +	Retrieves the number of errors found in the code. Note that the number of errors does not include the number of erasures. This result type is available for composite codes and 2D code types, with the exception of Data Matrix code types that do not use M_ERROR_CORRECTION with M_ECC_200 . (summarize)																		
<input type="checkbox"/> M_STATUS +	Retrieves the status of a read or verify operation. It can be helpful to obtain the number of codes read using McodeGetResult() with M_NUMBER , along with inquiring about M_STATUS . For example, if you set a code context to read two code occurrences and only one was found, the status returned is M_STATUS_NOT_FOUND . Calling McodeGetResult() with M_NUMBER will return 1. To retrieve the confidence score of a read operation, use M_SCORE . To retrieve the confidence score of a verify operation as a grade, use M_OVERALL_SYMBOL_GRADE . Note that when reading multiple code occurrences and more than one read operation fails, the least-common error is returned. For example, if you read two codes and one fails because the code was not found (M_STATUS_NOT_FOUND) and the other fails because the encoding type was unknown (M_STATUS_ENC_UNKNOWN), the latter is returned. (summarize) <i>ResultPtr info</i> Return values: <table> <tr> <td>M_STATUS_CRC_FAILED</td><td>Code read operation failed when validating the CRC (only for read operations).</td></tr> <tr> <td>M_STATUS_ECC_UNKNOWN</td><td>Unknown error correction type (only for read operations).</td></tr> <tr> <td>M_STATUS_ENC_UNKNOWN</td><td>Unknown encoding type (only for read operations).</td></tr> <tr> <td>M_STATUS_NOT_FOUND</td><td>Code was not found (only for read operations).</td></tr> <tr> <td>M_STATUS_NO_RESULT_AVAILABLE</td><td>There are no results available (for read and verify operations).</td></tr> <tr> <td>M_STATUS_READ_OK</td><td>Code read operation was successful (only for read operations).</td></tr> <tr> <td>M_STATUS_TIMEOUT_END</td><td>Code operation timed out (only for read and verify operations).</td></tr> <tr> <td>M_STATUS_VERIFY_FAILED</td><td>Code verify operation failed (only for verify operations).</td></tr> <tr> <td>M_STATUS_VERIFY_OK</td><td>Code verify was successful (only for verify operations).</td></tr> </table>	M_STATUS_CRC_FAILED	Code read operation failed when validating the CRC (only for read operations).	M_STATUS_ECC_UNKNOWN	Unknown error correction type (only for read operations).	M_STATUS_ENC_UNKNOWN	Unknown encoding type (only for read operations).	M_STATUS_NOT_FOUND	Code was not found (only for read operations).	M_STATUS_NO_RESULT_AVAILABLE	There are no results available (for read and verify operations).	M_STATUS_READ_OK	Code read operation was successful (only for read operations).	M_STATUS_TIMEOUT_END	Code operation timed out (only for read and verify operations).	M_STATUS_VERIFY_FAILED	Code verify operation failed (only for verify operations).	M_STATUS_VERIFY_OK	Code verify was successful (only for verify operations).
M_STATUS_CRC_FAILED	Code read operation failed when validating the CRC (only for read operations).																		
M_STATUS_ECC_UNKNOWN	Unknown error correction type (only for read operations).																		
M_STATUS_ENC_UNKNOWN	Unknown encoding type (only for read operations).																		
M_STATUS_NOT_FOUND	Code was not found (only for read operations).																		
M_STATUS_NO_RESULT_AVAILABLE	There are no results available (for read and verify operations).																		
M_STATUS_READ_OK	Code read operation was successful (only for read operations).																		
M_STATUS_TIMEOUT_END	Code operation timed out (only for read and verify operations).																		
M_STATUS_VERIFY_FAILED	Code verify operation failed (only for verify operations).																		
M_STATUS_VERIFY_OK	Code verify was successful (only for verify operations).																		
<input type="checkbox"/> M_TIMEOUT_END +	Retrieves whether the timeout was reached. You can set the timeout limit using McodeControl() with M_TIMEOUT . (summarize) <i>ResultPtr info</i> Return values: <table> <tr> <td>M_FALSE</td><td>Specifies that the timeout limit was not reached.</td></tr> <tr> <td>M_TRUE</td><td>Specifies that the timeout limit was reached.</td></tr> </table>	M_FALSE	Specifies that the timeout limit was not reached.	M_TRUE	Specifies that the timeout limit was reached.														
M_FALSE	Specifies that the timeout limit was not reached.																		
M_TRUE	Specifies that the timeout limit was reached.																		

To retrieve a result that is returned for a code read operation, the [ResultType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For retrieving results from a code read operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ANGLE +	Retrieves the angle at which the code was read. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
<input type="checkbox"/> M_BOTTOM_LEFT_X +	Retrieves the X-coordinate of the bottom-left corner of the code that was read. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
<input type="checkbox"/> M_BOTTOM_LEFT_Y +	Retrieves the Y-coordinate of the bottom-left corner of the code that was read. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
<input type="checkbox"/> M_BOTTOM_RIGHT_X +	Retrieves the X-coordinate of the bottom-right corner of the code that was read.

	For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
<input type="checkbox"/> M_BOTTOM_RIGHT_Y +	Retrieves the Y-coordinate of the bottom-right corner of the code that was read. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
<input type="checkbox"/> M_CELL_NUMBER_X +	Retrieves the number of cells in the X-direction.
<input type="checkbox"/> M_CELL_NUMBER_Y +	Retrieves the number of cells in the Y-direction.
<input type="checkbox"/> M_CELL_SIZE +	Retrieves the size of the cell in X (module size (Z)) of the code that was read.
<input type="checkbox"/> M_FOREGROUND_VALUE +	Retrieves the foreground color of the code that was read. (summarize)
	<div>ResultPtr info</div> Return values: Please see McodeControl() with M_FOREGROUND_BLACK . (details) Please see McodeControl() with M_FOREGROUND_WHITE . (details)
<input type="checkbox"/> M_POSITION_X +	Retrieves the X-coordinate of the occurrence. For all types of codes, except the Data Matrix and composite code types, this result is the center of the code in the X-direction. For Data Matrix codes, this result is the center, in the X-direction, of the top-left cell. For composite codes, this result is the center, in the X-direction, of the 1D component of the code. Note that to retrieve the top-left and bottom-right coordinates, use M_BOTTOM... and M_TOP... For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
<input type="checkbox"/> M_POSITION_Y +	Retrieves the Y-coordinate of the occurrence. For all types of codes, except the Data Matrix and composite code types, this result is the center of the code in the Y-direction. For Data Matrix codes, this result is the center, in the Y-direction, of the top-left cell. For composite codes, this result is the center, in the Y-direction, of the 1D component of the code. Note that to retrieve the top-left and bottom-right coordinates, use M_BOTTOM... and M_TOP... For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
<input type="checkbox"/> M_RECOMMENDED_APERTURE_SIZE +	Retrieves the recommended aperture size for the verify operation, following the ISO 15416 specification. Note that this value depends on McodeControl() with M_PIXEL_SIZE_IN_MM . If M_PIXEL_SIZE_IN_MM is not set (or is set to M_UNKNOWN), the value returned is (0.5 x cell size) . If the M_PIXEL_SIZE_IN_MM is set to a value, the recommended aperture size is computed using the ISO 15416 specification. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)
<input type="checkbox"/> M_SCORE +	Retrieves the confidence score of a read operation. The code type determines how this value is calculated. <ul style="list-style-type: none"> For linear 1D and RSS code types, the score is based on: <ul style="list-style-type: none"> The number of redundant scan lines. Redundant scan lines can be omitted from a code, while still yielding the same result. The difference between the actual code and theoretical code model. For 1D Planet and Postnet code types, the score is 1 if the code was read and 0 if it was not. For 2D code types, the score is calculated using: $((\text{Number of errors} + \text{Number of erasures}) / (\text{Number of codewords}))$. If the code could not be decoded successfully, the score is 0. For composite codes, the score is the score of either the 1D part or the 2D part; whichever was lower. To retrieve the error status of the verify operation, use M_STATUS . To retrieve the confidence score of a verify operation as a grade, use M_OVERALL_SYMBOL_GRADE . (summarize) <div>ResultPtr info</div> Return values: 0 <= Value <= 1 Specifies the confidence score of a read operation. 0 indicates 0% confidence and 1 indicates 100% confidence.
<input type="checkbox"/> M_SIZE_X +	Retrieves the size of the code in the X-direction. For composite code types, this result is the width of the 1D component of the code. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY . (summarize)
<input type="checkbox"/> M_SIZE_Y +	Retrieves the size of the code in the Y-direction. For composite code types, this result is the height of the 1D component of the code. For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY .

	(summarize)
<input type="checkbox"/> M_STRING +	<p>Retrieves the decoded string.</p> <p>If the string is decoded from a composite code model, the string is returned in the following format: 1D string 2D string, where is used as the separator.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_TEXT_CHAR</p> <p>Array size: Large enough to contain the decoded string (using M_STRING_SIZE +1).</p>
<input type="checkbox"/> M_STRING_SIZE +	Retrieves the length of the decoded string.
<input type="checkbox"/> M_THRESHOLD +	<p>Retrieves the threshold used to internally binarize the source image of a read operation.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Return values: 0<=Value<=255; M_ADAPTIVE; (details)</p>
<input type="checkbox"/> M_TOP_LEFT_X +	<p>Retrieves the X-coordinate of the top-left corner of the code that was read.</p> <p>For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY.</p> <p>(summarize)</p>
<input type="checkbox"/> M_TOP_LEFT_Y +	<p>Retrieves the Y-coordinate of the top-left corner of the code that was read.</p> <p>For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY.</p> <p>(summarize)</p>
<input type="checkbox"/> M_TOP_RIGHT_X +	<p>Retrieves the X-coordinate of the top-right corner of the code that was read.</p> <p>For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY.</p> <p>(summarize)</p>
<input type="checkbox"/> M_TOP_RIGHT_Y +	<p>Retrieves the Y-coordinate of the top-right corner of the code that was read.</p> <p>For 1D code types, this value is dependent on McodeControl() with M_POSITION_ACCURACY.</p> <p>(summarize)</p>
<input type="checkbox"/> M_TOTAL_STRING_SIZE +	<p>Retrieves the total length of all decoded strings. This is equal to the sum of the individual decoded string lengths (including the null terminator character of each string).</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: MIL_DOUBLE</p>

Combination constant for [M_STRING](#); [M_STRING_SIZE](#); [M_TOTAL_STRING_SIZE](#);

You can add the following value to the above-mentioned values to specify what type of symbols should be returned when dealing with unprintable characters.

● For use with string result types	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ESCAPE_SEQUENCE	<p>Retrieves the string with its unprintable characters represented by their ASCII character codes, when combined with M_STRING. Any back slashes (\) in the string are doubled, while unprintable characters, such as carriage returns, are put in a \xNN format, where NN is the hexadecimal value of the ASCII character code for the unprintable character.</p> <p>When combined with M_STRING_SIZE, it returns the size of the string including its unprintable characters.</p> <p>(summarize)</p>

To retrieve a result that is returned for a code verify operation, the [ResultType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter the address of a MIL_DOUBLE.

● For retrieving results from a code verify operation	
<input type="checkbox"/> Value	Description

<div>M_AXIAL_NONUNIFORMITY +</div>	<p>Retrieves the axial non-uniformity of the code. The axial non-uniformity is calculated using the following formula:</p> $\frac{\text{abs}(X_{avg} - Y_{avg})}{0.5 \times (X_{avg} + Y_{avg})}$ <p>Where X_{avg} is the average cell size in the X-dimension and Y_{avg} is the average measurement cell size in the Y-dimension.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Return values: 0 <= Value <= 1 The axial nonuniformity measure. M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code. </div> </div>
<div>M_AXIAL_NONUNIFORMITY_GRADE +</div>	<p>Retrieves the axial nonuniformity of the code, as a grade.</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Return values: M_CODE_GRADE_A Best grade for the result. M_CODE_GRADE_B Good grade for the result. M_CODE_GRADE_C Fair grade for the result. M_CODE_GRADE_D Poor grade for the result. M_CODE_GRADE_F Worst grade for the result. M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code. </div> </div>
<div>M_CODEWORD_DECODABILITY +</div>	<p>Retrieves the measure of the print quality of each codeword relative to the decoding algorithm. Note that this excludes any start/stop patterns.</p> <p>To retrieve the decodability of the start/stop patterns of your code, use M_SCAN_DECODABILITY.</p> <p>This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS. </div> </div>
<div>M_CODEWORD_DECODABILITY_GRADE +</div>	<p>Retrieves the decodability of each codeword in the code, as a grade.</p> <p>To retrieve the decodability of the start/stop patterns of your code as a grade, use M_SCAN_DECODABILITY_GRADE.</p> <p>This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS. Return values: M_CODE_GRADE_A Best grade for the result. M_CODE_GRADE_B Good grade for the result. M_CODE_GRADE_C Fair grade for the result. M_CODE_GRADE_D Poor grade for the result. M_CODE_GRADE_F Worst grade for the result. M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code. </div> </div>

<div> <div></div> <div>M_CODEWORD_DEFECTS +</div> </div>	<p>Retrieves and ranks the deviations in the expected signal that denotes a codeword in your code. The larger the result, the greater the defect in the codeword and the less likely that the codeword can be decoded without error.</p> <p>To retrieve the defects for your code as a single grade use M_DEFECTS_GRADE. To retrieve the defects of the start/stop patterns use M_SCAN_DEFECTS. This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS. </div> </div>
<div> <div></div> <div>M_CODEWORD_DEFECTS_GRADE +</div> </div>	<p>Retrieves the codeword defects, as a grade for each codeword.</p> <p>To retrieve the defects for your code as a single grade use M_DEFECTS_GRADE. To retrieve the defects of the start/stop patterns use M_SCAN_DEFECTS_GRADE. This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS. Return values: <div> <div>M_CODE_GRADE_A</div> <div>Best grade for the result.</div> </div> <div> <div>M_CODE_GRADE_B</div> <div>Good grade for the result.</div> </div> <div> <div>M_CODE_GRADE_C</div> <div>Fair grade for the result.</div> </div> <div> <div>M_CODE_GRADE_D</div> <div>Poor grade for the result.</div> </div> <div> <div>M_CODE_GRADE_F</div> <div>Worst grade for the result.</div> </div> <div> <div>M_CODE_GRADE_NOT_AVAILABLE</div> <div>The result is not available for the code.</div> </div> </div> </div>
<div> <div></div> <div>M_CODEWORD_MODULATION +</div> </div>	<p>Retrieves the modulation (MOD) of each codeword. Modulation is the ratio of the minimum edge contrast to symbol contrast within the code.</p> <p>To retrieve the modulation from the start/stop patterns, use M_SCAN_MODULATION.</p> <p>This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS. </div> </div>
<div> <div></div> <div>M_CODEWORD_MODULATION_GRADE +</div> </div>	<p>Retrieves the modulation (MOD) of each codeword, as a grade.</p> <p>To retrieve the modulation for your code as a single grade use M_MODULATION_GRADE. To retrieve the modulation from the start/stop patterns, as a grade, use M_SCAN_MODULATION_GRADE.</p> <p>This result type is part of the codeword quality used in the overlay procedure for the codeword print quality, described in the ISO 15415 specification.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <div> <div>ResultPtr info</div> <div> Data type: array of type MIL_DOUBLE Array size: The number of codewords scanned. This number can be obtained using McodeGetResult() with M_NUMBER_OF_CODEWORDS. Return values: <div> <div>M_CODE_GRADE_A</div> <div>Best grade for the result.</div> </div> <div> <div>M_CODE_GRADE_B</div> <div>Good grade for the result.</div> </div> <div> <div>M_CODE_GRADE_C</div> <div>Fair grade for the result.</div> </div> <div> <div>M_CODE_GRADE_D</div> <div>Poor grade for the result.</div> </div> <div> <div>M_CODE_GRADE_F</div> <div>Worst grade for the result.</div> </div> </div> </div>

	M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.
<input type="checkbox"/> M_CODEWORD_YIELD +	<p>Retrieves the codeword yield result. The codeword yield result determines how well your code can be read at an angle relative to the horizontal and vertical axis of code. When all other results are good, a poor codeword yield result can indicate a problem along the Y-axis of your code. For more information, refer to the ISO/IEC 15415 specification.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <p>0 <= Value <= 1 The codeword yield.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_CODEWORD_YIELD_GRADE +	<p>Retrieves the codeword yield, as a grade.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_DECODABILITY_GRADE +	<p>Retrieves the measure of the print quality of the code relative to the decoding algorithm, as a grade.</p> <p>To determine the decodability for each codeword in the code, use M_CODEWORD_DECODABILITY. To retrieve the decodability of the start/stop patterns of your code as a grade, use M_SCAN_DECODABILITY_GRADE.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_DECODE_GRADE +	<p>Retrieves the code decodability, as a grade.</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <p>M_CODE_GRADE_A The code can be read.</p> <p>M_CODE_GRADE_F The code cannot be read.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_DEFECTS_GRADE +	<p>Retrieves the measure of defects in the code, as a grade.</p> <p>To retrieve the measure of defects for each codeword as a grade, use M_CODEWORD_DEFECTS_GRADE. To retrieve the measure of defects in the start/stop patterns as a grade, use M_SCAN_DEFECTS_GRADE.</p>

	<p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<input type="checkbox"/> M_MODULATION_GRADE +	<p>Retrieves the codeword print quality assessment of the modulation, as a grade. For more information, refer to the ISO/IEC 15415 specification.</p> <p>To retrieve the modulation of each codeword as a grade, use M_CODEWORD_MODULATION_GRADE. To retrieve the modulation for the start/stop patterns as a grade, use M_SCAN_MODULATION_GRADE.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<input type="checkbox"/> M_NUMBER_OF_ROWS +	<p>Retrieves the number of rows verified in the code.</p> <p>For RSS code types, the number of rows is the number of stacks in the code. For cross-row code types, the number of rows is the number of rows verified in the start/stop pattern grading.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>												
<input type="checkbox"/> M_NUMBER_OF_SCANS +	<p>Retrieves the total number of scan reflectance profiles analyzed in the code. This is the sum of the number of scans per row, for every row within the code. The number of scans determines the number of results within an array for all the M_SCAN ... values described in this function.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>												
<input type="checkbox"/> M_NUMBER_OF_SCANS_PER_ROW +	<p>Retrieves the number of scan reflectance profiles per row. The number of scans determines the number of results within an array for all the M_SCAN ... values described in this function.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>												
<input type="checkbox"/> M_OVERALL_SYMBOL_GRADE +	<p>Retrieves the overall symbol grade. The code type determines the calculation used to obtain the grade.</p> <p>For 1D code types, the overall symbol grade is the average grade of the scan reflectance profile (using M_SCAN_REFLECTANCE_PROFILE_GRADE).</p> <p>For composite code types, the overall symbol grade is the worst grade overall, obtained when analyzing either the 1D or the 2D portions.</p> <p>For 2D code types, the overall symbol grade is the worst of all the returned grades.</p> <p>For cross-row codes, the overall symbol grade is the worst of all possible grades (with the exception of M_SCAN...GRADE).</p> <p>To retrieve the overall symbol grade for each row in the code, use M_ROW_OVERALL_GRADE.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.						
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												

	<p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_PRINT_GROWTH +	<p>Retrieves the print growth value. The print growth is a value determining whether the graphical features of the code have either grown or shrunk relative to theoretical model sufficiently to hinder readability.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p> <p>Value The print growth value.</p>
<input type="checkbox"/> M_PRINT_GROWTH_GRADE +	<p>Retrieves the print growth value, as a grade.</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_ROW_OVERALL_GRADE +	<p>Retrieves the overall grade for each row in the code.</p> <p>To retrieve the average of all the row overall grades, use M_OVERALL_SYMBOL_GRADE.</p> <p>For 1D code types, the overall row grade is the overall grade for the entire code.</p> <p>For 2D code types, the overall row grade is the overall grade per stack of the code.</p> <p>For cross-row code types, the overall row grade is the worst grade from the row.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of rows in the code. This number can be obtained using McodeGetResult() with M_NUMBER_OF_ROWS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_SCAN_DECODABILITY +	<p>Retrieves the decodability result of the scan reflectance profile.</p> <p>To retrieve the decodability for each codeword, use M_CODEWORD_DECODABILITY. To retrieve the decodability for your code as a grade, use M_DECODABILITY_GRADE.</p> <p>Note that certain code types have additional decodability measures that are taken into account when computing the decodability result. See the ISO 15417 (Code 128) and ISO 15420 (EAN and UPC code types) specifications.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p>

	<p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: 0 <= Value <= 1 The decodability parameter.</p>
<input type="checkbox"/> M_SCAN_DECODABILITY_GRADE +	<p>Retrieves the decodability result of the scan reflectance profile, as a grade.</p> <p>To retrieve the decodability for each codeword as a grade, use M_CODEWORD_DECODABILITY_GRADE. To retrieve the decodability for your cross-row code as a grade, use M_DECODABILITY_GRADE.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: M_CODE_GRADE_A Best grade for the result. M_CODE_GRADE_B Good grade for the result. M_CODE_GRADE_C Fair grade for the result. M_CODE_GRADE_D Poor grade for the result. M_CODE_GRADE_F Worst grade for the result. M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_SCAN_DECODE_GRADE +	<p>Retrieves whether the decoding algorithm succeeded or failed, as a grade.</p> <p>To retrieve the error status of the verify operation, use M_STATUS. To retrieve the confidence score of a read operation, use M_SCORE. Note that M_SCAN_EDGE_DETERMINATION_GRADE is used in calculating this value.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: M_CODE_GRADE_A The decoding algorithm succeeded. M_CODE_GRADE_F The decoding algorithm failed. M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_SCAN_DEFECTS +	<p>Retrieves and ranks the defects in the scan reflectance profile. The result is based on the ERN maximum (M_SCAN_ERN_MAXIMUM) and symbol contrast (M_SCAN_SYMBOL_CONTRAST).</p> <p>To retrieve the ranking measure for each codeword, use M_CODEWORD_DEFECTS. To determine the measure of defects in the code as a grade, use M_DEFECTS_GRADE.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: 0 <= Value <= 1 The ranking of the defects.</p>
<input type="checkbox"/> M_SCAN_DEFECTS_GRADE +	<p>Retrieves the measure of defects of the scan reflectance profile, as a grade.</p> <p>To retrieve the ranking measure of defects in each codeword, as a grade, use M_CODEWORD_DEFECTS_GRADE. To determine the measure of defects in the code, use M_DEFECTS_GRADE.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p>

	<p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<input type="checkbox"/> M_SCAN_EDGE_CONTRAST_MINIMUM +	<p>Retrieves the minimum edge contrast of the scan reflectance profile. When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE). This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <table> <tr> <td>0 <= Value <= 1</td><td>The minimum edge contrast.</td></tr> </table>	0 <= Value <= 1	The minimum edge contrast.										
0 <= Value <= 1	The minimum edge contrast.												
<input type="checkbox"/> M_SCAN_EDGE_CONTRAST_MINIMUM_GRADE +	<p>Retrieves the minimum edge contrast, as a grade. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<input type="checkbox"/> M_SCAN_EDGE_DETERMINATION_GRADE +	<p>Retrieves the edge determination, as a grade. The edge determination examines the number of bars and spaces in your code. See the ANSI X3.182 - 1990 specification. Note that this grade is not considered when calculating the overall code grade (M_OVERALL_SYMBOL_GRADE), instead it is included in the decode grade (M_SCAN_DECODE_GRADE). This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.		
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												

	M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.
<input type="checkbox"/> M_SCAN_ERN_MAXIMUM +	<p>Retrieves the highest ERN (element reflectance non-uniformity) in the scan reflectance profile. The ERN is an unexpected peak in the scan reflectance profile. The result is expressed as the ratio of the maximum ERN relative to the symbol contrast (ERN^{max} / SC) .</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: 0 <= Value <= 1 The element reflectance non-uniformity maximum.</p>
<input type="checkbox"/> M_SCAN_GUARD_PATTERN +	<p>Retrieves the size of the interior guard pattern, expressed as a factor of the cell (module size (Z)) in the scan reflectance profile. Guard patterns consist of two one-cell wide groupings at the end of each code. RSS code types have guard patterns at the ends of each row of the code. See the ISO 24724 specification. Note that the value is negative if the left guard pattern is greater than the right guard pattern, to permit easy identification.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for RSS code types and composite code types containing a RSS code.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p>
<input type="checkbox"/> M_SCAN_GUARD_PATTERN_GRADE +	<p>Retrieves the largest interior guard pattern, as a grade.</p> <p>This result type is available for RSS code types and composite code types containing a RSS code.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: M_CODE_GRADE_A Best grade for the result. M_CODE_GRADE_B Good grade for the result. M_CODE_GRADE_C Fair grade for the result. M_CODE_GRADE_D Poor grade for the result. M_CODE_GRADE_F Worst grade for the result. M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_SCAN_INTERCHARACTER_GAP +	<p>Retrieves the size of the largest intercharacter gap in the scan reflectance profile, expressed as a factor of the cell size (module size (Z)) in the scan reflectance profile. See ISO 16388 and AIM BC3-1995 for more details.</p> <p>Note that this result type will always return the value of M_CODE_GRADE_NOT_AVAILABLE when McodeControl() with M_PIXEL_SIZE_IN_MM is not set, and the following condition is true: $(5.3 Z \leq \text{Largest intercharacter gap} \leq 3 Z)$.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for Code 39 and Codabar code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values: M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code. Value The maximum intercharacter gap.</p>
<input type="checkbox"/> M_SCAN_INTERCHARACTER_GAP_GRADE +	<p>Retrieves the largest intercharacter gap, as a grade.</p> <p>This result type is available for Code 39 and Codabar code types.</p> <p>(summarize)</p>

	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <p>M_CODE_GRADE_A The maximum intercharacter gap is under three times the module size (3Z). M_CODE_GRADE_F The maximum intercharacter gap is over 5.3 times the module size (5.3Z). M_CODE_GRADE_NOT_AVAILABLE The maximum intercharacter gap cannot be calculated because the M_PIXEL_SIZE_IN_MM is not set.</p>
M_SCAN_MODULATION +	<p>Retrieves the modulation (MOD) of the scan reflectance profile. The modulation is the ratio of the minimum edge contrast (M_SCAN_EDGE_CONTRAST_MINIMUM) to the symbol contrast (M_SCAN_SYMBOL_CONTRAST).</p> <p>To retrieve the modulation of each codeword, use M_CODEWORD_MODULATION. This result type is available for 2D cross-row and composite code types. (summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <p>0 <= Value <= 1 The modulation parameter.</p>
M_SCAN_MODULATION_GRADE +	<p>Retrieves the modulation (MOD) of the scan reflectance profile, as a grade.</p> <p>To retrieve the modulation of each codeword as a grade, use M_CODEWORD_MODULATION_GRADE. To retrieve the modulation for the code as a grade, use M_MODULATION_GRADE. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <p>M_CODE_GRADE_A Best grade for the result. M_CODE_GRADE_B Good grade for the result. M_CODE_GRADE_C Fair grade for the result. M_CODE_GRADE_D Poor grade for the result. M_CODE_GRADE_F Worst grade for the result. M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SCAN_PRINT_CONTRAST_SIGNAL +	<p>Retrieves the print contrast signal (PCS) of the scan reflectance profile. Note that this is an approximation based on the maximum reflectance (M_SCAN_REFLECTANCE_MAXIMUM) and the code contrast (M_SCAN_SYMBOL_CONTRAST). For more details, see the ISO 15416 specification. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <p>0 <= Value <= 1 The print contrast signal.</p>
M_SCAN_PROFILE_END_X +	<p>Retrieves the X-coordinate of the end of the scan reflectance profile.</p> <p>This result type relates to the position of the scan profiles, enabling you to visualize or draw the available analysis profiles using McodeDraw() with M_DRAW_SCAN_PROFILES. Note that the start and stop positions of any given scan are related to the steps of the analysis were performed. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p>
M_SCAN_PROFILE_END_Y +	<p>Retrieves the Y-coordinate of the end of the scan reflectance profile.</p> <p>This result type relates to the position of the scan profiles, enabling you to visualize or draw the available analysis profiles using McodeDraw() with M_DRAW_SCAN_PROFILES. Note that the start and stop positions of any given scan are related to the steps of the analysis were performed. This result type is available for 1D, 2D cross-row, and composite code types. (summarize)</p>

M_SCAN_PROFILE_START_X +	<p>Retrieves the X-coordinate of the start of the scan reflectance profile.</p> <p>This result type relates to the position of the scan profiles, enabling you to visualize or draw the available analysis profiles using McodeDraw() with M_DRAW_SCAN_PROFILES. Note that the start and stop positions of any given scan are related to the steps of the analysis were performed.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
M_SCAN_PROFILE_START_Y +	<p>Retrieves the Y-coordinate of the start of the scan reflectance profile.</p> <p>This result type relates to the position of the scan profiles, enabling you to visualize or draw the available analysis profiles using McodeDraw() with M_DRAW_SCAN_PROFILES. Note that the start and stop positions of any given scan are related to the steps of the analysis were performed.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p>
M_SCAN_QUIET_ZONE +	<p>Retrieves the ratio between the expected quiet zone from theoretical code model and the measured quiet zone of your code as a ratio of (<i>measured quiet zone size / expected quiet zone size</i>).</p> <p>The measurement is taken from both sides of the code. Note that if the returned value is negative then the left quiet zone is worse than the right. Refer to the appropriate specifications for more details (ISO 15417, ISO 15420, ISO 16388, and ISO 16390, respectively).</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 1D code types (except RSS) and composite code types encoded with a format of EAN/UPC or UCC/EAN120.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available.</p> <p>Value The scan quiet zone vale.</p> </div>
M_SCAN_QUIET_ZONE_GRADE +	<p>Retrieves the quiet zone size, as a grade.</p> <p>This result type is available for 1D code types (except RSS) and composite code types encoded with a format of EAN/UPC or UCC/EAN120.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p> </div>
M_SCAN_REFLECTANCE_MAXIMUM +	<p>Retrieves the highest reflectance (R_{max}) of the scan reflectance profile, as a percentage.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>0 <= Value <= 1 The maximum reflectance.</p> </div>
M_SCAN_REFLECTANCE_MINIMUM +	<p>Retrieves the lowest reflectance (R_{min}) of the scan reflectance profile, as a percentage.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <div> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> </div>

	<p>Return values:</p> <p>0 <= Value <= 1 The minimum reflectance.</p>
M_SCAN_REFLECTANCE_MINIMUM_GRADE +	<p>Retrieves a pass or fail grade for the scan reflectance profile.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>M_CODE_GRADE_A The scan reflectance profile passed.</p> <p>M_CODE_GRADE_F The scan reflectance profile failed.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SCAN_REFLECTANCE_PROFILE_GRADE +	<p>Retrieves the lowest grade from all the scan reflectance profile validation tests.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
M_SCAN_REFLECTANCE_PROFILE_LENGTH +	<p>Retrieves the number of values in the scan reflectance profile.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p>
M_SCAN_REFLECTANCE_PROFILE_VALUES +	<p>Retrieves the reflectance values of the scan reflectance profile.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of values in the scan reflectance profile. This number can be obtained using McodeGetResult() with M_SCAN_REFLECTANCE_PROFILE_LENGTH.</p>
M_SCAN_SYMBOL_CONTRAST +	<p>Retrieves the code (symbol) contrast (SC) of the scan reflectance profile. Note that this value is used in conjunction with the maximum reflectance (M_SCAN_REFLECTANCE_MAXIMUM) to determine the print contrast signal (M_SCAN_PRINT_CONTRAST_SIGNAL). For more details, see ISO 15416.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>When dealing with 2D matrix codes, to retrieve the code contrast for the entire code, use M_SYMBOL_CONTRAST.</p> <p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The sum of the length of the scan profiles. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p> <p>Return values:</p> <p>0 <= Value <= 1 The symbol contrast.</p>
M_SCAN_SYMBOL_CONTRAST_GRADE +	<p>Retrieves the code contrast of the scan reflectance profile, as a grade.</p>

	<p>This result type is available for 1D, 2D cross-row, and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<input type="checkbox"/> M_SCAN_WIDE_TO_NARROW_RATIO +	<p>Retrieves the ratio between the average of the widest and the average of the narrowest bar/space in the scan reflectance profile. See ISO 16388 and ISO 16390 for more details.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for Code 39, Codebar, and Interleaved 2/5.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS.</p>												
<input type="checkbox"/> M_SCAN_WIDE_TO_NARROW_RATIO_GRADE +	<p>Retrieves wide to narrow ratio, as a grade.</p> <p>This result type is available for Code 39, Codebar, and Interleaved 2/5.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of scans. This number can be obtained using McodeGetResult() with M_NUMBER_OF_SCANS. Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<input type="checkbox"/> M_START_STOP_PATTERN_GRADE +	<p>Retrieves the average of the worst scan reflectance profile grades of each row, as a grade. A start/stop pattern is a set pattern that marks the end points of the code. For each scan reflectance profile, the start/stop pattern grade is computed for each verified row, then the average start/stop pattern grade for all the scan reflectance profiles is computed.</p> <p>This result type is available for 2D cross-row and composite code types.</p> <p>(summarize)</p> <p><i>ResultPtr info</i> Return values:</p> <table> <tr> <td>M_CODE_GRADE_A</td><td>Best grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_B</td><td>Good grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_C</td><td>Fair grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_D</td><td>Poor grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_F</td><td>Worst grade for the result.</td></tr> <tr> <td>M_CODE_GRADE_NOT_AVAILABLE</td><td>The result is not available for the code.</td></tr> </table>	M_CODE_GRADE_A	Best grade for the result.	M_CODE_GRADE_B	Good grade for the result.	M_CODE_GRADE_C	Fair grade for the result.	M_CODE_GRADE_D	Poor grade for the result.	M_CODE_GRADE_F	Worst grade for the result.	M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.
M_CODE_GRADE_A	Best grade for the result.												
M_CODE_GRADE_B	Good grade for the result.												
M_CODE_GRADE_C	Fair grade for the result.												
M_CODE_GRADE_D	Poor grade for the result.												
M_CODE_GRADE_F	Worst grade for the result.												
M_CODE_GRADE_NOT_AVAILABLE	The result is not available for the code.												
<input type="checkbox"/> M_SYMBOL_CONTRAST +	<p>Retrieves the code (symbol) contrast (SC) value for the code. In the case of matrix code types, the code contrast is calculated as the average 10% of the lighter pixels minus the average 10% of the darker pixels.</p>												

	<p>To retrieve the code contrast from scan reflectance profiles, use M_SCAN_SYMBOL_CONTRAST.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <p>0 <= Value <= 1 The SC value.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_SYMBOL_CONTRAST_GRADE +	<p>Retrieves the code (symbol) contrast (SC) value for the code as a grade. The higher the contrast, the better it is for scanning purposes and the better the grade.</p> <p>To retrieve the code contrast of the scan reflectance profile as a grade, use M_SCAN_SYMBOL_CONTRAST_GRADE.</p> <p>This result type is available for 2D matrix code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_UNUSED_ERROR_CORRECTION +	<p>Retrieves the ratio of unused error correction to the total amount of error correction available for the code. This tests the extent to which regional or spot damage in the code has eroded the reading safety margin that error correction provides.</p> <p>When dealing with measures that have associated grades, verify the associated grade first. This allows you to determine whether the result is available or not (M_CODE_GRADE_NOT_AVAILABLE).</p> <p>This result type is available for all 2D and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: The number of cells scanned. This number can be obtained using McodeGetResult() with M_CELL_NUMBER_X.</p> <p>Return values:</p> <p>0 <= Value <= 1 The ratio of unused error correction within the code.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>
<input type="checkbox"/> M_UNUSED_ERROR_CORRECTION_GRADE +	<p>Retrieves the ratio of unused error correction to the total amount of error correction available for the code, as a grade.</p> <p>This result type is available for all 2D and composite code types.</p> <p>(summarize)</p>
	<p><i>ResultPtr info</i></p> <p>Return values:</p> <p>M_CODE_GRADE_A Best grade for the result.</p> <p>M_CODE_GRADE_B Good grade for the result.</p> <p>M_CODE_GRADE_C Fair grade for the result.</p> <p>M_CODE_GRADE_D Poor grade for the result.</p> <p>M_CODE_GRADE_F Worst grade for the result.</p> <p>M_CODE_GRADE_NOT_AVAILABLE The result is not available for the code.</p>

Combination constants for the values listed in [For retrieving results from a code read or verify operation](#); and for the following values: [M_CODEWORD_DECODABILITY](#); [M_CODEWORD_DECODABILITY_GRADE](#); [M_CODEWORD_DEFECTS](#); [M_CODEWORD_DEFECTS_GRADE](#); [M_CODEWORD_MODULATION](#); [M_CODEWORD_MODULATION_GRADE](#); [M_CODEWORD_YIELD](#); [M_CODEWORD_YIELD_GRADE](#); [M_DECODABILITY_GRADE](#); [M_DECODE_GRADE](#); [M_DEFECTS_GRADE](#); [M_MODULATION_GRADE](#); [M_NUMBER_OF_ROWS](#); [M_NUMBER_OF_SCANS](#); [M_NUMBER_OF_SCANS_PER_ROW](#); [M_OVERALL_SYMBOL_GRADE](#); [M_ROW_OVERALL_GRADE](#);

M_SCAN_DECODABILITY; M_SCAN_DECODABILITY_GRADE; M_SCAN_DECODE_GRADE; M_SCAN_DEFECTS; M_SCAN_DEFECTS_GRADE; M_SCAN_EDGE_CONTRAST_MINIMUM; M_SCAN_EDGE_CONTRAST_MINIMUM_GRADE; M_SCAN_EDGE_DETERMINATION_GRADE; M_SCAN_ERN_MAXIMUM; M_SCAN_GUARD_PATTERN; M_SCAN_GUARD_PATTERN_GRADE; M_SCAN_MODULATION; M_SCAN_MODULATION_GRADE; M_SCAN_PRINT_CONTRAST_SIGNAL; M_SCAN_PROFILE_START_X; M_SCAN_PROFILE_START_Y; M_SCAN_PROFILE_END_X; M_SCAN_PROFILE_END_Y; M_SCAN_QUIET_ZONE; M_SCAN_QUIET_ZONE_GRADE; M_SCAN_REFLECTANCE_MAXIMUM; M_SCAN_REFLECTANCE_MINIMUM; M_SCAN_REFLECTANCE_MINIMUM_GRADE; M_SCAN_REFLECTANCE_PROFILE_GRADE; M_SCAN_REFLECTANCE_PROFILE_LENGTH; M_SCAN_REFLECTANCE_PROFILE_VALUES; M_SCAN_SYMBOL_CONTRAST; M_SCAN_SYMBOL_CONTRAST_GRADE; M_UNUSED_ERROR_CORRECTION; M_UNUSED_ERROR_CORRECTION_GRADE;

You can add one of the following values to the above-mentioned values to determine the verification result of a composite code.

If neither of these combination constants is added to a result type, the 1D results are returned followed by the 2D part. Note that this does not include general results (number of rows, number of scanlines) where results are compounded into a single value.

Composite code verification results	
Value	Description
M_2D_COMPONENT	Access the verification result of the 2D component of the composite code.
M_LINEAR_COMPONENT	Access the verification result of the linear component of the composite code.

Combination constants for the values listed in all tables **except For displaying results in an interactive dialog box**

You can add one of the following values to the above-mentioned values to cast the requested results to a required data type.

For specifying the data type	
Value	Description
M_TYPE_DOUBLE	<p>Casts the requested results to a <i>double</i>. (summarize)</p> <p><i>ResultPtr info</i> Data type: double</p>
M_TYPE_LONG	<p>Casts the requested results to a <i>long</i>. (summarize)</p> <p><i>ResultPtr info</i> Data type: long</p>
M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>ResultPtr info</i> Data type: MIL_DOUBLE</p>
M_TYPE_MIL_INT	<p>Casts the requested results to a <i>MIL_INT</i>. (summarize)</p> <p><i>ResultPtr info</i> Data type: MIL_INT</p>
M_TYPE_MIL_INT32	<p>Casts the requested results to a <i>MIL_INT32</i>. (summarize)</p> <p><i>ResultPtr info</i> Data type: MIL_INT32</p>
M_TYPE_MIL_INT64	<p>Casts the requested results to a <i>MIL_INT64</i>. (summarize)</p> <p><i>ResultPtr info</i> Data type: MIL_INT64</p>
M_TYPE_TEXT_CHAR	<p>Casts the requested results to a <i>MIL_TEXT_CHAR</i>.</p>

	(summarize)
	<i>ResultPtr info</i> Data type: array of type MIL_TEXT_CHAR Array size: Size depends on the contents of the array

ResultPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• array of type MIL_DOUBLE• array of type MIL_TEXT_CHAR• double• long• M_NULL• MIL_DOUBLE• MIL_INT• MIL_INT32• MIL_INT64

Specifies the address in which to place the specified result.

Set this parameter to **M_NULL** if you are setting the [ResultType](#) parameter to **M_INTERACTIVE** or if you are putting the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeInquire

Synopsis

Inquire about a code context or code model setting.

Syntax

```
MIL_INT McodeInquire(  
    MIL_ID CodeId,  
    MIL_INT InquireType,  
    void *UserVarPtr  
)
```

Description

This function inquires about a setting of the specified code context or code model.

Note that **McodeInquire()** in combination with **McodeWrite()** can also be used to determine the required image buffer size to encode a given string. First, call **McodeWrite()** with **ImageBufId** set to **M_NULL** and then call **McodeInquire()** to retrieve the minimum width (**M_WRITE_SIZE_X**) and height (**M_WRITE_SIZE_Y**) required for the code. This same method can be used to obtain the number of cells required to write your code (**M_WRITE_CELL_NUMBER_X** and **M_WRITE_CELL_NUMBER_Y**).

Parameters

CodeId
Specifies the identifier of the code context or code model.

InquireType
Specifies the setting about which to inquire.

To display the setting of the inquire types in an interactive dialog box, select the following:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter *M_NULL*.

● For displaying results in an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] Opens a read-only dialog which allows the user to see all the current settings of the context's inquire types.

To get the number of code models in a code context, select the following:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For inquiring about the number code models in a code context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NUMBER_OF_CODE_MODELS +	Returns the number of code models in the code context. (summarize)
	<i>UserVarPtr</i> info Return values:

	Value >= 0	Specifies the number of code models in the code context.
--	------------	----------------------------------------------------------

For a code context, the [InquireType](#) parameter can be set to one of the following to inquire about control settings relating only to read operations:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

● For inquiring about a code context's global settings relating to McodeRead operations		
Value	Description	
M_TIMEOUT +	Returns the maximum decoding time for a read operation. (summarize)	
	UserVarPtr info	Return values: M_DISABLE; Value >= 0; (details)
M_TOTAL_NUMBER +	Returns the total number of codes to be read in one image. This number is also limited by M_NUMBER . (summarize)	
	UserVarPtr info	Return values: M_ALL; M_DEFAULT; Value >= 0; (details)

For a code context, the [InquireType](#) parameter can be set to one of the following to inquire about control settings relating to read and verify operations:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

● For inquiring about a code context's global settings relating to McodeRead and McodeVerify operations		
Value	Description	
M_MINIMUM_CONTRAST +	Returns the minimum contrast between the foreground and background in the target image for 1D codes (excluding Planet and Postnet) when using the M_ADAPTIVE threshold mode. (summarize)	
	UserVarPtr info	Return values: 1 to 255; M_DEFAULT; (details)
M_SCANLINE_HEIGHT +	Returns the scanline height for scanline-based bar codes. (summarize)	
	UserVarPtr info	Return values: 0<Value<=256; (details)
M_SCANLINE_STEP +	Returns the scanline interval for scanline-based bar codes. (summarize)	
	UserVarPtr info	Return values: Value > 1; (details)
M_SEARCH_ANGLE_MODE +	Returns whether to perform calculations specific to angular-range strategies for the code context. (summarize)	
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
M_SPEED +	Returns the search speed. (summarize)	
	UserVarPtr info	Return values: M_HIGH; M_LOW; M_MEDIUM; M_VERY_HIGH; M_VERY_LOW; (details)
M_THRESHOLD +	Returns the threshold value used to internally binarize the source image. (summarize)	

UserVarPtr info

Return values: 0<=Value<=255; M_ADAPTIVE; ([details](#))

To get the code type of a code model, select the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For inquiring about a code model's code type	
☐ Value	Description
☐ M_CODE_TYPE +	Returns the code type of the specified model. (summarize)
	<i>UserVarPtr info</i> Return values: M_ANY; M_BC412; M_CODABAR; M_CODE39; M_CODE93; M_CODE128; M_COMPOSITECODE; M_DATAMATRIX; M_EAN8; M_EAN13; M_INTERLEAVED25; M_MAXICODE; M_MICROPDF417; M_PDF417; M_PHARMACODE; M_PLANET; M_POSTNET; M_QRCODE; M_RSSCODE; M_TRUNCATED_PDF417; M_UPC_A; M_UPC_E; (details)

For a code model, the [InquireType](#) parameter can be set to one of the following to inquire about control settings relating only to read operations:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For inquiring about a code model's settings relating to McodeRead operations	
☐ Value	Description
☐ M_ECC_CORRECTED_NUMBER +	Returns whether or not a more robust read operation will be performed. Note that M_DEFAULT is not returned. (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)
☐ M_NUMBER +	Returns the maximum number of codes to be read for the specified code model. This number is also limited by M_TOTAL_NUMBER . (summarize)
	<i>UserVarPtr info</i> Return values: M_ALL; M_DEFAULT; Value >= 0; (details)
☐ M_POSITION_ACCURACY +	Returns the positional accuracy of the read operation. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_HIGH; M_LOW; (details)

For a code model, the [InquireType](#) parameter can be set to one of the following to inquire about control settings relating to read and verify operations:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For inquiring about a code model's settings relating to McodeRead and McodeVerify operations	
☐ Value	Description
☐ M_BEARER_BAR +	Returns whether bearer bars surround the codes to read. (summarize)
	<i>UserVarPtr info</i> Return values: M_ABSENT; M_PRESENT; (details)
☐ M_CELL_NUMBER_X_MAX +	Returns the maximum number of cells to search for in the X-direction. (summarize)

	<i>UserVarPtr Info</i> Return values: M_ANY; Value > 0; (details)
<input type="checkbox"/> M_CELL_NUMBER_X_MIN +	Returns the minimum number of cells to search for in the X-direction. (summarize)
	<i>UserVarPtr Info</i> Return values: M_ANY; Value > 0; (details)
<input type="checkbox"/> M_CELL_NUMBER_Y_MAX +	Returns the maximum number of cells to search for in the Y-direction. (summarize)
	<i>UserVarPtr Info</i> Return values: M_ANY; Value > 0; (details)
<input type="checkbox"/> M_CELL_NUMBER_Y_MIN +	Returns the minimum number of cells to search for in the Y-direction. (summarize)
	<i>UserVarPtr Info</i> Return values: M_ANY; Value > 0; (details)
<input type="checkbox"/> M_CHECK_FINDER_PATTERN +	Returns whether or not the check for false Data Matrix codes is enabled. (summarize)
	<i>UserVarPtr Info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_DOT_SPACING +	Returns the distance between 2 dots in a matrix code composed of dots. (summarize)
	<i>UserVarPtr Info</i> Return values: -256 to 256; (details)
<input type="checkbox"/> M_SCAN_REVERSE +	Returns whether the bar code reader will scan the bar code both left-to-right and right-to-left, or only left-to-right. (summarize)
	<i>UserVarPtr Info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SEARCH_ANGLE +	Returns the nominal search angle. (summarize)
	<i>UserVarPtr Info</i> Return values: 0.0 to 360.0; (details)
<input type="checkbox"/> M_SEARCH_ANGLE_DELTA_NEG +	Returns the negative angle range of the search. (summarize)
	<i>UserVarPtr Info</i> Return values: 0.0 to 180.0; (details)
<input type="checkbox"/> M_SEARCH_ANGLE_DELTA_POS +	Returns the positive angle range of the search. (summarize)
	<i>UserVarPtr Info</i> Return values: 0.0 to 180.0; M_DEFAULT; (details)
<input type="checkbox"/> M_STRING_SIZE_MAX +	Returns the maximum size of the string encoded to read in each code. (summarize)
	<i>UserVarPtr Info</i> Return values: 1 to 65, 535; M_ANY; M_DEFAULT; (details)
<input type="checkbox"/> M_STRING_SIZE_MIN +	Returns the minimum size of the string encoded to read in each code. (summarize)
	<i>UserVarPtr Info</i> Return values: 1 to 65, 535; M_ANY; M_DEFAULT; (details)

<input type="checkbox"/> M_SUB_TYPE +	<p>Returns the particular codes in a family of codes for which to search. (summarize)</p> <p><i>UserVarPtr Info</i> Return values: M_ANY; M_EAN8; M_EAN13; M_RSS14; M_RSS_EXPANDED; M_RSS_EXPANDED_STACKED; M_RSS_LIMITED; M_RSS14_STACKED; M_RSS14_STACKED_OMNI; M_RSS14_TRUNCATED; M_UCCEAN128; M_UPC_A; M_UPC_E; (details)</p>
<input type="checkbox"/> M_USE_PRESEARCH +	<p>Returns whether the localization operation is performed prior to read and verify operations. (summarize)</p> <p><i>UserVarPtr Info</i> Return values: M_DISABLE; M_ENABLE; (details)</p>

For a code model, the [InquireType](#) parameter can be set to one of the following to inquire about control settings relating to read, verify, and write operations:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For inquiring about a code model's settings relating to McodeRead, McodeVerify, and McodeWrite operations	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CELL_NUMBER_X +	<p>Returns the number of cells to be used in the X-direction. (summarize)</p> <p><i>UserVarPtr Info</i> Return values: M_ANY; Value > 0; (details)</p>
<input type="checkbox"/> M_CELL_NUMBER_Y +	<p>Returns the number of cells to be used in the Y-direction. (summarize)</p> <p><i>UserVarPtr Info</i> Return values: M_ANY; Value > 0; (details)</p>
<input type="checkbox"/> M_CELL_SIZE_MAX +	<p>Returns the maximum cell size. (summarize)</p> <p><i>UserVarPtr Info</i> Return values: M_DEFAULT; Value; (details)</p>
<input type="checkbox"/> M_CELL_SIZE_MIN +	<p>Returns the minimum cell size. (summarize)</p> <p><i>UserVarPtr Info</i> Return values: M_DEFAULT; Value > 0; (details)</p>
<input type="checkbox"/> M_DATAMATRIX_SHAPE +	<p>Returns the shape of the Data Matrix code. (summarize)</p> <p><i>UserVarPtr Info</i> Return values: M_ANY; M_RECTANGLE; M_SQUARE; (details)</p>
<input type="checkbox"/> M_ENCODING +	<p>Returns the type of encoding scheme. (summarize)</p> <p><i>UserVarPtr Info</i> Return values: M_ANY; M_ENC_ALPHA; M_ENC_ALPHANUM; M_ENC_ALPHANUM_PUNC; M_ENC_ASCII; M_ENC_EAN8; M_ENC_EAN13; M_ENC_ISO8; M_ENC_MODE2; M_ENC_MODE3; M_ENC_MODE4; M_ENC_MODE5; M_ENC_MODE6; M_ENC_NUM; M_ENC_QRCODE_MODEL1; M_ENC_QRCODE_MODEL2; M_ENC_RSS14; M_ENC_RSS_EXPANDED; M_ENC_RSS_EXPANDED_STACKED; M_ENC_RSS_LIMITED; M_ENC_RSS14_STACKED; M_ENC_RSS14_STACKED_OMNI; M_ENC_RSS14_TRUNCATED; M_ENC_STANDARD; M_ENC_UCCEAN128_MICROPDF417; M_ENC_UCCEAN128_PDF417; M_ENC_UPCA; M_ENC_UPCE; (details)</p>
<input type="checkbox"/> M_ERROR_CORRECTION +	<p>Returns the type of error correction. (summarize)</p> <p><i>UserVarPtr Info</i> Return values: M_ANY; M_ECC_200; M_ECC_CHECK_DIGIT; M_ECC_COMPOSITE; M_ECC_H; M_ECC_L; M_ECC_M; M_ECC_NONE; M_ECC_Q; M_ECC_REED_SOLOMON; M_ECC_REED_SOLOMON_n; M_ECC_n; (details)</p>

<input type="checkbox"/> M_FOREGROUND_VALUE +	Returns the foreground color of the code. (summarize)
	<i>UserVarPtr info</i> Return values: M_FOREGROUND_ANY; M_FOREGROUND_BLACK; M_FOREGROUND_WHITE; (details)

For a code model, the [InquireType](#) parameter can be set to one of the following to inquire about control settings relating only to verify operations:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For inquiring about a code model's settings relating to McodeVerify operations	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ABSOLUTE_APERTURE_SIZE	Returns the absolute size (diameter) of the aperture. (summarize)
	<i>UserVarPtr info</i> Return values: Value >= 2; (details)
<input type="checkbox"/> M_APERTURE_MODE	Returns the mode to use to determine the aperture size. (summarize)
	<i>UserVarPtr info</i> Return values: M_ABSOLUTE; M_DISABLE; M_RELATIVE; (details)
<input type="checkbox"/> M_MAXIMUM_CALIBRATED_REFLECTANCE	Returns the maximum grayscale value in the target image. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 255; (details)
<input type="checkbox"/> M_MINIMUM_CALIBRATED_REFLECTANCE	Returns the minimum grayscale value in the target image. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 255; (details)
<input type="checkbox"/> M_PIXEL_SIZE_IN_MM	Returns the scale between a pixel and its physical measurement, in millimeters per pixel units. (summarize)
	<i>UserVarPtr info</i> Return values: Value > 0; (details)
<input type="checkbox"/> M_RELATIVE_APERTURE_FACTOR	Returns the relative aperture factor. (summarize)
	<i>UserVarPtr info</i> Return values: 0 <= Value <= 2; M_AUTO; (details)

For a code model, the [InquireType](#) parameter can be set to one of the following to inquire about control settings relating only to write operations:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For inquiring about a code model's settings relating to McodeWrite operations	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_WRITE_CELL_NUMBER_X +	Returns the number of cells in the X-direction required to encode the string passed to a write operation.
<input type="checkbox"/> M_WRITE_CELL_NUMBER_Y +	Returns the number of cells in the Y-direction required to encode the string passed to a write operation.
<input type="checkbox"/> M_WRITE_SIZE_X +	Returns the minimum width required for the destination image of a write operation.
<input type="checkbox"/> M_WRITE_SIZE_Y +	Returns the minimum height required for the destination image of a write operation.

Combination constants for [the values listed in](#) all tables **except For displaying results in an interactive dialog box, For inquiring about a code model's settings relating to McodeVerify operations**

You can add one of the following values to the above-mentioned values to cast the requested information to a required data type.

● For specifying the data type	
Value	Description
<input type="checkbox"/> M_TYPE_DOUBLE	Casts the requested data to a <i>double</i> . (summarize)
	<i>UserVarPtr info</i> Data type: double
<input type="checkbox"/> M_TYPE_LONG	Casts the requested data to a <i>long</i> . (summarize)
	<i>UserVarPtr info</i> Data type: long
<input type="checkbox"/> M_TYPE_MIL_DOUBLE	Casts the requested data to a <i>MIL_DOUBLE</i> . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE
<input type="checkbox"/> M_TYPE_MIL_INT	Casts the requested data to a <i>MIL_INT</i> . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_INT
<input type="checkbox"/> M_TYPE_MIL_INT32	Casts the requested data to a <i>MIL_INT32</i> . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_INT32
<input type="checkbox"/> M_TYPE_MIL_INT64	Casts the requested data to a <i>MIL_INT64</i> . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_INT64

Combination constant for [the values listed in](#) all tables **except For displaying results in an interactive dialog box, For inquiring about a code model's settings relating to McodeVerify operations**

You can add the following value to the above-mentioned values to specify whether a result is supported.

● For a general result or occurrence	
Value	Description
<input type="checkbox"/> M_SUPPORTED	Returns whether the specified inquire type is supported. (summarize)
	<i>ResultType info</i> Return values: M_NULL The result is not supported. Value != 0 The result is supported.

Accepts the address of one of the following (see above for specifics on which is expected):

- double
- long
- M_NULL
- MIL_DOUBLE
- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address in which to return the value of the inquired setting. Since this function also returns the requested information, you can set this parameter to M_NULL.

Return value

The returned value is the requested information cast to *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeModel

Synopsis

Add, find, or delete a code model within a code context.

Syntax

```
MIL_ID McodeModel(  
    MIL_ID CodeContextId,  
    MIL_INT Operation,  
    MIL_INT CodeType,  
    MIL_INT Instance,  
    MIL_INT ControlFlag,  
    MIL_ID *CodeModelIdPtr  
)
```

Description

This function adds a code model to a code context, finds a code model of a particular type in a code context, or deletes a code model from a code context.

A code context can contain multiple models of 1D code types (excluding RSS, Planet, and Postnet); for 2D and composite code types, a code context can contain at most one code model. Contexts with multiple models are only supported for read and write operations ([McodeRead\(\)](#) and [McodeWrite\(\)](#)). Verify operations ([McodeVerify\(\)](#)) require a context with a single model. You can add multiple code models to the code context by calling **McodeModel()** multiple times.

Use [McodeControl\(\)](#) and [McodeInquire\(\)](#) to configure or retrieve information about a code model or code context.

The added code models cannot be individually freed using [McodeFree\(\)](#). They are either automatically freed when their context is freed, or when they are removed from the context by calling **McodeModel()** with [M_DELETE](#).

Parameters

CodeContextId

Specifies the code context in which to add, find, or delete the code model. The code context must have been previously allocated on the required system using [McodeAlloc\(\)](#).

Operation

Specifies the operation to perform within the specified code context. This parameter must be set to one of the following values:

● For specifying the type of operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ADD	Adds a new code model to the code context. Specify the code type of the code model using the CodeType parameter. (summarize)
<input type="checkbox"/> M_DELETE	Deletes a code model within the code context, whose model identifier is specified by the CodeModelIdPtr parameter. For a M_DELETE operation, the CodeType and Instance parameters must be passed M_NULL . (summarize)
<input type="checkbox"/> M_FIND	Finds a code model within the code context, whose code type is specified by the CodeType parameter. If there are many code models having the same specified code type, you must specify which instance of the specified type using the Instance parameter. (summarize)

CodeType

Specifies the code type of the code model to add or find from the code context. Note that if the [Operation](#) parameter is set to [M_DELETE](#), set this parameter to **M_NULL**. Possible values for [CodeType](#) are listed below.

To find a code model of any type, the [CodeType](#) parameter must be set to the following:

● For finding code models of any type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ANY	Matches all code types. This value can only be used when the Operation parameter is set to M_FIND . (summarize)

To specify the type of a 1D code model to add or find, the [CodeType](#) parameter can be set to one of the following:

● For specifying a 1D code type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BC412	Sets the code type to BC412.
<input type="checkbox"/> M_CODABAR	Sets the code type to Codabar.
<input type="checkbox"/> M_CODE39	Sets the code type to Code 39.
<input type="checkbox"/> M_CODE93	Sets the code type to Code 93.
<input type="checkbox"/> M_CODE128	Sets the code type to Code 128.
<input type="checkbox"/> M_EAN8	Sets the code type to EAN-8.
<input type="checkbox"/> M_EAN13	Sets the code type to EAN-13.
<input type="checkbox"/> M_INTERLEAVED25	Sets the code type to Interleaved 25. M_INTERLEAVED25 is best used in a fixed-length application, with all reading equipment programmed to accept messages only of the correct length. (summarize)
<input type="checkbox"/> M_PHARMACODE	Sets the code type to Pharmacode.
<input type="checkbox"/> M_PLANET	Sets the code type to Planet.
<input type="checkbox"/> M_POSTNET	Sets the code type to Postnet.
<input type="checkbox"/> M_RSSCODE	Sets the code type to RSS code.
<input type="checkbox"/> M_UPC_A	Sets the code type to UPC-A.
<input type="checkbox"/> M_UPC_E	Sets the code type to UPC-E.

To specify the type of a 2D code model to add or find, the [CodeType](#) parameter can be set to one of the following:

● For specifying a 2D code type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DATAMATRIX	Sets the code type to Data Matrix.
<input type="checkbox"/> M_MAXICODE	Sets the code type to Maxicode.
<input type="checkbox"/> M_MICROPDF417	Sets the code type to MicroPDF417.
<input type="checkbox"/> M_PDF417	Sets the code type to PDF417.
<input type="checkbox"/> M_QRCODE	Sets the code type to QR code.
<input type="checkbox"/> M_TRUNCATED_PDF417	Sets the code type to Truncated PDF417.

To specify that the code model to add or find is a composite code, the [CodeType](#) parameter must be set to the following:



● For specifying a composite code type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_COMPOSITECODE	Sets the code type to composite code. The code type is a composite of a 1D and a 2D code type; the particular combination of code types is determined by the specified encoding scheme.

	(summarize)
--	-----------------------------

Instance

Specifies the instance of the code model to find in the context. Note that if the [Operation](#) parameter is set to [M_ADD](#) or [M_DELETE](#), set this parameter to **M_NULL**.

Each code model is associated to a unique identifier, starting at zero, when it is added to a code context, this is a model's instance. If a code type is specified using the [CodeType](#) parameter, the [Instance](#) parameter must indicate the instance, starting at zero, of that particular code type.

● For specifying the instance of a code model	
 Value	Description
 Value	Specifies the instance of the individual code model on which to perform the operation.

ControlFlag

Specifies the function's control flag. This parameter must be set to **M_DEFAULT**.

CodeModelIdPtr

Specifies the address in which to return the code model identifier for [M_ADD](#) and [M_FIND](#) operations. For a [M_DELETE](#) operation, [CodeModelIdPtr](#) specifies the identifier of the model to be deleted. For [M_ADD](#) and [M_FIND](#) operations, you can set this parameter to **M_NULL**, since this function also returns the code model identifier.

Return value

The returned value is the identifier of the code model for a successful [M_ADD](#) operation; if the model could not be added, **M_NULL** is returned. The returned value is the identifier of the code model for a successful [M_FIND](#) operation; **M_NULL** is returned if no code model is found. The returned value is **M_NULL** for an [M_DELETE](#) operation.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeRead

Synopsis

Read the code(s) in an image as specified in the code context.

Syntax

```
void McodeRead(
    MIL_ID CodeContextId,
    MIL_ID ImageBufId,
    MIL_ID CodeResultId
)
```

Description

This function searches for the code(s) in an image, as specified in the code context, and writes the results in the designated results buffer. The code context's control settings determine how to perform the operation. Retrieve results using [McodeGetResult\(\)](#) or [McodeGetResultSingle\(\)](#).

It is strongly recommended to use child buffers for read operations, especially if your image contains more codes than your context is configured to read; otherwise, MIL will randomly select the specified number of codes to read. Using child buffers is also recommended because it achieves a fast and robust read operation if your image contains other information besides the codes.

Before performing a read operation, certain controls might have to be set globally for the entire code context or individually for each code model in [McodeControl\(\)](#), specifically:

Control	Notes
M_CELL_NUMBER_X , M_CELL_NUMBER_Y	For 2D code types only, improves the robustness of the operation.
M_CELL_SIZE_MIN , M_CELL_SIZE_MAX	To improve the robustness of read operations for some code types, especially Data Matrix and Maxicode.
	Note that MIL might have difficulty reading codes if the cell size is less than 2 pixels, even if the size is specified.
M_ENCODING	For code types where M_ANY is not supported.
M_ERROR_CORRECTION	For code types where M_ANY is not supported.
M_FOREGROUND_VALUE	This control is essential for all code types, and the image will not be decoded if the foreground value is not correctly set.
M_SEARCH_ANGLE , M_SEARCH_ANGLE_DELTA_NEG , M_SEARCH_ANGLE_DELTA_POS	If the code to read is not within 5° of the horizontal axis, and/or if the code has bearer bars.
M_STRING_SIZE_MIN and M_STRING_SIZE_MAX	For M_BC412 , the string size must be specified.
M_SUB_TYPE	For M_COMPOSITECODE and M_RSSCODE , specifying the code sub-type can help increase the speed of read operations.

Parameters

CodeContextId

Specifies the identifier of the code context.

ImageBufId

Specifies the image buffer in which to search. This buffer must be 8-bit unsigned.

CodeResultId

Specifies the result buffer in which to write the results of the read.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The code result buffer ([CodeResultId](#)) must be allocated on the same system as the code context ([CodeContextId](#)). If it is not, an error will occur.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.

DLL

Requires mil.dll; milcode.dll.

McodeRestore

Synopsis

Restore a code context previously saved to a file.

Syntax

```
MIL_ID McodeRestore(  
    MIL_CONST_TEXT_PTR Filename,  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *CodeContextIdPtr  
)
```

Description

This function restores a code context that has been saved to a file using, [McodeSave\(\)](#), [McodeStream\(\)](#). It can also restore the predefined semi code contexts available in MIL by specifying their file names (for example, *SEMI_T1-95r0303.mfo*). For more information, refer to the [Predefined SEMI code contexts](#) subsection in the [Supported code types](#) section in [Chapter 13: Codes](#).

This function restores all of the code context's controls and models that were in effect when the code context was saved.

Parameters

Filename

Specifies the name and path of the file from which to restore the code context. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

● For specifying the file name and path	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)
	<div>Parameters</div> <div>FileName</div> Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.

SystemId

Specifies the system on which to allocate the context. This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Specifies the function's control flag. This parameter must be set to **M_DEFAULT**.

CodeContextIdPtr

Specifies the address of the variable in which to return the identifier of the code context. Since the function also returns the identifier, this parameter can be set to **M_NULL**.

Return value

The returned value is the code context identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeSave

Synopsis

Save the specified code context in a file.

Syntax

```
void McodeSave(
    MIL_CONST_TEXT_PTR FileName,
    MIL_ID CodeContextId,
    MIL_INT ControlFlag
)
```

Description

This function saves a code context in a file so that it can be reloaded with [McodeRestore\(\)](#) or [McodeStream\(\)](#).

Parameters

FileName

Specifies the name and path of the file in which to save the code context. It is recommended that you use the MCO file extension for easier use with other Matrox Imaging software products. The function internally handles the opening and closing of this file. If this file already exists, it will be overwritten.

This parameter can be set to one of the following values:

For specifying the file name and path					
<input type="checkbox"/> Value	Description				
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p> <table><tr><th colspan="2">Parameters</th></tr><tr><td><i>FileName</i></td><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters		<i>FileName</i>	Specifies the drive, directory, and name of the file.
Parameters					
<i>FileName</i>	Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</p>				

CodeContextId

Specifies the identifier of the data buffer to save.

ControlFlag

Specifies the function's control flag This parameter must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeStream

Synopsis

Load, restore, or save a code context from/to a file or a memory stream, or save a verification report to a file.

Syntax

```
void McodeStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *CodeContextOrResultIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load or restore a code context from a file or memory stream. Moreover, this function can also save a code context to a file or memory stream, and save a verification report to a file.

To inquire the number of bytes necessary to save a code context to memory stream, you should call this function (**McodeStream()**) first with **M_INQUIRE_SIZE_BYTE**.

The content saved to memory stream is equivalent to the content saved to file. In addition, any file saved using this function is equivalent to a file saved with **McodeSave()**.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using **McodeStream()**, you can choose to save a backwards-compatible version of the code context, which will work using a specified version of MIL that is up to one major release older than the current version. For example, if you allocate a code context using MIL 9.0 and save it to version 8.0, you can restore this context on a computer where MIL 8.0 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore a code context saved using MIL version 7.0 or above. Settings that do not exist in the lower version will be filled with default values when the code context is loaded or restored using the current version.

Parameters

MemPtrOrFileName

Specifies the file or memory stream.

● For specifying the file name or memory stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file (for example, "C:\mydirectory\myfile") when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving a code context to a file, use the MCO file extension. The function internally handles the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p> <p>To open or save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:////" (for example, "remote:////C:\mydirectory\myfile").</p> <p>(summarize)</p>
	<div>Parameters</div> <div>FileName</div> <p>Specifies the drive, directory, and name of the file.</p>
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows]

	Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE .
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. The parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire context. To ascertain the required size, call this function (McodeStream()) first with M_INQUIRE_SIZE_BYTE . (summarize)

SystemId

Specifies the system on which to restore the code context.

For an [M_RESTORE](#) operation, this parameter should be set to one of the following values:

● For a restore operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

For an [M_SAVE](#) and [M_SAVE_REPORT](#) operation, this parameter should be set to the following value:

● For a save report or save operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NULL	Specifies that this parameter is not applicable.

Operation

Specifies the operation to perform on the code context. This parameter must be set to one of the following values:

● For specifying the operation to perform on the code context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save a code context to memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated code context.
<input type="checkbox"/> M_RESTORE	Restores a code context from a file or memory stream and assigns a MIL identifier to this context.
<input type="checkbox"/> M_SAVE	Saves a code context to a specified file or memory stream.
<input type="checkbox"/> M_SAVE_REPORT	Saves a report containing most of the results from a verify operation as a flat text file. This operation is only supported when the StreamType parameter is set to M_FILE . (summarize)

StreamType

Specifies the type of stream in which to store/from which to restore the code context. This parameter must be set to one of the following values:

● For specifying the type of stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream. Note that this is the only value available when performing an M_SAVE_REPORT operation. (summarize)

<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)
------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------

Version

Specifies the MIL version of the code context. This parameter must be set to one of the following values.

● For specifying the MIL version	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL. For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. For an M_SAVE_REPORT operation, no version information is stored in the text file. This is the only possible setting for this operation. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Specifies the function's control flag and should be set to **M_DEFAULT**.

CodeContextOrResultIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the code context.

For [M_INQUIRE_SIZE_BYTE](#), [M_SAVE](#) operations, **CodeContextOrResultIdPtr** specifies the address of the variable from which to read the code context identifier.

For an [M_LOAD](#) operation, **CodeContextOrResultIdPtr** specifies the address of the variable from which to read the identifier of the code context in which to load the file or memory stream content.

For an [M_RESTORE](#) operation, **CodeContextOrResultIdPtr** specifies the address of the variable from which to return the identifier of the restored code context. If the operation is not successful, **M_NULL** is returned.

For an [M_SAVE_REPORT](#) operation, **CodeContextOrResultIdPtr** specifies the address in which to read the identifier of the code results buffer. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the code context, in bytes. If the size is not required, or if performing an [M_SAVE_REPORT](#) operation, set this parameter to **M_NULL**.

Note that the size of a code context will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeVerify

Synopsis

Compute the different quality-grades of the code in the specified source image.

Syntax

```
void McodeVerify (
    MIL_ID CodeContextId,
    MIL_ID ImageBufId,
    MIL_CONST_TEXT_PTR String,
    MIL_INT CodeResultId
)
```

Description

This function computes the different quality-grades of the code in the specified source image.

You can read the results of the verify operation with [McodeGetResult\(\)](#).

Note that this function requires a code context with a single model and is not supported for Pharmacode, Postnet, and Planet code types.

Since the global context settings are required to perform a verify operation, **McodeVerify()** requires that you pass a code context identifier, and not a code model identifier, as a parameter. The code context passed to **McodeVerify()** must contain a single code model because certain code model settings are also required to perform a verify operation. For more information on code model and code context settings required by **McodeVerify()**, see the [Customizing read and verify operation settings](#) section in [Chapter 13: Codes](#).

Before performing a verify operation, you might have to set some controls.

Control	Notes
M_ENCODING	Specifies the type of encoding scheme for the code. This control has to be set for the code types where M_ANY is not supported.
M_ERROR_CORRECTION	Specifies the type of error correction for the code. This control has to be set for the code types where M_ANY is not supported.
M_FOREGROUND_VALUE	Specifies the foreground value for the code model. This control is essential and the code will not be graded if the foreground value is not correctly set.
M_SEARCH_ANGLE, M_SEARCH_ANGLE_DELTA_NEG, M_SEARCH_ANGLE_DELTA_POS	Specify the nominal search angle and the range to search in around the nominal search angle. This control has to be specified if the code to read is not within the angular range of -5 to 5 degrees.
M_STRING_SIZE	Specifies the size of the string to be decoded. This control has to be specified for the BC412 code type.

This function is not supported for Pharmacode, Postnet, and Planet code types.

You should note that if the code cannot be decoded by MIL, all verify grades will be returned as F.

If you cannot read a bar code with a cell size that is size 2 or smaller, you can enlarge the image using [MimResize\(\)](#) and try again.

Parameters

CodeContextId

Specifies the identifier of the code context.

ImageBufId

Specifies the identifier of the image buffer containing the string to be verified.

String

Reserved for future expansion and must be set to **M_NULL**.

CodeResultId

Specifies the result buffer in which to write the results of the verify operation. This parameter must be set to **M_DEFAULT**.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The code result buffer ([CodeResultId](#)) must be allocated on the same system as the code context ([CodeContextId](#)). If it is not, an error will occur.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

McodeWrite

Synopsis

Encode an ASCII string into a symbol.

Syntax

```
void McodeWrite(
    MIL_ID CodeModelId,
    MIL_ID ImageBufId,
    MIL_CONST_TEXT_PTR String,
    MIL_INT ControlFlag
)
```

Description

This function encodes a null-terminated ASCII string into a symbol (code) using the code type and encoding scheme, specified by the code model, and saves the symbol into an image. For a given code model and string, you can also use this function, with [McodeInquire\(\)](#), to inquire about the minimum buffer size required to encode the string.

The control type settings of the specified code model determine how to perform the operation. Before performing a write operation, you must set certain controls, using [McodeControl\(\)](#), specifically:

Control	Notes
M_CELL_NUMBER_X , M_CELL_NUMBER_Y	Specifies the number of cells in the X and Y directions. Improves the robustness of the operation, for 2D code types only. If these settings are set to M_DEFAULT , then the number of cells will be selected automatically so as to minimize the size of the resulting code.
M_CELL_SIZE_MIN	Specifies the cell size, in pixels, of a unit of the code. If this setting is set to M_DEFAULT , the code will be written so as to just fit into the destination image.
M_ENCODING	Specifies the encoding scheme. This control type must be set for code types where M_ANY is not supported as the encoding scheme.
M_ERROR_CORRECTION	Specifies the type of error correction. This control type must be set for code types where M_ANY is not a supported error correction type.
M_FOREGROUND_VALUE	Specifies the foreground color of the code. This control type is essential for all code types.

Parameters

CodeModelId

Specifies the identifier of the code model.

ImageBufId

Specifies the image buffer in which to write the symbol. Set this parameter to one of the following.

For specifying the image buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NULL	Allows you to inquire about the minimum buffer size required. Using this value means that McodeWrite() will not write into an image buffer. After calling McodeWrite() with M_NULL , call McodeInquire() to retrieve the minimum width (M_WRITE_SIZE_X) and height (M_WRITE_SIZE_Y) required for the image buffer. (summarize)
<input type="checkbox"/> MIL image buffer identifier	Specifies that McodeWrite() will write into the image buffer, and in which image buffer to write. This buffer must be 8-bit unsigned. In addition, it should be large enough to hold the symbol. (summarize)

String

Specifies the address of the string to be encoded.



● For specifying the string of the code							
☐ Value	Description						
☐ MIL_TEXT(MIL_TEXT_PTR String)	<p>Specifies the address of the string. You need to know the string specifications of the code to write.</p> <p>Note the following restrictions for certain code types:</p> <ul style="list-style-type: none"> • If the code type is a Codabar code, the string must be numeric, but can contain the following ASCII characters: minus (-), dollar sign (\$), colon (:), forward slash (/), period (.), and plus (+). In addition, the string must start and end with any of the following characters: a, b, c, or d. • If the code type is an Interleaved 2 of 5 code, the string must have an even number of characters. • If the code type is a Maxicode code, with M_ENCODING set to M_ENC_MODE2 or M_ENC_MODE3, the string must respect the structured carrier message format (the portion of the string that contains postal code, country code, and class of service information). • If the code type is a composite code, the string must be passed in the following format: 1D string 2D string, where is used as the separator. • If the code type is a composite code, the FNC1 separator is represented by the unprintable character with ASCII value 29. <p>In addition, add parentheses around the application identifiers in your string. Note that this is required by the write operation; the parentheses are not actually coded.</p> <p>(summarize)</p> <table> <tr> <td></td><td>Parameters</td></tr> <tr> <td></td><td>String</td></tr> <tr> <td></td><td>Specifies the address of the string.</td></tr> </table>		Parameters		String		Specifies the address of the string.
	Parameters						
	String						
	Specifies the address of the string.						

ControlFlag

Specifies the function's control flag.

● For specifying the function's control flag	
☐ Value	Description
☐ M_DEFAULT	<p>Specifies that the string to encode does not use ASCII character codes for unprintable characters. Unprintable characters are not encoded.</p> <p>(summarize)</p>
☐ M_ESCAPE_SEQUENCE	<p>Specifies that the string to encode uses ASCII character codes for unprintable characters.</p> <p>You must use a code type that supports unprintable characters that are specified using ASCII codes. When typing the string to encode, the unprintable character's ASCII character code must be in hexadecimal format, preceded by \x (for example, \x0D for ASCII 13, which is the carriage return).</p> <p>Note that if you want to encode the "\"" character in escape sequence mode, type "\\\".</p> <p>(summarize)</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcode.lib.
DLL	Requires mil.dll; milcode.dll.

Mcol functions

Synopsis

The functions prefixed with Mcol make up the Color Analysis module. The Color Analysis module allows you to perform color-based operations such as matching, projection, and distance. You can use these operations to quickly design powerful applications that can identify colors, segment colors (supervised), detect defects, enhance color-to-grayscale conversion, separate superimposed colors, and measure the relative presence of a color in an image.

Functions

- [McolAlloc](#)
- [McolAllocResult](#)
- [McolControl](#)
- [McolDefine](#)
- [McolDistance](#)
- [McolDraw](#)
- [McolFree](#)
- [McolGetResult](#)
- [McolInquire](#)
- [McolMask](#)
- [McolMatch](#)
- [McolPreprocess](#)
- [McolProject](#)
- [McolRestore](#)
- [McolSave](#)
- [McolSetMethod](#)
- [McolStream](#)

McolAlloc

Synopsis

Allocate a color matching context.

Syntax

```
MIL_ID McolAlloc(  
    MIL_ID SystemId,  
    MIL_INT ContextType,  
    MIL_INT WorkingColorSpace,  
    MIL_ID ColorProfileId,  
    MIL_INT ControlFlag,  
    MIL_ID *ContextIdPtr  
)
```

Description

This function allocates a color matching context on the specified system. A color matching context contains all the information necessary to perform [McolMatch\(\)](#), including global context settings and the individual color-samples. When matching, color data is considered to be within the context's color space ([WorkingColorSpace](#)). When the color matching context is no longer required, you should release its memory, using [McolFree\(\)](#).

You can add color-samples to a color matching context using [McolDefine\(\)](#). To adjust context and color-sample settings, use [McolControl\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the color matching context. This parameter should be set to one of the following values:

For specifying the system	
Value	Description
M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ContextType

Specifies the type of color context. This parameter must be set to the following value:

For specifying the context	
Value	Description
M_COLOR_MATCHING	Specifies a context that will be used for color matching.

WorkingColorSpace

Specifies the context's color space.

This parameter should be set to one of the following values:

For specifying the color space	
Value	Description

<input type="checkbox"/> M_DEFAULT	Same as M_RGB .
<input type="checkbox"/> M_CIELAB	Specifies a CIELAB (or LAB) color space. This is based on the color's luminance (L), its position between red and green (A), and its position between yellow and blue (B). In this case, you must provide the color component (band) information of color buffers in the following order: L, A, B. (summarize)
<input type="checkbox"/> M_HSL	Specifies an HSL color space. This is based on the color's hue (H), saturation (S), and luminance (L). In this case, you must provide the color component (band) information of color buffers in the following order: H, S, L. (summarize)
<input type="checkbox"/> M_RGB	Specifies an RGB color space. This is based on the color's red (R), green (G), and blue (B) properties. In this case, you must provide the color component (band) information of color buffers in the following order: R, G, B. When setting the color matching strategy with McolSetMethod() , you can also specify that, for internal calculations, you want to convert your RGB data to CIELAB before the match. (summarize)

ColorProfileId

Specifies the identifier of the context's reference color space. This is the standard used to interpret the color space data.

This parameter must be set to the following value:

● For specifying the context's color space profile	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies to use standard RGB specifications (sRGB), as defined by the International Electrotechnical Commission (IEC) Project Team 61966-2-1.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the color matching context identifier. Since **McolAlloc()** also returns the color matching context identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the color matching context's identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolAllocResult

Synopsis

Allocate a color matching result buffer.

Syntax

```
MIL_ID McolAllocResult(  
    MIL_ID SystemId,  
    MIL_INT ResultType,  
    MIL_INT ControlFlag,  
    MIL_ID *ContextIdPtr  
)
```

Description

This function allocates a result buffer, on the specified system, to store the results from an [McolMatch\(\)](#) operation. You can read the results from the result buffer, using [McolGetResult\(\)](#). When the result buffer is no longer required, release its memory, using [McolFree\(\)](#).

Parameters

SystemId
Specifies the system on which to allocate the result buffer. This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ResultType
Specifies the type of result buffer to allocate. This parameter can be set to one of the following values:

● For specifying the type of result buffer to allocate	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_COLOR_MATCHING_RESULT .
<input type="checkbox"/> M_COLOR_MATCHING_RESULT	Specifies to allocate a result buffer for color matching results.

ControlFlag
Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr
Specifies the address of the variable in which to write the color matching result identifier. If allocation fails, **M_NULL** is returned as the identifier. Since [McolAllocResult\(\)](#) also returns the color matching result identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolControl

Synopsis

Control a color matching context, or color-samples contained therein.

Syntax

```
void McolControl(  
    MIL_ID ColorObjectId,  
    MIL_INT IndexOrLabel,  
    MIL_INT ControlType,  
    MIL_DOUBLE ControlValue  
)
```

Description

This function sets the specified control type for a color matching context, or for one or all of the color-samples contained therein. These control type settings affect the execution of [McolMatch\(\)](#), and also [McolDraw\(\)](#). Typically, you can inquire the control type setting, using [McolInquire\(\)](#).

Changing certain control type settings require you to preprocess the color matching context again, using [McolPreprocess\(\)](#). To know if the color matching context needs to be preprocessed, call [McolInquire\(\)](#) with the [M_PREPROCESSED](#) inquire type.

Parameters

- ColorObjectId
- Specifies the identifier of the color matching context to control. The color matching context must have been previously allocated on the required system using [McolAlloc\(\)](#).
- IndexOrLabel
- Specifies the color matching context, or color-samples therein, to control. Set this parameter to one of the following values.

For specifying a context or color-sample	
Value	Description
M_DEFAULT	Same as M_CONTEXT .
M_SAMPLE_INDEX(MIL_INT SampleIndex)	<div>Specifies the index of an existing color-sample to which to apply the control type setting. (summarize)</div> <div>Parameters</div> <div>SampleIndex</div> <div>Specifies the index. The index must be greater than or equal to 0.</div>
M_SAMPLE_LABEL(MIL_INT SampleLabel)	<div>Specifies the label of an existing color-sample to which to apply the control type setting. (summarize)</div> <div>Parameters</div> <div>SampleLabel</div> <div>Specifies the label. The label must be greater than 0.</div>
M_ALL	<div>Applies the specified control setting to all the color-samples contained within the color matching context. In this case, the control type setting must be supported by all the color-samples. (summarize)</div>
M_CONTEXT	Controls a global setting of the color matching context.

ControlType

Specifies the setting to change.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the setting's new value.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For controlling general context settings](#)
- [For controlling context settings that are used for color space encoding](#)
- [For controlling color-samples defined in a color matching context](#)

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control general color matching context settings. In this case, you must set the **IndexOrLabel** parameter to [M_CONTEXT](#).

For controlling general context settings	
ControlType	Description
ControlValue	
M_ACCEPTANCE	<p>Sets the acceptance level for the color-sample's score (M_SCORE with McolGetResult()). This score indicates the similarity between the color of the color-sample and the color of the target area. The higher the acceptance, the closer the colors must be for them to match.</p> <p>For a match, the color-sample must have a score that is greater than or equal to this level.</p> <p>(summarize)</p>
M_DEFAULT	<p>Specifies the default value. The default value is 0.0 percent.</p> <p>(summarize)</p>
0.0 to 100.0	Specifies an acceptable score, in percent.
M_ACCEPTANCE_RELEVANCE	<p>Sets the acceptance level for the target area's relevance score (M_SCORE_RELEVANCE). This score indicates the significance (relevance) of the match score (M_SCORE). In statistics, this is similar to the confidence level. The higher the relevance acceptance, the more significant the match score results must be (relative to other possible results), to have a successful match for the target area.</p> <p>A target area can only have a successful matching status (M_STATUS with McolGetResult()) if its relevance score is above the relevance acceptance level. A target area with a successful matching status also requires at least one color-sample score to be above M_ACCEPTANCE.</p> <p>(summarize)</p>
M_DEFAULT	<p>Specifies the default value. The default value is 0.0 percent.</p> <p>(summarize)</p>
0.0 to 100.0	Specifies an acceptable relevance score, in percent.
M_BACKGROUND_DRAW_COLOR	<p>Sets the color to use to draw background pixels when drawing or generating the appropriate images, using McolDraw() or McolMatch(). M_BACKGROUND_DRAW_COLOR is applied to the colored images (M_DRAW_COLORED_....), the distance image (M_DRAW_DISTANCE_IMAGE), and to the grayscale label images (M_DRAW_LABEL_....).</p> <p>A pixel is considered background when it is outside the target areas. For background pixels to exist, you must specify an area identifier image, using McolMatch().</p> <p>(summarize)</p>
M_DEFAULT	Same as M_TRANSPARENT .
M_RGB888(MIL_INT ComponentBand0, MIL_INT ComponentBand1, MIL_INT ComponentBand2)	<p>Specifies an 8-bit color value with which to set the background pixels. The color value is taken as is (no conversion), according to the working color space. Typically, this should be RGB, since the background color is meant for display. If required, color values are remapped to 8-bit.</p> <p>When M_BACKGROUND_DRAW_COLOR is set with M_RGB888(), only the first component (for example, red) will be used to draw or generate</p>

	<p>M_DRAW_DISTANCE_IMAGE, which is a grayscale image. (summarize)</p> <table><tr><th>Parameters</th></tr><tr><td><p><i>ComponentBand0</i></p><p>Specifies the first component, as a value between 0 and 255.</p></td></tr><tr><td><p><i>ComponentBand1</i></p><p>Specifies the second component, as a value between 0 and 255.</p></td></tr><tr><td><p><i>ComponentBand2</i></p><p>Specifies the third component, as a value between 0 and 255.</p></td></tr></table>	Parameters	<p><i>ComponentBand0</i></p> <p>Specifies the first component, as a value between 0 and 255.</p>	<p><i>ComponentBand1</i></p> <p>Specifies the second component, as a value between 0 and 255.</p>	<p><i>ComponentBand2</i></p> <p>Specifies the third component, as a value between 0 and 255.</p>
Parameters					
<p><i>ComponentBand0</i></p> <p>Specifies the first component, as a value between 0 and 255.</p>					
<p><i>ComponentBand1</i></p> <p>Specifies the second component, as a value between 0 and 255.</p>					
<p><i>ComponentBand2</i></p> <p>Specifies the third component, as a value between 0 and 255.</p>					
<input type="checkbox"/> M_TRANSPARENT	Specifies that the value of the background pixels will be left as is.				
<input type="checkbox"/> Value > 0	<p>Specifies the value of the background pixels, as a <i>double</i> or <i>integer</i>.</p> <p>When you set M_BACKGROUND_DRAW_COLOR to a specific numerical value, its "color" will be the corresponding grayscale value.</p> <p>When drawing the distance image (M_DRAW_DISTANCE_IMAGE) in a 32-bit integer buffer, you should keep the M_BACKGROUND_DRAW_COLOR value under 24-bit; that is, less than 16,777,215. (summarize)</p>				
<input type="checkbox"/> M_BAND_MODE	Specifies the color bands to use when performing a match. This applies to all color-samples in the context and to the target areas. (summarize)				
<input type="checkbox"/> M_DEFAULT	Same as M_ALL_BANDS .				
<input type="checkbox"/> M_ALL_BANDS	Specifies to use all bands.				
<input type="checkbox"/> M_COLOR_BAND_0	Specifies to use band 0.				
<input type="checkbox"/> M_COLOR_BAND_0 + M_COLOR_BAND_1	Specifies to use bands 0 and 1.				
<input type="checkbox"/> M_COLOR_BAND_0 + M_COLOR_BAND_2	Specifies to use bands 0 and 2.				
<input type="checkbox"/> M_COLOR_BAND_1	Specifies to use band 1.				
<input type="checkbox"/> M_COLOR_BAND_1 + M_COLOR_BAND_2	Specifies to use bands 1 and 2.				
<input type="checkbox"/> M_COLOR_BAND_2	Specifies to use band 2.				
<input type="checkbox"/> M_DISTANCE_IMAGE_NORMALIZE	<p>Sets the normalization factor when drawing or generating M_DRAW_DISTANCE_IMAGE, using McolDraw() or McolMatch().</p> <p>After normalization, distances are saturated according to the distance buffer type. For <i>integer</i> buffers, distances are remapped according to the buffer's possible range of values. For <i>floating-point</i> buffers, the normalized values are left between 0 and 1. (summarize)</p>				
<input type="checkbox"/> M_DEFAULT	Same as M_NO_NORMALIZE .				
<input type="checkbox"/> M_MAX_NORMALIZE	Specifies to normalize the distances by the maximum distance calculated by the distance image (M_DRAW_DISTANCE_IMAGE). This value corresponds to the M_MAX_DISTANCE result (McolGetResult()). (summarize)				
<input type="checkbox"/> M_NO_NORMALIZE	Specifies no normalization.				
<input type="checkbox"/> Value > 0.0	Specifies the normalization factor.				
<input type="checkbox"/> M_DISTANCE_TOLERANCE_MODE	Sets the strategy with which to calculate M_DISTANCE_TOLERANCE . (summarize)				
<input type="checkbox"/> M_DEFAULT	Same as M_ABSOLUTE .				
<input type="checkbox"/> M_ABSOLUTE	Specifies a tolerance strategy based on an absolute distance value. In this case, the M_DISTANCE_TOLERANCE value is applied as is. (summarize)				
<input type="checkbox"/> M_RELATIVE	Specifies a tolerance strategy based on a relative distance between all color-samples. In this case, the smallest distance between each color-sample all the other color-samples is computed internally; M_DISTANCE_TOLERANCE is multiplied by half this distance. (summarize)				
<input type="checkbox"/> M_SAMPLE_STDDEV	Specifies a tolerance strategy based on the color-sample's standard deviation. In this case, M_DISTANCE_TOLERANCE is multiplied by the standard deviation of				

	the color-sample using the current color distance. Note that this standard deviation value is always greater than or equal to 1. (summarize)				
<div><div></div><div>M_GENERATE_DISTANCE_IMAGE</div></div>	Sets whether to generate the distance image and store it in the result buffer. This image contains the difference between the color of the target area and the color of its best-matched color-sample. (summarize)				
<div><div></div><div>M_DEFAULT</div></div>	Same as M_DISABLE .				
<div><div></div><div>M_DISABLE</div></div>	Specifies not to generate the image.				
<div><div></div><div>M_ENABLE</div></div>	Specifies to generate the image.				
<div><div></div><div>M_GENERATE_LABEL_PIXEL_IMAGE</div></div>	Sets whether to generate the image containing the label of the color-sample for which each pixel voted. This image is only available if you use McolSetMethod() with M_MIN_DIST_VOTE . (summarize)				
<div><div></div><div>M_DEFAULT</div></div>	Same as M_DISABLE .				
<div><div></div><div>M_DISABLE</div></div>	Specifies not to generate the image.				
<div><div></div><div>M_ENABLE</div></div>	Specifies to generate the image.				
<div><div></div><div>M_GENERATE_SAMPLE_COLOR_LUT</div></div>	Sets whether to generate the color-sample label LUT, in the result buffer. This image contains the label value of each color-sample and its associated average color. (summarize)				
<div><div></div><div>M_DEFAULT</div></div>	Same as M_DISABLE .				
<div><div></div><div>M_DISABLE</div></div>	Specifies not to generate the LUT.				
<div><div></div><div>M_ENABLE</div></div>	Specifies to generate the LUT.				
<div><div></div><div>M_OUTLIER_DRAW_COLOR</div></div>	<p>Sets the color to use to draw outlier pixels when drawing or generating the appropriate images, using McolDraw() or McolMatch(). M_OUTLIER_DRAW_COLOR is applied to the colored images (M_DRAW_COLORED_...) and the distance image (M_DRAW_DISTANCE_IMAGE).</p> <p>Outlier pixels are pixels inside a target area that do not match with any color-sample. (summarize)</p>				
<div><div></div><div>M_DEFAULT</div></div>	Specifies the default value. The default value is 0. (summarize)				
<div><div><div><div></div><div>M_RGB888(MIL_INT <i>ComponentBand0</i>, MIL_INT <i>ComponentBand1</i>, MIL_INT <i>ComponentBand2</i>)</div></div></div></div>	<p>Specifies an 8-bit color value with which to set the outlier pixels. The color value is taken as is (no conversion), according to the working color space. Typically, this should be RGB, since the outlier color is meant for display. If required, color values are remapped to 8-bit.</p> <p>When M_OUTLIER_DRAW_COLOR is set with M_RGB888(), only the first component (for example, red) will be used to draw or generate M_DRAW_DISTANCE_IMAGE, which is a grayscale image. (summarize)</p> <table><tr><td><i>Parameters</i></td></tr><tr><td><i>ComponentBand0</i> Specifies the first component, as a value between 0 and 255.</td></tr><tr><td><i>ComponentBand1</i> Specifies the second component, as a value between 0 and 255.</td></tr><tr><td><i>ComponentBand2</i> Specifies the third component, as a value between 0 and 255.</td></tr></table>	<i>Parameters</i>	<i>ComponentBand0</i> Specifies the first component, as a value between 0 and 255.	<i>ComponentBand1</i> Specifies the second component, as a value between 0 and 255.	<i>ComponentBand2</i> Specifies the third component, as a value between 0 and 255.
<i>Parameters</i>					
<i>ComponentBand0</i> Specifies the first component, as a value between 0 and 255.					
<i>ComponentBand1</i> Specifies the second component, as a value between 0 and 255.					
<i>ComponentBand2</i> Specifies the third component, as a value between 0 and 255.					
<div><div></div><div>M_TRANSPARENT</div></div>	Specifies that the value of the outlier pixels will be left as is.				
<div><div></div><div>Value > 0</div></div>	<p>Specifies the value of the outlier pixel, as a <i>double</i> or <i>integer</i>.</p> <p>When you set M_OUTLIER_DRAW_COLOR to a specific numerical value, its color will be the corresponding grayscale value.</p> <p>When drawing the distance image (M_DRAW_DISTANCE_IMAGE) in a 32-bit <i>integer</i> buffer, you should keep the M_OUTLIER_DRAW_COLOR value under 24-bit; that is, less than 16,777,215. (summarize)</p>				
<div><div></div><div>M_OUTLIER_LABEL</div></div>	Sets the label to use for outlier pixels when drawing or generating the appropriate images, using McolDraw() or McolMatch() . M_OUTLIER_LABEL is applied to				

	<p>the grayscale label images (M_DRAW_LABEL_...).</p> <p>Outlier pixels are pixels inside a target area that do not match with any color-sample.</p> <p>The M_OUTLIER_LABEL value is returned as an M_BEST_MATCH_LABEL result when a target area does not have any matching color-sample.</p> <p>Note that the M_OUTLIER_LABEL setting does not represent a valid color-sample and you cannot pass it as a color-sample label to McolInquire(), McolGetResult(), or McolDraw(). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the label of the outlier pixels, as an <i>integer</i> . The label must be unique among existing labels, and must be less than 2^{24} . (summarize)
<input type="checkbox"/> M_PRECONVERT_GAMMA	<p>Specifies whether to perform gamma correction when converting colors (for example, RGB) to CIELAB before the match (McolSetMethod() with M_CIELAB). Gamma correction adjusts an image to compensate for the non-linear transformation between a pixel's value and its displayed intensity.</p> <p>If the driver or camera does not apply a transformation between the incoming light level and a pixel's value, you should disable M_PRECONVERT_GAMMA. This is typically the case when, for example, the RGB data from a camera is linear, with respect to the incoming light level. However, if a transformation is applied (for example, the RGB data from a camera is non-linear), you should enable M_PRECONVERT_GAMMA. For more information, see MdigControl() with M_GAMMA. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_ENABLE .
<input type="checkbox"/> M_DISABLE	Specifies not to perform gamma correction.
<input type="checkbox"/> M_ENABLE	Specifies to perform gamma correction.
<input type="checkbox"/> M_SAVE_AREA_IMAGE	Sets whether to save the area identifier image in the result buffer. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies not to save the image.
<input type="checkbox"/> M_ENABLE	Specifies to save the image.

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control color space encoding, according to the actual dynamic range of your color data. In this case, you must set the **IndexOrLabel** parameter to **M_CONTEXT**.

Color space encoding defines how color is transformed from the range represented in an image buffer to its native (theoretical) range. For example, RGB color space data is represented in an image buffer as values between 0 and 255 (8-bit); these values are then mapped to their native data range, which consists of all real numbers between 0 and 1.

The Color Analysis module allows you to perform this transformation automatically, using **M_ENCODING** with **M_nBIT**, which is typically sufficient. You can also use **M_USER_DEFINED** to explicitly control the transformation according to offset (**M_OFFSET_BAND_n**) and scale (**M_SCALE_BAND_n**) values. For more information, see the *Color space encoding* subsection in the *Advanced color matching settings* section in *Chapter 17: Color processing and analysis*.

● For controlling context settings that are used for color space encoding	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_ENCODING	Sets the color space encoding, which defines how color is transformed from the range represented in an image buffer to its native (theoretical) range. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_8BIT .
<input type="checkbox"/> M_8BIT	Specifies an internally defined color space encoding for 8-bit images.
<input type="checkbox"/> M_10BIT	Specifies an internally defined color space encoding for 10-bit images.
<input type="checkbox"/> M_12BIT	Specifies an internally defined color space encoding for 12-bit images.
<input type="checkbox"/> M_14BIT	Specifies an internally defined color space encoding for 14-bit images.
<input type="checkbox"/> M_16BIT	Specifies an internally defined color space encoding for 16-bit images.

<input type="checkbox"/> M_USER_DEFINED	Specifies that the color space encoding will be performed by M_OFFSET_BAND_n and M_SCALE_BAND_n .
<input type="checkbox"/> M_OFFSET_BAND_0	Sets the offset for the color space encoding, for band 0. This only applies if M_ENCODING is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> Value	Specifies the offset, as a <i>double</i> .
<input type="checkbox"/> M_OFFSET_BAND_1	Sets the offset for the color space encoding, for band 1. This only applies if M_ENCODING is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> Value	Specifies the offset, as a <i>double</i> .
<input type="checkbox"/> M_OFFSET_BAND_2	Sets the offset for the color space encoding, for band 2. This only applies if M_ENCODING is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> Value	Specifies the offset, as a <i>double</i> .
<input type="checkbox"/> M_SCALE_BAND_0	Sets the scale for the color space encoding, for band 0. This only applies if M_ENCODING is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> Value	Specifies the scale, as a <i>double</i> .
<input type="checkbox"/> M_SCALE_BAND_1	Sets the scale for the color space encoding, for band 1. This only applies if M_ENCODING is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> Value	Specifies the scale, as a <i>double</i> .
<input type="checkbox"/> M_SCALE_BAND_2	Sets the scale for the color space encoding, for band 2. This only applies if M_ENCODING is set to M_USER_DEFINED . (summarize)
<input type="checkbox"/> Value	Specifies the scale, as a <i>double</i> .

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control color-samples defined in the context. In this case, you must set the **IndexOrLabel** parameter to **M_SAMPLE_INDEX()** or **M_SAMPLE_LABEL()**.

● For controlling color-samples defined in a color matching context	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_DISTANCE_TOLERANCE	Sets the acceptable tolerance for the color distance between the color-sample and a target area. The tolerance ranges from no color difference to M_DISTANCE_TOLERANCE . The greater the tolerance, the greater the distance (difference) between the colors can be, for them to match. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_AUTO .
<input type="checkbox"/> M_AUTO	Specifies the tolerance automatically. If you set M_DISTANCE_TOLERANCE_MODE to M_ABSOLUTE , M_AUTO specifies an infinite (M_INFINITE) tolerance. If you set M_DISTANCE_TOLERANCE_MODE to M_RELATIVE , M_AUTO specifies a tolerance of 1. If you set M_DISTANCE_TOLERANCE_MODE to M_SAMPLE_STDDEV , M_AUTO specifies a tolerance of 3. (summarize)
<input type="checkbox"/> M_INFINITE	Specifies an infinite tolerance.

<div><div></div>Value >= 0.0</div>	<p>Specifies the tolerance value.</p> <p>Set the tolerance according to the specified tolerance mode (M_DISTANCE_TOLERANCE_MODE) and distance type (McolSetMethod()). For example, a tolerance value of 1 when using an M_MAHALANOBIS distance type is not the same as when using an M_MANHATTAN distance type, even if the tolerance mode is M_ABSOLUTE.</p> <p>(summarize)</p>
<div><div></div>M_SAMPLE_LABEL_VALUE</div>	<p>Sets the color-sample's label.</p> <p>Initially, you specify a color-sample's label value when adding a color-sample to the context, using McolDefine(). You can use M_SAMPLE_LABEL_VALUE to change the label afterwards.</p> <p>(summarize)</p>
<div><div></div>Value > 0</div>	<p>Specifies the label, as an <i>integer</i>. The label must be unique among existing labels, and must be less than 2^{24}.</p> <p>(summarize)</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolDefine

Synopsis

Define or delete a color-sample.

Syntax

```
void McolDefine(
    MIL_ID ColorMatcherContextId,
    MIL_ID SrcImageOrArrayId,
    MIL_INT UserLabelOrIndex,
    MIL_INT ColorSampleType,
    MIL_DOUBLE Param1,
    MIL_DOUBLE Param2,
    MIL_DOUBLE Param3,
    MIL_DOUBLE Param4
)
```

Description

This function allows you to add a color-sample to, or remove a color-sample from, a color matching context. You can define a color-sample from an image or from three explicit color component values. You can then match the color of the target area with one of the color-samples, using [McolMatch\(\)](#).

For color-samples defined from an image, an estimation of the color is determined to represent the color-sample. This estimate is based on color statistics, such as the mean; it is considered the color of the color-sample and is used in the match. The size of the color-sample does not affect the match operation.

The color-sample index starts at 0, and each subsequent color-sample added to the color matching context is given a sequential index number, in the order that the color-sample was added. If a color-sample is deleted, all entries with higher indices are shifted down by one. You can also assign a label to a color-sample, or it can be assigned automatically. To change it afterwards, use [McolControl\(\)](#) with [M_SAMPLE_LABEL_VALUE](#).

When you add or delete a color-sample, you must preprocess the color matching context ([McolPreprocess\(\)](#)) before any subsequent call to [McolMatch\(\)](#).

You must add color-samples one at a time; however you can delete all color-samples at once.

The format of your color data must be consistent. For example, you will not receive an error if you match an RGB target area with an HSL color-sample, or a 16-bit target area with an 8-bit color-sample. Color data is used as is, and is considered to be within the context's color space ([McolAlloc\(\)](#)); there is no internal conversion. Color data is interpreted band per band, independently of the buffer's format.

You must set the color space encoding according to the actual dynamic range of your color data, using [McolControl\(\)](#) with [M_ENCODING](#). By default, the Color Analysis module assumes an 8-bit color space encoding (regardless of the buffer depth).

Parameters

ColorMatcherContextId

Specifies the identifier of the color matching context to which to add, or from which to delete, the color-sample. The color matching context must have been previously allocated on the required system using [McolAlloc\(\)](#) with [M_COLOR_MATCHING](#).

SrcImageOrArrayId

Specifies the identifier of the source image buffer from which to define the color-sample. If you are defining a color-sample from three explicit color component values, or if you are deleting color-samples, set this parameter to [M_NULL](#).

UserLabelOrIndex

Specifies the color-sample to add or delete from the color matching context. Set this parameter to one of the following values.

● For specifying the color-sample	
☐ Value	Description
☐ M_DEFAULT	Specifies to add a color-sample and automatically assign it a label. The label is the next available numerical value following the highest label already defined. (summarize)
☐ M_SAMPLE_INDEX(MIL_INT <i>SampleIndex</i>)	Specifies the index of an existing individual color-sample to delete. You cannot use M_SAMPLE_INDEX() to add a color-sample. (summarize)
	<i>Parameters</i>
	<i>SampleIndex</i> This parameter specifies the index of the individual color-sample to delete. You can set this parameter to the following:
	Value >= 0
	Specifies the index.
☐ M_SAMPLE_LABEL(MIL_INT <i>SampleLabel</i>)	Specifies either the label of the color-sample to add, or the label of an existing individual color-sample to delete. (summarize)
	<i>Parameters</i>
	<i>SampleLabel</i> This parameter specifies the label of the color-sample to add or delete. You can set this parameter to the following:
	Value > 0
	Specifies the label of the individual color-sample, as an <i>integer</i> . The label must be unique among existing labels, and must be less than 2^{24} .
☐ M_ALL	Specifies all color-samples. You can only use M_ALL to delete color-samples. (summarize)

ColorSampleType

Specifies the type of color-sample to define when adding color-samples to the color matching context, or specifies to delete a color-sample from the context.

See the [Parameter associations](#) section for possible values.

Param1

Specifies an attribute of the color-sample to add. Its definition is dependent on the color-sample type chosen.

See the [Parameter associations](#) section for possible values.

Param2

Specifies an attribute of the color-sample to add. Its definition is dependent on the color-sample type chosen.

See the [Parameter associations](#) section for possible values.

Param3

Specifies an attribute of the color-sample to add. Its definition is dependent on the color-sample type chosen.

See the [Parameter associations](#) section for possible values.

Param4

Specifies an attribute of the color-sample to add. Its definition is dependent on the color-sample type chosen.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ColorSampleType**, **Param1**, **Param2**, **Param3**, and **Param4** parameters are described in the following tables:

- [For adding a color-sample to the context](#)
- [For deleting color-samples from the context](#)

Note that you should set all unused parameters to **M_DEFAULT**.

For adding a color-sample to the context		
ColorSampleType	Description	
Param1		
Param2		
Param3		
Param4		
M_IMAGE	Specifies to add a color-sample from the specified region (ROI) of the source image. In this case, you must set the SrcImageOrArrayId parameter to the identifier of an 8- or 16-bit color (3-band) unsigned image buffer. (summarize)	
Param1	Sets the X-offset of the origin of the color-sample region from the source image. (summarize)	
M_DEFAULT	Specifies that the X-offset is equal to 0.	
Value	Specifies the X-offset, in pixels.	
Param2	Sets the Y-offset of the origin of the color-sample region from the source image. (summarize)	
M_DEFAULT	Specifies that the Y-offset is equal to 0.	
Value	Specifies the Y-offset, in pixels.	
Param3	Sets the width (X-size) of the color-sample region. (summarize)	
M_DEFAULT	Specifies that the width is from the origin of the color-sample region to the end of the source image, in the X-direction.	
Value	Specifies the width, in pixels.	
Param4	Sets the height (Y-size) of the color-sample region. (summarize)	
M_DEFAULT	Specifies that the height is from the origin of the color-sample region to the end of the source image, in the Y-direction.	
Value	Specifies the height, in pixels.	
M_TRIPLET	Specifies to add a color-sample using three explicit values for the color components (RGB, HSL, or CIELAB). In this case, you must set the SrcImageOrArrayId parameter to M_NULL . (summarize)	
Param1	Sets the value of the first color component. (summarize)	
Param2	Sets the value of the second color component. (summarize)	
Param3	Sets the value of the third color component. (summarize)	

The following **ColorSampleType** parameter setting can be used to remove color-samples from the context. In this case, you must set the **SrcImageOrArrayId** parameter to **M_NULL**, and **Param1** to **Param4** to **M_DEFAULT**.

For deleting color-samples from the context		
---------------------------------------------	--	--

<input type="checkbox"/> ColorSampleType		Description
	Param1	
	Param2	
	Param3	
	Param4	
<input type="checkbox"/> M_DELETE		Specifies to delete one or all color-samples, depending on the UserLabelOrIndex parameter setting. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolDistance

Synopsis

Calculate the distance between colors.

Syntax

```
void McolDistance(
    MIL_ID Src1Id,
    MIL_ID Src2Id,
    MIL_ID DestId,
    MIL_ID MaskId,
    MIL_ID ParametersArrayId,
    MIL_INT DistType,
    MIL_DOUBLE NormalizeValue,
    MIL_INT ControlFlag
)
```

Description

This function calculates the point-to-point distance between colors in two images. You can also calculate the point-to-point distance between the color in one image and: a color constant, a covariance matrix, or the covariance of a specified image. To ignore unwanted pixels in the distance calculation, you can apply a mask.

The color distance is typically calculated only for pixels within the intersection of the source (**Src1Id** and **Src2Id**), destination (**DestId**), and mask (**MaskId**) images, with the assumption of a common origin at the top-left corner.

Results are returned to this function's parameters.

After normalization (**NormalizeValue**), distance results are saturated according to the destination buffer type. For *integer* buffers, distances are remapped according to the buffer's possible range of values. For *floating-point* buffers, the values are left between 0 and 1.

The format of your color data must be consistent. For example, you will not receive an error if you try to calculate the distance between an RGB and an HSL image, or a 16-bit and an 8-bit image. Color data is used as is, and is considered to be within a common color space; there is no internal conversion. Color data is interpreted band per band, independently of the buffer's format.

You can also obtain distance results after calling [McolMatch\(\)](#), which unlike **McolDistance()**, considers the color matching context's color space. For more information on how distances differ between these two functions, see the [Distance between colors](#) section in [Chapter 17: Color processing and analysis](#).

Parameters

Src1Id

Specifies the identifier of the first source image buffer. The buffer must be an 8- or 16-bit color (3-band) unsigned image buffer.

Src2Id

Specifies the identifier of the second source buffer, which can be either an image or an array.

For an image, the buffer must be an 8- or 16-bit color (3-band) unsigned image buffer. For an array, the content and size depends on the type of distance operation chosen (**DistType**).

DestId

Specifies the identifier of the destination image buffer in which to write the results. This must be a one-band buffer; its type is not restricted.

MaskId

Specifies the identifier of the image buffer used to identify the masked (non-zero) pixels. This must be a 1- or 8-bit (1-band) unsigned buffer. To not apply a mask, set this parameter to **M_NULL**.

Source pixels present in the masked region (non-zero pixels) are used by the distance calculation. Resulting distances are written in the masked portion of the destination image.

Unmasked pixels are set to 0 in the mask. In the destination, these pixels remain untouched; in the source, they are ignored by the distance calculation.

ParametersArrayId

Reserved for future expansion and must be set to **M_DEFAULT**.

DistType

Specifies the type of distance operation to perform. This parameter must be set to one of the following values:

● For specifying the distance type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EUCLIDEAN	Specifies to use a Euclidean color distance. A Euclidean distance is the square root of the sum of the squared differences between the color of each pixel in the image specified in the first source (Src1Id) and either the color of each pixel in the image specified in the second source, or a color constant (Src2Id). For a color constant, you must use a 3 x 1, one-dimensional MIL array buffer. This buffer's type is unrestricted, and must contain three color component values (for example, RGB). (summarize)
<input type="checkbox"/> M_MAHALANOBIS	Specifies to use a Mahalanobis color distance. A Mahalanobis distance is calculated between the color of each pixel in the image specified in the first source (Src1Id) and either the covariance of the image specified in the second source, or a covariance matrix (Src2Id). This distance, between a color and a distribution of colors, is similar to a Euclidean distance between the mean of the two colors, but weighted by the inverse of the covariance of the distribution. This implies that the more a color distribution varies in a direction within the color space, the less important is the distance in that direction. For a covariance matrix, you must use a 4 x 3, two-dimensional MIL array buffer. (summarize)
<input type="checkbox"/> M_MANHATTAN	Specifies to use a Manhattan color distance. A Manhattan distance is the sum of the absolute value of the differences between the color of each pixel in the image specified in the first source (Src1Id) and either the color of each pixel in the image specified in the second source, or a color constant (Src2Id). For HSL colors, the distance between angular coordinates is equal to the smallest angular difference, rather than the absolute value of the difference. For a color constant, you must use a 3 x 1, one-dimensional MIL array buffer. The buffer's type is unrestricted, and must contain three color component values (for example, RGB). (summarize)

NormalizeValue

Specifies the factor with which to normalize the resulting distances.

This parameter should be set to one of the following values:

● For specifying the normalization of distances	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MAX_NORMALIZE	Specifies to normalize the distances by the maximum distance calculated.
<input type="checkbox"/> M_NO_NORMALIZE	Specifies no normalization.
<input type="checkbox"/> Value > 0.0	Specifies the normalization factor.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.

DLL	Requires mil.dll; milcol.dll.
-----	-------------------------------

McolDraw

Synopsis

Draw specific features of a color matching context, or of a color matching result.

Syntax

```
void McolDraw(
    MIL_ID GraphContId,
    MIL_ID ColorObjectId,
    MIL_ID DestImageId,
    MIL_INT DrawOperation,
    MIL_INT AreaLabel,
    MIL_INT SampleIndexOrLabel,
    MIL_INT ControlFlag
)
```

Description

This function draws specific features for a selected color-sample of a specified color matching context, or of a color matching result, in the destination image buffer.

The required format and size of the destination image depends on the drawing operation. For a color matching context, you can retrieve the required format and size using [McolInquire\(\)](#); for example, [M_SAMPLE_LUT_SIZE_X](#). For a color matching result buffer, you can retrieve this information using [McolGetResult\(\)](#); for example, [M_AREA_IMAGE_SIZE_X](#).

The bit depth of the destination image buffer must accommodate all the color-sample data drawn, including the background color, the outlier color, and the outlier labels. To specify these values, use [McolControl\(\)](#) with [M_BACKGROUND_DRAW_COLOR](#), [M_OUTLIER_DRAW_COLOR](#), and [M_OUTLIER_LABEL](#). The numerical value of the color-sample labels, the background pixels, and the outlier pixels and labels should be different so you can easily identify them.

Parameters

GraphContId

Specifies the graphics context to use when drawing. This parameter must be set to one of the following values:

For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

ColorObjectId

Specifies the color matching context or result buffer for which to perform the drawing operation. The color matching context or result buffer must have been previously allocated on the system using [McolAlloc\(\)](#) or [McolAllocResult\(\)](#), respectively.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
The color matching result buffer must be allocated on the same system as the graphics context (**GraphContId**). If it is not, an error will occur.

DestImageId

Specifies the identifier of the destination image buffer in which to draw. The buffer can be any supported MIL image buffer. By drawing into its display's overlay buffer, you can also annotate the image non-destructively.

For a color matching result, the background of the destination image is set to the background color ([McolControl\(\)](#) with [M_BACKGROUND_DRAW_COLOR](#)) that is in effect when you call [McolMatch\(\)](#). If you set a transparent background, the image background remains untouched.

DrawOperation

Specifies the type of drawing operation to perform.

For a selected color-sample of a specified color context, this parameter can be set to one of the following values:

● For drawing operations with a color matching context	
☐ Value	Description
☐ M_DRAW_SAMPLE_IMAGE	Draws a copy of the internal color-sample image, specified with McolDefine() , into the destination image buffer (DestImageId). If you defined a triplet color-sample, M_DRAW_SAMPLE_IMAGE fills the destination image buffer with the color of the triplet. The triplet values are saturated according to the destination buffer depth. (summarize)
☐ M_DRAW_SAMPLE_MASK_IMAGE	Draws the mask image of the color-sample defined in the context (McolDefine()). M_DRAW_SAMPLE_MASK_IMAGE draws a copy of the mask of the internal color-sample image into the destination image buffer (DestImageId). You can only perform this operation on M_IMAGE color-samples masked with McolMask() . To inquire if a sample has a mask, use McolInquire() with M_SAMPLE_MASKED . (summarize)

For a color matching result buffer, this parameter can be set to one of the following values.

Note that the bit depth of the destination image buffer must accommodate all the color-sample data drawn, including background and outlier pixels.

● For drawing operations with a color matching result	
☐ Value	Description
☐ M_DRAW_AREA_IMAGE	Draws a copy of the specified area (AreaLabel) of the area identifier image, specified with McolMatch() . If you specify a target area with the AreaLabel parameter, the background of the destination image (DestImageId) is untouched; if you specify M_ALL , the whole area identifier image, including a 0 background, is copied to the destination image. To perform this drawing operation, you must enable McolControl() with M_SAVE_AREA_IMAGE . (summarize)
☐ M_DRAW_COLORED_LABEL_AREA_IMAGE	Draws, for each target area, the color of the best-matched color-sample. The background color and outlier color are also drawn. To perform this drawing operation, you must enable McolControl() with M_SAVE_AREA_IMAGE and M_GENERATE_SAMPLE_COLOR_LUT . (summarize)
☐ M_DRAW_COLORED_LABEL_PIXEL_IMAGE	Draws, for each pixel in each target area, the color of the color-sample for which each pixel voted. The background color and outlier color are also drawn. To perform this drawing operation, you must enable McolControl() with M_GENERATE_LABEL_PIXEL_IMAGE and M_GENERATE_SAMPLE_COLOR_LUT . You must also use McolSetMethod() with M_MIN_DIST_VOTE . (summarize)
☐ M_DRAW_DISTANCE_IMAGE	Draws the distance between the color of the target area (for an M_STAT_MIN_DIST operation mode) or target pixel (for an M_MIN_DIST_VOTE operation mode), and the color of its best-matched color-sample. Operation modes are set with McolSetMethod() . The background color and outlier color are also drawn. To perform this drawing operation, you must enable McolControl() with M_GENERATE_DISTANCE_IMAGE . To normalize distances, use McolControl() with M_DISTANCE_IMAGE_NORMALIZE . Although outlier colors are drawn with M_DRAW_DISTANCE_IMAGE , it might be difficult to identify them. In this case, use M_DRAW_LABEL_... , which draws the outlier labels. (summarize)
☐ M_DRAW_LABEL_AREA_IMAGE	Draws, for each target area, the label of the best-matched color-sample. The background color and outlier label are also drawn. To perform this drawing operation, you must enable McolControl() with M_SAVE_AREA_IMAGE . (summarize)
☐ M_DRAW_LABEL_PIXEL_IMAGE	Draws, for each pixel in each target area, the label of the color-sample for which each pixel voted. The background color and outlier label are also drawn. To perform this drawing operation, you must enable McolControl() with M_GENERATE_LABEL_PIXEL_IMAGE . You must also use McolSetMethod() with M_MIN_DIST_VOTE . (summarize)

For either a color matching context or color matching result, this parameter can be set to the following value:

● For drawing operations with either a color matching context or color matching result	
☐ Value	Description
☐ M_DRAW_SAMPLE_COLOR_LUT	<p>Draws the 3-band color-sample label LUT, where the label value of each color-sample is associated to its average color.</p> <p>The average colors are calculated using the color data you provide when defining the color-sample (McolDefine()). The calculation takes into account all three bands, regardless of the setting of McolControl() with M_BAND_MODE.</p> <p>To perform this operation, you must enable McolControl() with M_GENERATE_SAMPLE_COLOR_LUT.</p> <p>M_DRAW_SAMPLE_COLOR_LUT requires that you set the SampleIndexOrLabel parameter to M_ALL or M_DEFAULT.</p> <p>When using M_DRAW_SAMPLE_COLOR_LUT, the destination image buffer (DestImageId) must be a 3-band, 8-bit unsigned LUT buffer, with a size of $N \times 1$, where N is equal to the M_SAMPLE_LUT_SIZE_X inquire type (McolInquire()). You can associate this color-sample label LUT to either a display using MdispLut(), or to an image buffer using MbufControl() with M_ASSOCIATED_LUT.</p> <p>(summarize)</p>

AreaLabel

Specifies the label of the target area for which to perform the drawing operation. Set this parameter to one of the following values.

● For specifying the target area's label or index	
☐ Value	Description
☐ M_DEFAULT	Same as M_ALL .
☐ M_ALL	<p>Specifies all target areas. You should specify M_ALL if you do not use an area identifier image (M_NULL) with McolMatch().</p> <p>(summarize)</p>
☐ Value > 0	<p>Specifies the label of the target area, as it appears in the area identifier image.</p> <p>When specifying a specific target area's label, you must use McolControl() with M_SAVE_AREA_IMAGE set to M_ENABLE.</p> <p>(summarize)</p>

SampleIndexOrLabel

Specifies the index or label of the color-sample for which to perform the drawing operation. This parameter can be set to one of the following values:

● For specifying the color-sample's label or index							
☐ Value	Description						
☐ M_DEFAULT	Same as M_ALL .						
☐ M_SAMPLE_INDEX(MIL_INT <i>SampleIndex</i>)	<p>Specifies the color-sample's index.</p> <p>You can use M_SAMPLE_INDEX for a color matching context or color matching result.</p> <p>In this case, you must enable McolControl() with M_GENERATE_LABEL_PIXEL_IMAGE. However, you need not do this when performing an M_DRAW_LABEL_AREA_IMAGE or M_DRAW_COLORED_LABEL_AREA_IMAGE drawing operation.</p> <p>(summarize)</p> <table> <tr> <th colspan="2">Parameters</th></tr> <tr> <td><i>SampleIndex</i></td><td>This parameter specifies the color-sample's index. You can set this parameter to the following:</td></tr> <tr> <td>0 <= Value <= color-samples - 1</td><td>Specifies the index.</td></tr> </table>	Parameters		<i>SampleIndex</i>	This parameter specifies the color-sample's index. You can set this parameter to the following:	0 <= Value <= color-samples - 1	Specifies the index.
Parameters							
<i>SampleIndex</i>	This parameter specifies the color-sample's index. You can set this parameter to the following:						
0 <= Value <= color-samples - 1	Specifies the index.						
☐ M_SAMPLE_LABEL(MIL_INT <i>SampleLabel</i>)	<p>Specifies the color-sample's label.</p> <p>You can use M_SAMPLE_LABEL for a color matching context or color matching result.</p> <p>In this case, you must enable McolControl() with M_GENERATE_LABEL_PIXEL_IMAGE. However, you need not do this when performing an M_DRAW_LABEL_AREA_IMAGE or M_DRAW_COLORED_LABEL_AREA_IMAGE drawing operation.</p> <p>(summarize)</p> <table> <tr> <th colspan="2">Parameters</th></tr> <tr> <td><i>SampleLabel</i></td><td>This parameter specifies the color-sample's label. You can set this parameter to the following:</td></tr> </table>	Parameters		<i>SampleLabel</i>	This parameter specifies the color-sample's label. You can set this parameter to the following:		
Parameters							
<i>SampleLabel</i>	This parameter specifies the color-sample's label. You can set this parameter to the following:						

		Value > 0	Specifies the label.
<input type="checkbox"/> M_ALL	<div>Specifies all color-samples.</div> <div>You can use M_ALL when specifying a color matching result buffer.</div> <div>When specifying a color matching context, you can only use M_ALL with M_DRAW_SAMPLE_COLOR_LUT.</div> <div>(summarize)</div>		

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolFree

Synopsis

Free a color matching context or a color matching result buffer.

Syntax

```
void McolFree(
    MIL_ID ColorObjectId
)
```

Description

This function deletes the specified color matching context (and all its color-samples) or result buffer identifier, and releases any memory associated with it. To only delete color-samples in the context, use [McolDefine\(\)](#) with [M_DELETE](#).

All color matching contexts and all result buffers allocated on a particular system must be freed before the system can be freed.

Parameter

ColorObjectId

Specifies the identifier of the color matching context or result buffer to free. These must have been successfully allocated (with [McolAlloc\(\)](#) or [McolAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolGetResult

Synopsis

Get the specified type of results from a color matching result buffer.

Syntax

```
void McolGetResult(
    MIL_ID ResultId,
    MIL_INT AreaLabel,
    MIL_INT ColorSampleIndexOrLabel,
    MIL_INT ResultType,
    void *ResultArrayPtr
)
```

Description

This function retrieves the results of the specified type from a color matching result buffer. Results are only available after calling [McolMatch\(\)](#), and are organized by target area labels and color-sample indices. You can get results obtained from a specific target area or for all target areas. Similarly, you can get results obtained from a specific color-sample or for all color-samples.

To determine the order of the results, use [M_AREA_LABEL_VALUE](#), and set the **AreaLabel** parameter to [M_ALL](#) and the **ColorSampleIndexOrLabel** parameter to [M_GENERAL](#).

For general results, set both the **AreaLabel** and **ColorSampleIndexOrLabel** parameter to [M_GENERAL](#). For target area results, set the **AreaLabel** parameter to a specific target area (or [M_ALL](#)) and the **ColorSampleIndexOrLabel** parameter to [M_GENERAL](#). For color-sample results, set the **AreaLabel** parameter to a specific target area (or [M_ALL](#)) and the **ColorSampleIndexOrLabel** parameter to a specific color-sample (or [M_ALL](#)). To retrieve color-sample results for all color-samples and all areas, set both the **AreaLabel** and **ColorSampleIndexOrLabel** parameter to [M_ALL](#), and use [M_NUMBER_OF_AREAS](#) and [M_NUMBER_OF_SAMPLES](#) to set the size of the result array (*NumberOfTargetAreas x NumberOfSamples*). Note that to get a color-sample result, you must first refer to its target area.

To decrease result retrieval time, especially for remote systems, you can get different types of results from the result buffer all at once. To do so, set **ResultArrayPtr** to [M_NULL](#). In this case, results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [McolGetResult\(\)](#) again and you pass an array to the **ResultArrayPtr** parameter. You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

Parameters

ResultId
Specifies the identifier of the color matching result buffer from which to get results.

AreaLabel
Specifies the label of the target area(s) for which to get results.

Target areas are specified using [McolMatch\(\)](#) with the [AreaLabelImageId](#) parameter. If you set [AreaLabelImageId](#) to [M_NULL](#), you must set the **AreaLabel** parameter to [M_ALL](#) or [M_DEFAULT](#).

Set this parameter to one of the following values.

● For specifying the target area's label	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> M_ALL	Specifies to retrieve results for each target area.
<input type="checkbox"/> M_GENERAL	Specifies to retrieve results relating to the entire color matching context. This type of result is not specific to any one area. (summarize)
<input type="checkbox"/> Value > 0	Specifies a specific target area's label, as an <i>integer</i> .

ColorSampleIndexOrLabel

Specifies the color-sample(s) for which to get results. Set this parameter to one of the following values.

● For specifying the color-sample	
☐ Value	Description
☐ M_SAMPLE_INDEX(MIL_INT <i>SampleIndex</i>)	Specifies the index value of an existing color-sample for which to get results. (summarize)
	<i>Parameters</i> <i>SampleIndex</i> Specifies the index of a specific color-sample. The index is from 0 (inclusive) to the number of defined color-samples minus 1.
☐ M_SAMPLE_LABEL(MIL_INT <i>SampleLabel</i>)	Specifies the label of an existing individual color-sample for which to get results. (summarize)
	<i>Parameters</i> <i>SampleLabel</i> Specifies the label of a specific color-sample. The label must be greater than 0.
☐ M_ALL	Specifies to retrieve results for each color-sample. The color-samples are returned according to their index value, in increasing order. (summarize)
☐ M_GENERAL	Specifies to return results relating to the entire color matching context. This type of result is not specific to any one color-sample. (summarize)

ResultType

Specifies the type of result to retrieve.

To retrieve general results relating to the entire color matching context, the **ResultType** parameter can be set to one of the following values. In this case, you must set the **AreaLabel** and **ColorSampleIndexOrLabel** parameters to **M_GENERAL**.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type MIL_DOUBLE with a size equal to what is required by the result type (for multiple results) or the address of a MIL_DOUBLE (for a single result).

● For retrieving general results relating to the entire color matching context	
☐ Value	Description
☐ M_AREA_IMAGE_SIGN +	Returns the data type required for the image buffer in which to draw M_DRAW_AREA_IMAGE (McolDraw()). This result is available after you specify an area identifier image with McolMatch() , enable the M_SAVE_AREA_IMAGE control, and find a match. (summarize)
	<i>ResultArrayPtr info</i> Return values: M_UNSIGNED Unsigned data type.
☐ M_AREA_IMAGE_SIZE_BAND +	Returns the number of surfaces (color bands) required for the image buffer in which to draw M_DRAW_AREA_IMAGE (McolDraw()). This result is available after you specify an area identifier image with McolMatch() , enable the M_SAVE_AREA_IMAGE control, and find a match. (summarize)
	<i>ResultArrayPtr info</i> Return values: 1 1 band.
☐ M_AREA_IMAGE_SIZE_BIT +	Returns the depth per band, in bits, required for the image buffer in which to draw M_DRAW_AREA_IMAGE (McolDraw()).

	<p>This result is available after you specify an area identifier image with McolMatch(), enable the M_SAVE_AREA_IMAGE control, and find a match. (summarize)</p> <table> <tr> <th><i>ResultArrayPtr info</i></th><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>1</td><td>1-bit data depth per band.</td></tr> <tr> <td>8</td><td>8-bit data depth per band.</td></tr> <tr> <td>16</td><td>16-bit data depth per band.</td></tr> </table>	<i>ResultArrayPtr info</i>		Return values:		1	1-bit data depth per band.	8	8-bit data depth per band.	16	16-bit data depth per band.
<i>ResultArrayPtr info</i>											
Return values:											
1	1-bit data depth per band.										
8	8-bit data depth per band.										
16	16-bit data depth per band.										
M_AREA_IMAGE_SIZE_X +	<p>Returns the width, in pixels, required for the image buffer in which to draw M_DRAW_AREA_IMAGE (McolDraw()). This result is available after you specify an area identifier image with McolMatch(), enable the M_SAVE_AREA_IMAGE control, and find a match. (summarize)</p>										
M_AREA_IMAGE_SIZE_Y +	<p>Returns the height, in pixels, required for the image buffer in which to draw M_DRAW_AREA_IMAGE (McolDraw()). This result is available after you specify an area identifier image with McolMatch(), enable the M_SAVE_AREA_IMAGE control, and find a match. (summarize)</p>										
M_AREA_IMAGE_TYPE +	<p>Returns the data type and depth required for the image buffer in which to draw M_DRAW_AREA_IMAGE (McolDraw()). This result is available after you specify an area identifier image with McolMatch(), enable the M_SAVE_AREA_IMAGE control, and find a match. (summarize)</p> <table> <tr> <th><i>ResultArrayPtr info</i></th><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>1 + M_UNSIGNED</td><td>1-bit unsigned.</td></tr> <tr> <td>8 + M_UNSIGNED</td><td>8-bit unsigned.</td></tr> <tr> <td>16 + M_UNSIGNED</td><td>16-bit unsigned.</td></tr> </table>	<i>ResultArrayPtr info</i>		Return values:		1 + M_UNSIGNED	1-bit unsigned.	8 + M_UNSIGNED	8-bit unsigned.	16 + M_UNSIGNED	16-bit unsigned.
<i>ResultArrayPtr info</i>											
Return values:											
1 + M_UNSIGNED	1-bit unsigned.										
8 + M_UNSIGNED	8-bit unsigned.										
16 + M_UNSIGNED	16-bit unsigned.										
M_COLORED_LABEL_AREA_IMAGE_SIGN +	<p>Returns the data type required for the image buffer in which to draw M_DRAW_COLORED_LABEL_AREA_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_SAMPLE_COLOR_LUT and M_SAVE_AREA_IMAGE controls (required for McolDraw()), and find a match. (summarize)</p> <table> <tr> <th><i>ResultArrayPtr info</i></th><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>M_UNSIGNED</td><td>Unsigned data type.</td></tr> </table>	<i>ResultArrayPtr info</i>		Return values:		M_UNSIGNED	Unsigned data type.				
<i>ResultArrayPtr info</i>											
Return values:											
M_UNSIGNED	Unsigned data type.										
M_COLORED_LABEL_AREA_IMAGE_SIZE_BAND +	<p>Returns the number of surfaces (color bands) required for the image buffer in which to draw M_DRAW_COLORED_LABEL_AREA_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_SAMPLE_COLOR_LUT and M_SAVE_AREA_IMAGE controls (required for McolDraw()), and find a match. (summarize)</p> <table> <tr> <th><i>ResultArrayPtr info</i></th><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>3</td><td>3 bands.</td></tr> </table>	<i>ResultArrayPtr info</i>		Return values:		3	3 bands.				
<i>ResultArrayPtr info</i>											
Return values:											
3	3 bands.										
M_COLORED_LABEL_AREA_IMAGE_SIZE_BIT +	<p>Returns the depth per band, in bits, required for the image buffer in which to draw M_DRAW_COLORED_LABEL_AREA_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_SAMPLE_COLOR_LUT and M_SAVE_AREA_IMAGE controls (required for McolDraw()), and find a match. (summarize)</p> <table> <tr> <th><i>ResultArrayPtr info</i></th><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>8</td><td>8-bit data depth per band.</td></tr> </table>	<i>ResultArrayPtr info</i>		Return values:		8	8-bit data depth per band.				
<i>ResultArrayPtr info</i>											
Return values:											
8	8-bit data depth per band.										
M_COLORED_LABEL_AREA_IMAGE_SIZE_X +	<p>Returns the width, in pixels, required for the image buffer in which to draw M_DRAW_COLORED_LABEL_AREA_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_SAMPLE_COLOR_LUT and M_SAVE_AREA_IMAGE controls (required for McolDraw()), and find a match. (summarize)</p>										
M_COLORED_LABEL_AREA_IMAGE_SIZE_Y +	<p>Returns the height, in pixels, required for the image buffer in which to draw M_DRAW_COLORED_LABEL_AREA_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_SAMPLE_COLOR_LUT and M_SAVE_AREA_IMAGE controls (required for McolDraw()), and find a match. (summarize)</p>										
M_COLORED_LABEL_AREA_IMAGE_TYPE +	<p>Returns the data type and depth required for the image buffer in which to draw M_DRAW_COLORED_LABEL_AREA_IMAGE (McolDraw() or McolMatch()).</p>										

	<p>This result is available after you enable the M_GENERATE_SAMPLE_COLOR_LUT and M_SAVE_AREA_IMAGE controls (required for McolDraw()), and find a match. (summarize)</p>
	<p><i>ResultArrayPtr info</i> Return values: 8 + M_UNSIGNED 8-bit unsigned.</p>
<input type="checkbox"/> M_COLORED_LABEL_PIXEL_IMAGE_SIGN +	<p>Returns the data type required for the image buffer in which to draw M_DRAW_COLORED_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE and M_GENERATE_SAMPLE_COLOR_LUT controls (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE. (summarize)</p>
	<p><i>ResultArrayPtr info</i> Return values: M_UNSIGNED Unsigned data type.</p>
<input type="checkbox"/> M_COLORED_LABEL_PIXEL_IMAGE_SIZE_BAND +	<p>Returns the number of surfaces (color bands) required for the image buffer in which to draw M_DRAW_COLORED_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE and M_GENERATE_SAMPLE_COLOR_LUT controls (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE. (summarize)</p>
	<p><i>ResultArrayPtr info</i> Return values: 3 3 bands.</p>
<input type="checkbox"/> M_COLORED_LABEL_PIXEL_IMAGE_SIZE_BIT +	<p>Returns the depth per band, in bits, required for the image buffer in which to draw M_DRAW_COLORED_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE and M_GENERATE_SAMPLE_COLOR_LUT controls (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE. (summarize)</p>
	<p><i>ResultArrayPtr info</i> Return values: 8 8-bit data depth per band.</p>
<input type="checkbox"/> M_COLORED_LABEL_PIXEL_IMAGE_SIZE_X +	<p>Returns the width, in pixels, required for the image buffer in which to draw M_DRAW_COLORED_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE and M_GENERATE_SAMPLE_COLOR_LUT controls (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE. (summarize)</p>
<input type="checkbox"/> M_COLORED_LABEL_PIXEL_IMAGE_SIZE_Y +	<p>Returns the height, in pixels, required for the image buffer in which to draw M_DRAW_COLORED_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE and M_GENERATE_SAMPLE_COLOR_LUT controls (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE. (summarize)</p>
<input type="checkbox"/> M_COLORED_LABEL_PIXEL_IMAGE_TYPE +	<p>Returns the data type and depth required for the image buffer in which to draw M_DRAW_COLORED_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE and M_GENERATE_SAMPLE_COLOR_LUT controls (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE. (summarize)</p>
	<p><i>ResultArrayPtr info</i> Return values: 8 + M_UNSIGNED 8-bit unsigned.</p>
<input type="checkbox"/> M_DISTANCE_IMAGE_SIGN +	<p>Returns the data type required for the image buffer in which to draw M_DRAW_DISTANCE_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_DISTANCE_IMAGE control (required for McolDraw()), and find a match. Note that if you have normalized distances by the maximum distance calculated, using McolControl() with M_DISTANCE_IMAGE_NORMALIZE set to M_MAX_NORMALIZE, you can draw the distance image in an unsigned image buffer. (summarize)</p>
	<p><i>ResultArrayPtr info</i> Return values: M_FLOAT Floating point data type.</p>

<div><div></div><div>M_DISTANCE_IMAGE_SIZE_BAND +</div></div>	<p>Returns the number of surfaces (color bands) required for the image buffer in which to draw <code>M_DRAW_DISTANCE_IMAGE</code> (<code>McolDraw()</code> or <code>McolMatch()</code>). This result is available after you enable the <code>M_GENERATE_DISTANCE_IMAGE</code> control (required for <code>McolDraw()</code>), and find a match.</p> <div><div></div><div><div><div>ResultArrayPtr info</div><div>Return values:</div><div>11 band.</div></div></div></div>
<div><div></div><div>M_DISTANCE_IMAGE_SIZE_BIT +</div></div>	<p>Returns the depth per band, in bits, required for the image buffer in which to draw <code>M_DRAW_DISTANCE_IMAGE</code> (<code>McolDraw()</code> or <code>McolMatch()</code>). This result is available after you enable the <code>M_GENERATE_DISTANCE_IMAGE</code> control (required for <code>McolDraw()</code>), and find a match. Note that if you have normalized distances by the maximum distance calculated, using <code>McolControl()</code> with <code>M_DISTANCE_IMAGE_NORMALIZE</code> set to <code>M_MAX_NORMALIZE</code>, you can draw the distance image in an unsigned image buffer.</p> <div><div></div><div><div><div>ResultArrayPtr info</div><div>Return values:</div><div>3232-bit data depth per band.</div></div></div></div>
<div><div></div><div>M_DISTANCE_IMAGE_SIZE_X +</div></div>	<p>Returns the width, in pixels, required for the image buffer in which to draw <code>M_DRAW_DISTANCE_IMAGE</code> (<code>McolDraw()</code> or <code>McolMatch()</code>). This result is available after you enable the <code>M_GENERATE_DISTANCE_IMAGE</code> control (required for <code>McolDraw()</code>), and find a match.</p> <div><div></div><div></div></div>
<div><div></div><div>M_DISTANCE_IMAGE_SIZE_Y +</div></div>	<p>Returns the height, in pixels, required for the image buffer in which to draw <code>M_DRAW_DISTANCE_IMAGE</code> (<code>McolDraw()</code> or <code>McolMatch()</code>). This result is available after you enable the <code>M_GENERATE_DISTANCE_IMAGE</code> control (required for <code>McolDraw()</code>), and find a match.</p> <div><div></div><div></div></div>
<div><div></div><div>M_DISTANCE_IMAGE_TYPE +</div></div>	<p>Returns the data type and depth required for the image buffer in which to draw <code>M_DRAW_DISTANCE_IMAGE</code> (<code>McolDraw()</code> or <code>McolMatch()</code>). This result is available after you enable the <code>M_GENERATE_DISTANCE_IMAGE</code> control (required for <code>McolDraw()</code>), and find a match. Note that if you have normalized distances by the maximum distance calculated, using <code>McolControl()</code> with <code>M_DISTANCE_IMAGE_NORMALIZE</code> set to <code>M_MAX_NORMALIZE</code>, you can draw the distance image in an unsigned image buffer.</p> <div><div></div><div><div><div>ResultArrayPtr info</div><div>Return values:</div><div>32 + M_FLOAT32-bit float.</div></div></div></div>
<div><div></div><div>M_LABEL_AREA_IMAGE_SIGN +</div></div>	<p>Returns the data type required for the image buffer in which to draw <code>M_DRAW_LABEL_AREA_IMAGE</code> (<code>McolDraw()</code> or <code>McolMatch()</code>). This result is available after you enable the <code>M_SAVE_AREA_IMAGE</code> control (required for <code>McolDraw()</code>), and find a match.</p> <div><div></div><div><div><div>ResultArrayPtr info</div><div>Return values:</div><div>M_UNSIGNEDUnsigned data type.</div></div></div></div>
<div><div></div><div>M_LABEL_AREA_IMAGE_SIZE_BAND +</div></div>	<p>Returns the number of surfaces (color bands) required for the image buffer in which to draw <code>M_DRAW_LABEL_AREA_IMAGE</code> (<code>McolDraw()</code> or <code>McolMatch()</code>). This result is available after you enable the <code>M_SAVE_AREA_IMAGE</code> control (required for <code>McolDraw()</code>), and find a match.</p> <div><div></div><div><div><div>ResultArrayPtr info</div><div>Return values:</div><div>11 band.</div></div></div></div>
<div><div></div><div>M_LABEL_AREA_IMAGE_SIZE_BIT +</div></div>	<p>Returns the depth per band, in bits, required for the image buffer in which to draw <code>M_DRAW_LABEL_AREA_IMAGE</code> (<code>McolDraw()</code> or <code>McolMatch()</code>). This result is available after you enable the <code>M_SAVE_AREA_IMAGE</code> control (required for <code>McolDraw()</code>), and find a match.</p> <div><div></div><div><div><div>ResultArrayPtr info</div><div>Return values:</div><div>11-bit data depth per band.</div><div>88-bit data depth per band.</div><div>1616-bit data depth per band.</div></div></div></div>

	32	32-bit data depth per band.
<input type="checkbox"/> M_LABEL_AREA_IMAGE_SIZE_X +	Returns the width, in pixels, required for the image buffer in which to draw M_DRAW_LABEL_AREA_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_SAVE_AREA_IMAGE control (required for McolDraw()), and find a match. (summarize)	
<input type="checkbox"/> M_LABEL_AREA_IMAGE_SIZE_Y +	Returns the height, in pixels, required for the image buffer in which to draw M_DRAW_LABEL_AREA_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_SAVE_AREA_IMAGE control (required for McolDraw()), and find a match. (summarize)	
<input type="checkbox"/> M_LABEL_AREA_IMAGE_TYPE +	Returns the data type and depth required for the image buffer in which to draw M_DRAW_LABEL_AREA_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_SAVE_AREA_IMAGE control (required for McolDraw()), and find a match. (summarize)	
	<i>ResultArrayPtr info</i> Return values: 1 + M_UNSIGNED 1-bit unsigned. 8 + M_UNSIGNED 8-bit unsigned. 16 + M_UNSIGNED 16-bit unsigned. 32 + M_UNSIGNED 32-bit unsigned	
<input type="checkbox"/> M_LABEL_PIXEL_IMAGE_SIGN +	Returns the data type required for the image buffer in which to draw M_DRAW_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE control (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE . (summarize)	
	<i>ResultArrayPtr info</i> Return values: M_UNSIGNED Unsigned data type.	
<input type="checkbox"/> M_LABEL_PIXEL_IMAGE_SIZE_BAND +	Returns the number of surfaces (color bands) required for the image buffer in which to draw M_DRAW_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE control (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE . (summarize)	
	<i>ResultArrayPtr info</i> Return values: 1 1 band.	
<input type="checkbox"/> M_LABEL_PIXEL_IMAGE_SIZE_BIT +	Returns the depth per band, in bits, required for the image buffer in which to draw M_DRAW_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE control (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE . (summarize)	
	<i>ResultArrayPtr info</i> Return values: 1 1-bit data depth per band. 8 8-bit data depth per band. 16 16-bit data depth per band. 32 32-bit data depth per band.	
<input type="checkbox"/> M_LABEL_PIXEL_IMAGE_SIZE_X +	Returns the width, in pixels, required for the image buffer in which to draw M_DRAW_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE control (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE . (summarize)	
<input type="checkbox"/> M_LABEL_PIXEL_IMAGE_SIZE_Y +	Returns the height required for the image buffer in which to draw M_DRAW_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE control (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE . (summarize)	

M_LABEL_PIXEL_IMAGE_TYPE +	<p>Returns the data type and depth required for the image buffer in which to draw M_DRAW_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()).</p> <p>This result is available after you enable the M_GENERATE_LABEL_PIXEL_IMAGE control (required for McolDraw()), and find a match. You must also use McolSetMethod() with M_MIN_DIST_VOTE. (summarize)</p> <table> <tr> <td><i>ResultArrayPtr info</i></td><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>1 + M_UNSIGNED</td><td>1-bit unsigned.</td></tr> <tr> <td>8 + M_UNSIGNED</td><td>8-bit unsigned.</td></tr> <tr> <td>16 + M_UNSIGNED</td><td>16-bit unsigned.</td></tr> <tr> <td>32 + M_UNSIGNED</td><td>32-bit unsigned.</td></tr> </table>	<i>ResultArrayPtr info</i>		Return values:		1 + M_UNSIGNED	1-bit unsigned.	8 + M_UNSIGNED	8-bit unsigned.	16 + M_UNSIGNED	16-bit unsigned.	32 + M_UNSIGNED	32-bit unsigned.
<i>ResultArrayPtr info</i>													
Return values:													
1 + M_UNSIGNED	1-bit unsigned.												
8 + M_UNSIGNED	8-bit unsigned.												
16 + M_UNSIGNED	16-bit unsigned.												
32 + M_UNSIGNED	32-bit unsigned.												
M_MAX_DISTANCE +	Returns the largest color distance among all distances between the target area and all matching color-samples.												
M_NUMBER_OF_AREAS +	Returns the number of target areas in the area identifier image with which you performed the matching operation.												
M_NUMBER_OF_SAMPLES +	Returns the number of color-samples defined in the context.												

To retrieve results relating to a target area, the **ResultType** parameter can be set to one of the following values. In this case, you must set the **AreaLabel** parameter to a specific value (or [M_ALL](#)), and the **ColorSampleIndexOrLabel** parameter to [M_GENERAL](#).

Unless otherwise specified, the following values require that you pass the *ResultArrayPtr* parameter the address of an array of type `MIL_DOUBLE` with a size equal to what is required by the result type (for multiple results) or the address of a `MIL_DOUBLE` (for a single result).

● For retrieving target area results							
Value	Description						
M_AREA_LABEL_VALUE +	<p>Returns the label of the target areas you used to obtain results. If you do not specify an area identifier image with McolMatch(), the whole target image is labeled as target area 1. (summarize)</p>						
M_AREA_PIXEL_COUNT +	Returns the total number of pixels in the target area.						
M_BEST_MATCH_INDEX +	<p>Returns the index of the target area's best-matched color-sample. If there is none, -1 is returned. (summarize)</p>						
M_BEST_MATCH_LABEL +	<p>Returns the label of the target area's best-matched color-sample. If no color-samples are matched, the value you have set using McolControl() with M_OUTLIER_LABEL is returned. (summarize)</p>						
M_OUTLIER_COVERAGE +	<p>Returns the outlier coverage, which is the percentage of the number of pixels in the target area that did not vote for any color-sample. To retrieve this result, you must use McolSetMethod() with M_MIN_DIST_VOTE. (summarize)</p>						
M_SCORE_RELEVANCE +	<p>Returns the relevance score of the target area, in percent. This value indicates the significance (relevance) of M_SCORE. In statistics, this is similar to the confidence level.</p> <p>The target area's relevance score is based on the operation mode specified, using McolSetMethod():</p> <ul style="list-style-type: none"> M_STAT_MIN_DIST. $RelevanceScore = (BestDistance)^{-1} / \sum (Distance^{-1})$, where <i>BestDistance</i> is the distance of the best-matched color-sample, and the sum is over all color-samples that have been matched by the target area. M_MIN_DIST_VOTE. $RelevanceScore = \sum(NumberOfVotes) / NumberOfPixelsInTheTargetArea$, where <i>NumberOfVotes</i> refers to the current color-sample's number of votes, and the sum is over the number of votes for all color-samples. (summarize) 						
M_STATUS +	<p>Returns the match status of the target area. (summarize)</p> <table> <tr> <td><i>ResultArrayPtr info</i></td><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>M_FAILURE</td><td>No color-samples have been matched.</td></tr> </table>	<i>ResultArrayPtr info</i>		Return values:		M_FAILURE	No color-samples have been matched.
<i>ResultArrayPtr info</i>							
Return values:							
M_FAILURE	No color-samples have been matched.						

	M_SUCCESS	At least one color-sample has been matched.
--	-----------	---------------------------------------------

To retrieve results relating to a color-sample, the **ResultType** parameter can be set to one of the following values. In this case, you must set the **AreaLabel** parameter to a specific target area (or [M_ALL](#)) and the **ColorSampleIndexOrLabel** parameter to a specific color-sample (or [M_ALL](#)).

Unless otherwise specified, the following values require that you pass the *ResultArrayPtr* parameter the address of an array of type *MIL_DOUBLE* with a size equal to what is required by the result type (for multiple results) or the address of a *MIL_DOUBLE* (for a single result).

● For retrieving color-sample results	
☐ Value	Description
☐ M_COLOR_DISTANCE +	Returns the color distance between the target area and the best-matched color-sample, when using a minimum distance operation mode (McolSetMethod() with M_STAT_MIN_DIST). You can use McolSetMethod() to set the type of distance to calculate. Note that matching in the CIELAB color space will return distance results in the CIELAB native range. (summarize)
☐ M_SAMPLE_COVERAGE +	Returns the proportion of pixels within the target area that voted for the color-sample, in percent. You must use McolSetMethod() with M_MIN_DIST_VOTE to calculate M_SAMPLE_COVERAGE . (summarize)
☐ M_SAMPLE_LABEL_VALUE +	Returns the label of the color-sample corresponding to the specified index.
☐ M_SAMPLE_MATCH_STATUS +	Returns whether the specified color-sample fulfills the match conditions, with respect to the distance tolerance and acceptance. To determine if the color-sample is the best-matched color-sample, use M_BEST_MATCH_INDEX or M_BEST_MATCH_LABEL . (summarize) <div> <i>ResultArrayPtr info</i> Return values: <div>M_MATCH The color-sample fulfills the match conditions.</div> <div>M_NO_MATCH The color-sample does not fulfill the match conditions.</div> </div>
☐ M_SAMPLE_PIXEL_COUNT +	Returns the number of pixels within the target area that voted for the color-sample. You must use McolSetMethod() with M_MIN_DIST_VOTE to calculate M_SAMPLE_PIXEL_COUNT . (summarize)
☐ M_SCORE +	Returns the match score, in percent. This score indicates the similarity between the color of the color-sample and the color of the target area. The color-sample's match score is based on the operation mode specified, using McolSetMethod() : <ul style="list-style-type: none"> M_STAT_MIN_DIST. $MatchScore = \lceil k / (k + Distance) \rceil \times [(MaxDistance - Distance) / MaxDistance]$, where <i>Distance</i> refers to the current color-sample distance and <i>MaxDistance</i> refers to the maximum distance possible in the working color space. M_MIN_DIST_VOTE. $MatchScore = NumberOfVotes / \text{sum}(NumberOfVotes)$, where <i>NumberOfVotes</i> refers to the current color-sample's number of votes, and the sum is over the number of votes for all color-samples. (summarize)

Combination constant for [the values listed in](#) all tables

You can add the following value to the above-mentioned values to determine whether a result is available.

● For any result	
☐ Value	Description
☐ M_AVAILABLE	Returns whether a result is available to be retrieved. (summarize) <div> <i>ResultArrayPtr info</i> Return values: </div>

M_NULL	The result is not available to be retrieved.
Value != 0	The result is available to be retrieved.

Combination constants for [the values listed in](#) all tables

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

Some results might be altered if they do not fit numerically in the data type you specify. For example, if you cast a result value of 300 to a *char*, information will be lost. The default data type, *double*, will never result in lost information.

For specifying the data type

Value	Description
M_TYPE_CHAR	<p>Casts the requested results to a <i>char</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>char</i> Array size: what is required by the result type Note: for multiple results • Data type: <i>char</i> Note: for a single result
M_TYPE_DOUBLE	<p>Casts the requested results to a <i>double</i>. This is the default value. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>double</i> Array size: what is required by the result type Note: for multiple results • Data type: <i>double</i> Note: for a single result
M_TYPE_FLOAT	<p>Casts the requested results to a <i>float</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>float</i> Array size: what is required by the result type Note: for multiple results • Data type: <i>float</i> Note: for a single result
M_TYPE_LONG	<p>Casts the requested results to a <i>long</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>long</i> Array size: what is required by the result type Note: for multiple results • Data type: <i>long</i> Note: for a single result
M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>MIL_DOUBLE</i> Array size: what is required by the result type Note: for multiple results • Data type: <i>MIL_DOUBLE</i> Note: for a single result

<input type="checkbox"/> M_TYPE_MIL_ID	Casts the requested results to a <i>MIL_ID</i> . (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_ID Array size: what is required by the result type Note: for multiple results • Data type: MIL_ID Note: for a single result
<input type="checkbox"/> M_TYPE_MIL_INT	Casts the requested results to a <i>MIL_INT</i> . (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: what is required by the result type Note: for multiple results • Data type: MIL_INT Note: for a single result
<input type="checkbox"/> M_TYPE_MIL_INT32	Casts the requested results to a <i>MIL_INT32</i> . (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: what is required by the result type Note: for multiple results • Data type: MIL_INT32 Note: for a single result
<input type="checkbox"/> M_TYPE_MIL_INT64	Casts the requested results to a <i>MIL_INT64</i> . (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT64 Array size: what is required by the result type Note: for multiple results • Data type: MIL_INT64 Note: for a single result
<input type="checkbox"/> M_TYPE_SHORT	Casts the requested results to a <i>short</i> . (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type short Array size: what is required by the result type Note: for multiple results • Data type: short Note: for a single result

ResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none"> • array of type char • array of type double • array of type float • array of type long • array of type MIL_DOUBLE • array of type MIL_ID • array of type MIL_INT • array of type MIL_INT32 • array of type MIL_INT64 • array of type short • char • double • float • long

- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT32
- MIL_INT64
- short

Specifies the address of the first element of the array in which to write the requested information.

This array must be large enough to hold all entries. For example, if you retrieve results for all target areas and color-samples, you must account for the following array size: *NumberOfTargetAreas* x *NumberOfSamples*.

You must set this parameter to **M_NULL** if you are putting the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolInquire

Synopsis

Inquire information about a specified color matching context or color-sample contained therein.

Syntax

```
MIL_INT McolInquire(
    MIL_ID ColorObjectId,
    MIL_INT IndexOrLabel,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires information about a specified color matching context, or a color-sample contained therein. You can inquire all context and color-sample control values that you have set using [McolControl\(\)](#).

If the inquired control value setting is **M_DEFAULT**, **McolInquire()** returns **M_DEFAULT**. To inquire the actual default value, add **M_DEFAULT** to the inquire type.

Parameters

ColorObjectId

Specifies the identifier of the color matching context about which to inquire information. The context must have been previously allocated on the required system using [McolAlloc\(\)](#).

IndexOrLabel

Specifies whether you will inquire information about a color matching context or an individual color-sample therein. This parameter should be set to one of the following values:

● For specifying a context, model, or result buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_CONTEXT .
<input type="checkbox"/> M_SAMPLE_INDEX(MIL_INT <i>SampleIndex</i>)	Specifies the index of an individual color-sample about which to inquire. (summarize)
	<div>Parameters</div> <div><i>SampleIndex</i> Specifies the index. The index is from 0 to the number of defined color-samples minus 1.</div>
<input type="checkbox"/> M_SAMPLE_LABEL(MIL_INT <i>SampleLabel</i>)	Specifies the label of an individual color-sample about which to inquire. (summarize)
	<div>Parameters</div> <div><i>SampleLabel</i> Specifies the label.</div>
<input type="checkbox"/> M_CONTEXT	Specifies to inquire information about a color matching context.

InquireType

Specifies the type of setting about which to inquire.

To inquire about color matching context settings, set this parameter to one of the following values. In this case, you must set the [IndexOrLabel](#) parameter to [M_CONTEXT](#).

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For color matching context settings	
☐ Value	Description
☐ M_ACCEPTANCE +	<p>Returns the acceptance level for the color-sample's score. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>
☐ M_ACCEPTANCE_RELEVANCE +	<p>Returns the acceptance level for the target area's relevance score. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>
☐ M_BACKGROUND_DRAW_COLOR +	<p>Returns the color used to draw background pixels when drawing or generating M_DRAW_COLORED_LABEL_AREA_IMAGE, M_DRAW_COLORED_LABEL_PIXEL_IMAGE, or M_DRAW_DISTANCE_IMAGE, using McolDraw() or McolMatch(). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_TRANSPARENT; Value > 0; (details)</p>
☐ M_BAND_MODE +	<p>Returns the color bands to use when matching colors. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_ALL_BANDS; M_COLOR_BAND_0; M_COLOR_BAND_1; M_COLOR_BAND_2; M_COLOR_BAND_0 + M_COLOR_BAND_1; M_COLOR_BAND_0 + M_COLOR_BAND_2; M_COLOR_BAND_1 + M_COLOR_BAND_2; M_DEFAULT; (details)</p>
☐ M_COLOR_SPACE +	<p>Returns the context's color space. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_CIELAB; M_DEFAULT; M_HSL; M_RGB; (details)</p>
☐ M_DISTANCE_IMAGE_NORMALIZE +	<p>Returns the normalization factor when drawing or generating M_DRAW_DISTANCE_IMAGE, using McolDraw() or McolMatch(). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_MAX_NORMALIZE; M_NO_NORMALIZE; Value > 0.0; (details)</p>
☐ M_DISTANCE_TOLERANCE_MODE +	<p>Returns the strategy with which to calculate M_DISTANCE_TOLERANCE (McolControl()). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_ABSOLUTE; M_DEFAULT; M_RELATIVE; M_SAMPLE_STDDEV; (details)</p>
☐ M_DISTANCE_TYPE +	<p>Returns the type of distance calculated when matching colors. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DELTA_E; M_EUCLIDEAN; M_MAHALANOBIS; M_MANHATTAN; (details)</p>
☐ M_GENERATE_DISTANCE_IMAGE +	<p>Returns whether M_GENERATE_DISTANCE_IMAGE (McolControl()) was enabled. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>
☐ M_GENERATE_LABEL_PIXEL_IMAGE +	<p>Returns whether M_GENERATE_LABEL_PIXEL_IMAGE was enabled. (summarize)</p> <p><i>UserVarPtr info</i></p>

	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)								
<input type="checkbox"/> M_GENERATE_SAMPLE_COLOR_LUT +	<p>Returns whether M_GENERATE_SAMPLE_COLOR_LUT was enabled. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>								
<input type="checkbox"/> M_LABEL_AREA_IMAGE_SIZE_BIT +	<p>Returns the depth per band, in bits, required for the image buffer in which to draw or generate M_DRAW_LABEL_AREA_IMAGE (McolDraw() or McolMatch()). (summarize)</p> <p><i>UserVarPtr info</i> Return values:</p> <table> <tr> <td>1</td><td>1-bit data depth per band.</td></tr> <tr> <td>8</td><td>8-bit data depth per band.</td></tr> <tr> <td>16</td><td>16-bit data depth per band.</td></tr> <tr> <td>32</td><td>32-bit data depth per band.</td></tr> </table>	1	1-bit data depth per band.	8	8-bit data depth per band.	16	16-bit data depth per band.	32	32-bit data depth per band.
1	1-bit data depth per band.								
8	8-bit data depth per band.								
16	16-bit data depth per band.								
32	32-bit data depth per band.								
<input type="checkbox"/> M_LABEL_PIXEL_IMAGE_SIZE_BIT +	<p>Returns the depth per band, in bits, required for the image buffer in which to draw or generate M_DRAW_LABEL_PIXEL_IMAGE (McolDraw() or McolMatch()). (summarize)</p> <p><i>UserVarPtr info</i> Return values:</p> <table> <tr> <td>1</td><td>1-bit data depth per band.</td></tr> <tr> <td>8</td><td>8-bit data depth per band.</td></tr> <tr> <td>16</td><td>16-bit data depth per band.</td></tr> <tr> <td>32</td><td>32-bit data depth per band.</td></tr> </table>	1	1-bit data depth per band.	8	8-bit data depth per band.	16	16-bit data depth per band.	32	32-bit data depth per band.
1	1-bit data depth per band.								
8	8-bit data depth per band.								
16	16-bit data depth per band.								
32	32-bit data depth per band.								
<input type="checkbox"/> M_MATCH_MODE +	<p>Returns the operation mode used when matching colors. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_MIN_DIST_VOTE; M_STAT_MIN_DIST; (details)</p>								
<input type="checkbox"/> M_MODIFICATION_COUNT +	<p>Returns the current value of the modification counter. The modification counter is increased by one each time settings for the color matching context are modified. Although you cannot identify the modification counter's contents, you can compare them throughout your application to inquire if the context has been altered. If the modification counter has changed you can, for example, prompt the user to save before closing the application. (summarize)</p>								
<input type="checkbox"/> M_NUMBER_OF_SAMPLES +	Returns the number of color-samples defined in the context with McolDefine() .								
<input type="checkbox"/> M_OUTLIER_DRAW_COLOR +	<p>Returns the color used to draw outlier pixels when drawing or generating M_DRAW_COLORED_LABEL_AREA_IMAGE, M_DRAW_COLORED_LABEL_PIXEL_IMAGE, or M_DRAW_DISTANCE_IMAGE, using McolDraw() or McolMatch(). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_TRANSPARENT; Value > 0; (details)</p>								
<input type="checkbox"/> M_OUTLIER_LABEL +	<p>Returns the label used for outlier pixels when drawing or generating M_DRAW_LABEL_AREA_IMAGE and M_DRAW_LABEL_PIXEL_IMAGE, using McolDraw() or McolMatch(). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)</p>								
<input type="checkbox"/> M_OWNER_SYSTEM +	<p>Returns the identifier of the system on which either the color matching context or color matching result buffer was allocated. (summarize)</p> <p><i>UserVarPtr info</i> Return values: MIL system identifier; M_DEFAULT_HOST; (details)</p>								
<input type="checkbox"/> M_PRECONVERT_GAMMA +	<p>Returns whether M_PRECONVERT_GAMMA (McolControl()) was enabled. (summarize)</p> <p><i>UserVarPtr info</i></p>								

	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_PRECONVERT_MODE +	<p>Returns the mode used to internally convert colors before the matching operation. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_CIELAB; M_DEFAULT; M_NONE; (details)</p>
<input type="checkbox"/> M_PREPROCESSED +	<p>Returns whether the color matching context is preprocessed. The context must be preprocessed before calling McolMatch(). This inquire type will indicate that the context is not preprocessed after certain settings of the color matching context are changed with McolControl(), and after a color-sample is added or removed with McolDefine(). (summarize)</p> <p><i>UserVarPtr info</i> Return values: Value!= 0 The context is preprocessed. 0 The context is not preprocessed.</p>
<input type="checkbox"/> M_SAMPLE_LUT_SIGN +	<p>Returns the data type required for the image buffer in which to draw M_DRAW_SAMPLE_COLOR_LUT (McolDraw()). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_UNSIGNED Unsigned data type.</p>
<input type="checkbox"/> M_SAMPLE_LUT_SIZE_BAND +	<p>Returns the number of surfaces (color bands) required for the image buffer in which to draw M_DRAW_SAMPLE_COLOR_LUT (McolDraw()). (summarize)</p> <p><i>UserVarPtr info</i> Return values: 3 3 bands.</p>
<input type="checkbox"/> M_SAMPLE_LUT_SIZE_BIT +	<p>Returns the depth per band, in bits, required for the image buffer in which to draw M_DRAW_SAMPLE_COLOR_LUT (McolDraw()). (summarize)</p> <p><i>UserVarPtr info</i> Return values: 8 8-bit data depth per band.</p>
<input type="checkbox"/> M_SAMPLE_LUT_SIZE_X +	Returns the width, in pixels, required for the image buffer in which to draw M_DRAW_SAMPLE_COLOR_LUT (McolDraw()).
<input type="checkbox"/> M_SAMPLE_LUT_SIZE_Y +	Returns the height, in pixels, required for the image buffer in which to draw M_DRAW_SAMPLE_COLOR_LUT (McolDraw()).
<input type="checkbox"/> M_SAMPLE_LUT_TYPE +	<p>Returns the data type and depth required for the image buffer in which to draw M_DRAW_SAMPLE_COLOR_LUT (McolDraw()). (summarize)</p> <p><i>UserVarPtr info</i> Return values: 8 + M_UNSIGNED 8-bit unsigned.</p>
<input type="checkbox"/> M_SAVE_AREA_IMAGE +	<p>Returns whether M_SAVE_AREA_IMAGE was enabled. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>

To inquire about color matching context settings that are used for color space encoding, set this parameter to one of the following values. In this case, you must set the [IndexOrLabel](#) parameter to [M_CONTEXT](#).

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For inquiring context settings that are used for color space encoding	
<input type="checkbox"/> Value	Description

M_ENCODING +	Returns the color space encoding. (summarize)
	<i>UserVarPtr info</i> Return values: M_8BIT; M_10BIT; M_12BIT; M_14BIT; M_16BIT; M_DEFAULT; M_USER_DEFINED; (details)
M_OFFSET_BAND_0 +	Returns the offset for the color space encoding, for band 0. This inquire is only applicable if you use McolControl() with M_ENCODING set to M_USER_DEFINED . (summarize)
	<i>UserVarPtr info</i> Return values: Please see McolControl() with M_OFFSET_BAND_0 . (details)
M_OFFSET_BAND_1 +	Returns the offset for the color space encoding, for band 1. This inquire is only applicable if you use McolControl() with M_ENCODING set to M_USER_DEFINED . (summarize)
	<i>UserVarPtr info</i> Return values: Please see McolControl() with M_OFFSET_BAND_1 . (details)
M_OFFSET_BAND_2 +	Returns the offset for the color space encoding, for band 2. This inquire is only applicable if you use McolControl() with M_ENCODING set to M_USER_DEFINED . (summarize)
	<i>UserVarPtr info</i> Return values: Please see McolControl() with M_OFFSET_BAND_2 . (details)
M_SCALE_BAND_0 +	Returns the scale for the color space encoding, for band 0. This inquire is only applicable if you use McolControl() with M_ENCODING set to M_USER_DEFINED . (summarize)
	<i>UserVarPtr info</i> Return values: Please see McolControl() with M_SCALE_BAND_0 . (details)
M_SCALE_BAND_1 +	Returns the scale for the color space encoding, for band 1. This inquire is only applicable if you use McolControl() with M_ENCODING set to M_USER_DEFINED . (summarize)
	<i>UserVarPtr info</i> Return values: Please see McolControl() with M_SCALE_BAND_1 . (details)
M_SCALE_BAND_2 +	Returns the scale for the color space encoding, for band 2. This inquire is only applicable if you use McolControl() with M_ENCODING set to M_USER_DEFINED . (summarize)
	<i>UserVarPtr info</i> Return values: Please see McolControl() with M_SCALE_BAND_2 . (details)

To inquire about a color-sample defined in a color matching context, set this parameter to one of the following values. In this case, you must set the [IndexOrLabel](#) parameter to [M_SAMPLE_INDEX\(\)](#) or [M_SAMPLE_LABEL\(\)](#).

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

For a color-sample defined in a color matching context	
Value	Description
M_DISTANCE_TOLERANCE +	Returns the acceptable tolerance for the color distance between the color-sample and a target area. (summarize)
	<i>UserVarPtr info</i>

	Return values: M_AUTO; M_DEFAULT; M_INFINITE; Value >= 0.0; (details)								
<input type="checkbox"/> M_MATCH_SAMPLE_COLOR_BAND_0 +	<p>Returns the preprocessed color-sample average value for band 0.</p> <p>This information is available only if you have selected the band for matching (McolControl() with M_BAND_MODE) and have preprocessed the color matching context (McolPreprocess()).</p> <p>(summarize)</p>								
<input type="checkbox"/> M_MATCH_SAMPLE_COLOR_BAND_1 +	<p>Returns the preprocessed color-sample average value for band 1.</p> <p>This information is available only if you have selected the band for matching (McolControl() with M_BAND_MODE) and have preprocessed the color matching context (McolPreprocess()).</p> <p>(summarize)</p>								
<input type="checkbox"/> M_MATCH_SAMPLE_COLOR_BAND_2 +	<p>Returns the preprocessed color-sample average value for band 2.</p> <p>This information is available only if you have selected the band for matching (McolControl() with M_BAND_MODE) and have preprocessed the color matching context (McolPreprocess()).</p> <p>(summarize)</p>								
<input type="checkbox"/> M_SAMPLE_ABSOLUTE_TOLERANCE +	<p>Returns the maximum distance a color-sample can have for it to be matched. To retrieve this result you must first preprocess the color matching context, using McolPreprocess().</p> <p>(summarize)</p>								
<input type="checkbox"/> M_SAMPLE_COLOR_BAND_0 +	<p>Returns the color value for band 0 of a triplet color-sample.</p> <p>This information is available only for color-samples that you define using McolDefine() with M_TRIPLET.</p> <p>(summarize)</p>								
<input type="checkbox"/> M_SAMPLE_COLOR_BAND_1 +	<p>Returns the color value for band 1 of a triplet color-sample.</p> <p>This information is available only for color-samples that you define using McolDefine() with M_TRIPLET.</p> <p>(summarize)</p>								
<input type="checkbox"/> M_SAMPLE_COLOR_BAND_2 +	<p>Returns the color value for band 2 of a triplet color-sample.</p> <p>This information is available only for color-samples that you define using McolDefine() with M_TRIPLET.</p> <p>(summarize)</p>								
<input type="checkbox"/> M_SAMPLE_IMAGE_SIGN +	<p>Returns the data type of the image from which the color-sample was defined.</p> <p>This information is available only for color-samples that you define using McolDefine() with M_IMAGE.</p> <p>(summarize)</p> <table> <tr> <td><i>UserVarPtr info</i></td><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>M_UNSIGNED</td><td>Unsigned data type.</td></tr> </table>	<i>UserVarPtr info</i>		Return values:		M_UNSIGNED	Unsigned data type.		
<i>UserVarPtr info</i>									
Return values:									
M_UNSIGNED	Unsigned data type.								
<input type="checkbox"/> M_SAMPLE_IMAGE_SIZE_BAND +	<p>Returns the number of surfaces (color bands) of the image from which the color-sample was defined.</p> <p>This information is available only for color-samples that you define using McolDefine() with M_IMAGE.</p> <p>(summarize)</p> <table> <tr> <td><i>UserVarPtr info</i></td><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>3</td><td>3 bands.</td></tr> </table>	<i>UserVarPtr info</i>		Return values:		3	3 bands.		
<i>UserVarPtr info</i>									
Return values:									
3	3 bands.								
<input type="checkbox"/> M_SAMPLE_IMAGE_SIZE_BIT +	<p>Returns the depth per band, in bits, of the image from which the color-sample was defined.</p> <p>This information is available only for color-samples that you define using McolDefine() with M_IMAGE.</p> <p>(summarize)</p> <table> <tr> <td><i>UserVarPtr info</i></td><td></td></tr> <tr> <td>Return values:</td><td></td></tr> <tr> <td>8</td><td>8-bit data depth per band.</td></tr> <tr> <td>16</td><td>16-bit data depth per band.</td></tr> </table>	<i>UserVarPtr info</i>		Return values:		8	8-bit data depth per band.	16	16-bit data depth per band.
<i>UserVarPtr info</i>									
Return values:									
8	8-bit data depth per band.								
16	16-bit data depth per band.								
<input type="checkbox"/> M_SAMPLE_IMAGE_SIZE_X +	<p>Returns the width, in pixels, of the image from which the color-sample was defined.</p> <p>This information is available only for color-samples that you define using McolDefine() with M_IMAGE.</p> <p>(summarize)</p>								
<input type="checkbox"/> M_SAMPLE_IMAGE_SIZE_Y +	<p>Returns the height, in pixels, of the image from which the color-sample was defined.</p> <p>This information is available only for color-samples that you define using McolDefine() with M_IMAGE.</p>								

	(summarize)
M_SAMPLE_IMAGE_TYPE +	<p>Returns the data type and depth of the image from which the color-sample was defined. This information is available only for color-samples that you define using McolDefine() with M_IMAGE. (summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div> <div>8 + M_UNSIGNED</div> <div>8-bit unsigned.</div> </div> <div> <div>16 + M_UNSIGNED</div> <div>16-bit unsigned.</div> </div> </div>
M_SAMPLE_LABEL_VALUE +	Returns the current label of the color-sample.
M_SAMPLE_MASK_SIGN +	<p>Returns the data type of the color-sample's mask buffer. This information is available only if you define a mask, using McolMask(). (summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div> <div>M_UNSIGNED</div> <div>Unsigned data type.</div> </div> </div>
M_SAMPLE_MASK_SIZE_BAND +	<p>Returns the number of surfaces (color bands) of the color-sample's mask buffer. This information is available only if you define a mask, using McolMask(). (summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div> <div>1</div> <div>1 band.</div> </div> </div>
M_SAMPLE_MASK_SIZE_BIT +	<p>Returns the depth per band, in bits, of the color-sample's mask buffer. This information is available only if you define a mask, using McolMask(). (summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div> <div>1</div> <div>1-bit data depth per band.</div> </div> <div> <div>8</div> <div>8-bit data depth per band.</div> </div> </div>
M_SAMPLE_MASK_SIZE_X +	<p>Returns the width of the color-sample's mask buffer, in pixels. This information is available only if you define a mask, using McolMask(). (summarize)</p>
M_SAMPLE_MASK_SIZE_Y +	<p>Returns the height of the color-sample's mask buffer, in pixels. This information is available only if you define a mask, using McolMask(). (summarize)</p>
M_SAMPLE_MASK_TYPE +	<p>Returns the data type and depth of the color-sample's mask buffer. This information is available only if you define a mask, using McolMask(). (summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div> <div>1 + M_UNSIGNED</div> <div>1-bit unsigned.</div> </div> <div> <div>8 + M_UNSIGNED</div> <div>8-bit unsigned.</div> </div> </div>
M_SAMPLE_MASKED +	<p>Returns whether a mask for the color-sample was defined. (summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div> <div>M_FALSE</div> <div>A mask was not defined.</div> </div> </div>

	M_TRUE	A mask was defined.
M_SAMPLE_TYPE +	Returns the type of the color-sample defined with McolDefine() . (summarize)	
	<i>UserVarPtr info</i> Return values:	
	M_IMAGE	The color-sample was added from an image.
	M_TRIPLET	The color-sample was added from three explicit color component values (RGB, HSL, or CIELAB).

Combination constant for [the values listed in](#) all tables

You can add the following value to the above-mentioned values to determine the default value of an inquire type.

● For inquiring the default value of an inquire type		
Value	Description	
M_DEFAULT	Returns the default value of the specified inquire type. (summarize)	
		<i>UserVarPtr info</i> Data type: MIL_DOUBLE

Combination constants for [the values listed in](#) all tables

You can add one of the following values to the above-mentioned values to cast the requested information to the required data type.

Some inquired information might be altered if it does not fit numerically in the data type you specify. For example, if you cast a value of 300 to a *char*, information will be lost. The default data type, *double*, will never result in lost information.

● For specifying the data type		
Value	Description	
M_TYPE_CHAR	Casts the requested information to a <i>char</i> . (summarize)	
		<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type char Array size: what is required by the inquire type Note: for multiple inquires • Data type: char Note: for a single inquire
M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . This is the default value. (summarize)	
		<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type double Array size: what is required by the inquire type Note: for multiple inquires • Data type: double Note: for a single inquire
M_TYPE_FLOAT	Casts the requested information to a <i>float</i> . (summarize)	
		<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type float

	<p>Array size: what is required by the inquire type Note: for multiple inquiries</p> <ul style="list-style-type: none"> • Data type: float Note: for a single inquire
<input type="checkbox"/> M_TYPE_LONG	<p>Casts the requested information to a <i>long</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type long Array size: what is required by the inquire type Note: for multiple inquiries • Data type: long Note: for a single inquire
<input type="checkbox"/> M_TYPE_MIL_DOUBLE	<p>Casts the requested information to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: what is required by the inquire type Note: for multiple inquiries • Data type: MIL_DOUBLE Note: for a single inquire
<input type="checkbox"/> M_TYPE_MIL_ID	<p>Casts the requested information to a <i>MIL_ID</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_ID Array size: what is required by the inquire type Note: for multiple inquiries • Data type: MIL_ID Note: for a single inquire
<input type="checkbox"/> M_TYPE_MIL_INT	<p>Casts the requested information to a <i>MIL_INT</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: what is required by the inquire type Note: for multiple inquiries • Data type: MIL_INT Note: for a single inquire
<input type="checkbox"/> M_TYPE_MIL_INT32	<p>Casts the requested information to a <i>MIL_INT32</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: what is required by the inquire type Note: for multiple inquiries • Data type: MIL_INT32 Note: for a single inquire
<input type="checkbox"/> M_TYPE_MIL_INT64	<p>Casts the requested information to a <i>MIL_INT64</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT64 Array size: what is required by the inquire type Note: for multiple inquiries • Data type: MIL_INT64 Note: for a single inquire
<input type="checkbox"/> M_TYPE_SHORT	<p>Casts the requested information to a <i>short</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type short Array size: what is required by the inquire type

Note: for multiple inquires

- Data type: short

Note: for a single inquire

Combination constant for [the values listed in](#) all tables

You can add the following value to the above-mentioned values to determine whether an inquire type is supported.

● For inquiring whether an inquire type is supported

Value	Description
M_SUPPORTED	Returns whether the specified inquire type is supported for the color matching context. (summarize)
	<i>UserVarPtr info</i> Return values: M_FALSE The inquire type is not supported. M_TRUE The inquire type is supported.

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type char
- array of type double
- array of type float
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_ID
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- array of type short
- char
- double
- float
- long
- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT32
- MIL_INT64
- short

Specifies the address in which to write the requested information. Since the **McolInquire()** function also returns the requested information, you can set this parameter to **M_NULL**.

Return value

The return value is the required information, cast to a *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolMask

Synopsis

Mask regions of a color-sample.

Syntax

```
void McolMask(  
    MIL_ID ColorContextId,  
    MIL_INT ColorSampleIndexOrLabel,  
    MIL_ID MaskBufferId,  
    MIL_INT MaskType,  
    MIL_INT ControlFlag  
)
```

Description

This function allows you to apply masks to a specified color-sample. To apply a mask, you must have added the color-sample to the color matching context using [McolDefine\(\)](#) with [M_IMAGE](#).

With each mask, you can set pixels in the specified color-sample to a "don't care" state. "Don't care" pixels in a color-sample will not be considered by the matching operation.

Parameters

ColorContextId

Specifies the color matching context of the color-sample to mask. You must first allocate the color matching context on the system using [McolAlloc\(\)](#).

ColorSampleIndexOrLabel

Specifies the color-sample for which to create a mask.

For specifying the individual color-sample	
Value	Description
<div><div></div><div>M_SAMPLE_INDEX(MIL_INT SampleIndex)</div></div>	Specifies the index of an existing color-sample for which to create a mask. (summarize)
	Parameters
	<div>SampleIndex</div> <div>Specifies the index. The indices range from 0 to the number of defined color-samples in the context minus 1.</div>
<div><div></div><div>M_SAMPLE_LABEL(MIL_INT SampleLabel)</div></div>	Specifies the label of an existing color-sample for which to create a mask. (summarize)
	Parameters
	<div>SampleLabel</div> <div>Specifies the label.</div>

MaskBufferId

Specifies the identifier of the buffer to use to identify the masked (non-zero) pixels in the color-sample. This buffer must be a 1-band 8-bit unsigned image buffer.

If the mask is not the same size as the color-sample, the masking operation considers only the common region between both images. A common origin at the top-left corner of the mask image is assumed. If the mask is smaller than the color-sample, the color-sample region exceeding the mask is considered unmasked. If the mask is greater than the color-sample, the mask is clipped. An error is generated if the color-sample is entirely masked.

To remove the mask, set the **MaskBufferId** parameter to **M_NULL**. An error is generated if you try removing a mask that does not exist.

MaskType

Specifies the type of mask to employ. Set this parameter to the following value:

● For specifying the type of mask	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DONT_CARES	Specifies that non-zero pixels in the color-sample's masked region are ignored by the matching operation. These pixels will therefore not affect any of the scores. (summarize)

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolMatch

Synopsis

Match target areas with color-samples.

Syntax

```
void McolMatch(
    MIL_ID ColorMatchingContextId,
    MIL_ID TargetColorImageId,
    MIL_ID TargetColorProfileId,
    MIL_ID AreaLabelImageId,
    MIL_ID ColorResultorDestImageId,
    MIL_INT ControlFlag
)
```

Description

This function matches the target areas, which are defined in a target image, with the color-samples, which are defined in a color matching context. This allows you to identify a target area's best-matched color-sample, or to determine the proportion of color in a target area, based on the color-samples provided.

To match multiple target areas, you must specify an area identifier image. Each unique, non-zero label in the area identifier image defines an independent target area. Touching target areas with different labels are considered distinct. If you use an area identifier image that does not contain any non-zero labels, the match is not performed, no resulting image is produced, and all results at the target area or color-sample level are unavailable.

To retrieve all results from the match, set the **ControlFlag** parameter to [M_DEFAULT](#), and then use [McolGetResult\(\)](#). To retrieve only an image result, set the **ControlFlag** parameter to **M_DRAW_..._IMAGE**; the image is written in the specified destination image buffer (**ColorResultorDestImageId**).

The bit depth of the destination image buffer must accommodate all the color-sample data drawn, including the background color, the outlier color, and the outlier labels. To specify these values, use [McolControl\(\)](#) with [M_BACKGROUND_DRAW_COLOR](#), [M_OUTLIER_DRAW_COLOR](#), and [M_OUTLIER_LABEL](#). The numerical value of the color-sample labels, the background pixels, and the outlier pixels and labels should be different so you can easily identify them.

By default, a minimum distance operation is used to match the color of the target area with the color of the color-sample. To change the default operation mode, use [McolSetMethod\(\)](#). You can also use [McolSetMethod\(\)](#) to change the default distance type of the match operation.

For color-samples defined from an image, an estimation of the color is determined to represent the color-sample. This estimate is based on color statistics, such as the mean; it is considered the color of the color-sample and is used in the match. The same kind of color estimation is also calculated for target areas, when matching with a minimum distance operation.

The format of your color data must be consistent. For example, you will not receive an error if you match an RGB target area with an HSL color-sample, or a 16-bit target area with an 8-bit color-sample. Color data is used as is, and is considered to be within the context's color space ([McolAlloc\(\)](#)); there is no internal conversion. Color data is interpreted band per band, independently of the buffer's format.

You must set the color space encoding according to the actual dynamic range of your color data, using [McolControl\(\)](#) with [M_ENCODING](#). By default, the Color Analysis module assumes an 8-bit color space encoding (regardless of the buffer depth).

Certain modifications to the color matching context, such as adding or deleting color-samples, or changing some control settings, require you to preprocess the color matching context ([McolPreprocess\(\)](#)) before any subsequent call to **McolMatch()**. To inquire whether you should preprocess, use [McolInquire\(\)](#) with [M_PREPROCESSED](#).

Parameters

ColorMatchingContextId

Specifies the identifier of the color matching context. The color matching context must have been previously allocated on the required system using [McolAlloc\(\)](#).

TargetColorImageId

Specifies the identifier of the target image buffer with which to match the color-samples.

Target images must be 3-band 8- or 16-bit unsigned images. The target image and the color-sample must have the same color space encoding.

TargetColorProfileId

Specifies the identifier of the target image's reference color space. This is the standard used to interpret the target's color data when performing an internal conversion before the match operation, using [McolSetMethod\(\)](#) with [M_CIELAB](#).

This parameter must be set to the following value:

● For specifying the target image's color space profile	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies to use standard RGB specifications (sRGB), as defined by the International Electrotechnical Commission (IEC) Project Team 61966-2-1.

AreaLabelImageId

Specifies the identifier of the image buffer containing the target area labels. Area identifier images can be unsigned 1-band, packed binary, 8- or 16-bit grayscale images.

To match the entire target image as one target area, set this parameter to **M_NULL**.

ColorResultorDestImageId

Specifies the identifier of the result buffer or the image buffer in which to write the results of the color matching operation.

Result buffers must have been previously allocated on the required system, using [McolAllocResult\(\)](#).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

You must allocate the result buffer (**ColorResultorDestImageId**) on the same system as the color matching context (**ColorMatchingContextId**), otherwise an error will occur.

ControlFlag

Specifies the type of results to produce.

For specifying all results, this parameter must be set to the following value. In this case, you must set the **ColorResultorDestImageId** parameter to the identifier of a result buffer.

● For specifying all results	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Calculates all possible results. For M_DEFAULT , use McolGetResult() to retrieve results; in this case, use McolDraw() to draw the output images (M_DRAW_____IMAGE). To do so, you must first enable the appropriate image controls, using McolControl() with M_GENERATE_____ and M_SAVE_AREA_IMAGE . (summarize)

For specifying that a specific image should be drawn (**M_DRAW_____IMAGE**), this parameter must be set to one the following values. In this case, you must set the **ColorResultorDestImageId** parameter to the identifier of an image buffer. The bit depth of this buffer must accommodate all the color-sample data drawn, including background and outlier pixels. You can only draw one image at a time.

● For specifying a specific image to draw	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_COLORED_LABEL_AREA_IMAGE	Draws, for each target area, the color of the best-matched color-sample. The background color and outlier color are also drawn. For M_DRAW_COLORED_LABEL_AREA_IMAGE , you must set the ColorResultorDestImageId parameter to the identifier of a 3-band color image buffer. (summarize)
<input type="checkbox"/> M_DRAW_COLORED_LABEL_PIXEL_IMAGE	Draws, for each pixel in each target area, the color of the color-sample for which each pixel voted. The background color and outlier color are also drawn. For M_DRAW_COLORED_LABEL_PIXEL_IMAGE , you must set the ColorResultorDestImageId parameter to the identifier of a 3-band color image buffer, and you must use McolSetMethod() with M_MIN_DIST_VOTE . (summarize)

☐ M_DRAW_DISTANCE_IMAGE	<p>Draws the distance between the color of the target area (for an M_STAT_MIN_DIST operation mode) or target pixel (for an M_MIN_DIST_VOTE operation mode), and the color of its best-matched color-sample. Operation modes are set with McolSetMethod(). The background color and outlier color are also drawn. For M_DRAW_DISTANCE_IMAGE, you must set the ColorResultorDestImageId parameter to the identifier of a grayscale image buffer.</p> <p>To normalize distances, use McolControl() with M_DISTANCE_IMAGE_NORMALIZE.</p> <p>Although outlier colors are drawn with M_DRAW_DISTANCE_IMAGE, it might be difficult to identify them. In this case, use M_DRAW_LABEL_..., which draws the outlier labels.</p> <p>(summarize)</p>
☐ M_DRAW_LABEL_AREA_IMAGE	<p>Draws, for each target area, the label of the best-matched color-sample. The background color and outlier label are also drawn. For M_DRAW_LABEL_AREA_IMAGE, you must set the ColorResultorDestImageId parameter to the identifier of a grayscale image buffer. To inquire the required buffer depth, use McolInquire() with M_LABEL_AREA_IMAGE_SIZE_BIT.</p> <p>(summarize)</p>
☐ M_DRAW_LABEL_PIXEL_IMAGE	<p>Draws, for each pixel in each target area, the label of the color-sample for which each pixel voted. The background color and outlier label are also drawn. For M_DRAW_LABEL_PIXEL_IMAGE, you must use McolSetMethod() with M_MIN_DIST_VOTE, and you must set the ColorResultorDestImageId parameter to the identifier of a grayscale image buffer. To inquire the required buffer depth, use McolInquire() with M_LABEL_PIXEL_IMAGE_SIZE_BIT.</p> <p>(summarize)</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolPreprocess

Synopsis

Preprocess a color matching context.

Syntax

```
void McolPreprocess(
    MIL_ID ColorContextId,
    MIL_INT ControlFlag
)
```

Description

This function prepares the specified color matching context for a color matching operation. It calculates the necessary data for each color-sample contained within the color matching context and sets internal processing settings to optimize future calculations for speed and robustness. You must preprocess all color matching contexts before the first call to [McolMatch\(\)](#).

Changes to control settings, the color matching context, or one of its color-samples often require you to preprocess the context again. To inquire if this is the case, use [McolInquire\(\)](#) with [M_PREPROCESSED](#).

When you save the color matching context, the preprocessing changes are not saved. Upon restoration ([McolRestore\(\)](#)), you must preprocess the context.

Parameters

- ColorContextId
- Specifies the color matching context to preprocess. You must first allocate the color matching context on the system using [McolAlloc\(\)](#).
- ControlFlag
- Specifies the function's control flag. This parameter must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolProject

Synopsis

Perform a color projection.

Syntax

```
void McolProject(
    MIL_ID SrcId,
    MIL_ID ColorsId,
    MIL_ID DestId,
    MIL_ID DestMaskId,
    MIL_INT Operation,
    MIL_INT ControlFlag,
    MIL_INT *StatusPtr
)
```

Description

This function performs a color projection (transformation) of the source image, or returns a color projection matrix, which you can then apply to transform other images, using [MimConvert\(\)](#). **McolProject()** allows you to separate a selected color from a rejected one, or to convert color images to grayscale with minimal loss of information. You can also use this function to perform analytical tasks, such as calculating an image's covariance matrix, or the principal components of an image's color information.

Results are returned to this function's parameters.

Color projection is an operation that requires data defined in a Cartesian coordinate system (RGB, CIELAB). A projection with HSL data would provide incoherent results.

Parameters

SrcId

Specifies the identifier of the source image buffer on which to perform the projection transformation. This must be a 3-band 8- or 16-bit unsigned image. Source images are interpreted per band, independently of the MIL buffer format.

When separating colors ([M_COLOR_SEPARATION](#)), you can also set this parameter to **M_NULL**. In this case, the destination buffer will contain the resulting projection matrix, rather than the transformed image. When [SrcId](#) is set to **M_NULL**, you must set the [ColorsId](#) and [DestId](#) parameters to an array.

ColorsId

Specifies the identifier of an image buffer or array buffer that controls how the source pixels are interpreted.

See the [Parameter associations](#) section for possible values.

DestId

Specifies the identifier of the destination buffer in which to write the color projection (transformation) results. The destination buffer can be either an image or an array.

Note that when the result is a matrix, you must set this parameter to an array buffer, allocated using [MbufAlloc....\(\)](#) with **M_ARRAY**.

See the [Parameter associations](#) section for possible values.

DestMaskId

Specifies the identifier of the destination mask image. Mask images must be monochrome. You can only specify a mask if the destination buffer is an image. If you do not require a mask, set this parameter to **M_NULL**.

The destination mask determines where to write the result of the projection operation. Only unmasked pixels (non-zero) will be written to, in the destination image. The destination mask does not affect the calculated matrix.

Operation

Specifies the color projection operation.

See the [Parameter associations](#) section for possible values.

ControlFlag

Specifies the dynamic range on which the resulting projection is mapped. This is only applicable for [M_PRINCIPAL_COMPONENT_PROJECTION](#).

See the [Parameter associations](#) section for possible values.

StatusPtr

Specifies the address of the variable in which to write the status of the color projection operation. If you do not require this information, set this parameter to **M_NULL**.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ColorsId**, **DestId**, **Operation**, **ControlFlag**, and **StatusPtr** parameters are described in the table below.

- [For implementing the color projection](#)

Note that you should set any unused parameters to **M_DEFAULT**.

For implementing the color projection									
Operation	Description								
ColorsId									
DestId									
ControlFlag									
StatusPtr									
M_COLOR_SEPARATION	Specifies the removal of a color in the source image. In this case, you must identify the background, selected, and rejected colors. (summarize)								
ColorsId	Specifies the identifier of the buffer containing the background, selected, and rejected pixels colors with which to perform the color separation. If SrcId is set to M_NULL , you must set ColorsId to an array. In this case, the destination buffer will contain the resulting matrix for the specified source image, rather than the transformed source image. (summarize)								
Array buffer identifier	Specifies the identifier of a 3 x 3 MIL array buffer, where each row defines, in the following order, the background, the selected, and the rejected colors, to use to compute the projection matrix.								
Image buffer identifier	<p>Specifies the identifier of an image buffer that identifies the corresponding source image pixels (SrcId) to use to compute the projection matrix. The pixels identified with this image buffer (ColorsId) must be the required background, selected, and rejected pixels (colors). This buffer must be a 1-band 8-bit unsigned image buffer.</p> <p>To identify the corresponding source image pixels, you must set the appropriate pixels in this buffer (ColorsId) to one of the label values below.</p> <table> <tr> <th>Label value</th><th>Definition</th></tr> <tr> <td>M_BACKGROUND_LABEL</td><td>Identifies a background color pixel.</td></tr> <tr> <td>M_REJECTED_LABEL</td><td>Identifies a rejected color pixel.</td></tr> <tr> <td>M_SELECTED_LABEL</td><td>Identifies a selected color pixel.</td></tr> </table> <p>You must assign at least one pixel to each of these values. Pixel values (in the source) that do not correspond to any of these values (ColorsId) are ignored. If this buffer (ColorsId) is smaller than the source, outer source pixels are ignored.</p>	Label value	Definition	M_BACKGROUND_LABEL	Identifies a background color pixel.	M_REJECTED_LABEL	Identifies a rejected color pixel.	M_SELECTED_LABEL	Identifies a selected color pixel.
Label value	Definition								
M_BACKGROUND_LABEL	Identifies a background color pixel.								
M_REJECTED_LABEL	Identifies a rejected color pixel.								
M_SELECTED_LABEL	Identifies a selected color pixel.								

	(summarize)																		
<input type="checkbox"/> DestId	<p>Specifies the identifier of the destination buffer in which to write the color separation results.</p> <p>Note that if SrcId is set to M_NULL, you must set DestId to an array.</p> <p>(summarize)</p>																		
<input type="checkbox"/> Array buffer identifier	<p>Specifies the identifier of a 4 x 3 MIL array buffer, of type <i>float</i>, to store the resulting projection matrix.</p> <p>You can use this matrix with MimConvert() to separate colors in any 3-band color image. Be certain that the color distribution of all source images that use this matrix is similar, otherwise unpredictable results can occur.</p> <p>(summarize)</p>																		
<input type="checkbox"/> Image buffer identifier	Specifies the identifier of a 3-band color image buffer, to store the transformed (color separated) source image.																		
<input type="checkbox"/> StatusPtr	<p>Specifies the address of the variable in which to write the resulting status of the M_COLOR_SEPARATION operation. The possible values for the status are:</p> <table> <tr> <th>Status</th><th>Definition</th></tr> <tr> <td>M_3_COLORS_COLLINEAR</td><td>Specifies that the selected, rejected, and background colors are collinear.¹</td></tr> <tr> <td>M_NO_BACKGROUND_DEFINED</td><td>Specifies that no background color was defined with the ColorsId parameter.¹</td></tr> <tr> <td>M_NO_REJECTED_DEFINED</td><td>Specifies that no rejected color was defined with the ColorsId parameter.¹</td></tr> <tr> <td>M_REJECTED_EQUAL_BACKGROUND</td><td>Specifies that the rejected color equals the background color.¹</td></tr> <tr> <td>M_REJECTED_EQUAL_SELECTED</td><td>Specifies that the rejected color equals the selected color.¹</td></tr> <tr> <td>M_NO_SELECTED_DEFINED</td><td>Specifies that no selected color was defined with the ColorsId parameter.¹</td></tr> <tr> <td>M_SELECTED_EQUAL_BACKGROUND</td><td>Specifies that the selected color equals the background color.</td></tr> <tr> <td>M_SUCCESS</td><td>Specifies a successful operation.</td></tr> </table> <p>¹In this case, the projection is not well defined and processing cannot occur. If the DestId parameter specifies an image, it is filled with a copy of the source image (SrcId). If the DestId parameter specifies an array, it is filled with the identity matrix such that, using this array with MimConvert() copies the source image into the destination image.</p> <p>(summarize)</p>	Status	Definition	M_3_COLORS_COLLINEAR	Specifies that the selected, rejected, and background colors are collinear. ¹	M_NO_BACKGROUND_DEFINED	Specifies that no background color was defined with the ColorsId parameter. ¹	M_NO_REJECTED_DEFINED	Specifies that no rejected color was defined with the ColorsId parameter. ¹	M_REJECTED_EQUAL_BACKGROUND	Specifies that the rejected color equals the background color. ¹	M_REJECTED_EQUAL_SELECTED	Specifies that the rejected color equals the selected color. ¹	M_NO_SELECTED_DEFINED	Specifies that no selected color was defined with the ColorsId parameter. ¹	M_SELECTED_EQUAL_BACKGROUND	Specifies that the selected color equals the background color.	M_SUCCESS	Specifies a successful operation.
Status	Definition																		
M_3_COLORS_COLLINEAR	Specifies that the selected, rejected, and background colors are collinear. ¹																		
M_NO_BACKGROUND_DEFINED	Specifies that no background color was defined with the ColorsId parameter. ¹																		
M_NO_REJECTED_DEFINED	Specifies that no rejected color was defined with the ColorsId parameter. ¹																		
M_REJECTED_EQUAL_BACKGROUND	Specifies that the rejected color equals the background color. ¹																		
M_REJECTED_EQUAL_SELECTED	Specifies that the rejected color equals the selected color. ¹																		
M_NO_SELECTED_DEFINED	Specifies that no selected color was defined with the ColorsId parameter. ¹																		
M_SELECTED_EQUAL_BACKGROUND	Specifies that the selected color equals the background color.																		
M_SUCCESS	Specifies a successful operation.																		
<input type="checkbox"/> M_COVARIANCE	<p>Specifies to compute the covariance matrix of the source image's color information. This is a measure of the color variation.</p> <p>(summarize)</p>																		
<input type="checkbox"/> ColorsId	<p>Specifies the identifier of the buffer that controls how the source pixels are interpreted to perform the covariance calculation.</p> <p>(summarize)</p>																		
<input type="checkbox"/> Image buffer identifier	<p>Specifies the identifier of an image buffer that identifies the corresponding source image pixels (SrcId) to use to compute the covariance matrix. This buffer must be a 1-band 8-bit unsigned image buffer. You must set the appropriate pixels in this buffer (ColorsId) to M_SOURCE_LABEL. You must assign at least one pixel to this value. Pixel values that do not correspond to M_SOURCE_LABEL are ignored.</p> <p>If this buffer (ColorsId) is smaller than the source, outer source pixels are ignored.</p> <p>(summarize)</p>																		
<input type="checkbox"/> M_NULL	Specifies to use the entire source image to compute the covariance matrix.																		
<input type="checkbox"/> DestId	<p>Specifies the identifier of the destination buffer in which to write the results of the covariance calculation.</p> <p>(summarize)</p>																		
<input type="checkbox"/> Array buffer identifier	<p>Specifies the identifier of a 3 x 3 or 4 x 3 MIL array buffer, of type <i>float</i>, to store the resulting covariance matrix.</p> <p>For 3 x 3 arrays, the following covariance matrix elements are returned:</p> $\begin{bmatrix} \text{var}(\text{Band0}) & \text{cov}(\text{Band0}, \text{Band1}) & \text{cov}(\text{Band0}, \text{Band2}) \\ \text{cov}(\text{Band1}, \text{Band0}) & \text{var}(\text{Band1}) & \text{cov}(\text{Band1}, \text{Band2}) \\ \text{cov}(\text{Band2}, \text{Band0}) & \text{cov}(\text{Band2}, \text{Band1}) & \text{var}(\text{Band2}) \end{bmatrix}$ <p>var = variance, cov = covariance; covariance matrix is symmetrical, so: $\text{CovMatrix}(i, j) = \text{CovMatrix}(j, i)$</p>																		

	<div>For 4 x 3 arrays, the mean color is also returned, in the fourth column:</div> <div><table><tr><td><i>var (Band0)</i></td><td><i>cov (Band0, Band1)</i></td><td><i>cov (Band0, Band2)</i></td><td><i>Band0MeanValue</i></td></tr><tr><td><i>cov (Band1, Band0)</i></td><td><i>var (Band1)</i></td><td><i>cov (Band1, Band2)</i></td><td><i>Band1MeanValue</i></td></tr><tr><td><i>cov (Band2, Band0)</i></td><td><i>cov (Band2, Band1)</i></td><td><i>var (Band2)</i></td><td><i>Band2MeanValue</i></td></tr></table></div> <div>(summarize)</div>	<i>var (Band0)</i>	<i>cov (Band0, Band1)</i>	<i>cov (Band0, Band2)</i>	<i>Band0MeanValue</i>	<i>cov (Band1, Band0)</i>	<i>var (Band1)</i>	<i>cov (Band1, Band2)</i>	<i>Band1MeanValue</i>	<i>cov (Band2, Band0)</i>	<i>cov (Band2, Band1)</i>	<i>var (Band2)</i>	<i>Band2MeanValue</i>
<i>var (Band0)</i>	<i>cov (Band0, Band1)</i>	<i>cov (Band0, Band2)</i>	<i>Band0MeanValue</i>										
<i>cov (Band1, Band0)</i>	<i>var (Band1)</i>	<i>cov (Band1, Band2)</i>	<i>Band1MeanValue</i>										
<i>cov (Band2, Band0)</i>	<i>cov (Band2, Band1)</i>	<i>var (Band2)</i>	<i>Band2MeanValue</i>										
<div><div>StatusPtr</div></div>	<div>Specifies the address of the variable in which to write the resulting status of the M_COVARIANCE operation. The possible values for the status are:</div> <div><table><tr><th>Status</th><th>Definition</th></tr><tr><td>M_NO_SOURCE_DEFINED</td><td>Specifies that the ColorsId parameter cannot identify a corresponding source pixel. In this case, the resulting matrix (DestId) is filled with zeros.</td></tr><tr><td>M_SUCCESS</td><td>Specifies a successful operation.</td></tr></table></div> <div>(summarize)</div>	Status	Definition	M_NO_SOURCE_DEFINED	Specifies that the ColorsId parameter cannot identify a corresponding source pixel. In this case, the resulting matrix (DestId) is filled with zeros.	M_SUCCESS	Specifies a successful operation.						
Status	Definition												
M_NO_SOURCE_DEFINED	Specifies that the ColorsId parameter cannot identify a corresponding source pixel. In this case, the resulting matrix (DestId) is filled with zeros.												
M_SUCCESS	Specifies a successful operation.												
<div><div>M_PRINCIPAL_COMPONENT_PROJECTION</div></div>	<div>Specifies a principal component projection of the source image. This converts each color pixel to a grayscale value with a point-to-point projection of the source image, onto its principal component, using a grayscale LUT. The projection is a mathematically optimal color-to-grayscale transformation, providing minimal loss of information.</div> <div>(summarize)</div>												
<div><div><div>ColorsId</div><div><div>Image buffer identifier</div></div></div></div>	<div>Specifies the identifier of the buffer that controls how the source pixels are interpreted to perform the principal component projection.</div> <div>(summarize)</div> <div>Specifies the identifier of an image buffer that identifies the corresponding source image pixels (SrcId) to use for the principal component projection. This buffer must be a 1-band 8-bit unsigned image buffer.</div> <div>To identify the corresponding source image pixels, you must set the appropriate pixels in this buffer (ColorsId) to one of the label values below.</div> <div><table><tr><th>Label value</th><th>Definition</th></tr><tr><td>M_BRIGHT_LABEL</td><td>Identifies a source pixel that represents a color to project on the brightest side of the grayscale palette.</td></tr><tr><td>M_DARK_LABEL</td><td>Identifies a source pixel that represents a color to project on the darkest side of the grayscale palette.</td></tr><tr><td>M_SOURCE_LABEL</td><td>Identifies a source pixel.</td></tr></table></div> <div>Pixel values (in the source) that do not correspond to any of these values (ColorsId) are ignored. You must assign at least one pixel to M_SOURCE_LABEL; M_BRIGHT_LABEL and M_DARK_LABEL are optional. If this buffer (ColorsId) is smaller than the source, outer source pixels are ignored.</div> <div>(summarize)</div>	Label value	Definition	M_BRIGHT_LABEL	Identifies a source pixel that represents a color to project on the brightest side of the grayscale palette.	M_DARK_LABEL	Identifies a source pixel that represents a color to project on the darkest side of the grayscale palette.	M_SOURCE_LABEL	Identifies a source pixel.				
Label value	Definition												
M_BRIGHT_LABEL	Identifies a source pixel that represents a color to project on the brightest side of the grayscale palette.												
M_DARK_LABEL	Identifies a source pixel that represents a color to project on the darkest side of the grayscale palette.												
M_SOURCE_LABEL	Identifies a source pixel.												
<div><div><div>M_NULL</div></div></div>	<div>Specifies to use the entire source image to perform the principal component projection.</div>												
<div><div><div>DestId</div></div></div>	<div>Specifies the identifier of the destination buffer in which to write the principal component results.</div> <div>(summarize)</div>												
<div><div><div><div>Array buffer identifier</div></div></div></div>	<div>Specifies the identifier of a 4 x 1 MIL array buffer, of type <i>float</i>, to store the resulting projection matrix.</div> <div>You can use this matrix with MimConvert() to perform a color-to-grayscale transformation of any 3-band color image. Be certain that the color distribution of all source images that use this matrix is similar, otherwise unpredictable results can occur.</div> <div>(summarize)</div>												
<div><div><div><div>Image buffer identifier</div></div></div></div>	<div>Specifies the identifier of a 1-band image buffer, to store the transformed source image.</div>												
<div><div><div>ControlFlag</div></div></div>	<div>Specifies the dynamic range on which the resulting projection is mapped.</div> <div>(summarize)</div>												
<div><div><div><div>M_DEFAULT</div></div></div></div>	<div>Specifies that the projection result is mapped according to the dynamic range of all the pixels in the source image (SrcId).</div>												
<div><div><div><div>M_MASK_CONTRAST_ENHANCEMENT</div></div></div></div>	<div>Specifies that the projection result is mapped according to the dynamic range of the pixels identified with the ColorsId parameter. Be careful when doing this, since the source pixels outside the dynamic range (ColorsId) might saturate and cause unpredictable results.</div> <div>(summarize)</div>												
<div><div><div>StatusPtr</div></div></div>	<div>Specifies the address of the variable in which to write the resulting status of the M_PRINCIPAL_COMPONENT_PROJECTION operation. The possible values for the status are:</div> <div><table><tr><td></td><td></td></tr></table></div>												

	<table> <tr> <th>Status</th><th>Definition</th></tr> <tr> <td>M_NO_SOURCE_DEFINED</td><td>Specifies that the ColorsId parameter cannot identify a corresponding source pixel.¹</td></tr> <tr> <td>M_UNSTABLE_POLARITY</td><td>Specifies that the polarity of the projection is unstable. In this case, it is difficult to determine which pixels should be projected to the brightest/darkest side of the grayscale palette.</td></tr> <tr> <td>M_SUCCESS</td><td>Specifies a successful operation.</td></tr> </table> <p>¹In this case, the projection is not well defined and processing cannot occur. If the DestId parameter specifies an image, it is filled with a copy of the source image (SrcId). If the DestId parameter specifies an array, it is filled with the identity matrix such that, using this array with MimConvert() copies the source image into the destination image.</p> <p>(summarize)</p>	Status	Definition	M_NO_SOURCE_DEFINED	Specifies that the ColorsId parameter cannot identify a corresponding source pixel. ¹	M_UNSTABLE_POLARITY	Specifies that the polarity of the projection is unstable. In this case, it is difficult to determine which pixels should be projected to the brightest/darkest side of the grayscale palette.	M_SUCCESS	Specifies a successful operation.						
Status	Definition														
M_NO_SOURCE_DEFINED	Specifies that the ColorsId parameter cannot identify a corresponding source pixel. ¹														
M_UNSTABLE_POLARITY	Specifies that the polarity of the projection is unstable. In this case, it is difficult to determine which pixels should be projected to the brightest/darkest side of the grayscale palette.														
M_SUCCESS	Specifies a successful operation.														
<input type="checkbox"/> M_PRINCIPAL_COMPONENTS	<p>Specifies to compute the principal components of the source image's color information. The principal components of an image are the three eigenvectors of its covariance matrix. M_PRINCIPAL_COMPONENTS also allows you to retrieve the source image's average color value.</p> <p>(summarize)</p>														
<input type="checkbox"/> ColorsId	<p>Specifies the identifier of the buffer that controls how the source pixels are interpreted to perform the principal component calculation.</p> <p>(summarize)</p>														
<input type="checkbox"/> Image buffer identifier	<p>Specifies the identifier of an image buffer that identifies the corresponding source image pixels (SrcId) to use to compute the principal components. This buffer must be a 1-band 8-bit unsigned image buffer. You must set the appropriate pixels in this buffer (ColorsId) to M_SOURCE_LABEL. You must assign at least one pixel to this value. Pixel values that do not correspond to M_SOURCE_LABEL are ignored.</p> <p>If this buffer (ColorsId) is smaller than the source, outer source pixels are ignored.</p> <p>(summarize)</p>														
<input type="checkbox"/> M_NULL	<p>Specifies to use the entire source image to compute the principal components.</p>														
<input type="checkbox"/> DestId	<p>Specifies the identifier of the destination buffer in which to write the principal components.</p> <p>(summarize)</p>														
<input type="checkbox"/> Array buffer identifier	<p>Specifies the identifier of a 3 x 3, 4 x 3, or 5 x 3 MIL array buffer, of type <i>float</i>, to store the resulting principal components (eigenvectors). The values returned depend on the array's dimensions:</p> <ul style="list-style-type: none"> • 3 x 3. The 3 eigenvectors are returned columnwise. • 4 x 3. In addition to the 3 x 3 array results, the 3 eigenvalues are also returned in the fourth column. • 5 x 3. In addition to the 4 x 3 array results, the 3 average color values are returned in the fifth column. <p>Eigenvectors and their eigenvalues are returned in decreasing order of importance.</p> <p>(summarize)</p>														
<input type="checkbox"/> StatusPtr	<p>Specifies the address of the variable in which to write the resulting status of the M_PRINCIPAL_COMPONENTS operation. The possible values for the status are:</p> <table> <tr> <th>Status</th><th>Definition</th></tr> <tr> <td>M_NO_SOURCE_DEFINED</td><td>Specifies that the ColorsId parameter cannot identify a corresponding source pixel.¹</td></tr> <tr> <td>M_UNSTABLE_POLARITY</td><td>Specifies that the direction of the principal component is unstable. In this case, it is difficult to align the principal component, in a consistent and repeatable way, with the gray axis (from black to white).</td></tr> <tr> <td>M_UNSTABLE_PRINCIPAL_COMPONENT_2</td><td>Specifies that the direction of the third principal component (least important) is unstable.²</td></tr> <tr> <td>M_UNSTABLE_PRINCIPAL_COMPONENTS_12</td><td>Specifies that the direction of the second and third principal components is unstable.²</td></tr> <tr> <td>M_UNSTABLE_PRINCIPAL_COMPONENTS_012</td><td>Specifies that all principal component directions are unstable.¹</td></tr> <tr> <td>M_SUCCESS</td><td>Specifies a successful operation.</td></tr> </table> <p>¹In this case, the resulting matrix (DestId) is filled with zeros.</p> <p>²This is typically due to small variances along the color component's direction.</p>	Status	Definition	M_NO_SOURCE_DEFINED	Specifies that the ColorsId parameter cannot identify a corresponding source pixel. ¹	M_UNSTABLE_POLARITY	Specifies that the direction of the principal component is unstable. In this case, it is difficult to align the principal component, in a consistent and repeatable way, with the gray axis (from black to white).	M_UNSTABLE_PRINCIPAL_COMPONENT_2	Specifies that the direction of the third principal component (least important) is unstable. ²	M_UNSTABLE_PRINCIPAL_COMPONENTS_12	Specifies that the direction of the second and third principal components is unstable. ²	M_UNSTABLE_PRINCIPAL_COMPONENTS_012	Specifies that all principal component directions are unstable. ¹	M_SUCCESS	Specifies a successful operation.
Status	Definition														
M_NO_SOURCE_DEFINED	Specifies that the ColorsId parameter cannot identify a corresponding source pixel. ¹														
M_UNSTABLE_POLARITY	Specifies that the direction of the principal component is unstable. In this case, it is difficult to align the principal component, in a consistent and repeatable way, with the gray axis (from black to white).														
M_UNSTABLE_PRINCIPAL_COMPONENT_2	Specifies that the direction of the third principal component (least important) is unstable. ²														
M_UNSTABLE_PRINCIPAL_COMPONENTS_12	Specifies that the direction of the second and third principal components is unstable. ²														
M_UNSTABLE_PRINCIPAL_COMPONENTS_012	Specifies that all principal component directions are unstable. ¹														
M_SUCCESS	Specifies a successful operation.														

	(summarize)
--	-----------------------------

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolRestore

Synopsis

Restore a color matching context from disk.

Syntax

```
MIL_ID McolRestore(
    MIL_CONST_TEXT_PTR Filename,
    MIL_ID SystemId,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr
)
```

Description

This function restores a color matching context that was previously saved to a file, using [McolSave\(\)](#) or [McolStream\(\)](#). This function restores all of the color matching context's settings (and color-samples) that were in effect when the color matching context was saved. A restored color matching context is not preprocessed; therefore, you must call [McolPreprocess\(\)](#) before performing any calculations with [McolMatch\(\)](#).

Parameters

Filename

Specifies the name and path of the file from which to restore the color matching context. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

● For specifying the file name and path		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	Specifies the drive, directory, and name of the file; for example, " C:\mydirectory\myfile ". To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with " remote:/// "; for example, " remote:///C:\mydirectory\myfile ". (summarize)	
	Parameters	
	FileName	Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.	

SystemId

Specifies the system on which to restore the color matching context. This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the color matching context identifier. If allocation fails, **M_NULL** is written as the identifier. Since **McolRestore()** also returns the color matching context identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the color matching context identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolSave

Synopsis

Save a color matching context to a file.

Syntax

```
void McolSave(  
    MIL_CONST_TEXT_PTR FileName,  
    MIL_ID ContextId,  
    MIL_INT ControlFlag  
)
```

Description

This function saves to disk all the information about the previously allocated color matching context, and the color-samples contained therein. However, preprocessing information is not saved. You can reload the saved information, using [McolRestore\(\)](#) or [McolStream\(\)](#).

Parameters

FileName

Specifies the name and path of the file in which to save the color matching context. It is recommended that you use the MCOL file extension for easier use with other Matrox Imaging software products. The function internally handles the opening and closing of this file. If this file already exists, it is overwritten.

This parameter can be set to one of the following values:

For specifying the file name and path		
Value	Description	
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	Specifies the drive, directory, and name of the file, for example: " C:\mydirectory\myfile ". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with " remote:/// ", for example: " remote:///C:\mydirectory\myfile ". (summarize)	
	Parameters	
	FileName	Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.	

ContextId

Specifies the identifier of the color matching context to save.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolSetMethod

Synopsis

Set the strategy with which to perform color matching.

Syntax

```
void McolSetMethod(
    MIL_ID ContextId,
    MIL_INT OperationMode,
    MIL_INT DistanceType,
    MIL_INT PreconvertMode,
    MIL_INT ControlFlag
)
```

Description

This function sets the strategy with which to match color samples, when calling [McolMatch\(\)](#). The strategy is based on the color matching's operation mode and the type of color distance calculated. If required, you can also internally convert your color to a different color space before the matching operation. By manipulating these settings, you can affect the resulting color-samples that are matched by the target areas.

Parameters

ContextId

Specifies the color matching context with which to apply the color matching strategy. The color matching context must have been previously allocated on the required system using [McolAlloc\(\)](#).

OperationMode

Specifies the operation mode to use when matching colors. Set this parameter to one of the following values.

● For setting the mode of the color matching operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_STAT_MIN_DIST .
<input type="checkbox"/> M_MIN_DIST_VOTE	<p>Specifies a minimum distance operation, based on pixel voting.</p> <p>Color statistics are calculated (typically the mean) for each color-sample defined in the context, and the color distance between each pixel in each target area and each color-sample is determined. Each target pixel then votes for the closest color-sample, which is also within the distance tolerance (McolControl() with M_DISTANCE_TOLERANCE).</p> <p>The number of votes that a color-sample accumulates determines its score. The color-sample with the highest score above the acceptance (McolControl() with M_ACCEPTANCE) is the target area's best-matched color-sample.</p> <p>(summarize)</p>
<input type="checkbox"/> M_STAT_MIN_DIST	<p>Specifies a minimum distance operation, based on pixel statistics.</p> <p>Color statistics are calculated (typically the mean) for each target area and for each color-sample defined in the context, and the color distance between each target area and each color-sample is determined.</p> <p>The resulting distances determine the score of the color-samples. If a distance is not within a color-sample's distance tolerance, this color-sample's score is 0%. The color-sample with the highest score above the acceptance (McolControl() with M_ACCEPTANCE) is the target area's best-matched color-sample.</p> <p>(summarize)</p>

DistanceType

Specifies the type of distance calculation to use when matching colors. The distance is the difference in color between the color-sample and the target area. Except for the [M_MAHALANOBIS](#), which uses the covariance of the color-sample, the mean of the colors are used to calculate the distance.

Set this parameter to one of the following values.

● For setting the distance type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default. For RGB and CIELAB color spaces, the default is the same as M_EUCLIDEAN . For an HSL color space, the default is the same as M_MANHATTAN . (summarize)
<input type="checkbox"/> M_DELTA_E	Specifies to use a Delta-E color distance, as defined by the International Commission on Illumination (CIE). A Delta-E color distance is the same as a Euclidean color distance (M_EUCLIDEAN), but specialized for a CIELAB color space or when using a CIELAB conversion mode (PreconvertMode). You cannot calculate an M_DELTA_E color distance in an RGB or HSL color space. (summarize)
<input type="checkbox"/> M_EUCLIDEAN	Specifies to use a Euclidean color distance. A Euclidean distance is the square root of the sum of the squared differences between the color of the color-sample and the color of the target area. You cannot calculate an M_EUCLIDEAN color distance in an HSL color space. (summarize)
<input type="checkbox"/> M_MAHALANOBIS	Specifies to use a Mahalanobis color distance. A Mahalanobis distance is calculated between the color of the target area and the covariance of the color-sample. This distance, between a color and a distribution of colors, is similar to a Euclidean distance between the mean of the two colors, but weighted by the inverse of the covariance of the distribution. This implies that the more a color distribution varies in a direction within the color space, the less important is the distance in that direction. You cannot calculate an M_MAHALANOBIS color distance in an HSL color space. (summarize)
<input type="checkbox"/> M_MANHATTAN	Specifies to use a Manhattan color distance. A Manhattan distance is the sum of the absolute value of the differences between the color of the color-sample and the color of the target area. For an HSL color space, the distance between angular coordinates is equal to the smallest angular difference, rather than the absolute value of the difference. (summarize)

PreconvertMode

Specifies the mode with which to internally convert colors before the matching operation. Set this parameter to one of the following values.

● For setting the internal conversion mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_NONE .
<input type="checkbox"/> M_CIELAB	Specifies to internally convert colors to CIELAB before the matching operation. You can only specify M_CIELAB if you are working in an RGB color space (McolAlloc()). (summarize)
<input type="checkbox"/> M_NONE	Specifies no internal conversion.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

McolStream

Synopsis

Load, restore, or save a MIL color matching context from/to a file or memory stream.

Syntax

```
void McolStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *ObjectIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load or restore a MIL color matching context from a file or memory stream. All of the color matching context's settings (and color-samples) that were in effect when the color matching context was saved will be restored. A loaded or restored color matching context is not preprocessed, therefore you must call [McolPreprocess\(\)](#) before performing a match with [McolMatch\(\)](#).

This function can also save a color matching context to a file or memory stream. All information about the previously allocated color matching context (and the color-samples therein) is saved. However, preprocessing information is not saved.

To inquire the number of bytes necessary to save a color matching context, you should call [McolStream\(\)](#) first with the **Operation** parameter set to [M_INQUIRE_SIZE_BYTE](#).

The content saved to a memory stream is equivalent to the content saved to a file. In addition, any file saved with [McolSave\(\)](#) is equivalent to a file saved using [McolStream\(\)](#).

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Parameters

MemPtrOrFileName

Specifies the file or memory stream.

- For specifying the name and path of a file, memory stream, or opening an interactive dialog box

Value	Description		
<div> <div> <div>MIL_TEXT(</div> <div>MIL_TEXT_PTR FileName</div> <div>)</div> </div> </div>	<div> <p>Specifies the drive, directory, and name of the file, for example: " C:\mydirectory\myfile ", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving a color matching context to a file, use the MCOL file extension. The function internally handles the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p> <p>To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with " remote:/// ", for example: " remote:///C:\mydirectory\myfile ".</p> <p>(summarize)</p> <table> <tr> <th>Parameters</th> </tr> <tr> <td> <div> <div>FileName</div> <div>Specifies the drive, directory, and name of the file.</div> </div> </td> </tr> </table> </div>	Parameters	<div> <div>FileName</div> <div>Specifies the drive, directory, and name of the file.</div> </div>
Parameters			
<div> <div>FileName</div> <div>Specifies the drive, directory, and name of the file.</div> </div>			
<div> <div>M_INTERACTIVE</div> </div>	<div> <p>[This is only applicable to Windows]</p> <p>Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE.</p> </div>		

<input type="checkbox"/> M_NULL	Specifies to ignore this parameter (MemPtrOrFileName). This parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (McolStream()) first with the Operation parameter set to M_INQUIRE_SIZE_BYTE . (summarize)

SystemId

Specifies the system on which to restore the color matching context. For **M_INQUIRE_SIZE_BYTE**, **M_LOAD**, and **M_SAVE**, **SystemId** is ignored and should be set to **M_NULL**.

For an **M_RESTORE** operation, set this parameter to one of the following values:

● For M_RESTORE	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform. This parameter must be set to one of the following values:

● For specifying the operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save a color matching context. This operation is not supported when you set the StreamType parameter to M_FILE . (summarize)
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated color matching context.
<input type="checkbox"/> M_RESTORE	Restores a color matching context from a file or memory stream and assigns it an identifier.
<input type="checkbox"/> M_SAVE	Saves a color matching context to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the color matching context. This parameter must be set to one of the following values:

● For the type of stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream.
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)

Version

Specifies the MIL version of the color matching context. This parameter must be set to one of the following values.

● For specifying the MIL version	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, M_DEFAULT sets the version to the current version of MIL. For an M_LOAD or M_RESTORE operation, M_DEFAULT reads the file or stream for the version information. This is the only possible setting for these operations.

	(summarize)
<input type="checkbox"/> M_PROC_VERSION_90_PP1	Sets the version to MIL 9.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ObjectIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the color matching context.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, **ObjectIdPtr** specifies the address of the variable from which to read the color matching context identifier.

For an [M_LOAD](#) operation, **ObjectIdPtr** specifies the address of the variable from which to read the identifier of the color matching context where the file or memory stream content will be loaded.

For an [M_RESTORE](#) operation, **ObjectIdPtr** specifies the address of the variable in which to return the identifier of the restored color matching context. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the color matching context, in bytes.

If the size is not required, you can set this parameter to **M_NULL**.

The size of the color matching context varies depending on the MIL version that you specify.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milcol.lib.
DLL	Requires mil.dll; milcol.dll.

Mdig functions

Synopsis

The functions prefixed with Mdig make up the Digitizer module. The Digitizer Control module allows you to manipulate and control the digitizer section of an imaging board. It also allows you to set the state required for either simple or complex acquisition and allows acquisition functions to be sent to the imaging board.

Functions

- MdigAlloc
- MdigChannel
- MdigControl
- MdigControlFeature
- MdigFocus
- MdigFree
- MdigGetHookInfo
- MdigGrab
- MdigGrabContinuous
- MdigGrabWait
- MdigHalt
- MdigHookFunction
- MdigInquire
- MdigInquireFeature
- MdigLut
- MdigProcess
- MdigReference

MdigAlloc

Synopsis

Allocate a digitizer.

Syntax

```
MIL_ID MdigAlloc(
    MIL_ID SystemId,
    MIL_INT DigNum,
    MIL_CONST_TEXT_PTR DataFormat,
    MIL_INT InitFlag,
    MIL_ID *DigIdPtr
)
```

Description

This function allocates a digitizer on the specified system so that it can be used by subsequent MIL digitizer functions. A digitizer on the target system must be allocated to acquire data from a camera. Its device number specifies the first acquisition path of the digitizer. Its digitizer configuration format (DCF) establishes the number of acquisition paths used. One digitizer might use several acquisition paths depending on the input format of the camera. Use [MdigGrab\(\)](#), [MdigGrabContinuous\(\)](#), or [MdigProcess\(\)](#) to grab.

Multiple digitizers can be allocated on the same system. If the system supports the possibility of independent acquisition paths, the digitizers' device number together with their DCF establishes if they represent dependent or independent acquisition path(s) on the system. If independent, grabs from these digitizer occur simultaneously. If dependent, grabs from these digitizer occur consecutively.

If multiple cameras of the same type are connected to the same acquisition path(s), instead of allocating multiple digitizers, you can allocate a single digitizer in the required format and then use [MdigChannel\(\)](#) to switch between the cameras. [MdigChannel\(\)](#) selects the active input channel of the digitizer. The switch occurs upon the next call to [MdigGrab\(\)](#), [MdigGrabContinuous\(\)](#), or [MdigProcess\(\)](#) with the digitizer. The default input channel is determined by the selected DCF (generally, [M_CH0](#)).

It is faster to allocate all the required digitizers first, perform the required grabs, and then free all the digitizers rather than allocate, grab, and then free each digitizer in succession.

Upon execution of this function, MIL ensures that the digitizer is present before allocating it and generates an error if it is not. Note that the corresponding hardware won't necessarily be programmed at the time of allocation.

When you have completely finished using a digitizer, you should free it, using [MdigFree\(\)](#).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
You can allocate a digitizer even if the frame grabber is physically connected to another system, as long as the systems are both part of the same cluster.

Parameters

SystemId

Specifies the identifier of the system on which to allocate the digitizer. This parameter must be given a valid system identifier.

DigNum

Specifies the device number of the digitizer that is required. The device number identifies the first acquisition path with which to associate a timing control unit (for example, a programmable synchronization generator (PSG) or a video capture controller) on your frame grabber.

This parameter can be set to one of the following:

● For specifying the device number	
<input checked="" type="checkbox"/> Value	Description
	vio (w)
	solios gige (v)
	solios ecl/xcl
	solios ea/xa i
	odyssey ed/x
	odyssey ecl/z
	odyssey ea/x
	nexis (p)
	morphis qxt i
	morphis (n)
	met-II /std (i
	met-II /mc (l
	met-II /dig (i
	met-II /c (f)
	iris (l)
	ieee 1394 iid
	helios ed/xcl
	helios ea/xa
	gpu processii
	gige vision (c
	cronosplus (t
	corona-II (a)

<input type="checkbox"/> M_DEFAULT	Specifies the default device number. This value is set during MIL installation and using MilConfig. Note that the end-user can always change this value. (summarize)	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEVn	Specifies the first acquisition path of the digitizer with which to associate a timing control unit of your frame grabber. (summarize)	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																						
	The value of <i>n</i> must be 0 since you can only grab from one digitizer at a time.	a	b				g		i	j	k	l	m										w
	The value of <i>n</i> must be either 0 or 1.															p							
	The value of <i>n</i> must be either 0 or 1 for Matrox Morphis Dual. The value of <i>n</i> must be from 0 to 3 for Matrox Morphis Quad.													n									
	The value of <i>n</i> must be from 0 to 15.		c												o							v	
	When dealing with a cluster of boards featuring Matrox Odyssey Xpro, <i>n</i> can be a value from 0 to 31. Use MsysInquire() with M_DIGITIZER_NUM to return the total number of possible independent acquisition paths available on the system. Use MsysInquire() with M_DIGITIZER_TYPE + M_DEVn to return the type of frame grabber on which the specified acquisition path is present.															q	r	s					
	When not dealing with a cluster of boards featuring Matrox Odyssey Xpro, the value of <i>n</i> must be from 0 to 3. For details regarding allocation of specific independent digitizers, see the Allocating independent MIL digitizers on Matrox Odyssey section in Chapter 13: Matrox Odyssey .															q		s					
	When dealing with Matrox Solios XA single analog, the value of <i>n</i> must be 0. When dealing with Matrox Solios XA quad analog, the value of <i>n</i> must be from 0 to 3. For details regarding allocation of specific independent digitizers, see the Allocating independent MIL digitizers on Matrox Solios section in Chapter 14: Matrox Solios .																		t				
	The value of <i>n</i> must be from 0 to 3.				e																		
	The value of <i>n</i> must be either 0 or 1. 1 is available to allocate a digitizer only in dual-Base mode.				f												r				u		
	The value of <i>n</i> must be from 0 to 3. The number of digitizers that can be allocated depends on the amount of bandwidth available using M_1394_BANDWIDTH() .						h																

DataFormat

Specifies the name of the digitizer configuration format (DCF) appropriate for your camera. Depending on the target system, different DCFs are supported.

One of the following values can be specified when allocating a digitizer for either a monochrome or a color camera:

● For monochrome or color cameras																								
<div><div></div>Value</div>	Description	corona-II (a)	coronoplus (b)	glige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ed/xd (f)	helios ed/xd (g)	helios ed/xd (g)	ieee 1394 i/dc (h)	iris (i)	met-II /cl (i)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey eel/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios eel/xd (u)	solios gige (v)	vio (w)
<div><div></div>MIL_TEXT(MIL_TEXT_PTR MonoOrColorDCF)</div>	Specifies the name of the digitizer configuration format (DCF) appropriate for your camera. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	Parameters																							
	MonoOrColorDCF Specifies the name of the digitizer configuration format (DCF).	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	"M_DEFAULT" Specifies the default digitizer format. This value is set during MIL installation and using MilConfig. Note that the end-user can always change this value.	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

🌐 For monochrome cameras

[illegible]

MdigChannel

Synopsis

Select the active input data or synchronization channel of a digitizer.

Syntax

```
void MdigChannel(
    MIL_ID DigId,
    MIL_INT64 Channel
)
```

Description

This function selects the active input data or synchronization channel for the specified digitizer. If the digitizer does not have the specified channel, an error is generated and the last selected channel remains effective. The default channel corresponds to the default channel for the specified digitizer's DCF.

When choosing the data channel, the default synchronization channel will be used. This value is specified in the DCF. For more channel information when grabbing, see the Board Specific notes of the appropriate board.

[Matrox Corona-II; Matrox Helios eA/XA; Matrox Meteor-II / Multi-Channel; Matrox Odyssey eA/XA; Matrox Solios eA/XA]

To override the default synchronization and data channels, make two calls to **MdigChannel()**. The first should specify the data input channel using **M_CHn + M_SIGNAL** and the second should specify the synchronization channel using **M_CHn+M_SYNC**.

[Matrox Corona-II; Matrox CronosPlus; Matrox Helios eA/XA; Matrox Meteor-II / Multi-Channel; Matrox Meteor-II / Standard; Matrox Morphis; Matrox Odyssey eA/XA; Matrox Solios eA/XA]

If you intend to switch channels, enable camera locking before the switch (use **MdigControl()** with **M_CAMERA_LOCK**). This avoids corrupt images or synchronization-lost errors when channel-switching.

[Matrox Morphis]

If you intend to switch channels after grabbing a single field or frame, enable the fast-channel switching mode to minimize loss of time between grabs; set **MdigControl()** **M_CAMERA_LOCK** control type to **M_ENABLE + M_FAST**.

Parameters

DigId

Specifies the identifier of the digitizer.

Channel

Specifies the channel on which the digitizer is to acquire data. This parameter can be set to one of the following values, depending on the number of channels available for the specified digitizer's data format.

● For specifying the channel																	
<input type="checkbox"/> Value	Description	corona-II (a)															vio (w)
<input type="checkbox"/> M_DEFAULT +	Corresponds to the default channel for the specified digitizer's data format or M_CH0 . If your digitizer has only one channel that supports the selected data format, this parameter can only be set to M_DEFAULT or M_CH0 . (summarize)	a	b			e	f	g	h	j	k	l	m	n	o	q	w
<input type="checkbox"/> M_CH0 +	Specifies channel 0 as the channel on which the digitizer is to receive input data. In cases where you can connect and switch between more than one camera, the signal names are mentioned in the specifics. (summarize)	a	b			e	f		h	j	k	l	m	n		q	v

[illegible]

	<div>camera 8).</div> <div>For composite color/monochrome input, this corresponds to VID_IN8.</div> <div>For Y/C input, this corresponds to VID_IN15 (Y of camera 8) and VID_IN16 (C of camera 8).</div>										m								
M_CH8 +	<div>Specifies channel 8 as the channel on which the digitizer is to receive input data. (summarize)</div> <div>Board specific</div> <div>For composite color/monochrome input, this corresponds to VID_IN8.</div> <div>For composite color/monochrome (decoder path) input, this corresponds to VID_IN9. This channel is only available to the Matrox Meteor-II /Standard for PCI and PC/104-Plus.</div>										m	n							
M_CH9 +	<div>Specifies channel 9 as the channel on which the digitizer is to receive input data. (summarize)</div> <div>Board specific</div> <div>For composite color/monochrome input, this corresponds to VID_IN9.</div> <div>For composite color/monochrome (decoder path) input, this corresponds to VID_IN10. This channel is only available to the Matrox Meteor-II /Standard for PCI and PC/104-Plus.</div>										m	n							
M_CH10 +	<div>Specifies channel 10 as the channel on which the digitizer is to receive input data. (summarize)</div> <div>Board specific</div> <div>For composite color/monochrome input, this corresponds to VID_IN10.</div> <div>For composite color/monochrome (decoder path) input, this corresponds to VID_IN11. This channel is only available to the Matrox Meteor-II /Standard for PCI and PC/104-Plus.</div>										m	n							
M_CH11 +	<div>Specifies channel 11 as the channel on which the digitizer is to receive input data. (summarize)</div> <div>Board specific</div> <div>For composite color/monochrome input, this corresponds to VID_IN11.</div> <div>For composite color/monochrome (decoder path) input, this corresponds to VID_IN12. This channel is only available to the Matrox Meteor-II /Standard for PCI and PC/104-Plus.</div>										m	n							
M_CH12 +	<div>Specifies channel 12 as the channel on which the digitizer is to receive input data. (summarize)</div> <div>For composite color/monochrome input, this corresponds to VID_IN12.</div>											n							
M_CH13 +	<div>Specifies channel 13 as the channel on which the digitizer is to receive input data. (summarize)</div> <div>For composite color/monochrome input, this corresponds to VID_IN13.</div>											n							
M_CH14 +	<div>Specifies channel 14 as the channel on which the digitizer is to receive input data. (summarize)</div> <div>For composite color/monochrome input, this corresponds to VID_IN14.</div>											n							
M_CH15 +	<div>Specifies channel 15 as the channel on which the digitizer is to receive input data. (summarize)</div> <div>For composite color/monochrome input, this corresponds to VID_IN15.</div>											n							
M_RGB +	<div>Specifies the RGB input source (if present) on which the digitizer is to receive input data. This is the same as M_CH0, except that the synchronization is set on channel 3. This selection can be used only for RGB input. This corresponds to VID1_IN1, VID1_IN2, VID1_IN3, and SYNC_IN (synchronization signal).</div>	a										i							

[\(summarize\)](#)

Combination constant for the values listed in [For specifying the channel](#)

You can add the following value to the above-mentioned values to specify that the selected channel will be used for data input.

● For any supported channel																									
Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cd (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nextis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	vio (w)	
M_SIGNAL	Specifies that the selected channel will be used for data input.	a				e							l				q			t					

The following values can be used to select a synchronization channel instead of a data channel.

[Matrox Helios eA/XA; Matrox Odyssey eA/XA; Matrox Solios eA/XA]

Note that using a digitizer as an external synchronization signal renders the digitizer of that signal "occupied" and not available for allocating as a digitizer.

● For a synchronization channel																									
Value	Description	corona-II (a)	crosoplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nextis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ec/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	vio (w)		
M_CH0 + M_SYNC	Specifies channel 0 as the channel on which the digitizer is to receive synchronization data. (summarize)	a				e						l				q			t						
	Board specific																								
	This corresponds to VID1_IN1. This corresponds to P0_VID_IN_ <i>n</i> , where <i>n</i> is either A or B. The synchronization signal must be received on a signal whose name ends with the same letter as the data signal or the specified digitizer must not be able to receive data on this signal.	a						l																	
M_CH1 + M_SYNC	Specifies channel 1 as the channel on which the digitizer is to receive synchronization data. (summarize)	a				e						l				q			t						
	Board specific																								
	This corresponds to VID1_IN2. This corresponds to P1_VID_IN_ <i>n</i> , where <i>n</i> is either A or B. The synchronization signal must be received on a signal whose name ends with the same letter as the data signal or the specified digitizer must not be able to receive data on this signal.	a						l																	
M_CH2 + M_SYNC	Specifies channel 2 as the channel on which the digitizer is to receive synchronization data. (summarize)	a				e						l				q			t						
	Board specific																								
	This corresponds to VID1_IN3. This corresponds to P2_VID_IN_ <i>n</i> , where <i>n</i> is either A or B. The synchronization signal must be received on a signal whose name ends with the same letter as the data signal or the specified digitizer must not be able to receive data on this signal.	a						l																	
M_CH3 + M_SYNC	Specifies channel 3 as the channel on which the digitizer is to receive synchronization data.	a				e						l				q			t						

[illegible]

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigControl

Synopsis

Control a digitizer setting.

Syntax

```
void MdigControl(
    MIL_ID DigId,
    MIL_INT64 ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function allows you to control various digitizer settings.

If changing a control type of a digitizer currently being used in a grab, the change does not affect the current grab. It does, however, affect the very next grab.

Note that you can use [MdigInquire\(\)](#) to inquire about specific digitizer settings.

[Matrox IEEE 1394 IIDC driver; Matrox Solios GigE]

The control types described in this function can control either the connected camera or the Matrox imaging board. In situations where the control type is available on both the camera and the board, you can use the [M_CAMERA](#) or the [M_BOARD](#) combination constant to specify which to control.

When the control type is only available on the camera, the standard feature name is provided as a board-specific statement within the control type description. If your camera does not support the standard feature, an error is generated.

When the control type is only available on the Matrox imaging board, it is handled automatically by your Matrox imaging board. When the control type is available on both the camera and the board, it is handled automatically by your camera. If the camera does not support the standard feature, an error is generated. In this case, to use the control type with the board, you must use the [M_BOARD](#) combination constant.

Parameters

DigId

Specifies the identifier of the digitizer.

ControlType

Specifies the digitizer setting to control.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the value to assign to the digitizer setting.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For controlling the general digitizer settings](#)
- [For controlling the input gain](#)
- [For routing signals and setting their mode](#)
- [For controlling user-defined signals](#)

[illegible]

[illegible]

[illegible]

<div><div></div><div>M_REVERSE</div></div>	Flips the grabbed image horizontally.	a	b	c	e	f	g		j	k	l	m	n	o	p	q	r	s	t	u	v		
<div><div></div><div>M_GRAB_DIRECTION_Y +</div></div>	Sets the vertical grab direction. (summarize)	a	b	c	e	f			i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																						
	This control type only has an effect if the grab buffer has an M_ON_BOARD attribute; there is no effect when grabbing to Host memory.									j	k												
	In addition, grab buffers with the M_DISP attribute are not supported.																						
	By default, this control type is GenICam camera specific. The camera must support the ReverseY GenICam standard feature for this control type to be available on the camera.			c																			v
	Note that this control type is also available on the Matrox Imaging board.																						v
<div><div></div><div>M_DEFAULT</div></div>	Same as M_FORWARD .	a	b	c	e	f			i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_FORWARD</div></div>	Grabs from top to bottom, in the vertical direction.	a	b	c	e	f			i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_REVERSE</div></div>	Flips the grabbed image vertically.	a	b	c	e	f			i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_GRAB_FAIL_CHECK</div></div>	Sets whether and when to log a grab-fail error. (summarize)	b																					
<div><div></div><div>M_DEFAULT</div></div>	Same as M_FINAL_GRAB .	b																					
<div><div></div><div>M_DISABLE</div></div>	Does not log grab failures.	b																					
<div><div></div><div>M_ENABLE</div></div>	Logs all grab failures.	b																					
<div><div></div><div>M_FINAL_GRAB</div></div>	Logs monoshot grab failures or that of the last frame of a continuous grab.	b																					
<div><div></div><div>M_GRAB_FAIL_RETRY_NUMBER</div></div>	Sets the number of retries when a field or frame grab fails. (summarize)	b																					
<div><div></div><div>M_DEFAULT</div></div>	Specifies the default value. The default value is 1. (summarize)	b																					
<div><div></div><div>0 to 255</div></div>	Specifies the required number.	b																					
<div><div></div><div>M_GRAB_FIELD_NUM</div></div>	Sets the number of fields to grab when grabbing data with MdigGrab() . This control type should only be used with interlaced cameras. When using progressive cameras, set this control type to 1. (summarize)	a	b	c	e	f	g	h	j	k	l	m	n	o						t	u	v	w
<div><div></div><div>M_DEFAULT</div></div>	Specifies the default value. (summarize)	a	b	c	e	f	g	h	j	k	l	m	n	o						t	u	v	w
	<i>Board specific</i>																						
	For interlaced cameras, the default is 2. For progressive cameras, the default is 1.	a	b		e	f	g		j	k	l	m	n	o						t	u		w
	For GigE Vision cameras, the default is 1.		c																			v	
<div><div></div><div>1</div></div>	Specifies to grab one field. The grabbed field is written to sequential rows of the grab buffer. When set to 1, each field is treated like a frame and the following digitizer events occur relative to the grabbed field: M_GRAB_FRAME_START , M_GRAB_END , and M_GRAB_FRAME_END . To achieve 60 fps in NTSC or 50 fps in PAL, M_GRAB_START_MODE must be set to M_FIELD_START . (summarize)	a	b	c	e	f	g	h	j	k	l	m	n	o						t	u		w
<div><div></div><div>2</div></div>	Specifies to grab two fields.	a	b		e	f	g		j	k	l	m	n	o						t	u		w
<div><div></div><div>M_GRAB_LINE_COUNTER</div></div>	Sets whether a function hooked to an M_GRAB_END , M_ROTARY_ENCODER , or M_GRAB_FRAME_END event should return the number of lines grabbed; the function could have been hooked using either MdigProcess() or MdigHookFunction() . (summarize)			c	e	f	g			k				o		q	r	s	t	u			

[illegible]

<div><div></div>M_SOURCE_OFFSET_Y +</div>	<div>Sets the Y-offset of the input signal capture window. (summarize)</div> <div><div>Board specific</div><div>By default, this control type is GenICam camera specific. The camera must support the OffsetY GenICam standard feature for this control type to be available on the camera.</div><div>Note that this control type is also available on the Matrox Imaging board.</div><div>By default, this control type is IIDC camera specific. The camera must support the format 7 video mode and the IMAGE_POSITION IIDC standard feature for this control type to be available on the camera. This control type is also available on the board.</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>Value</div>	<div>Specifies the Y-offset, in pixels.</div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_SOURCE_SIZE_X +</div>	<div>Sets the width of the input signal capture window. Note that certain source sizes, offsets, and destination buffer sizes can affect a grab. Refer to MdigGrab() and MdigGrabContinuous() for more information. (summarize)</div> <div><div>Board specific</div><div>This control type is always available on a GigE Vision camera. However, on some cameras you might not be able to set the width (that is, this control type might be read-only).</div><div>Note that this control type is also available on the Matrox Imaging board.</div><div>By default, this control type is IIDC camera specific. The camera must support the format 7 video mode and the IMAGE_SIZE IIDC standard feature for this control type to be available on the camera. This control type is also available on the board.</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>Value</div>	<div>Specifies the width, in pixels.</div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_SOURCE_SIZE_Y +</div>	<div>Sets the height of the input signal capture window. Note that certain source sizes, offsets, and destination buffer sizes can affect a grab. Refer to MdigGrab() and MdigGrabContinuous() for more information. (summarize)</div> <div><div>Board specific</div><div>This control type is always available on a GigE Vision camera. However, on some cameras you might not be able to set the height (that is, this control type might be read-only).</div><div>Note that this control type is also available on the Matrox Imaging board.</div><div>By default, this control type is IIDC camera specific. The camera must support the format 7 video mode and the IMAGE_SIZE IIDC standard feature for this control type to be available on the camera. This control type is also available on the board.</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>Value</div>	<div>Specifies the height, in pixels.</div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_SYNCHRONIZE_ON_STARTED</div>	<div>Forces the next grab with the current digitizer to wait and start at the end of a grab with a second digitizer, so that the grabs are consecutive. This allows you to grab sequential frames from two or more MIL digitizers. Note that to force MIL to wait for the start of a grab performed by a second digitizer instead, use MdigControl() with M_GRAB_WAIT. See the M_SYNCHRONIZE_ON_STARTED example below. (summarize)</div>																q	r	s				
<div><div></div>M_NULL</div>	<div>Specifies not to synchronize with a second digitizer. This is the default value. (summarize)</div>																	q	r	s			
<div><div></div>MIL digitizer identifier</div>	<div>Specifies the MIL identifier of the second digitizer.</div>																	q	r	s			

You can add the following value to the above-mentioned value to specify that ultra-fast camera locking is enabled.

Value	Description
M_FAST	Enables ultra-fast camera locking. This uses numerous optimizations to increase the speed of the lock. Use ultra-fast camera locking when high reliability is not an issue; increasing the speed of the camera lock affects the robustness of the subsequent unlock. (summarize)

- For controlling the input gain

ControlType		Description	corona-ii (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios eai/xa (e)	helios eai/xd (f)	helios eai/xcl (f)	iris (i)	leee 1394 i1dc (h)	met-ii /dlq (k)	met-ii /mc (l)	met-ii /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey eai/xa (q)	odyssey eai/xcl (r)	odyssey eai/xd (s)	solios eai/xa (t)	solios eai/xcl (u)	solios gige (v)	vio (w)	
ControlValue																									
M_GAIN		Sets the input gain with which to amplify the input signal. Note that these corrections affect the acquisition path before the data is digitized, and are applied to all the color bands of the image. For a more conventional technique of setting the gain, use MdigControl() with M_GRAB_INPUT_GAIN . (summarize)			c					h														v	
		<i>Board specific</i>																							
		Note that this control type is IIDC camera specific. The camera must support the GAIN IIDC standard feature for this control type to be supported.								h															
		Note that this control type is GenICam camera specific. The camera must support the GainRaw GenICam standard feature for this control type to be supported.			c																			v	
M_DEFAULT		Specifies the default value (if supported).			c					h														v	
Value		Specifies a value between the minimum and maximum values supported by the camera. Use MdigInquire() to determine these values. (summarize)			c					h														v	
M_GRAB_AUTOMATIC_INPUT_GAIN +		Sets whether the input gain should be automatically set. (summarize)		b	c										n	o								v	w
		<i>Board specific</i>																							
		If enabled, the on-board decoder automatically sets the input gain.		b											n	o								w	
		By default, this setting is applied to both Y (luminance) and C (chrominance) components (for composite, after separation). To apply this setting to a specific													n	o									

[illegible]

You can add one of the following values to the above-mentioned values to specify to which video signal component to apply the input-gain setting.

Note that for other Matrox imaging boards that have user-defined signals, but are not supported with the constants below, see [MsSysControl\(\)](#).

[illegible]

You can add one of the following values to the above-mentioned values to specify which component(s) to affect.

Value	Description
<div>via (w)</div>	<div>sollos glig sollos ect sollos ea odyssey odyssey odyssey nexis (p) morphis morphis met-II /s/ met-II /r/ met-II /d/ met-II /c/ iris (l) lee 139 helios ed helios ect helios ea helios ea gpu proc gige visic cromospl corona-II</div>

You must add one of the following values to the above-mentioned values to specify the signal(s) to affect.

For a listing of available user-defined or auxiliary signals and their corresponding numbers in MIL, see the Connectors and signal name section of the MIL Board-specific Notes chapter for your Matrox imaging board.

For specifying the signal to affect																									
Value	Description	corona-II (a)	gpu vision (c)	coronoplus (b)	gige vision (d)	gpu processing (d)	helios ea/xa (e)	helios ec/xcl (f)	helios ed/xd (g)	helioc 1394 iildc (h)	iris (i)	met-II /cd (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis gxt (o)	nextis (p)	odyssey ea/xa (q)	odyssey ed/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ed/xd (u)	solos gige (v)	via (w)
M_BIT_MASK(MIL_INT Bit_encoded_value)	<p>Specifies a bit-encoded mask value that identifies the group of user-defined signals to affect. Enabled bits in the mask value identify which user-defined signals to affect.</p> <p>Note that, the control value can only be set to a bit-encoded value. (summarize)</p>																		q	r	s				

[Matrox Morphis]
Triggered grabs are not supported on Matrox Morphis Quad.

For controlling the settings to grab using a trigger																								
ControlType		Description	corona-II (a)	coronaplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ed/xd (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis opt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ed/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ed/xd (u)	solios gige (v)	solios v (w)
ControlValue																								
M_GRAB_CONTINUOUS_END_TRIGGER		Sets whether to automatically generate a trigger after <code>MdigHalt()</code> is issued when performing a triggered continuous grab. (summarize)			c	e	f	g	h	i					n		p	q	r	s	t	u	v	
M_DEFAULT		Specifies the default value. <code>M_ENABLE</code> is the default for a software trigger grab. <code>M_DISABLE</code> is the default for a hardware trigger grab. (summarize)			c	e	f	g	h	i					n		p	q	r	s	t	u	v	
M_DISABLE		Does not generate the trigger automatically. With a software triggered continuous grab operation, <code>MdigHalt()</code> will wait indefinitely until a software trigger is issued on a separate thread.			c	e	f	g	h	i					n		p	q	r	s	t	u	v	

[illegible]

[Matrox Solios GigE]

To change the purpose of an auxiliary signal to a user-defined signal, use `M_AUX_SIGNAL_SOURCE` set to `M_USER_BIT`. To set the mode of an I/O auxiliary signal, use `M_AUX_SIGNAL_MODE`.

ControlType	Description	vio (w)	solios gige (v)	solios eci/xci (u)	solios ea/xa (t)	odysey ed/pd (s)	odysey eci/xci (r)	odysey ea/xa (q)	nexis (p)	morphis qxt (o)	morphis (n)	met-II/std (m)	met-II/mrc (l)	met-II/dig (k)	met-II/cd (j)	hrs (i)	ieee 1394 iildc (h)	helios ed/xci (g)	helios eci/xci (f)	helios ea/xa (e)	gpu processing (d)	gige vision (c)	cronosplus (b)	corona-II (a)
ControlValue																								
M_AUTO_EXPOSURE +	Sets whether the exposure should be set manually to the specified value or if the camera controls the exposure level automatically and continuously. (summarize)		v														h				c			
	Board specific																							
	Note that this control type is IIDC camera specific. The camera must support the AUTO_EXPOSURE IIDC standard feature for this control type to be supported.																h							
	Note that this control type is GenICam camera specific. The camera must support the ExposureAuto GenICam standard feature for this control type to be supported.																				c			v
M_DEFAULT	Specifies the default value.																h							
M_DISABLE	Specifies that the camera will not control the exposure level and duration automatically. Instead the exposure duration is manually controlled using M_GRAB_EXPOSURE_TIME. (summarize)																				c			v
M_ENABLE	Specifies that the camera controls the exposure level and duration automatically for best effect.																				c			v
Value	Specifies a value between the minimum and maximum values supported by the camera. Use MdigiInquire() to determine these values. (summarize)																h							

<div><div></div><div>M_EDGE_RISING</div></div>	<p>Specifies to arm the trigger upon a high-to-low signal variation.</p>																		v		
<div><div></div><div>M_GRAB_EXPOSURE_ARM_SOURCE +</div></div>	<p>Sets the signal source that will arm the exposure trigger.</p> <p>Note that this control type only has an affect if M_GRAB_EXPOSURE_ARM is set to M_ENABLE.</p> <p>To force the exposure to be armed, use M_GRAB_EXPOSURE_ARM_SOURCE set to M_SOFTWARE and then set M_GRAB_EXPOSURE_ARM to M_ACTIVATE.</p> <p>Note that this control type automatically controls the Matrox Imaging board. It is not available to control the camera.</p> <div>(summarize)</div>																		v		
<div><div></div><div>M_DEFAULT</div></div>	<p>The default value is M_HARDWARE_PORT0.</p>																		v		
<div><div></div><div>M_HARDWARE_PORTn</div></div>	<p>Uses hardware port <i>n</i> to arm the exposure, where <i>n</i> is a value from 0 to 9.</p> <p>For a list of the available MIL hardware port numbers and their associated signal names, see the Matrox Solios GigE connectors and signal names section in Chapter 14: Matrox Solios.</p> <p>Note that this control value is GenICam camera specific. The camera must support the Line <i>n</i> GenICam standard feature, where the line number (<i>n</i>) represents the MIL hardware port number. This feature is either zero-based, or one-based. If the feature is zero-based, the line number (<i>n</i>) is a number from 0 to 11. If the feature is one-based, the line number (<i>n</i>) is a number from 1 to 12.</p> <div>(summarize)</div>																		v		
<div><div></div><div>M_SOFTWARE</div></div>	<p>Uses a software generated signal to arm the exposure timer. In this mode, the Matrox imaging driver automatically arms the exposure when a grab is queued.</p> <div>(summarize)</div>																		v		
<div><div></div><div>M_TIMERn</div></div>	<p>Uses the MIL exposure timer signal to arm the exposure, where <i>n</i> is a value from 1 to 8.</p> <p>Two exposure timer groups exist, 1 to 4 and 5 to 8. An exposure timer from group 1 to 4 can be used as a arm source for a timer from group 1 to 4, but cannot be used as a arming source for a timer from group 5 to 8, and vice versa.</p> <p>For a list of the available timers and their associated signals, see the Connectors and signal names section of the MIL Board-specific Notes for your Matrox imaging board.</p> <div>(summarize)</div>																		v		
<div><div></div><div>M_GRAB_EXPOSURE_CLOCK_FREQUENCY +</div></div>	<p>Specifies the source of the clock that drives the specified exposure timer.</p> <p>Note that this control type can be used only if M_GRAB_EXPOSURE_CLOCK_SOURCE is set to M_HARDWARE_PORTn.</p> <div>(summarize)</div>																		v		
	<div>Board specific</div>																		v		
<div><div></div><div>0 <= Value</div></div>	<p>Sets the input clock frequency, in Hz.</p>																		v		
<div><div></div><div>M_GRAB_EXPOSURE_CLOCK_SOURCE +</div></div>	<p>Specifies the source of the clock that drives the specified exposure timer.</p> <p>The clock source must have a frequency greater than or equal to 1 Hz.</p> <div>(summarize)</div>	a				e	f	g				l				q	r	s	t	u	v
	<div>Board specific</div>																				
	<p>Note that this control type automatically controls the Matrox Imaging board. It is not available to control the camera.</p>																				v
<div><div></div><div>M_DEFAULT</div></div>	<p>Specifies the default value.</p> <div>(summarize)</div>	a				e	f	g				l				q	r	s	t	u	v
	<div>Board specific</div>																				

	Uses the most appropriate clock source (as determined by the driver).	a		e	f	g			l			q	r	s	t	u	
	Same as M_AUTOMATIC .																v
<input type="checkbox"/> M_AUTOMATIC	Uses the most appropriate clock source from the 17 possible internal clock sources.																v
<input type="checkbox"/> M_CLOCKn	Uses the specific internal clock source, where n is a value from 0 to 16. The frequency of the internal clock source is $100 \text{ MHz} / 2^n$. (summarize)																v
<input type="checkbox"/> M_HARDWARE_PORTn	Specifies that hardware port n will be used to specify the clock source, where n is a value from 0 to 9. For a list of the available MIL hardware port numbers and their associated signal names, see the Matrox Solios GigE connectors and signal names section in Chapter 14: Matrox Solios . Note that this control value is GenICam camera specific. The camera must support the Line n GenICam standard feature, where the line number (n) represents the MIL hardware port number. This feature is either zero-based, or one-based. If the feature is zero-based, the line number (n) is a number from 0 to 11. If the feature is one-based, the line number (n) is a number from 1 to 12. (summarize)																v
<input type="checkbox"/> M_HSYNC	Uses the horizontal synchronization frequency of your camera.	a		e	f	g			l			q	r	s	t	u	
<input type="checkbox"/> M_PIXCLK	Uses the pixel clock frequency of your camera.	a		e	f	g			l			q	r	s	t	u	
<input type="checkbox"/> M_SYSCLK	Uses the Matrox imaging board's clock source frequency. (summarize)	a		e	f	g			l			q	r	s	t	u	
	<i>Board specific</i>																
	Uses a 25 MHz clock source frequency.	a							l								
	Uses a 16.666 MHz clock source frequency.			e	f	g						q	r	s	t	u	
<input type="checkbox"/> M_TIMERn	Uses the frequency of the specified timer. For a list of the available timers and their associated signals, see the Connectors and signal names section of the MIL Board-specific Notes for your Matrox imaging board. (summarize)	a		e	f	g			l			q	r	s	t	u	v
	<i>Board specific</i>																
	When controlling exposure timer 1, n can be 2. That is, you can use the frequency of timer 2 to clock timer 1. When controlling exposure timer 2, n can be 1. That is, you can use the frequency of timer 1 to clock timer 2.	a		e	f	g			l			q	r	s	t		
	When controlling exposure timer 1, n can be 2. That is, you can use the frequency of timer 2 to clock timer 1. When controlling exposure timer 2, 3, or 4, n can be 1. That is, you can use the frequency of timer 1 to clock timer 2, 3, or 4.															u	
	Uses the frequency of a specific timer, where n is a value from 1 to 8.																v
	Two exposure timer groups exist, 1 to 4 and 5 to 8. An exposure timer from group 1 to 4 can be used as a clock source for a timer from group 1 to 4, but cannot be used as a clock source for a timer from group 5 to 8, and vice versa.																
<input type="checkbox"/> M_VSYNC	Uses the vertical synchronization frequency of your camera.			e	f	g						q	r	s	t	u	
<input type="checkbox"/> M_GRAB_EXPOSURE_FORMAT +	Sets whether to enable the transmitters for exposures output signals, on systems whose transmitters are enabled through software and where the option of multiple signal types are possible. Refer to the Technical information appendix in the Installation and Hardware Reference manual for your Matrox imaging board. For a list of the available signals, see the Connectors and signal names section of the MIL Board-specific Notes chapter for your Matrox imaging board. (summarize)											q	r	s			
<input type="checkbox"/> M_DEFAULT	Same as the one specified by the DCF.											q	r	s			

[illegible]

[illegible]

You can add one of the following values to the above-mentioned values to specify to control the camera or the board.

For the rotary counter of the quadrature decoder					
<div><div></div>ControlType</div>		Description	vio (w)	slios gge (v)	slios eci/xcl (u)
<div><div></div>ControlValue</div>			helios ea/xa (t)	odyssey ed/xd (s)	odyssey ecj/xc (r)
M_ROTARY_ENCODER	Sets whether to enable the rotary counter of the quadrature decoder. If enabled, the value of the rotary counter can be retrieved at the end of a grab using <code>MdigGetHookInfo()</code> with <code>M_ROTARY_ENCODER_FRAME_END_POSITION</code> . Note that the current value of the rotary counter can be inquired using <code>MdigInquire()</code> with <code>M_ROTARY_ENCODER_POSITON</code> .	Note that if the rotary encoder is not connected, the counter will not increment. (summarize)	e f g		u
M_DEFAULT	Same as M_DISABLE.		e f g		u
M_DISABLE	Disables the rotary counter.		e f g		u
M_ENABLE	Enables rotary counter.		e f g		u
M_ROTARY_ENCODER_DIRECTION	Specifies whether the rotary counter is incrementing or decrementing for the forward direction of the rotary encoder. (summarize)		e f g	r s	u
M_DEFAULT	Same as M_FORWARD.		e f g	r s	u
M_BACKWARD	Decrements the rotary counter. Note that the counter does not decrement to negative numbers. For example, if the rotary counter is zero, the next decremented value is 429496295 ($2^{32}-1$). (summarize)		e f g	r s	u
M_FORWARD	Increments the rotary counter. Note that the counter cannot increment beyond the value of 429496295 ($2^{32}-1$). In this case, the next incremented value is 0. (summarize)		e f g	r s	u
M_ROTARY_ENCODER_POSITION	Resets the rotary counter. (summarize)		e f g	r s	u
Value = 0	The rotary counter can only reset to zero. If a non-zero value is specified, an error is generated. (summarize)		e f g	r s	u
M_ROTARY_ENCODER_POSITION_TRIGGER	Specifies the rotary counter value upon which a trigger is generated. (summarize)		e f g	r s	u
0 to 429496295	Specifies the value of the rotary counter upon which a trigger is generated. If a value beyond this range is specified, an error is generated.		e f g	r s	u
Do not specify a negative number when the rotary direction is set to <code>M_BACKWARD</code> . You must instead specify a positive rotary counter value, calculated by subtracting the required number of increments from the current position. For example, to trigger after 8 decrements when the counter is reset to 0, specify the rotary counter as (429496295 - 8) 429496287. (summarize)					

Note that for other Matrox imaging boards that have a UART, but are not supported with the constants below, see [MsysControl\(\)](#).

Note that for other Matrox imaging boards that have a UART, but are not supported with the constants below, see [MsysControl\(\)](#).

For controlling UART settings																								
ControlType		Description																						
ControlValue																								
M_UART_DATA_LENGTH +		Sets the number of data bits per character that is sent or received by the UART. (summarize)		a							j	k	l	m			q	r	s					
M_DEFAULT		Same as 8 .		a							j	k	l	m			q	r	s					
7		Specifies data length to 7 bits.		a							j	k	l	m			q	r	s					
8		Specifies data length to 8 bits.		a							j	k	l	m			q	r	s					
M_UART_PARITY +		Sets whether character data is sent or received with a parity bit and how the parity bit is set. The parity bit is an extra data bit (0 or 1) that is added to each character for error checking purposes. (summarize)		a							j	k	l	m			q	r	s					
M_DEFAULT		Same as M_DISABLE .		a							j	k	l	m			q	r	s					
M_DISABLE		Species that no extra bit is added (no parity).		a							j	k	l	m			q	r	s					
M_EVEN		Specifies that the number of 1's will be even.		a							j	k	l	m			q	r	s					
M_ODD		Specifies that the number of 1's will be odd.		a							j	k	l	m			q	r	s					
M_UART_READ_CHAR +		Reads one 8-bit ASCII character from the UART input buffer. If the input buffer is empty, the function will wait for the amount of time specified by M_UART_TIMEOUT . If a time-out occurs, the '?' character will be returned. Note that the UART input buffer is stored in the <i>MIL_INT8</i> data type. (summarize)		a							j	k	l	m			q	r	s					
M_PTR_TO_DOUBLE(char MPTYPE * PTR)		Specifies the address of the variable in which to save the 8-bit ASCII character read from the UART. (summarize)		a							j	k	l	m			q	r	s					
		<i>Parameters</i>																						
		<i>PTR</i> The address of the variable in which to save the 8-bit ASCII character.		a								j	k	l	m			q	r	s				
M_UART_READ_STRING +		Reads a string of incoming data from the UART. The number of 8-bit ASCII characters to read can be specified with M_UART_READ_STRING_LENGTH or M_UART_STRING_DELIMITER . M_UART_TIMEOUT specifies the maximum time to wait between each byte when reading incoming data. (summarize)		a							j	k	l	m			q	r	s					
M_PTR_TO_DOUBLE(char MPTYPE * PTR)		Specifies the address of the 8-bit ASCII character array in which to save the string read from the UART. The size of this array must be set to the same value as the M_UART_READ_STRING_MAXIMUM_LENGTH control type. (summarize)		a							j	k	l	m			q	r	s					
		<i>Parameters</i>																						
		<i>PTR</i> The address of the 8-bit ASCII character array in which to save the string.		a								j	k	l	m			q	r	s				
M_UART_READ_STRING_LENGTH +		Sets the length of the string to be read using M_UART_READ_STRING . (summarize)		a							j	k	l	m			q	r	s					
M_DEFAULT		Specifies the use of M_UART_STRING_DELIMITER to delineate the end of the string to be read.		a							j	k	l	m			q	r	s					
Value		Specifies the string length, in bytes.		a							j	k	l	m			q	r	s					
M_UART_READ_STRING_MAXIMUM_LENGTH +		Sets the maximum length to be read. This prevents global protection faults from happening		a							j	k	l	m			q	r	s					

For controlling audio signals																									
ControlType		Description	corona-II (a)	cronosplus (b)	glige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ec/Xd (f)	helios ed/Xd (g)	heli1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morpheus (n)	morpheus qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ec/Xd (r)	odyssey ed/Xd (s)	solos ea/Xa (t)	solos ec/Xd (u)	solos glige (v)	vio (w)
ControlValue																									
M_AUDIO_CONTROL		Sets the state of audio acquisition when grabbing with the specified digitizer. Note that, with the exception of M_AUDIO_GAIN and M_AUDIO_VOLUME , audio control types cannot be changed once this control is enabled. (summarize)															0								
M_DEFAULT		Same as M_DISABLE .															0								
M_DISABLE		Disables audio acquisition.															0								
M_ENABLE		Enables audio acquisition.															0								
M_AUDIO_ENCODING_FORMAT		Sets the audio encoding format. Note that this control type can only be changed when audio acquisition is disabled. (summarize)															0								
M_DEFAULT		Same as M_AUDIO_RAW_16 .															0								
M_AUDIO_ADPCM		Specifies the format as 4-bit ADPCM.															0								
M_AUDIO_ALAW		Specifies the format as 8-bit a-Law.															0								
M_AUDIO_MULAW		Specifies the format as 8-bit mu-Law.															0								
M_AUDIO_RAW_16		Specifies the format as 16-bit raw audio.															0								
M_AUDIO_GAIN		Sets the audio gain. Note that this control can only be changed when audio acquisition is disabled. This control type should be modified only if the initial signal is weak. (summarize)															0								
M_DEFAULT		The default value is 0.0 db. This results in a 1:1 ratio, which applies no audio gain to the output signal. (summarize)															0								
-12.0 to 35.25		Specifies the audio gain, in dB.															0								
M_AUDIO_SAMPLING_RATE		Sets the audio sampling rate. This control type can help clarify the sound. Typically higher rates produce larger buffers, but better results. (summarize)															0								
M_DEFAULT		Specifies the default value. The default value is 48000 Hz. (summarize)															0								
8000; 12000; 16000; 24000; 32000; 48000		Specifies the sampling rate, in Hz.															0								
M_AUDIO_VOLUME		Sets the audio volume. Note that this is a volume control and not an increase in audio gain, so this maximum volume cannot exceed the initial strength of the acquired signal. If, after modifying the volume, the signal is still too muted, try adjusting the audio gain, using M_AUDIO_GAIN . (summarize)															0								
M_DEFAULT		Same as M_MAX_LEVEL .															0								

The following example uses [M_GRAB_WAIT](#) to grab the same frame from multiple digitizers allocated on different Matrox Odyssey systems in the same cluster. Note that if the specified digitizer is already grabbing when this sample is invoked, the frame that is grabbed will be the one currently in on-board memory and not necessarily the one intended.

```

/* Synopsis: This program grabs the same frame to three buffers on different
 * systems. To do so, it uses 3 digitizers, each of which is allocated on
 * a different system within a cluster. Note that Dig1, Dig2, and Dig3 are
 * different digitizers and their grab mode is already set to M_ASYNCHRONOUS.
 */

/* Enable Dig2 and Dig3 to wait for the grab of Dig1.
 * Note that this needs to be done only once.
 */
MdigControl(Dig2, M_GRAB_WAIT, Dig1);
MdigControl(Dig3, M_GRAB_WAIT, Dig1);

/* Queue the grabs. */
MdigGrab(Dig2, Buf2);
MdigGrab(Dig3, Buf3);

/* Now queue the controlling grab. */
MdigGrab(Dig1, Buf1);

```

```

/* Synopsis: This program grabs sequential frames to three buffers on different
 * systems. To do so, it uses 3 digitizers, each of which is allocated on a
 * different system within a cluster. Note that Dig1, Dig2, and Dig3 are
 * different digitizers (on separate independent acquisition paths) and their
 * grab mode is already set to M_ASYNCHRONOUS.
 */

/* Queue the first grab. */
MdigGrab(Dig1, Buf1);

/* Queue the synchronization.
 * The grab of Dig2 will not be queued until Dig1 has started grabbing.
 */
MdigControl(Dig2, M_SYNCHRONIZE_ON_STARTED, Dig1);
MdigGrab(Dig2, Buf2);

/* Now synchronize Dig3 with Dig2. */
MdigControl(Dig3, M_SYNCHRONIZE_ON_STARTED, Dig2);
MdigGrab(Dig3, Buf3);

```

[Matrox Helios eA/XA; Matrox Helios eCL/XCL; Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD; Matrox Solios eA/XA; Matrox Solios eCL/XCL]

The following example uses [M_USER_BIT_VALUE](#) + [M_BIT_MASK\(\)](#). It sets user-defined signal 0 to 1 (on), 1 to 1 (on), and 5 to 0 (off):

```
MdigControl(MilDigitizer, M_USER_BIT_VALUE+M_BIT_MASK(0x23), 0x03);
```

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigControlFeature

Synopsis

Controls a feature of the camera.

Syntax

```
void MdigControlFeature(
    MIL_ID DigitizerId,
    MIL_ID ControlFlag,
    MIL_TEXT_PTR FeatureName,
    MIL_INT FeatureDataType,
    void *FeatureValuePtr
)
```

Description

This function allows you to directly control various manufacturer-specific features of the camera. When used with a GiGE Vision camera, it allows you to directly control various manufacturer-specific features specified with the camera's device description file (XML). To inquire a camera's manufacturer-specific feature, use [MdigInquireFeature\(\)](#). Note that for a complete list of available feature names and associated possible values, refer to your camera's documentation.

Parameters

DigitizerId

Specifies the identifier of the digitizer on which to apply the feature. This parameter must be given a valid digitizer identifier.

ControlFlag

Specifies the function's control flag. This parameter must be set to one of the following:

For specifying the function's control flag					
Value	Description	corona-II (a)	gige vision (c)	gpu processing (d)	vio (w)
M_DEFAULT	Sets the default value.		c		v
M_MAX	Sets the maximum value possible. Note that this value can only be used when FeatureDataType is set to M_TYPE_MIL_INT32 , M_TYPE_MIL_INT64 , or M_TYPE_DOUBLE . (summarize)		c		v
M_MIN	Sets the minimum value possible. Note that this value can only be used when FeatureDataType is set to M_TYPE_MIL_INT32 , M_TYPE_MIL_INT64 , or M_TYPE_DOUBLE . (summarize)		c		v

FeatureName

Sets the name of the camera feature to control.

For specifying the name of the feature	
Value	Description
solios gige	via (w)
solios ed/x	solios ed/x
solios ea/x	solios ea/x
odyssey ed	odyssey ed
odyssey ea	odyssey ea
nexis (p)	nexis (p)
morphis q	morphis q
met-II/stu	met-II/stu
met-II/m	met-II/m
met-II/d/c	met-II/d/c
met-II/cl	met-II/cl
lee 1394	lee 1394
helios ed/x	helios ed/x
helios ed/	helios ed/
helios ea/x	helios ea/x
cpu proces	cpu proces
glige vision	glige vision
corona-II	corona-II

[illegible]

FeatureValuePtr

Accepts the address of one of the following (see above for specifics on which is expected):

- bool
- M_NULL
- MIL_DOUBLE
- MIL_INT32
- MIL_INT64
- MIL_TEXT_PTR

Specifies the address of the variable in which to write the value of the feature.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigFocus

Synopsis

Adjust a camera's lens motor to a position which provides optimum focus.

Syntax

```
void MdigFocus(
    MIL_ID DigId,
    MIL_ID DestImageBufId,
    MIL_ID FocusImageRegionBufId,
    MIL_FOCUS_HOOK_FUNCTION_PTR FocusHookPtr,
    void *UserDataPtr,
    MIL_INT MinPosition,
    MIL_INT StartPosition,
    MIL_INT MaxPosition,
    MIL_INT MaxPositionVariation,
    MIL_INT ProcMode,
    MIL_INT *ResultPtr
)
```

Description

This function adjusts the lens motor of the camera, attached to the specified digitizer, to a position which produces optimum focus. **MdigFocus()** determines the optimum focus position by grabbing an image at an initial lens position, analyzing the focus quality of the grabbed image, calling a user-defined function that changes the position of the lens motor, and then grabbing and analyzing another image. The process repeats until the optimum focus position is found.

A specified search strategy determines how the position of the lens motor is updated (in which direction and by how much) between grabs. For more information, see the [Auto-focusing](#) section in [Chapter 22: Grabbing with your digitizer](#).

By default, before an image is analyzed, it is subsampled and filtered.

Parameters

DigId

Specifies the identifier of the digitizer. If you want the user-defined function to perform the grab, as well as move the lens motor, set this parameter to **M_NULL**.

DestImageBufId

Specifies the identifier of the buffer in which to place the grabbed images. This buffer should be of an appropriate type to hold the grabbed images. This buffer cannot be a signed 3-band buffer, a 32-bit buffer, or a 1-bit packed buffer.

FocusImageRegionBufId

Specifies the identifier of a child buffer of the destination buffer. Analysis of each grabbed image is limited to the region specified by this buffer. If you want the entire image analyzed, set this parameter to **M_DEFAULT**.

FocusHookPtr

Specifies the address of the user-defined hook function to call before each grab. This function must be declared as follows:

```
MIL_INT MFTYPE FocusHook(
    MIL_INT HookType,
    MIL_INT Position,
    void *UserDataPtr
)
```

Parameters:

HookType

Indicates whether the optimum focus position was found or whether a new position needs to be analyzed.

For determining whether the optimum focus position was found or a new position needs to be analyzed

<input type="checkbox"/> Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ed/xcl (f)	helios ed/xd (g)	helios i1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xcl (u)	solios gige (v)	vio (w)
<input type="checkbox"/> M_CHANGE	Specifies that the focus position will be set to the Position parameter.	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ON_FOCUS	Specifies that the optimum focus position was found and is specified by the Position parameter.	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Position

Focus position.

UserDataPtr

Pointer to data.

Upon successful completion, the hook function should return **M_NULL**. Otherwise, the hook function should return **M_STOP_FOCUS** to abort the **MdigFocus()** operation. In this case, the focus position returned by **MdigFocus()** will be undetermined.

UserDataPtr

Sets the address of the data that you want to make available to the user-defined function.

For specifying the address of the data

<input type="checkbox"/> Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ed/xcl (f)	helios ed/xd (g)	helios i1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xcl (u)	solios gige (v)	vio (w)
<input type="checkbox"/> M_DEFAULT	Specifies that the function does not require data.	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> Value	Specifies the address of the data.	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

MinPosition

Sets the minimum focus position of the search.

For specifying the minimum focus

<input type="checkbox"/> Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ed/xcl (f)	helios ed/xd (g)	helios i1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xcl (u)	solios gige (v)	vio (w)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> Value	Specifies the position, in lens motor steps.	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
Value	Description																							
M_DEFAULT +	Same as M_SMART_SCAN.	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_BISECTION +	Specifies that the bisection search strategy will be used. This strategy breaks down the given positional range, step-by-step, until it finds the optimum focus position. You must specify a combination value from the table below. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_EVALUATE +	Returns the focus indicator value for the image passed instead of finding the optimum focus position. To grab an image at the current lens position into DestImageBufId and evaluate this image, pass a valid digitizer identifier to DigId. Note that in this processing mode, the user-defined function, specified by the FocusHookPtr parameter, is not called. You must specify a combination value from the table below. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_REFOCUS +	Specifies that the refocus search strategy will be used. This strategy scans upward or downward until it finds the optimum focus position or until it reaches the minimum or maximum position. You must specify a combination value from the table below. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_SCAN_ALL +	Specifies that the scan_all search strategy will be used. This strategy scans, by 1, all positions between the minimum and maximum and returns the position which produced the highest focus indicator. You must specify a combination value from the table below. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_SMART_SCAN +	Specifies that the smart scan search strategy will be used. This strategy performs three refocus searches, each with a smaller positional increment. You must specify a combination value from the table below. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constant for M_REFOCUS; M_SMART_SCAN;

You can add the following value to the above-mentioned values to specify the default number of positions to verify a peak.

For M_REFOCUS or M_SMART_SCAN																								
Value	Description	corona-II (a)	cronoplus (b)	gpu vision (c)	gpu processing (d)	helios ea/xa (e)	helios ed/xd (f)	helios ed/xd (g)	helios i394_iildc (h)	iris (i)	met-II/d (j)	met-II/dlg (k)	met-II/mc (l)	met-II/std (m)	morphis (n)	morphis qxt (o)	morphis qxt (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ec/xd (u)	solos gige (v)	solos gige (w)
Value	Specifies the required number of positions. The default value is 2. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constants for any of the possible values of the [ProcMode](#) parameter

You must add one or more of the following values to the above-mentioned values to specify which images to skip.

For the ProcMode parameter	
Value	Description
coron	corona
cronc	cron
glge	gl
gpu	gpu
hello	hello
hello	hello
hello	hello
leee	leee
lirs (lirs (
met-	met-
met-	met-
met-	met-
met-	met-
morf	morf
morf	morf
nextis	nextis
odys	odys
odys	odys
solio	solio
solio	solio
solio	solio

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

ResultPtr

Specifies the address in which to return the optimum focus position or the focus indicator value, depending on the mode of operation.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigFree

Synopsis

Free a digitizer.

Syntax

```
void MdigFree(  
    MIL_ID DigId  
)
```

Description

This function deallocates a digitizer previously allocated with [MdigAlloc\(\)](#).

Parameter

DigId
Specifies the identifier of the digitizer.

Remark

- If you are creating a DLL that includes a call to **MdigFree()**, ensure that the call is not made from the **DllMain()** function, because **MdigFree()** might unload any DLL loaded with [MdigAlloc\(\)](#) and you cannot unload a DLL from **DllMain()**. If necessary, call **MdigFree()** from a clean-up function in your DLL instead.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigGetHookInfo

Synopsis

Get information about a digitizer hook event.

Syntax

```
MIL_INT MdigGetHookInfo(
    MIL_ID EventId,
    MIL_INT InfoType,
    void *UserVarPtr
)
```

Description

This function allows you to get information about the event that caused the hook-handler function to be called. **MdigGetHookInfo()** should only be called within the scope of a digitizer hook-handler function.

[Matrox Solios GigE]

Note that the following events (information types) are only available if they are supported by the camera. Refer to your camera's documentation for more details.

Parameters

EventId

Specifies the digitizer event identifier received by the hook-handler function (see [MdigProcess\(\)](#), [MdigHookFunction\(\)](#) and [MdigControl\(\)](#)). Note that this value cannot be set to NULL in the hook-handler function.

InfoType

Specifies a combination of two values: the event type and the type of information about the event to return. Set this parameter to the one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

[illegible]

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigGrab

Synopsis

Grab data from a camera into a buffer.

Syntax

```
void MdigGrab(
    MIL_ID DigId,
    MIL_ID DestImageBufId
)
```

Description

This function uses the specified digitizer to acquire a frame of data and stores this data in the destination image buffer.

When acquiring data from a line-scan type of camera, the exact number of rows grabbed is determined by the DCF.

In all cases, if the grab destination buffer is smaller than the digitizer's frame size, the buffer will be filled and all other data will be lost. If the grab destination buffer is larger than the digitizer's frame size, the buffer will be filled up to the digitizer's frame size. The rest of the buffer will remain untouched.

When acquiring data from an interlaced camera, both the odd and even fields are grabbed unless otherwise set with [MdigControl\(\)](#) using [M_GRAB_FIELD_NUM](#).

You can use [MdigGrabContinuous\(\)](#) to continuously acquire frames of data, or [MdigProcess\(\)](#) for its multiple buffering capabilities. Refer to [MbufAlloc...\(\)](#) for potential restrictions relating to your destination buffer.

Parameters

DigId


Specifies the identifier of the digitizer.

DestImageBufId

Specifies the identifier of the destination image buffer.

This parameter should be set to one of the following values:

For the identifier of the destination image buffer																											
Value	Description	corona-II (a) cronosplus (b) gige vision (c) gpu processing (d) helios ea/Xa (e) helios ecd/Xd (f) helios ecd/Xd (g) ieee 1394 i1dc (h) iris (i) met-II /cl (j) met-II /dig (k) met-II /mc (l) met-II /std (m) morphis (n) morphis qxt (o) nexis (p) odyssey ea/Xa (q) odyssey ecd/Xcl (r) odyssey ecd/Xcl (s) solios ea/Xa (t) solios ecd/Xcl (u) solios gige (v) vio (w)																									
MIL color image buffer identifier	Specifies the identifier of a color destination image buffer. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
	Board specific																										
	The width and the horizontal position of the grab destination buffer and the grab region (set with <code>MdigControl()</code> using <code>M_SOURCE_SIZE_X</code> and <code>M_SOURCE_SIZE_Y</code> , or <code>M_SOURCE_OFFSET_X</code> and <code>M_SOURCE_OFFSET_Y</code>) must be a multiple of 4 bytes. If not, the grab will be clipped accordingly.	a									j	k	l	m													
	The grab buffer's depth can only be 8 bits.									i					n	o	p					t	u		w		

	The grab buffer's depth can be 8 or 16 bits.			c		e	f	g			j	k											v	
	The width of the grab destination buffer must be at least 16 bytes. The height must be at least 4 lines. If not, an error will occur.		b																					
	The width of the grab destination buffer must be at least 16 bytes. If not, an error will occur.		a										l	m										
	The grab buffer's depth can be 8, 16 or 32 bits.																q	r	s					
	The width of the grab buffer must be at least 17 bytes. If not, an error will occur.																				t	u	v	
 MIL monochrome image buffer identifier	Specifies the identifier of a monochrome destination image buffer. It is not possible to grab into a color-band child buffer when the buffer is packed. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	The width and the horizontal position of the grab destination buffer and the grab region (set with <code>MdigControl()</code> using <code>M_SOURCE_SIZE_X</code> and <code>M_SOURCE_SIZE_Y</code> , or <code>M_SOURCE_OFFSET_X</code> and <code>M_SOURCE_OFFSET_Y</code>) must be a multiple of 4 bytes. If not, the grab will be clipped accordingly.	a									j	k	l	m										
	When using a color camera, the color data is first converted into YUV, and then the Y (luminance) component is stored in the buffer. The color space conversion is done in the hardware.	a	b			e	f	g		i					n	o	p	q	r	s	t	u	v	w
	When using a color camera, the red (alpha) band is grabbed.										j	k												
	When using a color camera, the green band is grabbed.											l												
	The grab buffer's depth can only be 8 bits.									i					n	o	p							w
	The grab buffer's depth can be 8 or 16 bits.	a		c		e	f	g			j	k										t	u	v
	The grab buffer's depth can be 8, 16 or 32 bits.																q	r	s					
	The width of the grab destination buffer must be at least 16 bytes. The height must be at least 4 lines. If not, an error will occur.		b																					
	The width of the grab destination buffer must be at least 16 bytes. If not, an error will occur.	a										l	m											
	The DCF must also be monochrome.							g																
	The width of the grab buffer must be at least 17 bytes. If not, an error will occur.																					t	u	v

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigGrabContinuous

Synopsis

Grab data continuously from a camera.

Syntax

```
void MdigGrabContinuous(
    MIL_ID DigId,
    MIL_ID DestImageBufId
)
```

Description

This function uses the specified digitizer to continuously acquire frames of data and store this data in the destination image buffer, until `MdigHalt()` is called. If the destination buffer is selected to the display, the display is continuously updated, however only the last frame grabbed is written to the specified buffer.

When acquiring data from a line-scan type of camera, the exact number of rows grabbed is determined by the DCF.

In all cases, if the grab destination buffer is smaller than the digitizer's frame size, the buffer will be filled and all other data will be lost. If the grab destination buffer is larger than the digitizer's frame size, the buffer will be filled up to the digitizer's frame size. The rest of the buffer will remain untouched.

If you need to save/process each grabbed frame, you should perform the grab using `MdigGrab()`; alternatively, you can use `MdigProcess()` for its multiple buffering capabilities. Refer to `MbufAlloc...()` for potential restrictions relating to your destination buffer.

Parameters

DigId
Specifies the identifier of the digitizer.

DestImageBufId
Specifies the identifier of the destination image buffer.

This parameter should be set to one of the following values:

For the identifier of the destination buffer image																											
Value	Description	corona-11 (a)	cronosplus (b)	glige vision (c)	gpu processing (d)	hellios ea/xa (e)	hellios eci/xc1 (f)	hellios ed/xd (g)	ieee 1394 i4dc (h)	iris (i)	met-11 /cd (j)	met-11 /dlig (k)	met-11 /fmc (l)	met-11 /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (b)	odyssey eci/xc1 (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios eci/xc1 (u)	solios glige (v)	vio (w)			
MIL color image buffer identifier	Specifies the identifier of a color destination image buffer. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
	Board specific																										
	The width and the horizontal position of the grab destination buffer and the grab region (set with <code>MdigControl()</code> using <code>M_SOURCE_SIZE_X</code> and <code>M_SOURCE_SIZE_Y</code> , or <code>M_SOURCE_OFFSET_X</code> and <code>M_SOURCE_OFFSET_Y</code>) must be a multiple of 4 bytes. If not, the grab will be clipped accordingly.	a									j	k	l	m													
	The grab buffer's depth can only be 8 bits.									i					n	o	p				t	u		w			

	The grab buffer's depth can be 8 or 16 bits.		c	e	f	g		j	k														v	
	The grab buffer's depth can be 8, 16 or 32 bits.																q	r	s					
	The width of the grab destination buffer must be at least 16 bytes. The height must be at least 4 lines. If not, an error will occur.	b																						
	The width of the grab destination buffer must be at least 16 bytes. If not, an error will occur.	a											l	m										
	The width of the grab buffer must be at least 17 bytes. If not, an error will occur.																					t	u	v
<input type="checkbox"/> MIL monochrome image buffer identifier	Specifies the identifier of a monochrome destination image buffer. It is not possible to grab into a color-band child buffer when the buffer is packed. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	The width and the horizontal position of the grab destination buffer and the grab region (set with <code>MdigControl()</code> using <code>M_SOURCE_SIZE_X</code> and <code>M_SOURCE_SIZE_Y</code> , or <code>M_SOURCE_OFFSET_X</code> and <code>M_SOURCE_OFFSET_Y</code>) must be a multiple of 4 bytes. If not, the grab will be clipped accordingly.	a									j	k	l	m										
	When using a color camera, the color data is first converted into YUV, and then the Y (luminance) component is stored in the buffer. The color space conversion is done in the hardware.	a	b			e	f	g		i				m	n	o	p	q	r	s	t	u	v	w
	When using a color camera, the red (alpha) band is grabbed.										j	k												
	When using a color camera, the green band is grabbed.												l											
	The grab buffer's depth can only be 8 bits.									i					n	o	p							w
	The grab buffer's depth can be 8 or 16 bits.			c		e	f	g			j	k										t	u	v
	The grab buffer's depth can be 8, 16 or 32 bits.																	q	r	s				
	The width of the grab destination buffer must be at least 16 bytes. The height must be at least 4 lines. If not, an error will occur.	b																						
	The width of the grab destination buffer must be at least 16 bytes. If not, an error will occur.	a											l	m										
	The DCF must also be monochrome.						g																	
	The width of the grab buffer must be at least 17 bytes. If not, an error will occur.																					t	u	v

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigGrabWait

Synopsis

Wait for the end of the grab in progress.

Syntax

```
void MdigGrabWait (
    MIL_ID DigId,
    MIL_INT ControlFlag
)
```

Description

This function allows you to temporarily override the grab mode on the specified digitizer (see [MdigControl\(\)](#) with [M_GRAB_MODE](#)).

Using this function forces the grab to wait until the grab timeout value has expired (set using [MdigControl\(\)](#) with [M_GRAB_TIMEOUT](#)). Note that if the grab timeout is set to infinite, the grab will never end and this function will wait indefinitely. If the grabbed frame is not returned within the period of the timeout, an error is generated.

Parameters

DigId

The [DigId](#) parameter specifies the identifier of the digitizer.

ControlFlag

Specifies the function's control flag. This parameter must be set to one of the following:

For specifying the function's control flag																									
<input type="checkbox"/> Value	Description																								
<input type="checkbox"/> M_GRAB_END	Wait for the end of all queued grabs. This value should not be used when grabbing data with MdigGrabContinuous() . (summarize)																								
<input type="checkbox"/> M_GRAB_FRAME_END	Waits for the end of the current grab.																								
	corona-II (a)	gige vision (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ecl/xd (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II/cl (j)	met-II/dig (k)	met-II/mc (l)	met-II/std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ecl/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecl/xd (u)	solios gige (v)	vio (w)		
	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigHalt

Synopsis

Halt a continuous grab from an input device.

Syntax

```
void MdigHalt (
    MIL_ID DigId
)
```

Description

This function stops the specified digitizer from grabbing data. It should be used when performing a continuous grab with [MdigGrabContinuous\(\)](#).

This function will wait for the end of the current frame before returning, to ensure the last frame is always valid.

Parameter

DigId
Specifies the identifier of the digitizer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigHookFunction

Synopsis

Hook a function to a digitizer event.

Syntax

```
void MdigHookFunction(
    MIL_ID DigId,
    MIL_INT HookType,
    MIL_DIG_HOOK_FUNCTION_PTR HookHandlerPtr,
    void *UserDataPtr
)
```

Description

This function allows you to attach or detach a user-defined function to a specified digitizer event. Once a hook-handler function is defined and hooked to an event, it is automatically called when the event occurs.

You can hook more than one function to an event by making separate calls to **MdigHookFunction()** for each function that you want to hook. MIL automatically chains and keeps an internal list of all these hooked functions. When a function is hooked, this new function is added to the end of the list. When the event happens, all user-defined functions in the list will be executed in the same order that they were hooked to the event. You can also remove any function from the list; in this case, MIL preserves the order of the remaining functions in the list.

You can obtain more information about the event from within the hook handler function using [MdigGetHookInfo\(\)](#).

Note that functions hooked to an event execute on a distinct thread. This permits the functions to run asynchronously from the operation that fired the event and from functions hooked to other events. Although there is a small queue to permit a certain amount of overlap, hooked functions should not take longer to execute than the period in which two of their associated events can occur. Typically, a hooked function performs the minimum number of operations required and, if necessary, performs longer processes by launching other threads.

[Matrox IEEE 1394 IIDC driver; Matrox Solios GigE]
Note that the following hook types are only available if they are supported by the camera. Refer to your camera's documentation for more details.

Parameters

DigId

Specifies the identifier of the digitizer.

HookType

Specifies the digitizer event to which to hook the function. This parameter can be set to one of the following values:

For specifying the digitizer event to which to hook the function															
Value	Description	corona-II (a)	coronaplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios edl/xd (f)	helios edl/xd (g)	ieee 1394 iidc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mnc (l)	met-II /std (m)	vio (w)
<input checked="" type="checkbox"/> M_CAMERA_LOCK +	Hooks the function to the event that occurs when the camera is locked. Use MdigInquire() with M_CAMERA_LOCKED to inquire if the camera is currently locked. (summarize)	a			e								n	o	w
<input checked="" type="checkbox"/> M_CAMERA_PRESENT +	Hooks the function to the presence of the camera. Use MdigInquire() with M_CAMERA_PRESENT to inquire if the camera is currently present. (summarize)	a	c		e			h					n	o	w

[illegible]

Upon successful completion, the hook-handler function should return **M_NULL**. Note, MFTYPE and MIL_DIG_HOOK_FUNCTION_PTR are reserved MIL predefined types for functions and data pointers.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
Note that a valid event identifier is available only when creating a hook on the end of a grab (that is, with the [HookType](#) parameter set to [M_GRAB_END](#)).

UserDataPtr

Specifies the address of the user data that you want to make available to the hook-handler function. This address is passed to the hook-handler function, through its [UserDataPtr](#) parameter, when the specified event occurs. Set this parameter to **M_NULL** if not used.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigInquire

Synopsis

Inquire about a digitizer setting.

Syntax

```
MIL_INT MdigInquire(  
    MIL_ID DigId,  
    MIL_INT64 InquireType,  
    void *UserVarPtr  
)
```

Description

This function allows you to inquire about various digitizer settings.

Note that you can use [MdigControl\(\)](#) to inquire specific digitizer settings.

[Matrox IEEE 1394 IIDC driver; Matrox Solios GigE]
The inquire types described in this function can inquire either the connected camera or the Matrox imaging board. In situations where the inquire type is available on both the camera and the board, you can use the [M_CAMERA](#) or the [M_BOARD](#) combination constant to specify which to inquire.

When the inquire type is only available on the camera, the standard feature name is provided as a board-specific statement within the inquire type description. If your camera does not support the standard feature, an error is generated.

When the inquire type is only available on the Matrox imaging board, it is handled automatically by your Matrox imaging board. When the inquire type is available on both the camera and the board, it is handled automatically by your camera. If the camera does not support the standard feature, an error is generated. In this case, to use the inquire type with the board, you must use the [M_BOARD](#) combination constant.

Parameters

DigId

Specifies the identifier of the digitizer.

InquireType

Specifies the digitizer setting to inquire. This parameter can be set to one of the following values:

The following inquire types allow you to inquire various digitizer settings.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

For inquiring the general digitizer settings																
Value	Description	corona-II (a)	coronoplus (b)	gige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ecd/Xcl (f)	helios ed/Xd (g)	ieee 1394 iIIC (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)
M_BAYER_COEFFICIENTS_ID	Returns a MIL identifier of the internal buffer containing the white balance coefficients used when grabbing from a camera with a Bayer color filter. Note that this buffer should not be freed. (summarize)			c			f	g		i					p	r s

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

	Returns the grab synchronization with the Host. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<div>UserVarPtr info</div> <div>Return values: M_ASYNCHRONOUS; M_ASYNCHRONOUS_QUEUED; M_SYNCHRONOUS; (details)</div> <div>Board specific</div> <div>Note that this inquire value is only available to inquire the Matrox Imaging board.</div>								h															
<div><div></div>M_GRAB_PERIOD</div>	Returns the duration of a frame (as specified in the DCF), in msec. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<div>Board specific</div> <div>Note that this inquire type is GenICam camera specific. The camera must support the AcquisitionFroms and RateAbs GenICam standard features for this inquire type to be supported.</div>			c																			v	
<div><div></div>M_GRAB_SAMPLING_POSITION</div>	Returns the delay applied to the pixel clock signal to offset it a bit. (summarize)												l											
	<div>UserVarPtr info</div> <div>Return values: 0 to 255; (details)</div>																							
<div><div></div>M_GRAB_SCALE_X +</div>	Returns the horizontal scaling factor. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> <div>Return values: 1; M_FILL_DESTINATION; Value = 1.0/n; Value > 1; (details)</div> <div>Board specific</div> <div>Note that, by default, this inquire type is GenICam camera specific. The camera must support the DecimationHorizontal GenICam standard feature for this inquire type to be available on the camera.</div> <div>Note that this inquire type is also available on the Matrox Imaging board.</div> <div>By default, this inquire type is IIDC camera specific. The camera must support the ability to scale grabbed data for this inquire type to be available on the camera. This inquire type is also available on the board.</div>			c																		v		
																							v	
									h															
<div><div></div>M_GRAB_SCALE_Y +</div>	Returns the vertical scaling factor. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> <div>Return values: 1; M_FILL_DESTINATION; Value = 1.0/n; Value > 1; (details)</div> <div>Board specific</div> <div>Note that, by default, this inquire type is GenICam camera specific. The camera must support the DecimationVertical GenICam standard feature for this inquire type to be available on the camera.</div> <div>Note that this inquire type is also available on the Matrox Imaging board.</div> <div>By default, this inquire type is IIDC camera specific. The camera must support the ability to scale grabbed data for this inquire type to be available on the camera. This inquire type is also available on the board.</div>			c																		v		
																							v	
									h															
<div><div></div>M_GRAB_START_MODE</div>	Returns the type of field on which to grab. (summarize)	a	b			e	f	g			j	k	l	m	n	o		q	r	s	t	u		w
	<div>UserVarPtr info</div> <div>Return values: M_FIELD_START; M_FIELD_START_EVEN;</div>																							

	M_FIELD_START_ODD; (details)	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_GRAB_TIMEOUT</div>	Returns the maximum time to wait for a frame before generating an error, in msecs. (summarize)																						
	UserVarPtr info Return values: M_INFINITE; Value > 0; (details)																						
	Board specific																						
	Note that this inquire value is only available to inquire the Matrox Imaging board.							h															
<div><div></div>M_GRAB_WAIT</div>	Returns whether the grab of a digitizer is to wait and start upon a grab of a second digitizer (MIL digitizer identifier), or not (<i>M_NULL</i>). (summarize)																q	r	s				
	UserVarPtr info Return values: MIL digitizer identifier; M_NULL; (details)																						
<div><div></div>M_HOOK_MASTER_THREAD_HANDLE</div>	Returns the handle of the main interrupt thread that dispatches the user-defined functions attached to a specified digitizer event to their respective threads.	a	b									l	m										
<div><div></div>M_HOOK_MASTER_THREAD_ID</div>	Returns the identifier of the main interrupt thread that dispatches the handling of the user-defined functions attached to a specified digitizer event to their respective threads.	a	b									l	m										
<div><div></div>M_HUE +</div>	Returns the color phase of the picture. Note that this inquire type is IIDC camera specific. The camera must support the HUE_INQ IIDC standard feature for this inquire type to be supported. (summarize)							h															
	UserVarPtr info Return values: Please see MdigControl() with M_HUE . (details)																						
<div><div></div>M_HUE_REF</div>	Returns the digitizer hue reference level. Note that this inquire type is not supported for monochrome cameras. (summarize)		b					h					m	n	o								w
	UserVarPtr info Return values: M_MAX_LEVEL; M_MIN_LEVEL; M_MIN_LEVEL <= Value <= M_MAX_LEVEL; (details)																						
	Board specific																						
	This is only supported on Matrox Vio /Analog DCF. Note that this inquire type is IIDC camera specific. The camera must support the HUE_INQ IIDC standard feature for this inquire type to be supported.							h															w
<div><div></div>M_INIT_FLAG</div>	Returns the digitizer initialization flag. (summarize)	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	UserVarPtr info Return values: M_DEFAULT; (details)																						
<div><div></div>M_INPUT_FILTER +</div>	Returns the low-pass filter applied to incoming data for the specified acquisition path(s) used by the specified digitizer. (summarize)	a			e												q			t			
	UserVarPtr info Return values: M_BYPASS; M_LOW_PASS_0; M_LOW_PASS_1; M_LOW_PASS_2; (details)																						
	Board specific																						
	For more information regarding the frequencies of low-pass filters, refer to the Installation and hardware reference manual for your Matrox imaging board.				e													q			t		

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

<div><div></div><div>M_SIZE_BIT</div></div>	<div>Returns the depth per band, in bits. (summarize)</div> <div><div>Board specific</div><div>This will be either 8-bit or 16-bit. Note that the 1394-compliant camera must send the data in packets that are a multiple of 8-bits in size.</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_SIZE_X</div></div>	<div>Returns the width of the image, in pixels. (summarize)</div> <div><div>Board specific</div><div>Note that this inquire type is GenICam camera specific. The camera must support the Width GenICam standard feature for this inquire type to be available on the camera.</div><div>Note that this inquire type is also available on the Matrox Imaging board.</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_SIZE_Y</div></div>	<div>Returns the height of the image, in pixels. (summarize)</div> <div><div>Board specific</div><div>Note that this inquire type is GenICam camera specific. The camera must support the Height GenICam standard feature for this inquire type to be available on the camera.</div><div>Note that this inquire type is also available on the Matrox Imaging board.</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_SOURCE_COMPENSATION +</div></div>	<div>Returns whether source compensation for cropping an input signal capture window is enabled or disabled. Note that this inquire value is only available to inquire the Matrox Imaging board. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values: M_DISABLE; M_ENABLE; (details)</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_SOURCE_OFFSET_X +</div></div>	<div>Returns the X-offset of the input-signal capture window. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values: Please see MdigControl() with M_SOURCE_OFFSET_X. (details)</div><div>Board specific</div><div>By default, this inquire type is IIDC camera specific. The camera must support the format 7 video mode and the IMAGE_POSITION IIDC standard feature for this inquire type to be available on the camera. This inquire! type is also available on the board.</div><div>Note that, by default, this inquire type is GenICam camera specific. The camera must support the OffsetX GenICam standard feature for this inquire type to be available on the camera.</div><div>Note that this inquire type is also available on the Matrox Imaging board.</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_SOURCE_OFFSET_Y +</div></div>	<div>Returns the Y-offset of the input-signal capture window. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values: Please see MdigControl() with M_SOURCE_OFFSET_Y. (details)</div><div>Board specific</div><div>By default, this inquire type is IIDC camera specific. The camera must support the format 7 video mode and the IMAGE_POSITION IIDC standard feature for this inquire type to be available on the camera. This inquire type is also available on the board.</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

[illegible]

[illegible]

Combination constants for `M_BLACK_REF`; `M_WHITE_REF`;

You can add one of the following values to the above-mentioned values to get information about a specific acquisition path used by the digitizer.

[Matrox Vio]
This is only supported on Matrox Vio /Analog DCF.

- For specifying which acquisition path is returned

[illegible]

Combination constant for M_BLACK_REF; M_WHITE_REF;

You can add the following value to the above-mentioned values to determine the reference level in Volts.

[Matrox Vio]

This is only supported on Matrox Vio /Analog DCF.

● For specifying the black and white reference level in Volts

Value	Description	vio (w)	solios gige (v)	solios eci/xd (u)	solios ea/xa (t)	odyssey ed/xd (s)	odyssey eci/xd (r)	odyssey ea/xa (q)	nexis (p)	morphis qxt (o)	morphis (n)	met-II/std (m)	met-II/mc (l)	met-II/dig (k)	met-II/cl (j)	iris (i)	ieee 1394 i1dc (h)	helios ed/xd (g)	helios eci/xd (f)	helios ea/xa (e)	gpu processing (d)	gige vision (c)	cronosplus (b)	corona-II (a)
M_VOLTAGE	Specifies the reference level in Volts. (summarize)	w			t			q												e				
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE																							

Combination constant for [M_CAMERA_LOCK](#);

You can add the following value to the above-mentioned value to determine whether ultra-fast camera locking is enabled.

● For locking the camera

Value	Description	vio (w)	solios gige (v)	solios eci/xd (u)	solios ea/xa (t)	odyssey ed/xd (s)	odyssey eci/xd (r)	odyssey ea/xa (q)	nexis (p)	morphis qxt (o)	morphis (n)	met-II/std (m)	met-II/mc (l)	met-II/dig (k)	met-II/cl (j)	iris (i)	ieee 1394 i1dc (h)	helios ed/xd (g)	helios eci/xd (f)	helios ea/xa (e)	gpu processing (d)	gige vision (c)	cronosplus (b)	corona-II (a)
M_FAST	Returns whether ultra-fast camera locking is enabled.									n														

Combination constant for [M_DIGITIZER_TYPE](#);

You must add the following value to the above-mentioned value to get the frame grabber(s) available to allocate a digitizer.

● For specifying the frame grabber(s) available to allocate a digitizer

Value	Description	vio (w)	solios gige (v)	solios eci/xd (u)	solios ea/xa (t)	odyssey ed/xd (s)	odyssey eci/xd (r)	odyssey ea/xa (q)	nexis (p)	morphis qxt (o)	morphis (n)	met-II/std (m)	met-II/mc (l)	met-II/dig (k)	met-II/cl (j)	iris (i)	ieee 1394 i1dc (h)	helios ed/xd (g)	helios eci/xd (f)	helios ea/xa (e)	gpu processing (d)	gige vision (c)	cronosplus (b)	corona-II (a)
M_DEVn	Specifies the number of the acquisition path to inquire, where n either 0 or 1 and represents the acquisition path number. Note that if n is greater than the number of acquisition paths available, an error is generated. To inquire the number of frame grabbers inside your Matrox 4SightM, use MsysInquire() with M_DIGITIZER_TYPE + M_DEVn . (summarize)							p																

The following inquire types allow you to inquire the input gain.

The following values require that you pass the value listed in the data-type area of the specified parameter.

● For inquiring the input gain

[illegible]

● For inquiring user-defined signals																									
Value	Description	corona-II (a)	cronosplus (b)	gpu vision (c)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ieee 1394 i/dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ec/xd (u)	solos gige (v)	vio (w)		
M_USER_BIT_COUNT	Returns the total number of user-defined signals. This count includes the number of input, output, and I/O user-defined signals. (summarize)	a	c		e	f	g			j	k	l	m				q	r	s	t	u	v			
	<i>UserVarPtr info</i> Return values: ValueTotal number of user-defined signals.																								
	<i>Board specific</i>																								
	Note that this inquire type is GenICam camera specific. The camera must support the LineSelector GenICam standard feature for this inquire type to be supported.		c																				v		
M_USER_BIT_COUNT_IN	Returns the number of user-defined signals that can be used for input. This count includes the number of input and I/O user-defined signals. (summarize)				e	f	g			j	k						q	r	s	t	u	v			
	<i>UserVarPtr info</i> Return values: ValueNumber of user-defined signals that can be used for input.																								
	<i>Board specific</i>																								
	Note that this inquire value is only available to inquire the Matrox Imaging board.																						v		
M_USER_BIT_COUNT_OUT	Returns the number of user-defined signals that can be used for output. This count includes the number of output and I/O user-defined signals. (summarize)				e	f	g			j	k						q	r	s	t	u	v			
	<i>UserVarPtr info</i> Return values: ValueNumber of user-defined signals that can be used for output.																								
	<i>Board specific</i>																								
	Note that this inquire value is only available to inquire the Matrox Imaging board.																						v		
M_USER_BIT_FORMAT +	Returns the type of transmitter/receivers enabled for the specified user-defined input signal, on systems whose transmitter/receivers are enabled through software and where the option of two or more signal types are possible. Refer to M_USER_IN_FORMAT and M_USER_OUT_FORMAT to change formats on other Matrox Imaging boards. (summarize)		c		e	f	g										q	r	s	t	u	v			
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; M_LVDS; M_OPTO; M_TRI_STATE; M_TTL; (details)																								
	<i>Board specific</i>																								
	Note that, by default, this inquire type is GenICam camera specific. The camera must support the LineFormat and the LineSelector GenICam standard features for this inquire type to be available on the camera.		c																				v		

M_USER_BIT_INTERRUPT_MODE +	Returns the type of functional-state change upon which to generate an interrupt. Note that this only applies to user-defined input signals. (summarize)	e f g	q r s t u v
	UserVarPtr info Return values: M_ANY_EDGE; M_DEFAULT; M_EDGE_FALLING; M_EDGE_RISING; (details)		
	Board specific		
	Note that this inquire value is only available to inquire the Matrox Imaging board.		v
M_USER_BIT_INTERRUPT_STATE +	Returns whether to generate an interrupt when the functional state for the user-defined signal changes. Note that this only applies to user-defined input signals. (summarize)	e f g	q r s t u v
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)		
	Board specific		
	Note that this inquire value is only available to inquire the Matrox Imaging board.		v
M_USER_BIT_MODE +	Returns the mode of the specified I/O user-defined signal. (summarize)	c e f g	q r s t u
	UserVarPtr info Return values: M_INPUT; M_OUTPUT; (details)		
	Board specific		
	Note that, by default, this inquire type is GenICam camera specific. The camera must support the LineMode and the LineSelector GenICam standard features for this inquire type to be available on the camera.	c	
M_USER_BIT_VALUE +	Returns the current state of the specified user-defined signal(s). Note that if there is no device connected to the pin associated with the specified user-defined signal, indeterminate results will result when that user-defined signal is inquired. (summarize)	a c e f g j k l m	q r s t u v
	UserVarPtr info Return values: Bit-encoded value; M_OFF; M_ON; (details)		
	Board specific		
	Note that, by default, this inquire type is GenICam camera specific. The camera must support the LineStatus and LineSelector GenICam standard features for this inquire type to be available on the camera.	c	v
	Note that this inquire type is also available on the Matrox Imaging board.		v
M_USER_IN_FORMAT +	Returns the type of the receivers for triggers and user-defined input signals on the specified acquisition path(s), on systems whose receivers are enabled through software and where the option of both TTL and RS-422/LVDS are possible. For a listing of available user-defined or auxiliary signals, see the Connectors and signal name section of the MIL Board-specific Notes chapter for your Matrox imaging board. Refer to M_USER_BIT_FORMAT , and M_GRAB_TRIGGER_FORMAT to inquire formats on other Matrox Imaging boards. (summarize)	j k	
	UserVarPtr info Return values: M_DISABLE; M_LVDS; M_TTL; (details)		
M_USER_OUT_FORMAT +	Returns the type of the exposure or user-defined output signal(s) on the specified acquisition path(s), on systems whose I/O drivers are enabled through software and where the option of both TTL and RS-422/LVDS are possible. For a listing of available user-defined or auxiliary signals, see the Connectors and signal name section of the MIL Board-specific Notes chapter for your Matrox imaging board. Refer to M_USER_BIT_FORMAT , and M_GRAB_TRIGGER_FORMAT to inquire formats on other Matrox Imaging boards. (summarize)	j k	

The following inquire types specify the settings for inquiring exposure signals. For more information, see the [Grabbing with triggers and exposures](#) section in [Chapter 22: Grabbing with your digitizer](#).

	<p>third-party IEEE 1394 IIDC-compliant network boards. In addition, this inquire type is only available on the Matrox Imaging board, and not on the camera.</p> <p>Note that, by default, this inquire type is GenICam camera specific. The camera must support the TriggerSelector GenICam standard feature for this inquire type to be available on the camera.</p> <p>Note that this inquire type is also available on the Matrox Imaging board.</p>	a	c	e	f	g	h	i	j	k	l	p	q	r	s	t	u	v
M_GRAB_EXPOSURE_TIME +	Returns the time for the active portion of the exposure signal (that is, the exposure time). (summarize) UserVarPtr info Data type: MIL_DOUBLE Return values: Please see MdigControl() with M_GRAB_EXPOSURE_TIME. (details) Board specific This inquire type is only supported on Matrox 4Sight-M 1394b; it is not supported on third-party IEEE 1394 IIDC-compliant network boards. In addition, this inquire type is only available on the Matrox Imaging board, and not on the camera. Note that, by default, this inquire type is GenICam camera specific. The camera must support the ExposureTimeAbs GenICam standard feature for this inquire type to be available on the camera. Note that this inquire type is also available on the Matrox Imaging board.	a	c	e	f	g	h	i	j	k	l	p	q	r	s	t	u	v
M_GRAB_EXPOSURE_TIME_DELAY +	Returns the delay between the trigger and the active portion of the exposure signal. (summarize) UserVarPtr info Data type: MIL_DOUBLE Return values: Please see MdigControl() with M_GRAB_EXPOSURE_TIME_DELAY. (details) Board specific This inquire type is only supported on Matrox 4Sight-M 1394b; it is not supported on third-party IEEE 1394 IIDC-compliant network boards. In addition, this inquire type is only available on the Matrox Imaging board, and not on the camera. Note that, by default, this inquire type is GenICam camera specific. The camera must support the TriggerDelayAbs GenICam standard feature for this inquire type to be available on the camera. Note that this inquire type is also available on the Matrox Imaging board.	a	c	e	f	g	h	i	j	k	l	p	q	r	s	t	u	v
M_GRAB_EXPOSURE_TRIGGER_MISSED +	Returns the trigger activation mode for specified timer. (summarize) UserVarPtr info Return values: M_DISABLE; M_ENABLE; (details)			e	f	g										t	u	
M_GRAB_EXPOSURE_TRIGGER_MODE +	Returns the trigger activation mode for specified timer. (summarize) UserVarPtr info Return values: M_EDGE_FALLING; M_EDGE_RISING; (details) Board specific This inquire type is only supported on Matrox 4Sight-M 1394b; it is not supported on third-party IEEE 1394 IIDC-compliant network boards. In addition, this inquire type is only available on the Matrox Imaging board, and not on the camera. Note that, by default, this inquire type is GenICam camera specific. The camera must support the TriggerActivation and TriggerSelector GenICam standard features for this inquire type to be available on the camera. Note that this inquire type is also available on the Matrox Imaging board.	a	c	e	f	g	h		j	k	l	q	r	s	t	u	v	

M_UART_PARITY +	Returns the UART parity setting. (summarize)	a							j	k	l	m			q	r	s		
	UserVarPtr info Return values: M_DISABLE; M_EVEN; M_ODD; (details)																		
M_UART_READ_STRING_LENGTH +	Returns the number of bytes to be read when using MdigControl() with M_UART_READ_STRING . If this length is specified by M_UART_STRING_DELIMITER , the returned result will be M_DEFAULT . (summarize)	a							j	k	l	m			q	r	s		
	UserVarPtr info Return values: M_DEFAULT Returns that the length is limited by M_UART_STRING_DELIMITER . Value Returns the length of the string, in bytes.																		
M_UART_READ_STRING_MAXIMUM_LENGTH +	Returns the maximum length of the string to be read using M_UART_READ_STRING . (summarize)	a							j	k	l	m			q	r	s		
	UserVarPtr info Return values: Please see MdigControl() with M_UART_READ_STRING_MAXIMUM_LENGTH . (details) M_DEFAULT Returns the maximum length of the string, in bytes.																		
M_UART_SPEED +	Returns the baud rate of the UART. (summarize)	a							j	k	l	m			q	r	s		
	UserVarPtr info Return values: 600;1200;1800;2400;3600;4800;7200;9600;14400;19200;28800;38400;57600;76800;115200;230400; (details)																		
M_UART_STOP_BITS +	Returns the number of extra data bit(s) that are added to each character to indicate the end of a character. (summarize)	a							j	k	l	m			q	r	s		
	UserVarPtr info Return values: 1; 2; (details)																		
M_UART_STRING_DELIMITER +	Returns the character used to terminate strings of incoming or outgoing data. The delimiter is used but not sent when writing data with M_UART_WRITE_STRING ; it is read for incoming data with M_UART_READ_STRING . (summarize)	a							j	k	l	m			q	r	s		
	UserVarPtr info Return values: Please see MdigControl() with M_UART_STRING_DELIMITER . (details)																		
M_UART_TIMEOUT +	Returns the maximum time, in msec, to wait between each byte when reading incoming data. (summarize)	a							j	k	l	m			q	r	s		
	UserVarPtr info Return values: M_INFINITE; Value; (details)																		
M_UART_WRITE_STRING_LENGTH +	Returns the length of the string to be sent to the UART for transmission. (summarize)	a							j	k	l	m			q	r	s		
	UserVarPtr info Return values: Please see MdigControl() with M_UART_WRITE_STRING_LENGTH . (details)																		

The following inquire types allow you to inquire the audio acquisition channel.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_INT](#).

● For inquiring audio signals																			
Value	Description	coro	gige	gpu	hello	hello	hello	leee	met-	met-	met-	met-	next	odys	odys	solio	solio	solio	v/o (

[illegible]

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type MIL_TEXT_CHAR
- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT64

Specifies the address in which to write the requested information. Since the **MdigiInquire()** function also returns the requested information, you can set this parameter to **M_NULL**.

Return value

The returned value is the requested information, usually cast to a *MIL_INT*. For inquire types cast to *MIL_TEXT_CHAR*, the returned value is *M_NULL*.

Example

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The following example uses `M_USER_BIT_VALUE + M_BIT_MASK()`. The returned value is also a bit-encoded value defining the state of the specified user-defined signals as being on (1) or off (0). For example, to inquire the value of user-defined signals 9, 10, 11, and 12, use:

```
MdigInquire(MilDigitizer, M_USER_BIT_VALUE+M_BIT_MASK(0x1E00), &UserBits);
```

If 9 and 10 are set to 1 and 11 and 12 are set to 0, the return value will be 0x0600.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigInquireFeature

Synopsis

Inquires a feature of the camera.

Syntax

```
void MdigInquireFeature(
    MIL_ID DigitizerId,
    MIL_ID InquireFlag,
    MIL_TEXT_PTR FeatureName,
    MIL_INT FeatureDataType,
    void *FeatureValuePtr
)
```

Description

This function allows you to directly inquire various manufacturer-specific features of the camera. When used with a GiGE Vision camera, it allows you to directly inquire various manufacturer-specific features specified with the camera's device description file (XML). To control a camera's manufacturer-specific feature, use `MdigiControlFeature()`. Note that for a complete list of available feature names and associated possible values, refer to your camera's documentation.

Parameters

DigitizerId

Specifies the identifier of the digitizer on which to acquire about the feature. This parameter must be given a valid digitizer identifier.

InquireFlag

Specifies the function's inquire flag. This parameter must be set to one of the following:

● For specifying the function's inquire flag					
Value	Description	corona-II (a)	cromoplus (b)	glge vision (c)	vio (w)
M_DEFAULT	Returns the current value.			c	v
M_MAX	Returns the maximum value possible. Note that this value can only be used when FeatureData Type is set to M_TYPE_MIL_INT32, M_TYPE_MIL_INT64, or M_TYPE_DOUBLE. (summarize)			c	v
M_MIN	Returns the minimum value possible. Note that this value can only be used when FeatureData Type is set to M_TYPE_MIL_INT32, M_TYPE_MIL_INT64, or M_TYPE_DOUBLE. (summarize)			c	v

FeatureName

Sets the name of the camera feature to inquire.

For specifying the name of the feature	
Value	Description
	<p>solios gjege</p> <p>vio (w)</p> <p>solios ed/k/</p> <p>solios ea/k/</p> <p>odyssey ed</p> <p>odyssey ea</p> <p>odyssey ed</p> <p>odyssey ea</p> <p>nexis (p)</p> <p>met-II /ste</p> <p>met-II /cl</p> <p>met-II /d/c</p> <p>met-II /cl</p> <p>lree 1394</p> <p>helios ed/k/</p> <p>helios ed/</p> <p>helios ea/k/</p> <p>gpu proces</p> <p>glige vision</p> <p>cnosoplus</p> <p>corona-II</p>

(summarize)

FeatureValuePtr info
Data type: MIL_TEXT_PTR

FeatureValuePtr

Accepts the address of one of the following (see above for specifics on which is expected):

- bool
- M_NULL
- MIL_DOUBLE
- MIL_INT32
- MIL_INT64
- MIL_TEXT_PTR

Specifies the address of the variable in which to return the feature value.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigLut

Synopsis

Copy a LUT buffer to a digitizer's physical LUT.

Syntax

```
void MdigLut(
    MIL_ID DigId,
    MIL_ID LutBufId
)
```

Description

This function copies a LUT buffer to the specified digitizer's physical LUT. MIL uses the data format (DCF) of the digitizer to determine whether a physical LUT is supported. If it is not, an error is generated.

To create a LUT buffer, use **MbufAlloc...**() with an **M_LUT** attribute.

The following table indicates how the digitizer's physical LUT is configured.

Configuration of the digitizer's LUT	Determining factor
Number of entries in the digitizer's physical LUT	Data being grabbed (DCF)
Depth (8- or 16-bit)	LUT buffer
Number of bands	DCF & LUT buffer

The digitizer's physical LUT is typically configured to have the same number of components (bands) as either the LUT buffer or the data to be grabbed (determined by the digitizer's DCF), depending on which has more bands. The digitizer's physical LUT is also configured to have the same number of entries as the maximum possible value per band of the data to grab. The depth of a digitizer's physical LUT is configured to be either 8- or 16-bits per band, depending on if the LUT buffer depth is 8-bit or 16-bit, respectively. Cameras, however, are not so limited. When dealing with a 10- or 12-bit camera, use a 16-bit destination grab buffer and load the digitizer's physical LUT with the difference zero padded. Note that if the digitizer's physical LUT cannot support the depth of the LUT buffer, an error will occur.

If the destination grab buffer depth is larger than that of the digitizer's physical LUT, the destination grab buffer's least significant bits are set to zero when the data is grabbed. If the digitizer's physical LUT depth is greater than that of the destination grab buffer, the most-significant bits of the data (the non-zero values) are used when the data is grabbed.

To copy the data from a LUT buffer to the digitizer's physical LUT, the number of entries in the LUT buffer must match those of the digitizer's physical LUT.

LUT buffer data is loaded into the digitizer's physical LUT, as follows:

LUT buffer	Digitizer's physical LUT	Result
1 band	1 band	The LUT buffer is copied directly into the digitizer's physical LUT.
1 band	3 band	The LUT buffer is copied into each component of the digitizer's physical LUT.
3 band	1 band	The first band of the LUT buffer is copied into the digitizer's physical LUT.
3 band	3 band	Each of the LUT buffer's bands are copied into the corresponding component of the digitizer's physical LUT.

[Matrox Corona-II; Matrox Helios eA/XA; Matrox Helios eCL/XCL; Matrox Meteor-II / Camera Link; Matrox Meteor-II / Digital; Matrox Meteor-II / Multi-Channel; Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD; Matrox Solios eA/XA; Matrox Solios eCL/XCL]

Note that if 14-, 16-, or 32-bit data is grabbed, the digitizer's physical LUT is not used.

[Matrox Corona-II]

If 12-, 14-, or 16-bit digital data is grabbed, the digitizer's physical LUT is not available.

Regardless of the depth at which data is grabbed, YUV data bypasses the digitizer's physical LUT.

Parameters

DigId

Specifies the identifier of the digitizer.

LutBufId

Specifies the LUT buffer identifier. This parameter should be set to one of the following values:

● For specifying the LUT buffer																										
<div><input type="checkbox"/> Value</div>	Description	corona-11 (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xcl (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-11 /cl (j)	met-11 /dlg (k)	met-11 /mc (l)	met-11 /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	vio (w)		
<div><input type="checkbox"/> M_DEFAULT</div>	Specifies that the digitizer's physical LUT is not used.	a				e	f	g		i	j	k	l					q	r	s	t	u				
<div><input type="checkbox"/> MIL LUT buffer identifier</div>	Specifies a LUT buffer. (summarize)	a				e	f	g		i	j	k	l					q	r	s	t	u				
	<i>Board specific</i>																									
	When grabbing 8-bit data, the digitizer's physical LUT has 256 entries.	a					f	g		i	j	k	l						r	s		u				
	When grabbing 10-bit data, the digitizer's physical LUT has 1024 entries.	a																								
	When grabbing 10-bit data, the digitizer's physical LUT has 1024 entries. When grabbing 12-bit data, the digitizer's physical LUT has 4096 entries.					e	f	g			j							q	r	s	t	u				

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigProcess

Synopsis

Grabs a sequence of images and processes them with a user-defined function as they are grabbed.

Syntax

```
void MdigProcess(
    MIL_ID DigId,
    const MIL_ID *DestImageArrayPtr,
    MIL_INT ImageCount,
    MIL_INT Operation,
    MIL_INT OperationFlag,
    MIL_DIG_HOOK_FUNCTION_PTR HookHandlerPtr,
    void *UserDataPtr
)
```

Description

This function uses the specified digitizer to acquire images and store them sequentially in a list of buffers. It also hooks a user-defined function to the modification of any buffer in the specified list. So every time a new frame is grabbed in a buffer in the list, the user-defined function is called.

If set synchronously, **MdigProcess()** will block the thread until the list of sequential grabs has completed. If set asynchronously, the buffers are prepared, and then control is returned to the calling thread while, in the background, the grabs will begin.

The grabs can stop at the end of the list of buffers, after a predefined number of grabs occur, or when a stop instruction is explicitly given. Round-robin grabbing will occur when the number of grabs exceeds the number of buffers available. To grab until a stop instruction, you must call **MdigProcess()** with **M_START**; when you want to stop grabbing, you must call **MdigProcess()** with **M_STOP**. Note that in this case, if you do not issue a stop instruction, an endless loop is created.

When grabbing round-robin, if the average time it takes to process a frame is greater than the frame rate of a camera, frames will eventually be missed.

The buffers are filled with the grabbed data in the order in which they are stored in the list. The index of the buffer modified can be inquired using **MdigGetHookInfo()** with **M_MODIFIED_BUFFER** + **M_BUFFER_INDEX**. The MIL identifier of the buffer modified can be inquired using **MdigGetHookInfo()** with **M_MODIFIED_BUFFER** + **M_BUFFER_ID**.

Note that, by default, if a buffer modification event occurs while a buffer is being processed by the hooked user-defined function, the event is queued. When running on a computer with multiple CPUs, you can have the event handled by different instances of the user defined function running on different threads. To do so, use **MsysControl()** with **M_MODIFIED_BUFFER_HOOK_MODE** set to **M_MULTI_THREAD**.

When the specified digitizer is set to perform triggered grabs, the default behavior of the function is to wait for a trigger event before grabbing each frame. To grab all or a specific number of frames per trigger, set the **OperationFlag** parameter to **M_TRIGGER_FOR_FIRST_GRAB** and use the **Operation** parameter to specify how many frames to grab per trigger.

It is not recommended to change the digitizer settings of the digitizer currently being used when grabs are still pending.

Parameters

DigId

Specifies the identifier of the digitizer to be used to perform the grabs.

DestImageArrayPtr

Specifies the address of the array containing the identifiers of the buffers in which to place the grabbed images. The buffers should be of an appropriate type and depth to hold the grabbed images. To allocate each buffer, use **MbufAlloc...Q** with **M_GRAB**. When a buffer is no longer required, release it, using **MbufFree()**.

ImageCount

Specifies the number of buffers in the array of destination image buffers.

Operation

Specifies the type of operation to perform.

This parameter should be set to one of the following values:

● For specifying the type of operation to perform																									
<div><input type="checkbox"/> Value</div>	Description	corona-II (a)	cronosplus (b)	gpu vision (c)	hellios ea/xa (e)	hellios ec/xd (f)	hellios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II/c1 (j)	met-II/d1g (k)	met-II/mc (l)	met-II/std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios g1ge (v)	v1o (w)		
<div><input type="checkbox"/> M_SEQUENCE +</div>	<p>Grabs a specific number of frames, storing them sequentially in a list of buffers. Grabbing ends once the specified number of frames have been captured.</p> <p>When using a digitizer set to perform triggered grabs and the OperationFlag parameter is set to M_TRIGGER_FOR_FIRST_GRAB, the function will grab the specified number of frames when the first trigger event occurs.</p> <p>(summarize)</p>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<div><input type="checkbox"/> M_START +</div>	<p>Starts grabbing round-robin into the list of buffers; the grabs will continue until stopped (using MdigProcess() with M_STOP).</p> <p>When using a digitizer set to perform triggered grabs and the OperationFlag parameter is set to M_TRIGGER_FOR_FIRST_GRAB, the function will, by default, start grabbing all frames when the first trigger event occurs.</p> <p>(summarize)</p>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<div><input type="checkbox"/> M_STOP +</div>	<p>Stops the grab. By default, it also cancels all grabs that might have been queued. Note that cancelled grabs will not trigger the hooked function, but the events for the previous grabs are still processed.</p> <p>To stop a grab started in synchronous mode, the Operation parameter must be set to M_STOP and the MdigProcess() function called from a separate thread.</p> <p>(summarize)</p>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

Combination constant for **M_STOP**;

You can add the following value to the above-mentioned value to specify that previously queued grabs should not be cancelled when stopping.

● For M_STOP																									
<input type="checkbox"/> Value	Description	corona-II (a)	cronosplus (b)	gpu processing (d)	glige vision (c)	hellios ea/xa (e)	hellios ec/xc/ (f)	hellios ed/xd (g)	hellios ee/xe (h)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (i)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xc/ (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	
<input type="checkbox"/> M_WAIT	Stops queuing new grabs and waits for all previously queued grabs to finish before returning control to the Host.	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Combination constant for **M_SEQUENCE**;

You can add the following value to the above-mentioned value to specify the number of frames to grab.

● For M_SEQUENCE	
<input type="checkbox"/> Value	Description
	corona-II cronosplu gige visio gpu proce hellios ea/ hellios ec/ hellios ed/ ieee 1394 iris (I) met-II /cl met-II /di met-II /m met-II /st morphis morphis s nexis (p) odyssey e odyssey e odyssey e solios ea/ solios ecl solios gige vco (w)

<div><div><div></div></div><div>M_COUNT(MIL_INT <i>n</i>)</div></div>	<div><div>Specifies the number of frames to grab in the sequence.</div><div>This combination constant will cause MdigProcess() to stop grabbing when the grab count (<i>n</i>) is reached. If the grab count is greater than the buffer count, the grab will be done in round-robin style. By default, the grab count is equal to the number of buffers in the destination image array.</div><div>(summarize)</div></div>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<div><div>Parameters</div><div><div><i>n</i></div><div>Specifies the number of frames to grab. <i>n</i> is a positive integer that can be a maximum of 2 to the power of 28.</div></div></div>																						

Combination constant for **M_START**;

You can add the following value to the above-mentioned value to specify the number of frames to grab sequentially when a digitizer trigger event occurs.

● For M_START																								
Value	Description	corona-II (a)	crotoplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ieee 1394 ilbc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ec/xd (u)	solos gige (v)	vio (w)
<input type="checkbox"/> M_FRAMES_PER_TRIGGER (MIL_INT <i>n</i>)	Specifies the number of frames to grab sequentially when a digitizer trigger event occurs. Note that to use this setting, the OperationFlag parameter must be set to M_TRIGGER_FOR_FIRST_GRAB and the digitizer used must be configured to grab upon a trigger. (summarize)					e	f	g		i					n	o	p	q	r	s	t	u	w	
	<i>Parameters</i>																							
	<i>n</i> Specifies the number of frames to grab sequentially for each digitizer trigger event; <i>n</i> can be a maximum of 2 to the power of 28.					e	f	g		i					n	o	p	q	r	s	t	u	w	

OperationFlag

Allows you to provide more information about the synchronization and triggering of the grab.

● For controlling the order in which processing occurs																										
<input type="checkbox"/> Value	Description	corona-II (a)	crotoplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ieee 1394 ilbc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	sollos ea/xa (t)	sollos ec/xd (u)	sollos gige (v)	vio (w)		
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default execution mode.</p> <p>If the Operation parameter is set to M_START, the default is the same as M_ASYNCHRONOUS.</p> <p>If the Operation parameter is set to M_SEQUENCE, the default is the same as M_SYNCHRONOUS.</p> <p>If the Operation parameter is set to M_STOP, the default is the same as M_SYNCHRONOUS unless MdigProcess() is called from the user-defined function, in which case the default is the same as M_ASYNCHRONOUS. Note that when the Operation parameter is set to M_STOP, M_DEFAULT is the only possible setting for the OperationFlag parameter.</p> <p>(summarize)</p>	a	b	c	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

<input type="checkbox"/> M_ASYNCHRONOUS +	Allows your thread to continue after initiating the start of the grabs, rather than waiting for MdigProcess() to finish. Note than when you are grabbing and processing a sequence of images asynchronously, you must make an additional call to MdigProcess using M_STOP to free internally allocated resources. (summarize)	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_SYNCHRONOUS +	Synchronizes your thread with the end of MdigProcess() .	a	b	c		e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constant for [M_ASYNCHRONOUS](#); [M_SYNCHRONOUS](#);

You can add the following value to the above-mentioned values to specify that trigger events only control when to start grabbing all frames or every nth frame, when the digitizer is triggered. The default behavior for the function (when M_TRIGGER_FOR_FIRST_GRAB is not set) is to wait for a trigger event for every frame to grab.

● For M_ASYNCHRONOUS and M_SYNCHRONOUS																									
<input type="checkbox"/> Value	Description	corona-II (a)	gige vision (c)	crnosplus (b)	gpu processing (d)	helios ea/xa (e)	helios eel/xd (f)	helios ed/xd (g)	ilee 1394 ilic (h)	iris (i)	met-II /cl (j)	met-II /mc (l)	met-II /dig (k)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ed/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios eel/xd (u)	solios gige (v)	vio (w)	
<input type="checkbox"/> M_TRIGGER_FOR_FIRST_GRAB	Specifies that a trigger will only be used to control the grab of the first frame. When there is no preset end to the list of grabs (M_START), you can also force every <i>nth</i> grab to wait for a trigger. To do so, set the Operation parameter to M_START + M_FRAMES_PER_TRIGGER() . Note that if the digitizer is not configured for triggered grabs, M_TRIGGER_FOR_FIRST_GRAB is ignored. (summarize)					e	f	g		i					n	o	p	q	r	s	t	u		w	

HookHandlerPtr

Specifies the address of the function that is called when a buffer is modified in the array of destination image buffers.

The hook-handler function, pointed to by **HookHandlerPtr**, must be declared as follows:

```
MIL_INT MFTYPE HookHandler(  
    MIL_INT HookType,  
    MIL_ID EventId,  
    void MPTYPE *UserDataPtr  
)
```

Parameters:

HookType

Type of event that generated the call ([M_MODIFIED_BUFFER](#)).

EventId

Event identifier. You can pass the identifier to [MdigGetHookInfo\(\)](#) to inquire about the hooked event.

UserDataPtr

User data pointer that was passed (as **UserDataPtr**) to **MdigProcess()**.

Upon successful completion, the hook-handler function should return *M_NULL*. Note *MFTYPE* and *MIL_DIG_HOOK_FUNCTION_PTR* are reserved MIL predefined types for functions and data pointers.

If the hook-handler function modifies the buffer that called it, **MdigProcess()** will not be called again for that buffer.

UserDataPtr

Specifies the address of the user data that you want to make available to the hook-handler function. This address is passed to the hook-handler function, through its **UserDataPtr** parameter, when the specified event occurs. Set this parameter to **M_NULL** if not used.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MdigReference

Synopsis

Select a digitization reference level.

Syntax

```
void MdigReference(
    MIL_ID DigId,
    MIL_INT64 ReferenceType,
    MIL_DOUBLE ReferenceLevel
)
```

Description

This function sets (if available) the reference levels used to digitize the analog signal received from a camera. This function is specific to analog input devices. Depending on the type of digitizer and input signal, some reference types are not applicable.

[Matrox Helios eA/XA; Matrox Odyssey eA/XA; Matrox Solios eA/XA]

The following settings are for the XA only. Refer to Adjusting the reference levels, in the Frame Grabbers chapter of your board's Installation and Hardware Guide. There are three ways to change the input signal's gain:

The first method is to set [M_BLACK_REF](#) and [M_WHITE_REF](#) to specific values (between [M_MIN_LEVEL](#) and [M_MAX_LEVEL](#)). This method is commonly used interactively (for example, in an application that allows you to change the settings until you are satisfied with the image quality). Note that the black reference level must always be less than the white level. Do not use [MdigControl\(\)](#) with [M_GRAB_INPUT_GAIN](#) if you are using this method since the gain will be adjusted automatically.

The second method is to set the black and the white reference levels in Volts. By adding [M_VOLTAGE](#) to the [ReferenceType](#), the optimum levels for the offset and gain controllers will be calculated automatically. This is more useful if you know the actual input signal voltages.

[Matrox Helios eA/XA; Matrox Solios eA/XA]

The third method is to set [M_WHITE_REF](#) and then call [MdigControl\(\)](#) with [M_GRAB_INPUT_GAIN](#) set to a value from [M_MIN_LEVEL](#) to [M_MAX_LEVEL](#). This method corresponds more closely to the actual hardware that controls the reference levels. The hardware has an offset controller followed by a gain controller; the analog-to-digital converter cannot be programmed directly with the reference levels. Note that when using this method, you should not specify a white reference level because it would cause the gain to be altered.

[Matrox Odyssey eA/XA]

The third method is to call [MdigControl\(\)](#) with [M_GRAB_INPUT_GAIN](#) set to a value from [M_MIN_LEVEL](#) to [M_MAX_LEVEL](#). This method corresponds more closely to the actual hardware that controls the reference levels. The hardware has an offset controller followed by a gain controller; the analog-to-digital converter cannot be programmed directly with the reference levels. Note that when using this method, you should not specify a white reference level because it would cause the gain to be altered.

[Matrox GigE Vision driver; Matrox IEEE 1394 IIDC driver; Matrox Solios GigE]

Note that the following reference types are only available if they are supported by the camera. Refer to your camera's documentation for more details.

Note that the following reference types are only available when grabbing from an analog input.

Parameters

DigId

Specifies the identifier of the digitizer on which to set the reference level. An error is generated if the specified digitizer does not support the type of programmable digitization reference levels specified.

ReferenceType

Specifies the reference level type to adjust for the specified digitizer.

This parameter can be set to one of the following:

For specifying the reference level type																									
Value	Description	corona-II (a)	crotoplus (b)	glge vision (c)	gpu processing (d)	helios ea/xa (e)	helios ecl/xcl (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ed/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ecl/xd (u)	solios gige (v)	vio (w)	
M_BLACK_REF +	Sets the input signal's digitization black reference level. Note that when setting black and white reference levels, always set black reference levels first. (summarize)	a	c	e									l				q			t		v	w		
	Board specific																								
	Note that some consecutive ReferenceLevel settings might produce the same result due to the fact that there are only 98 distinct adjustments (adjustments of 10.23 mV each).	a											l												
	The minimum voltage is 0.6V. The maximum voltage is 1.6V.	a											l												
	The minimum voltage level is -1 V. The maximum voltage level is +1 V. Note that the white level should be higher than the black level.			e													q			t					
	Note that this reference type is only available when your DCF uses either an analog RGB video signal or a monochrome via RGB video signal.																							w	
M_BRIGHTNESS_REF	Sets the brightness level for composite input signals. (summarize)	b						h					m	n	o									w	
	Board specific																								
	Note that this reference type is only available when using a DCF that uses a composite video signal.																							w	
M_CONTRAST_REF	Sets the contrast level for composite input signals. (summarize)	b											m	n	o									w	
	Board specific																								
	Note that this reference type is only available when using a DCF that uses a composite video signal.																							w	
M_HUE_REF	Sets the hue level for composite input signals. M_HUE_REF is available when grabbing color data (not monochrome data). (summarize)	b						h					m	n	o									w	
	Board specific																								
	Note that this reference type is only available when using a DCF that uses a composite video signal.																							w	
M_SATURATION_REF	Sets the saturation level for composite input signals. M_SATURATION_REF is available when grabbing color data (not monochrome data). (summarize)	b						h					m	n	o									w	
	Board specific																								
	Note that this reference type is only available when using a DCF that uses a composite video signal.																							w	
M_WHITE_REF +	Sets the input signal's digitization white reference level. Note that when setting black and white reference levels, always set black reference levels first. (summarize)	a	c	e									l				q			t		v	w		
	Board specific																								
	Note that some consecutive ReferenceLevel settings might produce the same result due to the fact that there are only 98 distinct adjustments (adjustments of 10.23 mV each).	a											l												
	The minimum voltage is 1.6V. The maximum voltage is 2.6V.	a											l												
	The minimum voltage level is -1 V. The maximum voltage level is +1 V. Note, the white level should be higher than the black level.			e													q			t					

For automatically specifying the reference level																									
<div><input type="checkbox"/> Value</div>	Description	corona-II (a)	cronoplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ec/xd (u)	solos gige (v)	vio (w)		
<div><input type="checkbox"/> M_AUTOMATIC</div>	Sets the reference level automatically. (summarize)			c																		v	w		
	<i>Board specific</i>																								
	Only supported when grabbing from the analog path.																						w		

ReferenceLevel

Specifies the level of reference.

The smallest voltage increment supported by your board can differ such that consecutive reference-level settings might produce the same result.

Note, some digitizers might take a few milliseconds before the reference level stabilizes.

For specifying the reference level																							
Value	Description	corona-II (a)	coronoplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xci (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	odyssey ea/xa (p)	odyssey ec/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ec/xd (u)	solos gige (v)	vio (w)
M_DEFAULT	Sets the reference levels to the defaults for the specified digitizer data format.	a	b	c		e			h				l	m	n	o	q			t		v	w
M_MAX_LEVEL	Sets the value to the maximum level.	a	b			e							l	m	n	o	q			t			w
M_MIN_LEVEL	Sets the value to the minimum level.	a	b			e							l	m	n	o	q			t			w
M_MIN_LEVEL <= Value <= M_MAX_LEVEL	Sets a value between M_MIN_LEVEL and M_MAX_LEVEL , inclusive. To calculate the value to pass to MdigReference() , use the following equation with the appropriate voltages for your particular board. Value to pass to MdigReference() = $\left(\frac{\text{Voltage needed} - \text{minimum voltage}}{\text{Maximum voltage} - \text{minimum voltage}} \right) (\text{M_MAX_LEVEL} - \text{M_MIN_LEVEL})$ (summarize) <div><i>Board specific</i></div> <div>If not using M_BLACK_REF + M_VOLTAGE or M_WHITE_REF + M_VOLTAGE, this value will map to a value within the given range.</div> <div>When using M_VOLTAGE, the reference levels are from -1.0V to +1.0V.</div> <div>When using M_VOLTAGE, the reference levels are from 0.0V to 1.0V.</div>	a	b			e							l	m	n	o	q			t			w
Minimum voltage <= Value <= Maximum voltage	Sets a value between the minimum voltage and the maximum voltage, inclusive. Note that this value can only be used with M_BLACK_REF + M_VOLTAGE or M_WHITE_REF + M_VOLTAGE . The reference levels are from -1.0V to +1.0V (for example, use 0.7 for 700 mV). (summarize)					e											q			t			

Compilation information

Header	Include mil.h.

Library	Use mil.lib.
DLL	Requires mil.dll.

Mdisp functions

Synopsis

The functions prefixed with Mdisp make up the Display module. The Display module allows you to display images and manipulate their display. The Display module offers many display effects, such as annotation, LUTs, and panning and zooming

Functions

- [MdispAlloc](#)
- [MdispControl](#)
- [MdispFree](#)
- [MdispHookFunction](#)
- [MdispInquire](#)
- [MdispLut](#)
- [MdispPan](#)
- [MdispSelect](#)
- [MdispSelectWindow](#)
- [MdispZoom](#)

MdispAlloc

Synopsis

Allocate a display.

Syntax

```
MIL_ID MdispAlloc(  
    MIL_ID SystemId,  
    MIL_INT DispNum,  
    MIL_CONST_TEXT_PTR DispFormat,  
    MIL_INT InitFlag,  
    MIL_ID *DisplayIdPtr  
)
```

Description

This function allocates a display on the specified system so that it can be used by subsequent MIL display functions. Use [MdispSelect\(\)](#) to select the image buffer to display. Note that the buffer and the display should be allocated on the same system.

In general, the display is presented on the computer running the main MIL application; however there are two ways to display an image buffer on the remote computer, depending on your application requirements:

- When developing a Distributed MIL application and you allocate a display on a DMIL remote system, you can select image buffers allocated on that DMIL remote system to the display. By default, these buffers are displayed on the master computer. However, to display the buffers on the remote computer, combine [M_AUXILIARY](#) or [M_WINDOWED](#) with [M_REMOTE_DISPLAY](#) when allocating the display. This type of display is referred to as a remote display.
- When developing a MIL application and wanting to publish a display so that image buffers selected to the display can be viewed at any remote computer in, for example, a web browser, use [M_NETWORK](#) when allocating the display. This type of display is referred to as a network display.

[Matrox Iris]
There can be 31 network displays allocated simultaneously.

When a display is no longer required, you should free it, using [MdispFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the display. This parameter should be set to one of the following values:

For specifying the system	
Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

DispNum

Specifies the number (or rank) of the display. This parameter must be set to the following value:

For specifying the number of the display	
Value	Description
<input type="checkbox"/> M_DEFAULT	Allows MIL to find the best device to use for a windowed display or an auxiliary display. If your imaging board has a display section, and it is available, MIL will typically use it for display purposes.

([summarize](#))

DispFormat

Specifies the format of the display. For windowed displays, this parameter has no effect and should be set to [MIL_TEXT\(\)](#) with "M_DEFAULT". For auxiliary displays, this parameter specifies the screen resolution. For network displays, this parameter specifies color format in which to transmit the data.

For any type of display, this parameter can be set to the following value:

● For specifying the default display format	
☐ Value	Description
☐ MIL_TEXT(MIL_TEXT_PTR DefaultDispFormat)	Specifies the default display format. For windowed displays, this is the only supported setting. For auxiliary displays, this specifies the default video configuration format (VCF), which can be set during installation and later modified using the MILConfig utility (Defaults tab). For network displays, this is the same as "M_REMOTEVIEW". (summarize)
	Parameters
	DefaultDispFormat Specifies the default display format.
	"M_DEFAULT" Specifies the default display format.

When programming for your Matrox Iris, there is no local display to show the images or other data. The only display available is remote, which is displayed on the client computer (in a web page). Your Matrox Iris does not support windowed or auxiliary displays.

For network displays, the [DispFormat](#) parameter can be set to one of the following values:

● For network displays	
☐ Value	Description
☐ MIL_TEXT(MIL_TEXT_PTR NetworkDispFormat)	Specifies the format of the network display to be allocated. (summarize)
	Parameters
	NetworkDispFormat Specifies the format of the network display.
	"M_REMOTEVIEW" Sets the display format to the most appropriate type of display available.
	"M_REMOTEVIEW_MONO" Sets the display format to monochrome.
	"M_REMOTEVIEW_PSEUDO" Sets the display format to pseudo-color. To display the monochrome data in color, you must allocate your display with this setting. This will map images through a LUT upon display. This display format can support a default 256 monochrome LUT with upper and lower 10 LUT entries programmed with Windows reserved 32-bit color values. Note that this default LUT cannot be changed.
	"M_REMOTEVIEW_RGB" Sets the display format to RGB.

For auxiliary displays that do not use an encoder, the [DispFormat](#) parameter can be set to the following:

buffer selected for display; that is, if the window moves or is occluded, the window is updated with the image buffer accordingly.
([summarize](#))

Combination constants for [M_AUXILIARY](#); [M_WINDOWED](#);

You can add one or more of the following values to the above-mentioned values to set where and how the display is allocated.

For controlling where and how the display is allocated																									
Value	Description	corona-II (a)	cronosplus (b)	glige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ec/Xd (f)	helios ed/Xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ec/Xd (r)	odyssey ed/Xd (s)	solos ea/Xa (t)	solos ec/Xcl (u)	solos glige (v)	vio (w)	
M_GDI_OVERLAY	Forces the display's overlay buffer to be compatible with GDI operations, when the overlay-display mechanism is enabled. You can then allocate and use a device context (DC) for drawing in the buffer. Note that performing MIL operations on the display's overlay buffer is more efficient when the display's overlay buffer is not forced to be GDI compatible. (summarize)	a	b	c		e	f	g	h		j	k	l	m	n	o	p	q	r	s	t	u	v	w	
	<i>Board specific</i>																								
	Note that this value is only supported for windowed displays.																p								
M_REMOTE_DISPLAY	Allocates the display so that it is displayed on the remote computer. This setting is only available when the display is allocated on a DMIL remote system. If the display is allocated on a DMIL remote system but this setting is not specified, the display is presented on the master computer. To present the display on the remote computer, the Distributed MIL server cannot be running as a service; you must start it manually. To do so, you must first enable the Manually launched Distributed MIL option during ActiveMIL installation (or on the Distributed MIL Server Settings page of MILConfig) on the remote computer. Then, you must logon to the remote computer and run the executable of the Distributed MIL server (<i>MilNetworkServer.exe</i>); to run the executable automatically upon logon, add it to the remote computer's Startup folder. For more information, see the Preparing the master and remote computers section in Chapter 24: Distributed MIL . (summarize)	a	b	c		e	f	g	h		j	k	l	m	n	o	p	q	r	s	t	u	v	w	

DisplayIdPtr

Specifies the address of the variable in which to write the display identifier. Since the **MdispAlloc()** function also returns the display identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the display identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Remark

- If you are creating a DLL that includes a call to **MdispAlloc()**, ensure that the call is not made from the **DIIMain()** function, because **MdispAlloc()** might load a required DLL and you cannot load a DLL from **DIIMain()**. If necessary, call **MdispAlloc()** from an initialization function in your DLL instead.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll; mildisplay.dll.

MdispControl

Synopsis

Control a MIL display setting.

Syntax

```
void MdispControl(
    MIL_ID DisplayId,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function allows you to control the specified MIL display setting.

Parameters

- DisplayId
- Specifies the identifier of the target display.
- ControlType
- Specifies the type of display setting to control. The control types for windowed displays can control the default MIL or user-specified window of the display ([MdispSelect\(\)](#) or [MdispSelectWindow\(\)](#)).
- See the [Parameter associations](#) section for possible values.
- ControlValue
- Specifies the new value to assign to the display setting specified by the [ControlType](#) parameter.
- See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For display settings](#)
- [For windowed and auxiliary displays](#)
- [For windowed displays](#)
- [For displays allocated on a DMIL remote system but displayed on the master computer](#)
- [For MIL windowed displays](#)
- [For specifying settings when using auxilliary displays](#)
- [For specifying settings when using the encoder](#)

Unless otherwise specified, the following [ControlType](#) and corresponding [ControlValue](#) parameter settings are available for windowed, auxiliary, and network displays.

For display settings		
ControlType	Description	
ControlValue		corona-II (a) cronsoplus (b) glige vision (c) gpu processing (d) helios ea/xa (e) helios ec/Xcl (f) helios ed/Xd (g) ieee 1394 i1dc (h) iris (i) met-II /cl (j) met-II /dlg (k) met-II /mc (l) met-II /std (m) morphis (n) morphis qxt (o) nexis (p) odyssey ea/xa (q) odyssey ec/Xcl (r) odyssey ed/Xd (s) sollos aa/xa (t) sollos ed/Xcl (u) sollos glige (v) v10 (w)

<input type="checkbox"/> M_OVERLAY	<p>Sets whether MIL's overlay-display mechanism is enabled. This mechanism allows you to annotate displayed image buffers non-destructively. For more information on MIL's overlay buffers, see the Annotating the displayed image non-destructively section in Chapter 20: Displaying an image.</p> <p>Once enabled, use MdispInquire() with the M_OVERLAY_ID inquire type to determine the MIL identifier of the overlay buffer. Note that once created, the overlay buffer is like any other image buffer. That is, you can perform operations on it as you would on a normal buffer (except for grabbing).</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISABLE	<p>Disables MIL's overlay-display mechanism.</p> <p>This is the default value.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ENABLE	Enables MIL's overlay-display mechanism.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_OVERLAY_CLEAR	<p>Sets the value to which the overlay buffer associated with the display should be cleared. The buffer is cleared to the specified value immediately after this control type setting is changed.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Same as M_TRANSPARENT_COLOR .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_RGB888(MIL_INT <i>red</i>, MIL_INT <i>green</i>, MIL_INT <i>blue</i>)	<p>Specifies the RGB value to which the display's overlay buffer will be cleared for a non 8-bit display mode.</p> <p>Specifies the RGB value. The buffer must be a non 8-bit 3-band buffer.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Parameters</i>																							
	<i>red</i> Specifies the red component, as a value between 0 and 255.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>green</i> Specifies the green component, as a value between 0 and 255.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>blue</i> Specifies the blue component, as a value between 0 and 255.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_BLACK	Specifies the color black.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_BLUE	Specifies the color blue.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_BRIGHT_GRAY	Specifies the color bright gray.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_CYAN	Specifies the color cyan.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_BLUE	Specifies the color dark blue.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_CYAN	Specifies the color dark cyan.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_GREEN	Specifies the color dark green.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_MAGENTA	Specifies the color dark magenta.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_RED	Specifies the color dark red.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_DARK_YELLOW	Specifies the color dark yellow.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_GRAY	Specifies the color gray.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_GREEN	Specifies the color green.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_LIGHT_BLUE	Specifies the color light blue.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_LIGHT_GRAY	Specifies the color light gray.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_LIGHT_GREEN	Specifies the color light green.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_LIGHT_WHITE	Specifies the color light white.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

<input type="checkbox"/> M_COLOR_MAGENTA	Specifies the color magenta.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_RED	Specifies the color red.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_WHITE	Specifies the color white.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_YELLOW	Specifies the color yellow.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_TRANSPARENT_COLOR	Specifies that the display's overlay buffer will be cleared to the transparency color. Set the transparency color with the M_TRANSPARENT_COLOR control type. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_OVERLAY_SHOW	Sets whether the display's overlay buffer is visible. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISABLE	Specifies that the overlay buffer is not visible. Display the image buffer selected on the display only. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ENABLE	Specifies that the overlay buffer is visible. Annotations or other changes made to the overlay buffer in a color other than the transparency color will annotate the image selected to the display. Set the transparency color with the M_TRANSPARENT_COLOR control type. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_OPAQUE	Specifies that only the overlay buffer is visible. The image buffer selected will be hidden. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_TITLE	Sets the display's title to a specified string. For windowed displays, set whether the window's title bar is visible with the M_WINDOW_TITLE_BAR control type. For auxiliary displays, the title bar is not visible but the display's title can still be inquired. If no string is specified, the window is given a default title ("MIL Display #x"). (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_PTR_TO_DOUBLE(void MPTYPE * PTR)	Specifies the address of the character array in which the window title is stored. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Parameters</i> <i>PTR</i> The address of the character array in which to read the window title.																							
<input type="checkbox"/> M_TRANSPARENT_COLOR	Sets the transparency (keying) color. Pixels of the image buffer selected to the display are displayed only where the corresponding pixels of the overlay buffer are set to the transparency color. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Specifies the optimal transparency color.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_RGB888(MIL_INT Red, MIL_INT Green, MIL_INT Blue)	Specifies the RGB value to which to set the transparency color for a non 8-bit display mode. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Parameters</i>																							
	<i>Red</i> Specifies the red component, as a value between 0 and 255.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Green</i> Specifies the green component, as a value between 0 and 255.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Blue</i> Specifies the blue component, as a value between 0 and 255.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_COLOR_BLACK	Specifies the color black.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

	<div>Board specific</div> <div>The default mode corresponds to M_ASYNCHRONOUS, if the user-defined window thread is the same as the current one. Otherwise, it is M_SYNCHRONOUS.</div> <div>Same as M_ASYNCHRONOUS.</div>	a	b	c	d	e	f	g	h	j	k	l	m	n	o	p	q	r	s	t	u	v	w	
<div><div>☐</div>M_ASYNCHRONOUS</div>	<div>Specifies that the display is updated when possible. (summarize)</div> <div>Board specific</div> <div>The paint messages are queued and that the application's window procedure (WndProc) will update the display whenever possible. Basically, when a paint message is issued, MIL calls the Windows <code>InvalidateRect()</code> function. For more information on <code>InvalidateRect()</code>, refer to the MSDN library.</div> <div>The published image data stream to be asynchronous. The function will return immediately while internal threads are sending data to all clients that are listening for the data stream. MdispControl() using M_UPDATE_SYNCHRONIZATION with a valid display identifier will result in a call to the Remote module's <code>RemoteSetAttribute</code> function, with the proper parameters set.</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div>☐</div>M_SYNCHRONOUS</div>	<div>Specifies that the display is updated immediately.</div> <div>The display is updated immediately upon reception of a paint message. Basically, when a paint message is issued, MIL calls the Windows <code>InvalidateRect()</code> function followed by the Windows <code>UpdateWindow()</code> function. For more information on <code>InvalidateRect()</code> and <code>UpdateWindow()</code>, refer to the MSDN library. (summarize)</div> <div>Board specific</div> <div>The published image data stream will be synchronous. The function will return only when the information is sent to all clients that are listening for the data stream. The call to MdispControl() using M_UPDATE_SYNCHRONIZATION with a valid display ID will result in a call to the Remote module's <code>RemoteSetAttribute</code> function, with the proper parameters set.</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

The following **ControlType** and corresponding **ControlValue** parameter settings are available for windowed and auxiliary displays.

For windowed and auxiliary displays	
ControlType	Description
ControlValue	
M_CENTER_DISPLAY	<p>Sets whether a selected image buffer will be centered in the display, both in X and Y.</p> <p>If enabled, the image buffer will be offset by $(\text{WindowSizeX} - \text{BufferSizeX})/2$, $(\text{WindowSizeY} - \text{BufferSizeY})/2$ in the window.</p> <p>(summarize)</p>
M_DEFAULT	<p>Specifies the default value. The default value is M_ENABLE for a MIL default window except if the window is maximized. The default value is M_DISABLE for a user-defined window.</p> <p>(summarize)</p>
M_DISABLE	<p>Specifies that the image buffer will not be centered in the display.</p>
M_ENABLE	<p>Specifies that the image buffer will be centered in the display, both in X and Y.</p>
M_FILL_DISPLAY	<p>Sets whether the display will be filled with the selected image buffer using an automatically calculated zoom factor.</p> <p>To retain the original aspect ratio of the image buffer, while filling the display, use M_SCALE_DISPLAY.</p> <p>(summarize)</p>
M_DISABLE	<p>Specifies that the display will not be filled with the image buffer using an automatically calculated zoom factor. Other zoom factors can be applied, using MdispZoom().</p> <p>This is the default value.</p> <p>(summarize)</p>

<input type="checkbox"/> M_ENABLE	<p>Specifies that the display will be filled with the image buffer using an automatically calculated zoom factor.</p> <p>This setting overrides zoom factors (MdispZoom()), disables the window's zoom buttons (M_WINDOW_ZOOM), and overrides pan and scroll settings (MdispPan()). (summarize)</p>
<input type="checkbox"/> M_INTERPOLATION_MODE	<p>Sets the type of interpolation used to display a zoomed image. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_FAST .
<input type="checkbox"/> M_FAST	<p>Specifies to use the fastest interpolation mode available to update the display. This might be something other than nearest neighbor (for example, it could be bilinear) depending on the hardware accelerated mode used and how the final blit is performed. (summarize)</p>
<input type="checkbox"/> M_NEAREST_NEIGHBOR	Specifies the nearest neighbor interpolation.
<input type="checkbox"/> M_NO_TEARING	<p>Allows you to enable and set the no-tearing mode for the selected display.</p> <p>When using DirectDraw 7 (DirectX version 7.0), no-tearing requires special hardware, such as a Matrox Millennium G550 graphics controller; when using a non-Matrox display board, you must enable this feature using MILConfig before you can use it in MIL. If the appropriate hardware is not available and you try to enable no-tearing, MIL will permit tearing instead of generating an error. When using Direct3D (DirectX version 9.0), no special hardware is required; this technology handles no-tearing directly. However, Direct3D enables no-tearing for the entire screen; therefore, enabling no-tearing for one MIL display activates it for all displays at the same time. Standard hardware-acceleration display mode does not support no-tearing and an error is returned if you try to activate it.</p> <p>To determine if the graphics controller is currently implementing a specified display with no-tearing, use MdispInquire() with M_NO_TEARING_ACTIVE.</p> <p>You must disable support for direct window annotations (MdispControl() with M_WINDOW_ANNOTATIONS) before you can enable no-tearing; otherwise, an error is generated.</p> <p>For more information on screen tearing, see the Screen tearing section in Chapter 20: Displaying an image. (summarize)</p> <div> <div>Operating system specific</div> <div>[<i>This is only applicable to Linux.</i>]</div> </div> <p>Under Linux, M_NO_TEARING is only supported for auxiliary displays.</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is M_DISABLE, except for auxiliary displays using an encoder and displaying a continuous grab and using DirectDraw 7. In which case, the default value is M_ENABLE. (summarize)</p>
<input type="checkbox"/> M_ADVANCED +	<p>Specifies a mode that uses an advanced no-tearing mechanism for all updates to the display. In this mode, screen tearing does not occur regardless of whether the rate at which display memory is updated is slower, equal, or faster than the rate at which the screen is refreshed. When more than one update to display memory occurs between two vertical synchronization pulses of the screen, the updates are queued using multiple buffers.</p> <p>Note that during a vertical synchronization pulse of the screen, only one update is copied to the screen. By default, MIL copies every update in the queue to the screen; this results in a lag in the time it takes for the updates to be displayed. When using DirectDraw 7, MIL can skip certain updates in the queue to avoid this lag; see combination values below. When using Direct3D, this lag might occur because no updates are skipped. (summarize)</p>
<input type="checkbox"/> M_BASIC	<p>Specifies a mode that uses a basic no-tearing mechanism for all updates to the display. This mechanism compensates only when the rate at which display memory is updated is equal to or slower than the rate at which the screen is refreshed. Therefore, screen tearing might still occur when MdispInquire() with M_NO_TEARING_ACTIVE returns M_YES. (summarize)</p>
<input type="checkbox"/> M_DISABLE	Specifies not to use a no-tearing mechanism to update the display.
<input type="checkbox"/> M_ENABLE	Specifies to use the optimal no-tearing mode for the current situation (varies depending on such factors as the board used and CPU availability).
<input type="checkbox"/> M_GRAB_CONTINUOUS_ONLY	<p>Specifies a mode that uses M_BASIC no-tearing mode when displaying a continuous grab. For all other display updates (that is, updates that are necessary due to modifications made to the display's overlay buffer or to the buffer selected to the display), it does not compensate for tearing. Therefore, MdispInquire() with M_NO_TEARING_ACTIVE returns M_YES only during a continuous grab, indicating that the graphics controller is updating the screen with a no-tearing mechanism. After MdigHalt() is called, MdispInquire() with M_NO_TEARING_ACTIVE returns M_NO.</p> <p>This mode is not supported when using Direct3D. (summarize)</p>
<input type="checkbox"/> M_SAFE_MODE	Sets how to display grabbed data. This is useful when dealing with displays used for continuous grabs. If CTRL-ALT-DEL is pressed or if a screen saver starts,

	MIL loses all display surfaces. Enabling M_SAFE_MODE ensures that grabbed data is not grabbed directly into display memory, if possible. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the grabbed data is grabbed directly into display memory, if possible.
<input type="checkbox"/> M_ENABLE	Specifies that the grabbed data is not grabbed directly into display memory, if possible.
<input type="checkbox"/> M_SCALE_DISPLAY	Sets whether to fill the display with the selected image buffer, while keeping the same aspect ratio. To fill the display, without ensuring that the aspect ratio is maintained, use M_FILL_DISPLAY . (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the image will not be scaled to fit the display. Unless a zoom factor has been applied, using MdispZoom() , the image will retain its original size and aspect ratio, leaving unused portions of the display black. This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies to automatically scale the selected image buffer to fit the display, while keeping the same aspect ratio. In this way, the image is not distorted. This setting overrides zoom factors (MdispZoom()), disables the window's zoom buttons (M_WINDOW_ZOOM), and overrides pan and scroll settings (MdispPan()). (summarize)
<input type="checkbox"/> M_UPDATE_RATE_DIVIDER	Sets after how many buffer modifications to update the display, regardless of the time lapsed. For example, if you set M_UPDATE_RATE_DIVIDER to 10, one out of every 10 buffer modifications will cause a display update. If fewer than the specified number of buffer modifications occur, the display is not updated. The display is only updated with the last buffer modification. This control type is useful when performing a continuous grab. If the display is allocated on a DMIL remote system but displayed on the master computer, another mechanism is also available to limit the number of display updates. You can set an upper limit to the number of display updates to perform per second, using M_UPDATE_RATE_MAX . In this case, updates of modified areas of the buffer are accumulated into one update between transmissions. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1. (summarize)
<input type="checkbox"/> Value	Specifies the frequency of the update.
<input type="checkbox"/> M_VIEW_BIT_SHIFT	Sets the number of bits by which to bit-shift the pixel values of the image buffer selected to the display, when the M_VIEW_MODE control type is set to M_BIT_SHIFT . (summarize)
<input type="checkbox"/> Value	Specifies the number of bits. This value should be set to the number of significant bits in the image buffer minus 8. For example, if a 16-bit image buffer contains data grabbed from a 10-bit digitizer, a shift of 2 should be used. The default value is 0. (summarize)
<input type="checkbox"/> M_VIEW_MODE	Sets the view mode of the display. The view mode establishes how an image buffer gets remapped to the display; this is especially useful when displaying a non 8-bit image buffer. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is M_TRANSPARENT unless the buffer selected on the display is a 1-bit buffer; when a 1-bit buffer is used, the default value is M_AUTO_SCALE . (summarize)
<input type="checkbox"/> M_AUTO_SCALE	Remaps the pixel values to the display so that the minimum and maximum values in the image (not the full range of the image buffer) are set to 0 and 255, respectively. If the image buffer contains a single value, the displayed value is determined by linearly re-mapping the full range of the image buffer to that of the display (for example, (0 to 64K) to (0 to 255)). (summarize)
<input type="checkbox"/> M_BIT_SHIFT	Bit-shifts the pixel values of the image buffer by the specified number of bits upon updating the display. Specify the number of bits with the M_VIEW_BIT_SHIFT control type. (summarize)
<input type="checkbox"/> M_MULTI_BYTES	Displays each byte of the image buffer in separate display pixels. In other words, each pixel of a 16-bit image buffer will occupy two consecutive display pixels. Each pixel of a 32-bit image buffer will occupy four consecutive display pixels. This mode is only supported for 16-bit and 32-bit 1-band buffer. This mode is primarily useful when grabbing from a multi-tap camera.

	(summarize)
<input type="checkbox"/> M_TRANSPARENT	Specifies that no pixel remapping will be performed. Note that only the 8 least-significant bits of the image buffer will be displayed. (summarize)

Combination constants for [M_ADVANCED](#) (of [M_NO_TEARING](#));

You can add one of the following values to the above-mentioned value to specify which updates to display memory, if any, are skipped (not copied to the screen).

These combination values have an effect when the rate at which display memory is updated is faster than the rate at which the screen is refreshed.

You cannot use these combination values when using Direct3D.

For specifying the type of M_ADVANCED no-tearing mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NO_SKIP	Specifies that none of the updates are skipped and every update in the queue waits to be copied to the screen. Note that since only one display update is copied to the screen during a vertical synchronization pulse of the screen, this produces a lag in displaying the updates. This is the default value. (summarize)
<input type="checkbox"/> M_SKIP_NEWEST	Specifies that the newest updates in the queue are skipped and only the oldest update is copied to the screen. By reducing the number of updates to the screen, the lag in displaying updates is avoided and CPU usage is reduced. (summarize)
<input type="checkbox"/> M_SKIP_OLDEST	Specifies that the oldest updates in the queue are skipped and only the newest update is copied to the screen. By reducing the number of updates to the screen, the lag in displaying updates is avoided and CPU usage is reduced. (summarize)

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are available for all windowed displays.

For windowed displays	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_ROI_BUFFER_OFFSET_X	Sets the X offset of the ROI, in buffer coordinates. Note that this offset is not affected by panning or zooming the image in the buffer. To set the X offset in display coordinates, see M_ROI_DISPLAY_OFFSET_X . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to (Buffer SizeX - 1)	Specifies the ROI OffsetX.
<input type="checkbox"/> M_ROI_BUFFER_OFFSET_Y	Sets the Y offset of the ROI, in buffer coordinates. Note that this offset is not affected by panning or zooming the image in the buffer. To set the Y offset in display coordinates, see M_ROI_DISPLAY_OFFSET_Y . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to (Buffer SizeY - 1)	Specifies the ROI OffsetY.
<input type="checkbox"/> M_ROI_BUFFER_SIZE_X	Sets the width of the ROI, in buffer coordinates. Note that this size is not affected by panning or zooming the image in the buffer. To set the width in display coordinates, see M_ROI_DISPLAY_SIZE_X . (summarize)

<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to (Buffer SizeX - 1)	Specifies the ROI SizeX.
<input type="checkbox"/> M_ROI_BUFFER_SIZE_Y	Sets the height of the ROI, in buffer coordinates. Note that this size is not affected by panning or zooming the image in the buffer. To set the height in display coordinates, see M_ROI_DISPLAY_SIZE_Y . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to (Buffer SizeY - 1)	Specifies the ROI SizeY.
<input type="checkbox"/> M_ROI_DEFINE	Enables or disables the defining mode. In this mode, you can define a region-of-interest using the mouse, by clicking and dragging the mouse to create a rectangle. Note that the ROI is automatically shown. Once the define mode is disabled, the ROI will be hidden until M_ROI_SHOW is enabled. (summarize)
<input type="checkbox"/> M_START +	Specifies the start of the defining mode. In this mode, you can define an ROI, move it and resize it. If an ROI has already been defined, it can be modified by moving it and/or resizing it. (summarize)
<input type="checkbox"/> M_STOP	Specifies the end of a defining mode. This stops the defining mode. No further moving or resizing of the ROI is possible. This is the default value. (summarize)
<input type="checkbox"/> M_ROI_DISPLAY_OFFSET_X	Sets the X offset of the ROI, in display coordinates. The ROI will be restricted to buffer area. Note that as the image is zoomed and panned, the ROI increases in size and moves accordingly, ensuring that the area encompassed by the ROI does not change. The offset will change value in display coordinates, but will remain the same in buffer coordinates. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to (Display SizeX - 1)	Specifies the ROI OffsetX.
<input type="checkbox"/> M_ROI_DISPLAY_OFFSET_Y	Sets the Y offset of the ROI, in display coordinates. The ROI will be restricted to buffer area. Note that as the image is zoomed and panned, the ROI increases in size and moves accordingly, ensuring that the area encompassed by the ROI does not change. The offset will change value in display coordinates, but will remain the same in buffer coordinates. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to (Display SizeY - 1)	Specifies the ROI OffsetY.
<input type="checkbox"/> M_ROI_DISPLAY_SIZE_X	Sets the width of the ROI, in display coordinates. The ROI will be restricted to buffer area. Note that as the image is zoomed and panned, the ROI increases in size and moves accordingly, ensuring that the area encompassed by the ROI does not change. The offset will change value in display coordinates, but will remain the same in buffer coordinates. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to (Display SizeX - 1)	Specifies the ROI SizeX.
<input type="checkbox"/> M_ROI_DISPLAY_SIZE_Y	Sets the height of the ROI, in display coordinates. The ROI will be restricted to buffer area. Note that as the image is zoomed and panned, the ROI increases in size and moves accordingly, ensuring that the area encompassed by the ROI does not change. The offset will change value in display coordinates, but will remain the same in buffer coordinates. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to (Display SizeY - 1)	Specifies the ROI SizeY.
<input type="checkbox"/> M_ROI_HANDLE_COLOR	Sets the color used by MIL to draw the ROI sizing handles while in the defining mode. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_COLOR_BLACK .

<input type="checkbox"/> <code>M_RGB888(MIL_INT <i>red</i>, MIL_INT <i>green</i>, MIL_INT <i>blue</i>)</code>	Specifies the RGB value of the ROI sizing handles. (summarize)
	<i>Parameters</i>
	<i>red</i> Specifies the red component, as a value between 0 and 255.
	<i>green</i> Specifies the green component, as a value between 0 and 255.
	<i>blue</i> Specifies the blue component, as a value between 0 and 255.
<input type="checkbox"/> <code>M_COLOR_BLACK</code>	Specifies the color black.
<input type="checkbox"/> <code>M_COLOR_BLUE</code>	Specifies the color blue.
<input type="checkbox"/> <code>M_COLOR_BRIGHT_GRAY</code>	Specifies the color bright gray.
<input type="checkbox"/> <code>M_COLOR_CYAN</code>	Specifies the color cyan.
<input type="checkbox"/> <code>M_COLOR_DARK_BLUE</code>	Specifies the color dark blue.
<input type="checkbox"/> <code>M_COLOR_DARK_CYAN</code>	Specifies the color dark cyan.
<input type="checkbox"/> <code>M_COLOR_DARK_GREEN</code>	Specifies the color dark green.
<input type="checkbox"/> <code>M_COLOR_DARK_MAGENTA</code>	Specifies the color dark magenta.
<input type="checkbox"/> <code>M_COLOR_DARK_RED</code>	Specifies the color dark red.
<input type="checkbox"/> <code>M_COLOR_DARK_YELLOW</code>	Specifies the color dark yellow.
<input type="checkbox"/> <code>M_COLOR_GRAY</code>	Specifies the color gray.
<input type="checkbox"/> <code>M_COLOR_GREEN</code>	Specifies the color green.
<input type="checkbox"/> <code>M_COLOR_LIGHT_BLUE</code>	Specifies the color light blue.
<input type="checkbox"/> <code>M_COLOR_LIGHT_GRAY</code>	Specifies the color light gray.
<input type="checkbox"/> <code>M_COLOR_LIGHT_GREEN</code>	Specifies the color light green.
<input type="checkbox"/> <code>M_COLOR_LIGHT_WHITE</code>	Specifies the color light white.
<input type="checkbox"/> <code>M_COLOR_MAGENTA</code>	Specifies the color magenta.
<input type="checkbox"/> <code>M_COLOR_RED</code>	Specifies the color red.
<input type="checkbox"/> <code>M_COLOR_WHITE</code>	Specifies the color white.
<input type="checkbox"/> <code>M_COLOR_YELLOW</code>	Specifies the color yellow.
<input type="checkbox"/> <code>M_ROI_LINE_COLOR</code>	Sets the color used by MIL to draw the outline of the ROI. (summarize)
<input type="checkbox"/> <code>M_DEFAULT</code>	Same as M_COLOR_RED .
<input type="checkbox"/> <code>M_RGB888(MIL_INT <i>red</i>, MIL_INT <i>green</i>, MIL_INT <i>blue</i>)</code>	Specifies the RGB value of the ROI lines. (summarize)
	<i>Parameters</i>
	<i>red</i> Specifies the red component, as a value between 0 and 255.
	<i>green</i> Specifies the green component, as a value between 0 and 255.

	<i>blue</i> Specifies the blue component, as a value between 0 and 255.
<input type="checkbox"/> M_COLOR_BLACK	Specifies the color black.
<input type="checkbox"/> M_COLOR_BLUE	Specifies the color blue.
<input type="checkbox"/> M_COLOR_BRIGHT_GRAY	Specifies the color bright gray.
<input type="checkbox"/> M_COLOR_CYAN	Specifies the color cyan.
<input type="checkbox"/> M_COLOR_DARK_BLUE	Specifies the color dark blue.
<input type="checkbox"/> M_COLOR_DARK_CYAN	Specifies the color dark cyan.
<input type="checkbox"/> M_COLOR_DARK_GREEN	Specifies the color dark green.
<input type="checkbox"/> M_COLOR_DARK_MAGENTA	Specifies the color dark magenta.
<input type="checkbox"/> M_COLOR_DARK_RED	Specifies the color dark red.
<input type="checkbox"/> M_COLOR_DARK_YELLOW	Specifies the color dark yellow.
<input type="checkbox"/> M_COLOR_GRAY	Specifies the color gray.
<input type="checkbox"/> M_COLOR_GREEN	Specifies the color green.
<input type="checkbox"/> M_COLOR_LIGHT_BLUE	Specifies the color light blue.
<input type="checkbox"/> M_COLOR_LIGHT_GRAY	Specifies the color light gray.
<input type="checkbox"/> M_COLOR_LIGHT_GREEN	Specifies the color light green.
<input type="checkbox"/> M_COLOR_LIGHT_WHITE	Specifies the color light white.
<input type="checkbox"/> M_COLOR_MAGENTA	Specifies the color magenta.
<input type="checkbox"/> M_COLOR_RED	Specifies the color red.
<input type="checkbox"/> M_COLOR_WHITE	Specifies the color white.
<input type="checkbox"/> M_COLOR_YELLOW	Specifies the color yellow.
<input type="checkbox"/> M_ROI_SHOW	Enables or disables the show mode, the mode in which the ROI boundary is visible. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the ROI's boundary will be hidden. This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies that the ROI's boundary will be shown.
<input type="checkbox"/> M_WINDOW_ANNOTATIONS	Advises MIL whether annotations will be drawn directly to the window's device context (DC) using GDI functions. You must disable no-tearing (MdispControl() with M_NO_TEARING) before you can enable support for direct window annotations; otherwise, an error is generated. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is M_ENABLE except when the buffer's storage format is M_DIRECTX or its location is M_VIDEO_MEMORY (MbufAlloc...() or MbufCreate...()). In those cases, the default value is M_DISABLE . (summarize)
<input type="checkbox"/> M_DISABLE	Specifies to advise MIL that window annotations are not drawn into the DC using GDI functions. This allows MIL to use Direct3D for its display updates. If you still use GDI functions to draw, some annotations might be corrupted. No errors will be reported, but the resulting annotations' quality cannot be guaranteed. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies to advise MIL that window annotations might be drawn directly into the DC using GDI functions. If you use GDI functions (no MIL functions) to draw, MIL will ensure the annotations are displayed correctly. (summarize)
<input type="checkbox"/> M_NULL	Specifies to advise MIL that the X display connection has not been set (Linux operating system).
<input type="checkbox"/> Value	Specifies the pointer to the X display connection (Linux operating system).

<input type="checkbox"/> M_WINDOW_ROI_BUTTONS	Sets whether the ROI buttons should be present in the MIL default window. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the ROI buttons are not displayed. This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies that the ROI buttons are displayed.
<input type="checkbox"/> M_WINDOW_UPDATE_ON_PAINT	[<i>This is only applicable to Windows</i>] Sets when the display will be updated. (summarize)
<input type="checkbox"/> M_DEFAULT	Allows MIL to decide which message to receive before updating the display.
<input type="checkbox"/> M_DISABLE	Updates the display on reception of a WM_ERASEBKGD message in Windows.
<input type="checkbox"/> M_ENABLE	Updates the display on reception of a WM_PAINT message in Windows.

Combination constant for [M_START](#) (of [M_ROI_DEFINE](#));

You can add the following value to the above-mentioned value to specify the reset of the existing ROI, and the start of a new ROI definition.

● For M_ROI_DEFINE	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_RESET	Enables the reset mode. This allows resetting the existing ROI and the definition of a new one. (summarize)

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are available for displays allocated on a DMIL remote system but displayed on the master computer.

● For displays allocated on a DMIL remote system but displayed on the master computer	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_ASYNC_UPDATE	Sets whether to send display updates from the remote computer to the master computer in asynchronous or synchronous mode. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies to use the default mode, which can be selected using the MILConfig utility. In MILConfig, if you select the High quality display option, the default mode is M_DISABLE ; if you select the Optimized for bandwidth usage display option, the default mode is M_ENABLE . (summarize)
<input type="checkbox"/> M_DISABLE	Specifies to send display updates in synchronous mode. In synchronous mode, the display is updated after each modification to the displayed image buffer (or the overlay buffer), and the application waits for the data to be transferred to the display before continuing. When an image, located on a remote computer, is displayed on the master computer, the update time can significantly slow your application. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies that data will be sent asynchronously. In asynchronous mode, the required display updates are queued using multiple internal image buffers and the application continues once the update has been queued. This mode maximizes your bandwidth usage while minimizing processing delays. There will be a delay of $1 / \text{M_UPDATE_RATE_MAX}$ between two consecutive updates. (summarize)
<input type="checkbox"/> M_COMPRESSION_TYPE	[<i>For essential MIL-Lite information, see remarks.</i>] Sets how to compress data being transmitted from the remote computer to the master computer. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies to use the default compression type, which can be selected using the MILConfig utility. In MILConfig, if you select the High quality display option, compression is disabled (M_NULL); if you select the Optimized for bandwidth usage display option, compression is enabled and the compression type is set to M_JPEG_LOSSY .

	(summarize)
<input type="checkbox"/> M_JPEG2000_LOSSLESS	<p>Specifies that JPEG2000 lossless compression will be used. The supported data formats are: 1-band, 8- or 16-bit data. JPEG2000 compression supports much higher ratios of compression.</p> <p>You might need to use this compression type, for example, when there are fine details in your image or overlay buffer. Note, however, you should avoid this compression type unless absolutely necessary since compression/decompression time is high; you should only use it when the benefits of compression outweigh the overhead associated with the compression/decompression.</p> <p>(summarize)</p>
<input type="checkbox"/> M_JPEG2000_LOSSY	<p>Specifies JPEG2000 lossy compression will be used. JPEG2000 compression supports much higher ratios of compression without compromising image quality.</p> <p>You might need to use this compression type, for example, when there are fine details in your image or overlay buffer. Note, however, you should avoid this compression type unless absolutely necessary since compression/decompression time is high; you should only use it when the benefits of compression outweigh the overhead associated with the compression/decompression.</p> <p>You can manually set the compression factor using M_Q_FACTOR.</p> <p>(summarize)</p>
<input type="checkbox"/> M_JPEG_LOSSLESS	<p>Specifies JPEG lossless compression will be used. The supported data formats are: 1-band, 8- or 16-bit data.</p> <p>(summarize)</p>
<input type="checkbox"/> M_JPEG_LOSSLESS_INTERLACED	<p>Specifies that JPEG lossless compression will be used in separate fields. The supported data formats are 1-band, 8- or 16-bit data.</p> <p>(summarize)</p>
<input type="checkbox"/> M_JPEG_LOSSY	<p>Specifies that JPEG lossy compression will be used.</p> <p>You can manually set the compression factor using M_Q_FACTOR.</p> <p>(summarize)</p>
<input type="checkbox"/> M_JPEG_LOSSY_INTERLACED	<p>Specifies that JPEG lossy compression will be used in separate fields.</p> <p>You can manually set the compression factor using M_Q_FACTOR.</p> <p>(summarize)</p>
<input type="checkbox"/> M_NULL	<p>Specifies that compression is disabled.</p>
<input type="checkbox"/> M_Q_FACTOR	<p><i>[For essential MIL-Lite information, see remarks.]</i></p> <p>Sets the quantization factor. This control type is only applied when lossy compression is used.</p> <p>The Q factor is applied to all bands.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 80.</p> <p>(summarize)</p>
<input type="checkbox"/> 1 to 99	<p>Specifies the quantization factor. Only integer values are accepted. The higher the factor, the more the compression, but the lower the image quality.</p> <p>(summarize)</p>
<input type="checkbox"/> M_UPDATE_RATE_MAX	<p>Sets the maximum rate at which to update the display when it is allocated on a DMIL remote system but presented on the master computer. This control type is only respected when updating the display asynchronously (M_ASYNC_UPDATE is enabled). Between transmissions, updates are accumulated into one update.</p> <p>To update the display only after a specific number of buffer modifications, regardless of the time lapsed, use M_UPDATE_RATE_DIVIDER instead. This works whether updating the display synchronously or asynchronously, but updates are not accumulated between transmissions.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Same as M_INFINITE.</p>
<input type="checkbox"/> M_INFINITE	<p>Specifies to send updates as fast as possible (limited by the available bandwidth and transmission delay).</p>
<input type="checkbox"/> Value > 0	<p>Specifies the maximum number of updates per second. Only integer values are accepted. A minimum delay of $1/Value$ seconds will be respected between two consecutive updates.</p> <p>(summarize)</p>

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are only available for MIL windowed displays (selected using [MdispSelect\(\)](#)), and are not available for user windowed displays (selected using [MdispSelectWindow\(\)](#)):

For MIL windowed displays

ControlType	Description																								
ControlValue																									
M_FULL_SCREEN	Sets whether the window should be maximized and shown without the title bar or menu (full screen mode). (summarize)																								
M_DEFAULT	Same as M_DISABLE .																								
M_DISABLE	Specifies that the window will not be maximized and shown with the title bar and menu.																								
M_ENABLE	Specifies that the window will be maximized and shown without the title bar or menu.																								
M_WINDOW_INITIAL_POSITION_X	Sets the initial, left-most, X-coordinate of the window. (summarize)																								
Value	Specifies the X-coordinate, in pixels.																								
M_WINDOW_INITIAL_POSITION_Y	Sets the initial, top-most, Y-coordinate of the window. (summarize)																								
Value	Specifies the Y-coordinate, in pixels.																								
M_WINDOW_KEYBOARD_USE	<p>Sets whether to activate the keys associated with the display's window.</p> <p>The default key usage is:</p> <table border="1"> <tbody> <tr> <td>+</td><td>Increases the X- and Y-zoom factors.</td></tr> <tr> <td>-</td><td>Decreases the X- and Y-zoom factors.</td></tr> <tr> <td>Page-up</td><td>Scrolls the image buffer up to the previous display section.</td></tr> <tr> <td>Page-down</td><td>Scrolls the image buffer down to the next display section.</td></tr> <tr> <td>Up arrow</td><td>Scrolls the image buffer up to the previous line.</td></tr> <tr> <td>Down arrow</td><td>Scrolls the image buffer down to the next line.</td></tr> <tr> <td>Left arrow</td><td>Pans the image buffer left by one pixel.</td></tr> <tr> <td>Right arrow</td><td>Pans the image buffer right by one pixel.</td></tr> <tr> <td>Ctrl-Up arrow</td><td>Scrolls the image buffer up to the previous display section.</td></tr> <tr> <td>Ctrl-Down arrow</td><td>Scrolls the image buffer down to the next display section.</td></tr> <tr> <td>Ctrl-Left arrow</td><td>Pans the image buffer left to the previous display section.</td></tr> <tr> <td>Ctrl-Right arrow</td><td>Pans the image buffer right to the next display section.</td></tr> </tbody> </table> <p>(summarize)</p>	+	Increases the X- and Y-zoom factors.	-	Decreases the X- and Y-zoom factors.	Page-up	Scrolls the image buffer up to the previous display section.	Page-down	Scrolls the image buffer down to the next display section.	Up arrow	Scrolls the image buffer up to the previous line.	Down arrow	Scrolls the image buffer down to the next line.	Left arrow	Pans the image buffer left by one pixel.	Right arrow	Pans the image buffer right by one pixel.	Ctrl-Up arrow	Scrolls the image buffer up to the previous display section.	Ctrl-Down arrow	Scrolls the image buffer down to the next display section.	Ctrl-Left arrow	Pans the image buffer left to the previous display section.	Ctrl-Right arrow	Pans the image buffer right to the next display section.
+	Increases the X- and Y-zoom factors.																								
-	Decreases the X- and Y-zoom factors.																								
Page-up	Scrolls the image buffer up to the previous display section.																								
Page-down	Scrolls the image buffer down to the next display section.																								
Up arrow	Scrolls the image buffer up to the previous line.																								
Down arrow	Scrolls the image buffer down to the next line.																								
Left arrow	Pans the image buffer left by one pixel.																								
Right arrow	Pans the image buffer right by one pixel.																								
Ctrl-Up arrow	Scrolls the image buffer up to the previous display section.																								
Ctrl-Down arrow	Scrolls the image buffer down to the next display section.																								
Ctrl-Left arrow	Pans the image buffer left to the previous display section.																								
Ctrl-Right arrow	Pans the image buffer right to the next display section.																								
M_DISABLE	Does not activate the keys associated with the display's window.																								
M_ENABLE	<p>Activates the keys associated with the display's window.</p> <p>This is the default value. (summarize)</p>																								
M_WINDOW_MAXBUTTON	<p>[<i>This is only applicable to Windows</i>]</p> <p>Sets whether the window's maximize button is visible. (summarize)</p>																								
M_DISABLE	Specifies that the button is not visible.																								
M_ENABLE	<p>Specifies that the button is visible.</p> <p>This is the default value. (summarize)</p>																								
M_WINDOW_MENU_BAR	[<i>This is only applicable to Windows</i>]																								

	Sets whether the window's menu bar is visible. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the menu bar is not visible.
<input type="checkbox"/> M_ENABLE	Specifies that the menu bar is visible. This is the default value. (summarize)
<input type="checkbox"/> M_WINDOW_MENU_BAR_CHANGE	[<i>This is only applicable to Windows</i>] Sets whether the presence of the menu bar can be toggled. (summarize)
<input type="checkbox"/> M_DISABLE	Disallows toggling.
<input type="checkbox"/> M_ENABLE	Allows toggling. This is the default value. (summarize)
<input type="checkbox"/> M_WINDOW_MINBUTTON	[<i>This is only applicable to Windows</i>] Sets whether the window's minimize button is visible. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the button is not visible.
<input type="checkbox"/> M_ENABLE	Specifies that the button is visible. This is the default value. (summarize)
<input type="checkbox"/> M_WINDOW_MOVE	[<i>This is only applicable to Windows</i>] Sets whether window movement is allowed. (summarize)
<input type="checkbox"/> M_DISABLE	Disallows window movement.
<input type="checkbox"/> M_ENABLE	Allows window movement. This is the default value. (summarize)
<input type="checkbox"/> M_WINDOW_OVERLAP	[<i>This is only applicable to Windows</i>] Sets whether the window can be overlapped by another. (summarize)
<input type="checkbox"/> M_DISABLE	Disallows the window from being overlapped by another (keep window on top).
<input type="checkbox"/> M_ENABLE	Allows the window to be overlapped by another. This is the default value. (summarize)
<input type="checkbox"/> M_WINDOW_RESIZE	[<i>This is only applicable to Windows</i>] Sets whether window resizing is allowed. (summarize)
<input type="checkbox"/> M_DISABLE	Disallows window resizing.
<input type="checkbox"/> M_ENABLE	Allows window resizing. This is the default value.

	(summarize)
<input type="checkbox"/> M_FULL_SIZE	Forces a full-size display. Resizing is disallowed. (summarize)
<input type="checkbox"/> M_NORMAL_SIZE	Same as M_ENABLE .
<input type="checkbox"/> M_WINDOW_SCROLLBAR	[<i>This is only applicable to Windows</i>] Sets whether the scroll bars are visible. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the scroll bars are not visible.
<input type="checkbox"/> M_ENABLE	Specifies that the window's scroll bars are visible. This is the default value. (summarize)
<input type="checkbox"/> M_WINDOW_SYSBUTTON	[<i>This is only applicable to Windows</i>] Sets whether the window's system buttons are visible. The system buttons refer to the collection of the window's minimize, maximize, and close buttons. To individually set whether the minimize and maximize buttons are visible, change the M_WINDOW_MAXBUTTON or M_WINDOW_MINBUTTON control type settings. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the button is not visible.
<input type="checkbox"/> M_ENABLE	Specifies that the button is visible. This is the default value. (summarize)
<input type="checkbox"/> M_WINDOW_TITLE_BAR	Sets whether the window's title bar is visible. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the title bar is not visible.
<input type="checkbox"/> M_ENABLE	Specifies that the title bar is visible. This is the default value. (summarize)
<input type="checkbox"/> M_WINDOW_TITLE_BAR_CHANGE	Sets whether the presence of the window's title bar can be toggled. (summarize)
<input type="checkbox"/> M_DISABLE	Disallows toggling.
<input type="checkbox"/> M_ENABLE	Allows toggling. Double-clicking on the image will toggle the visibility of the application window's title bar. If the title bar is present, you can also right-click on the title bar to select Title Off . This is the default value. (summarize)
<input type="checkbox"/> M_WINDOW_ZOOM	Sets whether to allow window zoom buttons (+, -). (summarize)
<input type="checkbox"/> M_DISABLE	Disallows window zoom buttons (+, -).
<input type="checkbox"/> M_ENABLE	Allows window zooming buttons (+,-). This is the default value. (summarize)

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings can be specified only when using an auxilliary display. Otherwise, an error will occur.

• For specifying settings when using auxilliary displays

 For specifying settings when using the encoder

For specifying settings when using the encoder	
ControlType	Description
ControlValue	
M_ENCODER_FILTER	Sets a filter for luminance. (summarize)
M_DEFAULT	Same as M_LOW_PASS_0.
M_LOW_PASS_0	Specifies a low-pass filter type A.
M_LOW_PASS_1	Specifies a low-pass filter type B.
M_LOW_PASS_2	Specifies a 5.5 MHz low-pass filter. (summarize)
M_NOTCH	Specifies a subcarrier frequency low-pass filter.
M_ENCODER_PEDESTAL	Sets whether a pedestal is to be generated in the composite video signal. (summarize)
M_DEFAULT	Same as M_ENABLE.
M_DISABLE	Does not generate a pedestal in the output.
M_ENABLE	Generates a pedestal in the composite video signal.
M_SYNC_TYPE	Sets the type of output synchronization signal. This control type is only supported when the display format is RGB. (summarize)
M_DEFAULT	Specifies the default output synchronization signal.

Remarks

- *[MIL-Lite]*
MIL-Lite does not support displaying a 32-bit image buffer in a display that has its `M_VIEW_MODE` control type set to `M_AUTO_SCALE`, unless you have purchased the MIL Image Analysis license.
- *[MIL-Lite]*
Note that MIL-Lite compression support is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll; mildisplay.dll.

MdispFree

Synopsis

Free a display.

Syntax

```
void MdispFree(
    MIL_ID DisplayId
)
```

Description

This function deallocates a display previously allocated with [MdispAlloc\(\)](#).

If an image buffer was selected to the display using [MdispSelect\(\)](#), **MdispFree()** also closes the associated window for windowed displays, or leaves the display blank for auxiliary displays. If an image buffer was selected to the display using [MdispSelectWindow\(\)](#), the associated window is left open but it is left blank.

Parameter

DisplayId

Specifies the identifier of the display.

Remark

- If you are creating a DLL that includes a call to **MdispFree()**, ensure that the call is not made from the **DIIMainQ()** function, because **MdispFree()** might unload any DLL loaded with [MdispAlloc\(\)](#) and you cannot unload a DLL from **DIIMainQ()**. If necessary, call **MdispFree()** from a clean-up function in your DLL instead.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll; mildisplay.dll.

MdispHookFunction

Synopsis

Hook a function to a display event.

Syntax

```
void MdispHookFunction(
    MIL_ID DisplayId,
    MIL_INT HookType,
    MIL_DISP_HOOK_FUNCTION_PTR HookHandlerPtr,
    void *UserDataPtr
)
```

Description

This function allows you to attach or detach a user-defined function to a specified display event. Once a hook-handler function is defined and hooked to an event, it is automatically called when the event occurs.

Note that functions hooked to an event execute on a distinct thread. This permits the functions to run asynchronously from the operation that fired the event and from functions hooked to other events. Although there is a small queue to permit a certain amount of overlap, hooked functions should not take longer to execute than the period in which two of their associated events can occur. You cannot determine the instance of the event that fired the function, and even if this were possible, this information would generally not be very useful. Typically, a hooked function performs the minimum number of operations required and, if necessary, performs longer processes by launching other threads.

You can hook more than one function to an event by making separate calls to **MdispHookFunction()** for each function that you want to hook. MIL automatically chains and keeps an internal list of all these hooked functions. When a function is hooked, this new function is added to the end of the list. When the event happens, all user-defined functions in the list will be executed in the same order that they were hooked to the event. You can also remove any function from the list; in this case, MIL preserves the order of the remaining functions in the list.

Note that this function is not available on network displays.

Parameters

DisplayId
Specifies the identifier of the target display.

HookType
Specifies the display event type. This parameter can be set to the following:

● For specifying the display event type																											
<input checked="" type="checkbox"/> Value	Description	corona-II (a)	cronoplus (b)	glige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xcl (f)	helios ed/xd (g)	ieee 1394 ilbc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xcl (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xcl (u)	solios glige (v)	v/o (w)			
<input checked="" type="checkbox"/> M_FRAME_START +	Calls the hook-handler function each time a new frame is displayed. You can only hook to this event if you are using DirectDraw 7 (DirectX version 7.0) and using a windowed display. You can specify the version of DirectX to use for display using MappAlloc() with M_DX_VERSION() . (summarize)	a	b	c	d	e	f	g	h		j	k	l	m	n	o		q	r	s	t	u	v	w			
<input checked="" type="checkbox"/> M_ROI_CHANGE +	Calls the hook-handler function each time the ROI changes. When the ROI is resized using the mouse, the hooked function is called on each mouse movement. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input checked="" type="checkbox"/> M_ROI_CHANGE_END +	Calls the hook-handler function when ROI defining mode is switched from enabled to	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

disabled (that is, when in ROI defining mode and you do one of the following: click outside the ROI, click on the **ShowROI** button in the default MIL window's toolbar, or call `MdispControl()` with `M_ROI_DEFINE` and `M_STOP`).

Combination constant for the values listed in [For specifying the display event type](#)

You can add the following value to the above-mentioned values to specify to unhook a function.

● For M_FRAME_START	
☐ Value	Description
☐ M_UNHOOK	Unhooks a hooked function.

HookHandlerPtr

Specifies the address of the function that should be called when the specified event occurs. The hook-handler function, pointed to by [HookHandlerPtr](#), must be declared as follows:

```
MIL_INT MFTYPE HookHandler(  
    MIL_INT HookType,  
    MIL_ID EventId,  
    void *UserDataPtr  
)  
Parameters:  
  
    HookType  
        Type of event hooked.  
  
    EventId  
        Reserved for future use.  
  
    UserDataPtr  
        Data pointer that was passed to MdispHookFunction().
```

Upon successful completion, the hook-handler function should return `M_NULL`. Note, MFTYPE and MIL_DISP_HOOK_FUNCTION_PTR are reserved MIL predefined types for functions and data pointers.

UserDataPtr

Specifies the address of the user data that you want to make available to the hook-handler function. This address is passed to the hook-handler function, through its [UserDataPtr](#) parameter, when the specified event occurs. Set this parameter to `M_NULL` if not used.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll; mildisplay.dll.

MdispInquire

Synopsis

Inquire about a display setting.

Syntax

```
MIL_INT MdispInquire(
    MIL_ID DisplayId,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires about a specified display setting.

Parameters

DisplayId

Specifies the identifier of the display.

InquireType

Specifies the type of display setting about which to inquire. This parameter can be set to one of the following values.

Unless otherwise specified, the following **InquireType** parameter settings are available for windowed, auxiliary and network displays.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

[illegible]

The following inquire types are only available for windowed displays:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

● For windowed displays	
☐ Value	Description
☐ M_GDI_COMPATIBLE_OVERLAY	<p>Returns whether the display's overlay buffer is forced to be compatible with GDI operations. You specify this when allocating the display, using MdispAlloc(). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DISABLE Returns that the display's overlay buffer might not be compatible with GDI operations. M_ENABLE Returns that the display's overlay buffer is forced to be compatible with GDI operations.</p>
☐ M_ROI_BUFFER_OFFSET_X	<p>Returns the X offset of the ROI, in buffer coordinates. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 to (Buffer SizeX - 1); M_DEFAULT; (details)</p>
☐ M_ROI_BUFFER_OFFSET_Y	<p>Returns the Y offset of the ROI, in buffer coordinates. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 to (Buffer SizeY - 1); M_DEFAULT; (details)</p>
☐ M_ROI_BUFFER_SIZE_X	<p>Returns the width of the ROI, in buffer coordinates. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 to (Buffer SizeX - 1); M_DEFAULT; (details)</p>
☐ M_ROI_BUFFER_SIZE_Y	<p>Returns the height of the ROI, in buffer coordinates. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 to (Buffer SizeY - 1); M_DEFAULT; (details)</p>
☐ M_ROI_DEFINE	<p>Returns whether the defining mode is enabled or disabled. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_START; M_STOP; (details)</p>
☐ M_ROI_DISPLAY_OFFSET_X	<p>Returns the X offset of the ROI, in display coordinates. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 to (Display SizeX - 1); M_DEFAULT; (details)</p>
☐ M_ROI_DISPLAY_OFFSET_Y	<p>Returns the Y offset of the ROI, in display coordinates. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 to (Display SizeY - 1); M_DEFAULT; (details)</p>
☐ M_ROI_DISPLAY_SIZE_X	<p>Returns the width of the ROI, in display coordinates. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 to (Display SizeX - 1); M_DEFAULT; (details)</p>
☐ M_ROI_DISPLAY_SIZE_Y	<p>Returns the height of the ROI, in display coordinates. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 to (Display SizeY - 1); M_DEFAULT; (details)</p>
☐ M_ROI_HANDLE_COLOR	<p>Returns the color used by MIL to draw the ROI sizing handles. (summarize)</p>

	<i>UserVarPtr info</i> Return values: M_COLOR_BLACK; M_COLOR_BLUE; M_COLOR_BRIGHT_GRAY; M_COLOR_CYAN; M_COLOR_DARK_BLUE; M_COLOR_DARK_CYAN; M_COLOR_DARK_GREEN; M_COLOR_DARK_MAGENTA; M_COLOR_DARK_RED; M_COLOR_DARK_YELLOW; M_COLOR_GRAY; M_COLOR_GREEN; M_COLOR_LIGHT_BLUE; M_COLOR_LIGHT_GRAY; M_COLOR_LIGHT_GREEN; M_COLOR_LIGHT_WHITE; M_COLOR_MAGENTA; M_COLOR_RED; M_COLOR_WHITE; M_COLOR_YELLOW; (details) Byte-encoded RGB value This is a byte encoded value. To retrieve the R, G, and B components, use the M_RGB888_R, M_RGB888_G, and M_RGB888_B macros.
<input type="checkbox"/> M_ROI_LINE_COLOR	Returns the color used by MIL to draw the outline of the ROI. (summarize)
	<i>UserVarPtr info</i> Return values: M_COLOR_BLACK; M_COLOR_BLUE; M_COLOR_BRIGHT_GRAY; M_COLOR_CYAN; M_COLOR_DARK_BLUE; M_COLOR_DARK_CYAN; M_COLOR_DARK_GREEN; M_COLOR_DARK_MAGENTA; M_COLOR_DARK_RED; M_COLOR_DARK_YELLOW; M_COLOR_GRAY; M_COLOR_GREEN; M_COLOR_LIGHT_BLUE; M_COLOR_LIGHT_GRAY; M_COLOR_LIGHT_GREEN; M_COLOR_LIGHT_WHITE; M_COLOR_MAGENTA; M_COLOR_RED; M_COLOR_WHITE; M_COLOR_YELLOW; (details) Byte-encoded RGB value This is a byte encoded value. To retrieve the R, G, and B components, use the M_RGB888_R, M_RGB888_G, and M_RGB888_B macros.
<input type="checkbox"/> M_ROI_SHOW	Returns whether the ROI show mode is enabled or disabled, indicating whether the ROI is visible or hidden. (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_WINDOW_ROI_BUTTONS	Returns whether the ROI buttons should be present in the MIL default window. (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)

The following are only available for MIL windowed displays (selected using [MdispSelect\(\)](#)) and are not available for user windowed displays (selected using [MdispSelectWindow\(\)](#)):

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

● For MIL windowed displays	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FULL_SCREEN	Returns whether the window should be maximized and shown without the title bar or menu (full screen mode). (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_WINDOW_ANNOTATIONS	Returns whether annotations might be drawn directly to the device context (DC) using GDI functions. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; M_NULL; Value; (details)
<input type="checkbox"/> M_WINDOW_HANDLE	Returns the Windows handle (HWND) of the display's window.
<input type="checkbox"/> M_WINDOW_MAXBUTTON	[This is only applicable to Windows] Returns whether the window's maximize button should be visible. (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_WINDOW_MENU_BAR	[This is only applicable to Windows] Returns whether the menu bar should be present. (summarize)
	<i>UserVarPtr info</i>

	Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_WINDOW_MENU_BAR_CHANGE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Returns whether the presence of the menu bar can be toggled. (summarize)</p> <p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_WINDOW_MINBUTTON	<p>[<i>This is only applicable to Windows</i>]</p> <p>Returns whether the window's minimize button should be visible. (summarize)</p> <p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_WINDOW_MOVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Returns whether window movement is allowed. (summarize)</p> <p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_WINDOW_OVERLAP	<p>[<i>This is only applicable to Windows</i>]</p> <p>Returns whether the window can be overlapped by another. (summarize)</p> <p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_WINDOW_RESIZE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Returns whether window resizing is allowed. (summarize)</p> <p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; M_FULL_SIZE; M_NORMAL_SIZE; (details)</p>
<input type="checkbox"/> M_WINDOW_SCROLLBAR	<p>[<i>This is only applicable to Windows</i>]</p> <p>Returns whether the window's scroll bars should be visible. (summarize)</p> <p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_WINDOW_SIZE_X	Returns the width of the display's window client area.
<input type="checkbox"/> M_WINDOW_SIZE_Y	Returns the height of the display's window client area.
<input type="checkbox"/> M_WINDOW_SYSBUTTON	<p>[<i>This is only applicable to Windows</i>]</p> <p>Returns whether the window system buttons (minimize, maximize, and close buttons) should be visible. (summarize)</p> <p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_WINDOW_TITLE_BAR	<p>Returns whether the title bar should be visible. (summarize)</p> <p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_WINDOW_TITLE_BAR_CHANGE	<p>Returns whether the presence of the window's title bar can be toggled. (summarize)</p> <p><i>UserVarPtr info</i></p> <p>Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_WINDOW_ZOOM	

	Returns whether window zooming is allowed. (summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: M_DISABLE; M_ENABLE; (details)</div> </div>

The information returned by the following inquire types has no meaning unless the display is selected to the screen ([MdispSelect\(\)](#) or [MdispSelectWindow\(\)](#)):

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

● For windowed or auxiliary displays selected to the screen	
☐ Value	Description
☐ M_PAN_OFFSET_X	Returns the number of pixels in X by which the image buffer is panned. (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE Return values: Please see the XOffset parameter of MdispPan(). (details)</div> </div>
☐ M_PAN_OFFSET_Y	Returns the number of pixels in Y by which to the image buffer is panned. (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE Return values: Please see the YOffset parameter of MdispPan(). (details)</div> </div>
☐ M_SIZE_X	Returns the display's width. For windowed displays, the returned value is equal to the width of the client area of the window. For auxiliary displays, the returned width is that specified in the VCF file. (summarize)
☐ M_SIZE_Y	Returns the display's height. For windowed displays, the returned value is equal to the height of the client area of the window. For auxiliary displays, the returned height is that specified in the VCF file. (summarize)

The following inquire types are only available with [M_WINDOWED](#) displays. The information returned by these inquire types has no meaning unless the display is selected to the screen ([MdispSelect\(\)](#) or [MdispSelectWindow\(\)](#)):

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

● For windowed displays selected to the screen	
☐ Value	Description
☐ M_WINDOW_OFFSET_X	Returns the X-offset of the display's window client area, relative to the top-left of the screen (of the primary monitor).
☐ M_WINDOW_OFFSET_Y	Returns the Y-offset of the display's window client area, relative to the top-left of the screen (of the primary monitor).
☐ M_WINDOW_PAN_X	Returns the position of the horizontal scroll bar in the display's window.
☐ M_WINDOW_PAN_Y	Returns the position of the vertical scroll bar in the display's window.
☐ M_WINDOW_ZOOM_FACTOR_X	Returns the display's X-zoom factor (as controlled by zoom buttons). (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> </div>
☐ M_WINDOW_ZOOM_FACTOR_Y	Returns the display's Y-zoom factor (as controlled by zoom buttons). (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> </div>

The following inquire types are only available with windowed or auxiliary displays allocated on a remote system but displayed on the master computer.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_INT`.

For windowed or auxiliary displays allocated on a remote system but displayed on the master computer	
Value	Description
<input type="checkbox"/> M_ASYNC_UPDATE	<p>Returns whether the data from the remote computer is transmitted to the master computer asynchronously or synchronously for display. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_COMPRESSION_TYPE	<p>[For essential MIL-Lite information, see remarks.] Returns the way in which data is compressed when transmitted from the remote computer to the master computer for display. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_JPEG_LOSSLESS; M_JPEG2000_LOSSLESS; M_JPEG_LOSSLESS_INTERLACED; M_JPEG_LOSSY; M_JPEG2000_LOSSY; M_JPEG_LOSSY_INTERLACED; M_NULL; (details)</p>
<input type="checkbox"/> M_Q_FACTOR	<p>[For essential MIL-Lite information, see remarks.] Returns the quantization factor used when compressing data transmitted from the remote computer to the master computer using a lossy compression. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 1 to 99; M_DEFAULT; (details)</p>
<input type="checkbox"/> M_UPDATE_RATE_MAX	<p>Returns the maximum rate at which to send updates from remote computer to the master computer. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_INFINITE; Value > 0; (details)</p>

The following inquire types are only available with an auxiliary display.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_INT`.

For auxiliary displays		corona-II (a)	corosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ed/xd (f)	helios ed/xd (g)	ilee 1394 i1dc (h)	irs (i)	met-II /cl (j)	met-II /mc (l)	met-II /dig (k)	met-II /sdc (m)	morphis (n)	nexus qxt (o)	odyssey ea/xa (q)	odyssey ed/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ed/xd (u)	solios gige (v)	vio (w)
<input type="checkbox"/> M_AUXILIARY_KEEP_DISPLAY_ALIVE	<p>Returns whether the auxiliary display remains active after the board's video controller is freed. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)</p>																						w
<input type="checkbox"/> M_SELECT_VIDEO_SOURCE	<p>Returns whether the digitizer will use low-latency pass-through mode. (summarize)</p> <p><i>UserVarPtr info</i> Return values: MIL identifier; M_NULL; (details)</p>																						w

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type MIL_TEXT_CHAR
- MIL_DOUBLE
- MIL_ID
- MIL_INT

Specifies the address in which to write the requested information. Since the **MdisplInquire()** function also returns the requested information, you can set this parameter to **M_NULL**.

Return value

Except for the inquire types that require an address of an array, the returned value is the setting of the requested display setting, cast to *MIL_INT*. For the inquire types requiring an address of an array, the returned value is **M_NULL**.

Remark

- *[MIL-Lite]*
Note that MIL-Lite compression support is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll; mildisplay.dll.

MdispLut

Synopsis

Associate a LUT buffer with a display.

Syntax

```
void MdispLut (
    MIL_ID DisplayId,
    MIL_ID LutBufId
)
```

Description

This function associates a LUT buffer with the specified display. It can also be used to disassociate a LUT from the display.

If and when the display is selected ([MdispSelect\(\)](#)), the change required to produce the display (LUT) effect occurs. MIL checks the target display to determine whether or not a LUT is supported. If not, an error is generated.

When displaying using a LUT, note the following:

- If the LUT buffer values are changed while the image is selected on the display, the changes will not take effect until the next call is made to **MdispLut()**. That is, the LUT is not automatically updated when the LUT buffer is modified.
- You cannot select a 3-band image buffer to a display that is associated with a LUT buffer. In addition, the image buffer must be an 8- or 16-bit buffer. If these conditions are not respected, an error is generated.
- The view mode of the display cannot be set to [M_AUTO_SCALE](#) or [M_MULTI_BYTES](#).
- The number of LUT buffer entries must be the same as the maximum number of intensities that can be represented in the displayed buffer. In other words, if you want to map an 8-bit 1-band image (that is, an image that can have 256 intensities), your LUT must also have 256 entries.

To disassociate a LUT buffer from a display, call **MdispLut()** with [M_DEFAULT](#).

For more information, see the [Lookup tables in general](#) section in [Chapter 19: Lookup tables](#) and the [Mapping 1-band images through a LUT upon display](#) section in [Chapter 20: Displaying an image](#).

Parameters

DisplayId

Specifies the identifier of the display with which to associate the LUT buffer.

LutBufId

Specifies a LUT buffer. This parameter should be set to one of the following values:

● For the identifier of the previously allocated LUT buffer	
☐ Value	Description
☐ M_DEFAULT	Disassociates a LUT buffer from the specified display.
☐ M_PSEUDO	Specifies the default pseudo-color LUT buffer as the LUT buffer to associate with the display. When this LUT is used and displaying an 8-bit image buffer, each gray intensity is displayed in a different color. (summarize)
☐ MIL LUT buffer identifier	Specifies the identifier of the custom LUT buffer (allocated with MbufAlloc1d() or MbufAllocColor() with M_LUT) to associate with the display. If you associate a three-band LUT buffer (RGB) with the display, and then select an image buffer to the display, the same image data is mapped through each band of the LUT, typically resulting in a color display effect (depends on the LUT values). Note that you cannot use a custom LUT buffer with a network display . (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll; mildisplay.dll.

MdispPan

Synopsis

Pan (scroll) a display.

Syntax

```
void MdispPan(
    MIL_ID DisplayId,
    MIL_DOUBLE XOffset,
    MIL_DOUBLE YOffset
)
```

Description

This function associates X- and Y-panning offsets with the specified display. When an image buffer is selected for display, the image is displaced on the display, from the top-left corner of the window (for windowed displays) or the screen (for auxiliary displays), according to these offsets.

Note, the offsets ([XOffset](#) and [YOffset](#)) are in image pixels. Since the current zoom factors affect the displayed size of an image pixel, the panning offsets are also affected by the zoom factors. For example, if the display has an associated X-zoom factor of 4, panning by an X-offset of one image pixel results in panning by 4 pixels in the horizontal direction on the display.

To center the selected image buffer in the display, use [MdispControl\(\)](#) with [M_CENTER_DISPLAY](#), instead of using **MdispPan()**.

Note that this function is not available on network displays.

Parameters

- DisplayId**
- Specifies the identifier of the display.
- XOffset**
- Specifies the number of image pixels by which to pan an image buffer horizontally when it is displayed. Specify the offset relative to the top-left corner of the image buffer. Specify a positive offset value to displace the image to the left.
- YOffset**
- Specifies the number of image pixels by which to pan an image buffer vertically when it is displayed. Specify the offset relative to the top-left corner of the image buffer. Specify a positive offset value to displace the image upwards.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll; mildisplay.dll.

MdispSelect

Synopsis

Select an image buffer to display.

Syntax

```
void MdispSelect(
    MIL_ID DisplayId,
    MIL_ID ImageBufId
)
```

Description

This function outputs the content of the specified image buffer to the specified MIL display. You can only display one image buffer at a time on a specific display.

For a windowed display, the image is presented in a window that has the same size as the image, unless such an window cannot fit on the desktop. In which case, the window will have scroll bars to view other parts of the image. If the window is resized, causing it not to fit on the desktop, scroll bars will appear.

For an auxiliary display, the image is presented at the top-left corner of an auxiliary screen. The border outside the image is blanked out if the image is smaller in size than the screen area (if the hardware supports this). Otherwise, the right and bottom portion of the image, the part that exceeds the display size, is not displayed.

You can control the displayed area using `MdispControl()`, `MdispPan()`, or `MdispZoom()`. For example, you can center the selected image in the display using `MdispControl()` with `M_CENTER_DISPLAY`.

To remove an image buffer selected to the display using `MdispSelect()`, you can use `MdispSelect()` with `M_NULL`. For windowed displays, this closes the associated window; for auxiliary displays, this leaves the display blank.

You can only remove the entire image buffer from the display. Therefore, when displaying a parent buffer, you cannot remove one of its child buffers from the display.

Note that it is not necessary to stop displaying the image buffer before selecting another buffer for display.

Parameters

DisplayId

Specifies the identifier of the display.

ImageBufId

Specifies the image buffer to display. To be displayable, this buffer must be an image buffer that has an `M_IMAGE` + `M_DISP` attribute.

Set this parameter to `M_NULL` to stop displaying the currently selected image buffer on the specified display.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll; mildisplay.dll.

MdispSelectWindow

Synopsis

Select an image buffer to display in a user-defined window.

Syntax

```
void MdispSelectWindow(
    MIL_ID DisplayId,
    MIL_ID ImageBufId,
    HWND ClientWindowHandle
)
```

Description

This function outputs the content of the specified image buffer to the specified user-defined window, using the specified MIL display.

If the specified image buffer is smaller in size than the target window size, the border outside the image is not modified. If the specified buffer is larger in size than the target window, the right and bottom portion of the buffer, the part that exceeds the window, is not displayed.

You can control the displayed area using [MdispControl\(\)](#), [MdispPan\(\)](#), or [MdispZoom\(\)](#). For example, you can center the selected image in the display using [MdispControl\(\)](#) with [M_CENTER_DISPLAY](#).

To remove an image buffer selected to the display using [MdispSelectWindow\(\)](#), you can use [MdispSelectWindow\(\)](#) with [M_NULL](#). This leaves the associated window open but leaves it blank.

This function is valid only in a Windows environment.

This function does not support remote displays ([MdispAlloc\(\)](#) with [M_REMOTE_DISPLAY](#)).

Parameters

- DisplayId**
- Specifies the identifier of the display.
- ImageBufId**
- Specifies the image buffer to display. To be displayable, this buffer must be an image buffer that has an [M_IMAGE](#) + [M_DISP](#) attribute.
- Set this parameter to [M_NULL](#) to stop displaying the currently selected image buffer on the specified display.
- ClientWindowHandle**
- Specifies the handle of the user-defined window or child window. This window must have been created with the Windows API functions. If this parameter is set to zero, this function behaves like [MdispSelect\(\)](#).
- Set this parameter to [M_NULL](#) if the [ImageBufId](#) parameter is set to [M_NULL](#).

Compilation information

Header	Include windows.h; mil.h (windows.h must be included first).
Library	Use mil.lib.
DLL	Requires mil.dll; mildisplay.dll.

MdispZoom

Synopsis

Zoom a display.

Syntax

```
void MdispZoom(  
    MIL_ID DisplayId,  
    MIL_DOUBLE XFactor,  
    MIL_DOUBLE YFactor  
)
```

Description

This function associates a zoom factor in X and/or in Y with the specified display. When an image buffer is selected for display, it will be zoomed according to these factors. The image buffer is displayed starting from its top-left corner, unless panning is also associated with the display (using [MdispPan\(\)](#)).

Instead of explicitly specifying a zoom factor, you can automatically scale the selected image buffer to fit the display, using [MdispControl\(\)](#) with [M_SCALE_DISPLAY](#) or [M_FILL_DISPLAY](#). [M_SCALE_DISPLAY](#) maintains the image aspect ratio, whereas [M_FILL_DISPLAY](#) doesn't. Both these control types override the explicitly specified zoom factors ([MdispZoom\(\)](#)), disable the window's zoom buttons ([M_WINDOW_ZOOM](#)), and override pan settings ([MdispPan\(\)](#)).

Note that this function is not available on network displays.

Parameters

DisplayId

Specifies the identifier of the display.

XFactor

Sets the X-zoom factor. This parameter should be set to one of the following values:

For specifying the X-zoom factor	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 0.0 < Value <= 16.0	Specifies the zoom factor.

YFactor

Sets the Y-zoom factor. This parameter should be set to one of the following values:

For specifying the Y-zoom factor	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 0.0 < Value <= 16.0	Specifies the zoom factor.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll; mldisplay.dll.

Medge functions

Synopsis

The functions prefixed with Medge make up the Edge Finder module. The Edge Finder module provides powerful operations that allow you to extract different types of edges in images, such as object contours or thin curvilinear shapes. The module offers many settings including image processing controls, feature controls, and annotations controls. It also provides analysis capabilities allowing for the advanced manipulations of results. The Edge Finder module can also be used in conjunction with the MIL Geometric Model Finder module; that is, you can define models from the result of an edge extraction, or you can find occurrences in the result of an edge extraction. The Edge Finder module offers complete support for calibration. Extraction of edges can be performed in the calibrated real-world such that, even without physically correcting your images, results are in real-world units.

Functions

- [MedgeAlloc](#)
- [MedgeAllocResult](#)
- [MedgeCalculate](#)
- [MedgeControl](#)
- [MedgeDraw](#)
- [MedgeFree](#)
- [MedgeGetNeighbors](#)
- [MedgeGetResult](#)
- [MedgeInquire](#)
- [MedgeMask](#)
- [MedgePut](#)
- [MedgeRestore](#)
- [MedgeSave](#)
- [MedgeSelect](#)
- [MedgeStream](#)

MedgeAlloc

Synopsis

Allocate an Edge Finder context.

Syntax

```
MIL_ID MedgeAlloc(  
    MIL_ID SystemId,  
    MIL_INT EdgeFinderType,  
    MIL_INT ControlFlag,  
    MIL_ID *ContextIdPtr  
)
```

Description

This function allocates an Edge Finder context on the specified system. An Edge Finder context contains all the information necessary to perform an [MedgeCalculate\(\)](#) operation, including global processing settings, and edge features to calculate. When the Edge Finder context is no longer required, you should release its memory, using [MedgeFree\(\)](#).

Edge Finder context settings can be adjusted using [MedgeControl\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the Edge Finder context.

This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

EdgeFinderType

Specifies the type of Edge Finder context. This parameter must be set to one of the values below.

● For the Edge Finder context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CONTOUR	Specifies a contour context type, which is used to find object contours in images.
<input type="checkbox"/> M_CREST	Specifies a crest context type, which is used to find thin line crests in images.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the Edge Finder context identifier. Since the **MedgeAlloc()** function also returns the Edge Finder context identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the Edge Finder context identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeAllocResult

Synopsis

Allocate an Edge Finder result buffer.

Syntax

```
MIL_ID MedgeAllocResult(  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *EdgeResultIdPtr  
)
```

Description

This function allocates an Edge Finder result buffer, on the specified system, to store results obtained from an [MedgeCalculate\(\)](#) operation. When the Edge Finder result buffer is no longer required, release its memory, using [MedgeFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the Edge Finder result buffer.

This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

EdgeResultIdPtr

Specifies the address of the variable in which to write the Edge Finder result buffer identifier. Since the **MedgeAllocResult()** function also returns the Edge Finder result buffer identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the result identifier if the allocation is successful. If allocation fails, M_NULL is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeCalculate

Synopsis

Perform edge extraction and feature calculations.

Syntax

```
void MedgeCalculate(
    MIL_ID ContextId,
    MIL_ID SourceImageId,
    MIL_ID SourceDeriv1Id,
    MIL_ID SourceDeriv2Id,
    MIL_ID SourceDeriv3Id,
    MIL_ID EdgeResultId,
    MIL_INT ControlFlag
)
```

Description

This function extracts edges and calculates edge features in the source image buffer or in the derivative image buffers. You can also use this function to perform post-calculations on included edges of an Edge Finder result buffer.

Results are stored in the specified Edge Finder result buffer. Select which edge features to calculate with [MedgeControl\(\)](#). Include, exclude, or delete edges that meet a specified criterion with [MedgeSelect\(\)](#). Note that to calculate new features and/or select features of a new set of edges, **MedgeCalculate()** must be called after [MedgeControl\(\)](#) and/or [MedgeSelect\(\)](#).

If the source image or derivative images are calibrated, results are calculated in the world coordinate system, otherwise they are calculated in the image coordinate system.

If the source image or derivative images are calibrated, you can retrieve results in real-world or in pixel units. However, in the presence of distortion, some results are meaningless when converted from real-world to pixel units (for example, Feret angles). For example, if an edge appears warped in the source image, but the calibration object of the source image compensates for this during the extraction, the resulting Feret angles are meaningful in the real-world coordinate system, and meaningless in the image coordinate system.

Calculations are typically done on a source image buffer ([SourceImageId](#)), where **MedgeCalculate()** calculates the required derivative image buffers that are used to extract edges. In this case, the derivative image buffers ([SourceDeriv1Id](#), [SourceDeriv2Id](#), and [SourceDeriv3Id](#)) must be set to **M_NULL**. However, if you specify your own derivative image buffers, the edge extraction process is done with them instead. In this case, [SourceImageId](#) must be set to **M_NULL**. Note that, when specifying your own derivative image buffers, the majority of processing control settings remain in effect, except for filter settings (for example, [M_FILTER_SMOOTHNESS](#)).

When calculated by Edge Finder, derivative buffers are always consistent (for example, they have the same scaling factor) and are always normalized to 10-bit images such that the maximum output value is 1024 for 8-bit source images. When you provide the derivative buffers, they should respect these limitations for best performance.

To avoid repeatedly calculating time-consuming edge features for all edges in a source image, **MedgeCalculate()** can also perform post-calculations on included edges of an Edge Finder result buffer. That is, once the results of a calculation have been stored in an Edge Finder result buffer, you can select a subset of edges from that Edge Finder result buffer (with [MedgeSelect\(\)](#)) and/or add new features for calculation to that Edge Finder result buffer (with [MedgeControl\(\)](#)), and then perform the post-calculation with **MedgeCalculate()**. In this case, processing time greatly decreases since calculations are not unnecessarily repeated. The operation of selecting edges and adding new features can be repeated until the required result is calculated. To perform post-calculations on the included edges of an Edge Finder result buffer, set [SourceImageId](#), [SourceDeriv1Id](#), [SourceDeriv2Id](#), and [SourceDeriv3Id](#) to **M_NULL**.

Note that there are some restrictions when post-calculating edges; for more information, see the [Post-calculation](#) subsection in the [Calculating and retrieving results](#) section in [Chapter 9: Edge Finder](#).

Parameters

ContextId

Specifies the Edge Finder context to use for the extraction. The Edge Finder context must have been previously allocated on the required system using [MedgeAlloc\(\)](#).

SourceImageId

Specifies the source image buffer from which to extract edges. Typically, the source image buffer should be 1-band 8-bit unsigned. Other buffer depths and types are generally accepted, but can slightly decrease performance. Note that 3-band source image buffers are only supported when extracting edge contours. When the source image buffer is 32-bit float, Edge Finder uses floating-point precision calculations.

If you specify your own derivative image buffers, or if you are performing post-calculations on the included edges of an Edge Finder result buffer, this parameter must be set to **M_NULL**.

SourceDeriv1Id

Specifies the X-source derivative image buffer used to extract edges. When the Edge Finder context type is [M_CONTOUR](#), [SourceDeriv1Id](#) specifies the first derivative of the image in the X-direction. When the Edge Finder context type is [M_CREST](#), [SourceDeriv1Id](#) specifies the second derivative of the image in the X-direction.

Derivative image buffers must be 1-band 16-bit signed or 1-band 32-bit float. In addition, all derivative image buffers must be consistent (for example, they must have the same scaling factor), and they must be of the same type and size. When the derivative image buffers are 32-bit float, Edge Finder uses floating-point precision calculations.

If you specify a source image buffer, or if you are performing post-calculations on the included edges of an Edge Finder result buffer, this parameter must be set to **M_NULL**.

SourceDeriv2Id

Specifies the Y-source derivative image buffer used to extract edges. When the Edge Finder context type is [M_CONTOUR](#), [SourceDeriv2Id](#) specifies the first derivative of the image in the Y-direction. When the Edge Finder context type is [M_CREST](#), [SourceDeriv2Id](#) specifies the second derivative of the image in the Y-direction.

Derivative image buffers must be 1-band 16-bit signed or 1-band 32-bit float. In addition, all derivative image buffers must be consistent (for example, they must have the same scaling factor), and they must be of the same type and size. When the derivative image buffers are 32-bit float, Edge Finder uses floating-point precision calculations.

If you specify a source image buffer, or if you are performing post-calculations on the included edges of an Edge Finder result buffer, this parameter must be set to **M_NULL**.

SourceDeriv3Id

Specifies the source cross-derivative image buffer used to extract edges. The cross-derivative can only be specified if the Edge Finder context type is [M_CREST](#). If the Edge Finder context type is [M_CONTOUR](#), set [SourceDeriv3Id](#) to **M_NULL**.

Derivative image buffers must be 1-band 16-bit signed or 1-band 32-bit float. In addition, all derivative image buffers must be consistent (for example, they must have the same scaling factor), and they must be of the same type and size. When the derivative image buffers are 32-bit float, Edge Finder uses floating-point precision calculations.

If you specify a source image buffer, or if you are performing post-calculations on the included edges of an Edge Finder result buffer, this parameter must be set to **M_NULL**.

EdgeResultId

Specifies the Edge Finder result buffer in which to write the results of the extraction. The Edge Finder result buffer must have been previously allocated on the required system using [MedgeAllocResult\(\)](#).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD; Matrox Solios GigE; Matrox Solios eA/XA; Matrox Solios eCL/XCL]
The Edge Finder result buffer ([EdgeResultId](#)) must be allocated on the same system as the Edge Finder context ([ContextId](#)). If it is not, an error will occur.

ControlFlag

Specifies whether optimal performance during Edge Finder calculations is ensured. This parameter can only be used when providing derivative image buffers. When providing a source image buffer, optimal performance is always ensured; therefore, this parameter must be set to **M_DEFAULT**.

When providing derivative image buffers, this parameter must be set to one of the following values:

● For the derivative image buffers	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that optimal performance is not ensured.
<input type="checkbox"/> M_NO_CHECK	Specifies that optimal performance is ensured. M_NO_CHECK should only be used if each derivative image buffer has pixel values that do not go beyond the range of positive or negative 1024 (that is, 10-bit signed values). This ensures optimal performance since Edge Finder does not have to verify unnecessary values. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeControl

Synopsis

Control an Edge Finder context or an Edge Finder result buffer setting.

Syntax

```
void MedgeControl(
    MIL_ID ContextOrResultId,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function sets the specified control for either an Edge Finder context, or an Edge Finder result buffer. For Edge Finder contexts, these settings control the execution of [MedgeCalculate\(\)](#) operations and select which edge features [MedgeCalculate\(\)](#) should calculate. For Edge Finder result buffers, these settings control the post manipulation of results. For example, to draw a zoomed region of the source image that was used to calculate results, the drawing control values must be appropriately set. Similarly, to select edges based on the proximity of an edge or edges to a specified point, [M_NEAREST_NEIGHBOR_RADIUS](#) must be appropriately set. For more information, see [MedgeDraw\(\)](#) or [MedgeSelect\(\)](#).

All control settings can typically be inquired with [MedgeInquire\(\)](#).

For new context settings to take effect, you must calculate the settings, using [MedgeCalculate\(\)](#).

Note that some control settings have post-calculation restrictions. For more information, see the [Post-calculation](#) subsection in the [Calculating and retrieving results](#) section in [Chapter 9: Edge Finder](#).

By setting the [ControlType](#) parameter to [M_INTERACTIVE](#), you can set the control types interactively.

Parameters

ContextOrResultId

Specifies either the Edge Finder context or the Edge Finder result buffer whose settings you want to modify. The Edge Finder context or the Edge Finder result buffer must have been previously allocated on the required system using [MedgeAlloc\(\)](#) or [MedgeAllocResult\(\)](#), respectively.

ControlType

Specifies the setting to change.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the setting's new value.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

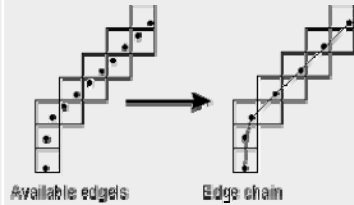
- [For opening an interactive dialog box](#)
- [For operation settings for object contours and line crests](#)
- [For operation settings for line crests](#)
- [For saving internal buffers in the Edge Finder result buffer](#)
- [For calculating edge features](#)

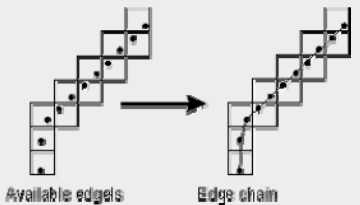
- For calculating Feret values
- For selecting groups of edge features
- For performing post-calculations on extracted edges
- For a result buffer
- For setting closest-edgel constraints

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings can be specified for an Edge Finder context.

For opening an interactive dialog box		
<input type="checkbox"/> ControlType	Description	
ControlValue		
<input type="checkbox"/> M_INTERACTIVE	<p>[This is only applicable to Windows]</p> <p>For an Edge Finder context, this setting opens a dialog box that allows you to edit the control types of the specified Edge Finder context interactively.</p> <p>(summarize)</p>	
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.	

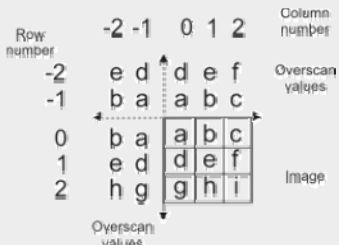
The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are used to control the Edge Finder context operation settings and can be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts, unless otherwise specified.

For operation settings for object contours and line crests		
<input type="checkbox"/> ControlType	Description	
ControlValue		
<input type="checkbox"/> M_ACCURACY	<p>Sets the edgel accuracy of the edge extraction. Accuracy depends on the image's dynamic range, sharpness, and noise. The best accuracy is achieved in well-contrasted noise-free images.</p> <p>(summarize)</p>	
<input type="checkbox"/> M_DEFAULT	Same as M_HIGH .	
<input type="checkbox"/> M_DISABLE	<p>Specifies that edgel accuracy will be disabled. Edgels will be calculated with pixel accuracy.</p> <p>(summarize)</p>	
<input type="checkbox"/> M_HIGH	<p>Specifies high accuracy. Edgels will be calculated with subpixel accuracy.</p> <p>(summarize)</p>	
<input type="checkbox"/> M_VERY_HIGH	<p>Specifies very high accuracy. Edgels will be calculated with very precise subpixel accuracy. M_VERY_HIGH uses a classical camera model to compensate for any pixel-distortion aberration.</p> <p>(summarize)</p>	
<input type="checkbox"/> M_CHAIN_ALL_NEIGHBORS	<p>Sets how edge chains are built. Edge chains are built using an 8-connected lattice.</p> <p>(summarize)</p>	
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .	
<input type="checkbox"/> M_DISABLE	<p>Specifies that edge chains are built with the least amount of edgel information possible.</p> <div data-bbox="583 1161 934 1364">  <p>Available edgels Edge chain</p> </div> <p>(summarize)</p>	
<input type="checkbox"/> M_ENABLE	<p>Specifies that edge chains are built with as much edgel information as possible.</p> <p>Enabling M_CHAIN_ALL_NEIGHBORS can result in slightly longer calculations, however, the edge chain will contain more edgel information.</p>	

 <p>Available edgels</p> <p>Edge chain</p> <p>(summarize)</p>	
<input type="checkbox"/> <code>M_DETAIL_LEVEL</code>	<p>Sets the level of details to extract from the image. The detail level determines what is considered an edge/background. A higher detail level will include more edges than a lower detail level.</p> <p>Essentially, <code>M_DETAIL_LEVEL</code> sets the threshold mode of the Edge Finder context. Note that an <code>M_DETAIL_LEVEL</code> setting overrides an <code>M_THRESHOLD_MODE</code> setting.</p> <p>Typically, <code>M_DETAIL_LEVEL</code> is used when interfacing with the MIL Geometric Model Finder module. Otherwise, <code>M_THRESHOLD_MODE</code> should be used instead.</p> <p>(summarize)</p>
<input type="checkbox"/> <code>M_DEFAULT</code>	Same as <code>M_MEDIUM</code> .
<input type="checkbox"/> <code>M_HIGH</code>	Sets the detail level to high.
<input type="checkbox"/> <code>M_MEDIUM</code>	Sets the detail level to medium.
<input type="checkbox"/> <code>M_VERY_HIGH</code>	Sets the detail level to very high.
<input type="checkbox"/> <code>M_EXTRACTION_SCALE</code>	<p>Sets the scale of the image at which to do the edge extraction. Once the extraction is complete, the results are scaled to the original scale of the image.</p> <p>An extraction scale less than one speeds up the calculation or the search but can result in a less reliable result, including, the loss of important details and/or a reduction in the accuracy of the search results.</p> <p><code>M_EXTRACTION_SCALE</code> is for advanced users of the EdgeFinder module. The default setting usually provides the most accurate search results.</p> <p>(summarize)</p>
<input type="checkbox"/> <code>M_DEFAULT</code>	Specifies the default value. The default value is 1.0.
<input type="checkbox"/> Value > 0	Specifies the extraction scale.
<input type="checkbox"/> <code>M_FILTER_MODE</code>	<p>Sets the mode in which to perform the edge extraction filter. Typically, <code>M_FILTER_MODE</code> only affects <code>M_DERICHE</code> and <code>M_SHEN</code> filter types. The filter type is set using <code>MedgeControl()</code> with <code>M_FILTER_TYPE</code>.</p> <p>(summarize)</p>
<input type="checkbox"/> <code>M_DEFAULT</code>	<p>Specifies the default filter mode. The default is typically set to the most efficient mode.</p> <p>If special hardware is available to perform the convolution, the default is <code>M_KERNEL</code>.</p> <p>If special hardware is not available to perform the convolution, the default is <code>M_RECURSIVE</code>.</p> <p>(summarize)</p>
<input type="checkbox"/> <code>M_KERNEL</code>	<p>Specifies the use of a non-recursive implementation of the filter.</p> <p>In this case, a kernel is used to perform the neighborhood operation. To set the size of the filter's convolution kernel, use <code>MedgeControl()</code> with <code>M_KERNEL_WIDTH</code>. Note that the size of the kernel can be constrained by the available hardware resources.</p> <p>When using an Infinite Impulse Response (IIR) filter in this mode, the filtering is done using a kernel approximation (Finite Impulse Response) of the filter.</p> <p>Typically, you would only use <code>M_KERNEL</code> if dedicated hardware is available to perform the convolution.</p> <p>When using an IIR filter, this mode is not as efficient as <code>M_RECURSIVE</code> for large kernels.</p> <p>(summarize)</p>
<input type="checkbox"/> <code>M_RECURSIVE</code>	Specifies the use of a recursive implementation of an Infinite Impulse Response (IIR) filter, when applicable.

	<p>If this mode actually used a kernel, the kernel size would be theoretically infinite.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FILTER_SMOOTHNESS	<p>Sets the degree of smoothness (strength of denoising) applied by the filter during the neighborhood operation.</p> <p>M_FILTER_SMOOTHNESS only has an effect if MedgeControl() with M_FILTER_TYPE is set to M_DERICHE or M_SHEN.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 50.0.</p> <p>(summarize)</p>
<input type="checkbox"/> 0.0 to 100.0	<p>Specifies the smoothness value.</p> <p>A value of 100.0 results in a strong noise reduction effect, while a value of 0.0 has almost no noise reduction effect.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FILTER_TYPE	<p>Sets the type of filter used when performing the neighborhood operation used to extract edges. The type of filter determines the distribution of the neighborhoods' influence.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Same as M_SHEN.</p>
<input type="checkbox"/> M_DERICHE	<p>Specifies a Canny-Deriche infinite support filter. This is an exponential weighting function, of the general form:</p> $k(a n + 1)e^{-a n }$ <p>Canny-Deriche is an Infinite Impulse Response (IIR) filter.</p> <p>For the Canny-Deriche filter, the neighborhoods' influence decreases much slower as the distance from the central pixel increases, compared to the Shen-Castan filter (M_SHEN).</p> <p>M_DERICHE can be used with both M_CONTOUR and M_CREST Edge Finder contexts. Typically, M_DERICHE is used for unusually thick crests.</p> <p>M_DERICHE allows you to use Edge Finder's smoothing capabilities. To do so, use MedgeControl() with M_FILTER_SMOOTHNESS.</p> <p>M_DERICHE allows you to control the filter mode used to calculate edges. To do so, use MedgeControl() with M_FILTER_MODE.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FREI_CHEN	<p>Specifies a Frei Chen filter. This is a Finite Impulse Response (FIR) filter that can be represented with the following convolution kernels:</p> $\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix}$ <p>M_FREI_CHEN can only be used with M_CONTOUR Edge Finder contexts. Also, when using M_FREI_CHEN, you cannot smooth images using MedgeControl() with M_FILTER_SMOOTHNESS.</p> <p>(summarize)</p>
<input type="checkbox"/> M_PREWITT	<p>Specifies a Prewitt filter. This is a Finite Impulse Response (FIR) filter that can be represented with the following convolution kernels:</p> $\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$ <p>M_PREWITT can only be used with M_CONTOUR Edge Finder contexts. Also, when using M_PREWITT, you cannot smooth images using MedgeControl() with M_FILTER_SMOOTHNESS.</p> <p>(summarize)</p>
<input type="checkbox"/> M_SHEN	<p>Specifies a Shen-Castan infinite support exponential filter. This is an exponential weighting function, of the general form:</p> $Ke^{-\beta n }$

	<p>Shen-Castan is an Infinite Impulse Response (IIR) filter.</p> <p>For the Shen-Castan filter, the neighborhoods' influence decreases much faster as the distance from the central pixel increases, compared to the Canny-Deriche filter (M_DERICHE).</p> <p>M_SHEN can be used with both M_CONTOUR and M_CREST Edge Finder contexts. Typically, M_SHEN performs an excellent edge extraction on most images; however, if you are extracting unusually thick crests that yield inappropriate results, you should use M_DERICHE.</p> <p>M_SHEN allows you to use Edge Finder's smoothing capabilities. To do so, use MedgeControl() with M_FILTER_SMOOTHNESS.</p> <p>M_SHEN allows you to control the filter mode used to calculate edges. To do so, use MedgeControl() with M_FILTER_MODE. (summarize)</p>
<input type="checkbox"/> M_SOBEL	<p>Specifies a Sobel filter. This is a Finite Impulse Response (FIR) filter that can be represented with the following convolution kernels:</p> $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ <p>M_SOBEL can only be used with M_CONTOUR Edge Finder contexts. Also, when using M_SOBEL, you cannot smooth images using MedgeControl() with M_FILTER_SMOOTHNESS. (summarize)</p>
<input type="checkbox"/> M_FLOAT_MODE	<p>Sets whether to force all the processing of edge extraction operations to be performed using floating-point precision calculations. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that all edge extractions are not forced to be performed using floating-point precision calculations.
<input type="checkbox"/> M_ENABLE	Specifies that all edge extractions are forced to be performed using floating-point precision calculations.
<input type="checkbox"/> M_KERNEL_DEPTH	<p>Sets the depth of the convolution kernel used for the kernel filtering mode. M_KERNEL_DEPTH is ignored unless MedgeControl() with M_FILTER_MODE is set to M_KERNEL.</p> <p>Typically, M_KERNEL_DEPTH only affects Infinite Impulse Response (IIR) type filters, which you can set using MedgeControl() with M_FILTER_TYPE. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 8 bit. (summarize)
<input type="checkbox"/> 8	Specifies an 8-bit kernel depth. This is faster, though less accurate, than using a 16-bit kernel depth. (summarize)
<input type="checkbox"/> 16	Specifies a 16-bit kernel depth. This is slower, though more accurate, than using an 8-bit kernel depth. (summarize)
<input type="checkbox"/> M_KERNEL_WIDTH	<p>Sets the maximum X and Y size of the convolution kernel used for kernel filtering modes.</p> <p>The size of the kernel can be constrained by the available hardware resources. Larger kernels result in longer processing times.</p> <p>M_KERNEL_WIDTH is ignored unless MedgeControl() with M_FILTER_MODE is set to M_KERNEL.</p> <p>Typically, M_KERNEL_WIDTH only affects Infinite Impulse Response (IIR) type filters, which you can set using MedgeControl() with M_FILTER_TYPE. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_AUTO .
<input type="checkbox"/> M_AUTO	<p>Specifies the maximum X and Y size of the convolution kernel automatically, for a given smoothness factor. You can set the smoothness factor using MedgeControl() with M_FILTER_SMOOTHNESS.</p> <p>The size of the kernel, for a given smoothness factor, is calculated as the size beyond which the quantified filter values are zero. If the smoothness factor calls for a kernel size that is beyond the limits of your hardware, the smoothness will be sacrificed to achieve the largest possible kernel size. (summarize)</p>
<input type="checkbox"/> Value >= 3	Specifies the maximum X and Y size of the convolution kernel directly. Only odd integer values are accepted.

	<p>If your hardware cannot handle the kernel size, the effective smoothness factor will be saturated to the maximum possible value for the specified kernel size. You can set the smoothness factor using MedgeControl() with M_FILTER_SMOOTHNESS. (summarize)</p>
<input type="checkbox"/> M_MAGNITUDE_TYPE	<p>Sets how to calculate the magnitude of the edge at each edgel position.</p> <p>Note that for M_CONTOUR Edge Finder contexts, the magnitude is the norm of the gradient vector at the edgel position. For M_CREST Edge Finder contexts, the magnitude is equal to the maximum eigenvalue of the Hessian matrix at the edgel position. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value.</p> <p>For M_CONTOUR Edge Finder contexts, the default is M_SQR_NORM. For M_CREST Edge Finder contexts, the default is M_NORM. (summarize)</p>
<input type="checkbox"/> M_NORM	<p>Specifies that the magnitude will be used.</p>
<input type="checkbox"/> M_SQR_NORM	<p>Specifies that the square of the magnitude will be used.</p> <p>This value optimizes the edge extraction operation while still preserving a very good edgel position accuracy. M_SQR_NORM is calculated faster than M_NORM, though it is less accurate. (summarize)</p>
<input type="checkbox"/> M_OVERSCAN	<p>Sets the type of overscan used to handle the source image's bordering pixels.</p> <p>M_OVERSCAN is ignored if MedgeControl() with M_FILTER_MODE is set to M_RECURSIVE. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Same as M_MIRROR.</p>
<input type="checkbox"/> M_DISABLE	<p>Specifies that no overscan will be used. When overscan is disabled, the border pixels of the source image are not processed by the neighborhood operation if additional processing time is needed. (summarize)</p>
<input type="checkbox"/> M_MIRROR	<p>Specifies that the border pixels of a source image are processed using overscan pixel values that mirror the source buffer pixel values. That is, the overscan pixel values will be a mirror copy of the source buffer's borders. For example:</p>  <p>(summarize)</p>
<input type="checkbox"/> M_REPLACE	<p>Specifies that the border pixels of a source image are processed using overscan pixel values set to the overscan replacement value (MedgeControl() with M_OVERSCAN_REPLACE_VALUE).</p>
<input type="checkbox"/> M_TRANSPARENT	<p>Specifies that the border pixels of a source image are processed using transparent overscan pixel values. That is, the overscan pixel values will be those of the parent buffer. If they are not available, a mirror type overscan is used instead. (summarize)</p>
<input type="checkbox"/> M_OVERSCAN_REPLACE_VALUE	<p>Sets a replacement value for the overscan pixel values. Note that M_OVERSCAN_REPLACE_VALUE is ignored unless M_OVERSCAN with MedgeControl() is set to M_REPLACE. In addition, M_OVERSCAN_REPLACE_VALUE is ignored if MedgeControl() with M_FILTER_MODE is set to M_RECURSIVE. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 0. (summarize)</p>
<input type="checkbox"/> M_REPLACE_MAX	<p>Specifies that the overscan neighborhood pixel values will be set to the maximum value of the source image buffer.</p>
<input type="checkbox"/> M_REPLACE_MIN	<p>Specifies that the overscan neighborhood pixel values will be set to the minimum value of the source image buffer.</p>
<input type="checkbox"/> Value	<p>Specifies the value of the overscan neighborhood pixels.</p>

<input type="checkbox"/> M_THRESHOLD_HIGH	Sets the upper bound of the hysteresis threshold value. M_THRESHOLD_HIGH is ignored unless MedgeControl() with M_THRESHOLD_MODE is set to M_USER_DEFINED . Note that lower threshold values result in a more sensitive edgel detection; that is, a higher (or equal) number of edgels are detected. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the upper bound of the hysteresis threshold.
<input type="checkbox"/> M_THRESHOLD_LOW	Sets the lower bound of the hysteresis threshold value. M_THRESHOLD_LOW is ignored unless MedgeControl() with M_THRESHOLD_MODE is set to M_USER_DEFINED . Note that lower threshold values result in a more sensitive edgel detection; that is, a higher (or equal) number of edgels are detected. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the lower bound of the hysteresis threshold.
<input type="checkbox"/> M_THRESHOLD_MODE	<p>Sets the threshold of the edge extraction. The Edge Finder module uses a classical hysteresis threshold to extract relevant edges in the image. Threshold values can either be set manually or determined automatically by the Edge Finder module. Note that lower threshold values result in a more sensitive edgel detection; that is, a higher (or equal) number of edgels are detected.</p> <p>When interfacing with the MIL Geometric Model Finder module, you should use M_DETAIL_LEVEL to set the threshold mode. Note that an M_DETAIL_LEVEL setting overrides an M_THRESHOLD_MODE setting. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_HIGH .
<input type="checkbox"/> M_DISABLE	Specifies no threshold. All edges are extracted. (summarize)
<input type="checkbox"/> M_HIGH	Specifies a high threshold. M_HIGH should be used for images with some contrast variations, noise, and non-uniform illumination. M_HIGH always results in a lower (or equal) number of edgels than M_MEDIUM . (summarize)
<input type="checkbox"/> M_LOW	Specifies a low threshold. All edges over a minimum noise-based estimated threshold are extracted. M_LOW always results in a lower (or equal) number of edgels than M_DISABLE . (summarize)
<input type="checkbox"/> M_MEDIUM	Specifies a medium threshold. M_MEDIUM should be used for multi-contrast images, or for images with a lot of noise or non-uniform illumination. M_MEDIUM always results in a lower (or equal) number of edgels than M_LOW . (summarize)
<input type="checkbox"/> M_USER_DEFINED	Specifies that the threshold values will be user-defined. Set the threshold values with M_THRESHOLD_LOW and M_THRESHOLD_HIGH . (summarize)
<input type="checkbox"/> M_VERY_HIGH	Specifies a very high threshold. Only the strongest edges in the image are extracted. M_VERY_HIGH always results in a lower (or equal) number of edgels than M_HIGH . (summarize)
<input type="checkbox"/> M_THRESHOLD_TYPE	<p>Sets the type of hysteresis threshold used when performing the edge extraction.</p> <p>Edge chains are built such that the magnitude values of all connected edgels are stronger than a lower bound threshold value and such that at least one of the edgel's magnitude in each edge chain is stronger than a upper bound threshold value.</p> <p>Note that M_THRESHOLD_TYPE sets how to use the lower and upper threshold bounds, while M_THRESHOLD_MODE defines what the bounds are. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_HYSTERESIS .
<input type="checkbox"/> M_FULL_HYSTERESIS	Specifies that the lower bound threshold value is 0.
<input type="checkbox"/> M_HYSTERESIS	Specifies that both the lower bound threshold value and the upper bound threshold value will be used.
<input type="checkbox"/> M_NO_HYSTERESIS	Specifies that the lower bound threshold value is equal to the upper bound threshold value.
<input type="checkbox"/> M_TIMEOUT	Sets the maximum edge extraction and calculation time for MedgeCalculate() , in msec. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies an infinite amount of edge extraction and calculation time.

<input type="checkbox"/> Value > 0	Specifies the maximum edge extraction and calculation time, in msec.
------------------------------------	----------------------------------------------------------------------

The following [ControlType](#) and corresponding [ControlValue](#) settings are used to control the Edge Finder context operation settings and can only be specified for [M_CREST](#) Edge Finder contexts.

For operation settings for line crests	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_FOREGROUND_VALUE	Sets the color of the line crests to extract from the image. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_FOREGROUND_BLACK .
<input type="checkbox"/> M_ANY	Specifies that the line crests are both lighter and darker than the image's background color. This corresponds to both valley-like and ridge-like lines extracted from the image. (summarize)
<input type="checkbox"/> M_FOREGROUND_BLACK	Specifies that the line crests are darker than the image's background color. This corresponds to valley-like lines extracted from the image (a valley-like Gaussian profile). (summarize)
<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that the line crests are lighter than the image's background color. This corresponds to ridge-like lines extracted from the image (a ridge-like Gaussian profile). (summarize)

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are used to save internal buffers in the Edge Finder result buffer and can be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts.

For saving internal buffers in the Edge Finder result buffer	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_SAVE_ANGLE	Sets whether the internal angle buffer, used when extracting edges, is saved in the Edge Finder result buffer. Note that the angle value is not meaningful in the whole image; it is only meaningful for edges whose magnitude value is above the lower bound threshold value. Note that the angle is measured between the horizontal axis and the perpendicular direction of the edge chain at each edgel location. For more information on angle convention in MIL, see the Internal processing buffers subsection in the Calculating and retrieving results section in Chapter 9: Edge Finder . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the angle buffer will not be saved.
<input type="checkbox"/> M_ENABLE	Specifies that the angle buffer will be saved.
<input type="checkbox"/> M_SAVE_CHAIN_ANGLE	Sets whether the angle value of the edge at each edgel position is saved in the Edge Finder result buffer. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the angle values will not be saved.
<input type="checkbox"/> M_ENABLE	Specifies that the angle values will be saved.
<input type="checkbox"/> M_SAVE_CHAIN_MAGNITUDE	Sets whether the magnitude value of the edge at each edgel position is saved in the Edge Finder result buffer. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the magnitude values will not be saved.
<input type="checkbox"/> M_ENABLE	Specifies that the magnitude values will be saved.
<input type="checkbox"/> M_SAVE_DERIVATIVES	Sets whether the internal derivative buffers used when extracting edges are saved in the Edge Finder result buffer. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .

<input type="checkbox"/> M_DISABLE	Specifies that the derivative buffers will not be saved.
<input type="checkbox"/> M_ENABLE	Specifies that the derivative buffers will be saved.
<input checked="" type="checkbox"/> M_SAVE_IMAGE	Sets whether the source image is saved in the Edge Finder result buffer. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the image will not be saved.
<input type="checkbox"/> M_ENABLE	Specifies that the image will be saved.
<input checked="" type="checkbox"/> M_SAVE_MAGNITUDE	Sets whether the internal magnitude buffer, used when extracting edges, is saved in the Edge Finder result buffer. For M_CONTOUR Edge Finder contexts, the magnitude is the norm of the gradient vector at the edgel position. For M_CREST Edge Finder contexts, the magnitude is equal to the maximum eigenvalue of the Hessian matrix at the edgel position. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the magnitude buffer will not be saved.
<input type="checkbox"/> M_ENABLE	Specifies that the magnitude buffer will be saved.
<input checked="" type="checkbox"/> M_SAVE_MASK	Sets whether the mask buffer is saved in the Edge Finder result buffer. For more information, see MedgeMask() . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the mask will not be saved.
<input type="checkbox"/> M_ENABLE	Specifies that the mask will be saved.

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are used to control the edge features calculated for each edge and can be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts.

🔍 For calculating edge features	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input checked="" type="checkbox"/> M_AVERAGE_STRENGTH +	Sets whether to calculate the average strength of each edge. This is the average energy of each edge, which is defined as the energy of the edge (M_STRENGTH) divided by its number of edgels. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the average strength will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the average strength will be calculated.
<input checked="" type="checkbox"/> M_BOX_X_MAX +	Sets whether to calculate the extreme right edgel coordinate of each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the extreme right edgel coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the extreme right edgel coordinate will be calculated.
<input checked="" type="checkbox"/> M_BOX_X_MIN +	Sets whether to calculate the extreme left edgel coordinate of each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the extreme left edgel coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the extreme left edgel coordinate will be calculated.
<input checked="" type="checkbox"/> M_BOX_Y_MAX +	Sets whether to calculate the extreme bottom edgel coordinate of each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .

<input type="checkbox"/> M_DISABLE	Specifies that the extreme bottom edgel coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the extreme bottom edgel coordinate will be calculated.
<input type="checkbox"/> M_BOX_Y_MIN +	Sets whether to calculate the extreme top edgel coordinate of each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the extreme top edgel coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the extreme top edgel coordinate will be calculated.
<input type="checkbox"/> M_CENTER_OF_GRAVITY_X +	Sets whether to calculate the X-position of each edge's center of gravity. This is equal to the average position of edgel positions in X. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the X-position of the center of gravity will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the X-position of the center of gravity will be calculated.
<input type="checkbox"/> M_CENTER_OF_GRAVITY_Y +	Sets whether to calculate the Y-position of each edge's center of gravity. This is equal to the average position of edgel positions in Y. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the Y-position of the center of gravity will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the Y-position of the center of gravity will be calculated.
<input type="checkbox"/> M_CIRCLE_FIT_CENTER_X +	Sets whether to calculate the X-coordinate of the center of the circle that is the best fit for each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the X-coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the X-coordinate will be calculated.
<input type="checkbox"/> M_CIRCLE_FIT_CENTER_Y +	Sets whether to calculate the Y-coordinate of the center of the circle that is the best fit for each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the Y-coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the Y-coordinate will be calculated.
<input type="checkbox"/> M_CIRCLE_FIT_COVERAGE +	Sets whether to calculate the coverage of the circle that is the best fit for each edge. The circle fit coverage indicates what angular fraction of the fitted circle is subtended by the radii going to the endpoints of the edge. If the edge is closed, the coverage will be 1. For a quarter circle the coverage would be 0.25. A perfectly straight line has a coverage of 0, it doesn't subtend much of an infinitely large circle. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the coverage of the circle will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the coverage of the circle will be calculated.
<input type="checkbox"/> M_CIRCLE_FIT_ERROR +	Sets whether to calculate the fit error of the circle that is the best fit for each edge. This is calculated as the average quadratic error. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the fit error of the circle will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the fit error of the circle will be calculated.
<input type="checkbox"/> M_CIRCLE_FIT_RADIUS +	Sets whether to calculate the radius of the circle that is the best fit for each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the radius will not be calculated.

<input type="checkbox"/> M_ENABLE	Specifies that the radius will be calculated.
<input checked="" type="checkbox"/> M_CLOSURE +	Sets whether to calculate if each edge forms a closed chain. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the edge's closure state will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the edge's closure state will be calculated.
<input checked="" type="checkbox"/> M_CONVEX_PERIMETER +	Sets whether to calculate the convex elongation of each edge. This is an approximation of the perimeter of the convex hull of an edge. It is derived from several Feret diameters; therefore, a larger number of Ferets gives a more accurate result (see M_NUMBER_OF_FERETS). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the convex elongation will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the convex elongation will be calculated.
<input checked="" type="checkbox"/> M_ELLIPSE_FIT_ANGLE +	Sets whether to calculate the angle of the ellipse that is the best fit for each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the angle will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the angle will be calculated.
<input checked="" type="checkbox"/> M_ELLIPSE_FIT_CENTER_X +	Sets whether to calculate the X-coordinate of the center of the ellipse that is the best fit for each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the X-coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the X-coordinate will be calculated.
<input checked="" type="checkbox"/> M_ELLIPSE_FIT_CENTER_Y +	Sets whether to calculate the Y-coordinate of the center of the ellipse that is the best fit for each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the Y-coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the Y-coordinate will be calculated.
<input checked="" type="checkbox"/> M_ELLIPSE_FIT_COVERAGE +	Sets whether to calculate the coverage of the ellipse that is the best fit for each edge. The coverage describes the portion of the ellipse covered by the edge. The value returned is between 0.0 and 1.0, inclusive, where 0.0 equals no coverage, 0.5 equals 50 percent coverage, and 1.0 equals 100 percent coverage. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the coverage of the ellipse will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the coverage of the ellipse will be calculated.
<input checked="" type="checkbox"/> M_ELLIPSE_FIT_ERROR +	Sets whether to calculate the fit error of the ellipse that is the best fit for each edge. This is calculated as the average quadratic error. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the fit error of the ellipse will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the fit error of the ellipse will be calculated.
<input checked="" type="checkbox"/> M_ELLIPSE_FIT_MAJOR_AXIS +	Sets whether to calculate the major axis of the ellipse that is the best fit for each edge. The major axis is the line passing through the foci, center, and vertices of an ellipse. It is also the principal axis of symmetry. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the major axis will not be calculated.

<input type="checkbox"/> M_ENABLE	Specifies that the major axis will be calculated.
<input type="checkbox"/> M_ELLIPSE_FIT_MINOR_AXIS +	Sets whether to calculate the minor axis of the ellipse that is the best fit for each edge. The minor axis is the line through the center of an ellipse that is perpendicular to the major axis. The minor axis is an axis of symmetry. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the minor axis will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the minor axis will be calculated.
<input type="checkbox"/> M_FAST_LENGTH +	Sets whether to quickly calculate the length of each edge. M_FAST_LENGTH gives a coarser but faster approximation of the edge's length than M_LENGTH . M_FAST_LENGTH is equal to: $(\sum \text{horizontal_edge_pairs}) + (\sum \text{vertical_edge_pairs}) + \sqrt{2} (\sum \text{diagonal_edge_pairs})$ (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that a fast length will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that a fast length will be calculated.
<input type="checkbox"/> M_FERET_ELONGATION +	Sets whether to calculate the Feret elongation of each edge. This is a measure of the shape of each edge. It is equal to M_FERET_MAX_DIAMETER / M_FERET_MIN_DIAMETER . It is accurate for reasonably compact objects, but becomes less accurate for very elongated objects (because M_FERET_MIN_DIAMETER becomes less accurate). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the Feret elongation will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the Feret elongation will be calculated.
<input type="checkbox"/> M_FERET_MAX_ANGLE +	Sets whether to calculate the maximum Feret angle of each edge. This is the angle at which the maximum Feret diameter is found. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the maximum Feret angle will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the maximum Feret angle will be calculated.
<input type="checkbox"/> M_FERET_MAX_DIAMETER +	Sets whether to calculate the maximum Feret diameter of each edge. This is the largest Feret diameter found after checking a certain number of angles (see M_NUMBER_OF_FERETS). More angles will give a more accurate result, but will take longer to calculate. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the maximum Feret diameter will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the maximum Feret diameter will be calculated.
<input type="checkbox"/> M_FERET_MEAN_DIAMETER +	Sets whether to calculate the average Feret diameter at all the angles checked (see M_NUMBER_OF_FERETS). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the average Feret diameter will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the average Feret diameter will be calculated.
<input type="checkbox"/> M_FERET_MIN_ANGLE +	Sets whether to calculate the minimum Feret angle of each edge. This is the angle at which the minimum Feret diameter is found. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the minimum Feret angle will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the minimum Feret angle will be calculated.
<input type="checkbox"/> M_FERET_MIN_DIAMETER +	Sets whether to calculate the minimum Feret diameter of each edge. This is the smallest Feret diameter found after checking a certain number of angles (see M_NUMBER_OF_FERETS). More angles will give a more accurate result, but will take longer to calculate. (summarize)

<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the minimum Feret diameter will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the minimum Feret diameter will be calculated.
<input type="checkbox"/> M_FERET_X +	Sets whether to calculate the X-Feret value of each edge. This is the dimension of the minimum bounding box of an edge in the horizontal direction; that is, M_BOX_X_MAX - M_BOX_X_MIN . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the X-Feret value will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the X-Feret value will be calculated.
<input type="checkbox"/> M_FERET_Y +	Sets whether to calculate the Y-Feret value of each edge. This is the dimension of the minimum bounding box of an edge in the vertical direction; that is, M_BOX_Y_MAX - M_BOX_Y_MIN . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the Y-Feret value will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the Y-Feret value will be calculated.
<input type="checkbox"/> M_FIRST_POINT_X +	Sets whether to calculate the X-coordinate of each edge's first point (starting point). Together with M_FIRST_POINT_Y , these values define a unique point for each edge, which is always the first point of each edge chain. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the first point's X-coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the first point's X-coordinate will be calculated.
<input type="checkbox"/> M_FIRST_POINT_Y +	Sets whether to calculate the Y-coordinate of each edge's first point (starting point). Together with M_FIRST_POINT_X , these values define a unique point for each edge, which is always the first point of each edge chain. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the first point's Y-coordinate will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the first point's Y-coordinate will be calculated.
<input type="checkbox"/> M_GENERAL_FERET +	Sets whether to calculate the general Feret of each edge. This is the Feret diameter calculated at M_GENERAL_FERET_ANGLE . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the general Feret will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the general Feret will be calculated.
<input type="checkbox"/> M_LABEL_VALUE +	Sets whether to calculate the label value of each edge in an image. The label value is a positive integer greater or equal to one; each edge in an image has a unique label value. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ENABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the label value will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the label value will be calculated.
<input type="checkbox"/> M_LENGTH +	Sets whether to calculate the length of each edge. M_LENGTH gives a more accurate but slower approximation of the edge's length than M_FAST_LENGTH . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the length of each edge will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the length of each edge will be calculated.
<input type="checkbox"/> M_LINE_FIT_A +	Sets whether to calculate the <i>A</i> variable of the line that is the best fit for each edge. The line fit approximation is based on the following equation:

	$A x + B y + C = 0$. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the A variable will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the A variable will be calculated.
<input type="checkbox"/> M_LINE_FIT_B +	Sets whether to calculate the B variable of the line that is the best fit for each edge. The line fit approximation is based on the following equation: $A x + B y + C = 0$. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the B variable will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the B variable will be calculated.
<input type="checkbox"/> M_LINE_FIT_C +	Sets whether to calculate the C variable of the line that is the best fit for each edge. The line fit approximation is based on the following equation: $A x + B y + C = 0$. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the C variable will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the C variable will be calculated.
<input type="checkbox"/> M_LINE_FIT_ERROR +	Sets whether to calculate the fit error of the line that is the best fit for each edge. This is calculated as the average quadratic error. The line fit approximation is based on the following equation: $A x + B y + C = 0$. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the fit error will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the fit error will be calculated.
<input type="checkbox"/> M_MOMENT_ELONGATION +	Sets whether to calculate the moment elongation of each edge. M_MOMENT_ELONGATION is defined as the ratio of the principal values of the edge's inertial matrix, which corresponds to the principal directions of the edge shape. This can be approximately defined as the ratio between the edge's minimum and maximum moment. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the moment elongation will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the moment elongation will be calculated.
<input type="checkbox"/> M_MOMENT_ELONGATION_ANGLE +	Sets whether to calculate the angle of the principal axis along each edge's moment elongation (M_MOMENT_ELONGATION). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the angle of the moment elongation will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the angle of the moment elongation will be calculated.
<input type="checkbox"/> M_POSITION_X +	Sets whether to calculate the X-position of each edge. The position of the edge is defined by the middle edgel of the edge chain. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the X-position will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the X-position will be calculated.
<input type="checkbox"/> M_POSITION_Y +	Sets whether to calculate the Y-position of each edge. The position of the edge is defined by the middle edgel of the edge chain. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the Y-position will not be calculated.

<input type="checkbox"/> M_ENABLE	Specifies that the Y-position will be calculated.
<input type="checkbox"/> M_SIZE +	Sets whether to calculate the number of edgels of each edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the number of edgels will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the number of edgels will be calculated.
<input type="checkbox"/> M_STRENGTH +	Sets whether to calculate the strength of each edge. This is the energy of each edge, which is defined as the sum of the square of the edgels' magnitude values. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the strength will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the strength will be calculated.
<input type="checkbox"/> M_TORTUOSITY +	Sets whether to calculate the tortuosity measure of each edge. The tortuosity measure is equal to the diagonal length of the edge's bounding box (M_BOX), divided by the length of the edge (M_LENGTH). Therefore, a non-tortuous edge (a straight line) will have a tortuosity of 1.0 while a tortuous edge will have its tortuosity decreasing toward zero. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the tortuosity measure will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the tortuosity measure will be calculated.
<input type="checkbox"/> M_X_MAX_AT_Y_MAX +	Sets whether to calculate the maximum X-coordinate at the maximum Y-coordinate of each edge. Together with M_BOX_Y_MAX , this is one of four contact points on the convex perimeter of the edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the X-maximum at Y-maximum will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the X-maximum at Y-maximum will be calculated.
<input type="checkbox"/> M_X_MIN_AT_Y_MIN +	Sets whether to calculate the minimum X-coordinate at the minimum Y-coordinate of each edge. Together with M_BOX_Y_MIN , this is one of four contact points on the convex perimeter of the edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the X-minimum at Y-minimum will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the X-minimum at Y-minimum will be calculated.
<input type="checkbox"/> M_Y_MAX_AT_X_MIN +	Sets whether to calculate the maximum Y-coordinate at the minimum X-coordinate of each edge. Together with M_BOX_X_MIN , this is one of four contact points on the convex perimeter of the edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the Y-maximum at X-minimum will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the Y-maximum at X-minimum will be calculated.
<input type="checkbox"/> M_Y_MIN_AT_X_MAX +	Sets whether to calculate the minimum Y-coordinate at the maximum X-coordinate of each edge. Together with M_BOX_X_MAX , this is one of four contact points on the convex perimeter of the edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the Y-minimum at X-maximum will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the Y-minimum at X-maximum will be calculated.

Combination constants for the values listed in [For calculating edge features](#)

You can add one of the following values to the above-mentioned value to specify a sorting key for result retrieval.

Note that only one edge feature can be selected as the first, second, or third sorting key.

● For specifying a sorting key	
☐ Value	Description
☐ M_NO_SORT	Removes the specified sorting key.
☐ M_SORTn_DOWN	Specifies the feature as the <i>n</i> th sorting key (in descending order), where <i>n</i> stands for an integer between 1 and 3.
☐ M_SORTn_UP	Specifies the feature as the <i>n</i> th sorting key (in ascending order), where <i>n</i> stands for an integer between 1 and 3.

The following **ControlType** and corresponding **ControlValue** parameter settings are used to calculate Feret values and can be specified for both **M_CONTOUR** and **M_CREST** Edge Finder contexts.

● For calculating Feret values	
☐ ControlType	Description
☐ ControlValue	
☐ M_FERET_ANGLE_SEARCH_MAX	Sets the end of the angular search region used for M_FERET_MAX_DIAMETER , M_FERET_MIN_DIAMETER , M_FERET_MAX_ANGLE , and M_FERET_MIN_ANGLE calculations. The search is done in the counter-clockwise direction in steps of $(\text{M_FERET_ANGLE_SEARCH_MAX} - \text{M_FERET_ANGLE_SEARCH_MIN}) / \text{M_NUMBER_OF_FERETS}$. (summarize)
☐ M_DEFAULT	Specifies the default value. The default value is 360.0 °. (summarize)
☐ 0.0 to 360.0	Specifies the end of the angular region.
☐ M_FERET_ANGLE_SEARCH_MIN	Sets the start of the angular search region used for M_FERET_MAX_DIAMETER , M_FERET_MIN_DIAMETER , M_FERET_MAX_ANGLE , and M_FERET_MIN_ANGLE calculations. The search is done in the counter-clockwise direction in steps of $(\text{M_FERET_ANGLE_SEARCH_MAX} - \text{M_FERET_ANGLE_SEARCH_MIN}) / \text{M_NUMBER_OF_FERETS}$. (summarize)
☐ M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
☐ 0.0 to 360.0	Specifies the start of the angular region.
☐ M_GENERAL_FERET_ANGLE	Sets the angle at which to calculate the M_GENERAL_FERET value. (summarize)
☐ M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
☐ 0.0 to 360.0	Specifies the angle, in degrees.
☐ M_MOMENT_ELONGATION_ANGLE	Sets whether to calculate the moment elongation of each edge. M_MOMENT_ELONGATION_ANGLE is defined as the ratio of the principal values of the edge's inertial matrix, which corresponds to the principal directions of the edge shape. An offset can be added to the M_MOMENT_ELONGATION_ANGLE . For example, if you want to get the general feret measure to be perpendicular to the moment elongation angle the value would be M_MOMENT_ELONGATION_ANGLE +90. (summarize)
☐ M_NUMBER_OF_FERETS	Sets the number of Ferets used to calculate M_FERET_MAX_DIAMETER , M_FERET_MIN_DIAMETER , M_FERET_MAX_ANGLE , M_FERET_MEAN_DIAMETER , M_FERET_MIN_ANGLE , M_FERET_ELONGATION , and M_CONVEX_PERIMETER . The first Feret angle used is always 0°, and the difference between successive angles is 180° / number of Ferets. Note that increasing the number of Ferets increases the accuracy of the results; however, it also increases the processing time. (summarize)
☐ M_DEFAULT	Specifies the default value. The default value is 8. (summarize)
☐ Value	Specifies the number of Ferets.

The following values allow you to select groups of edge features for calculation in a single call.

For selecting groups of edge features		
ControlType		Description
ControlValue		
<input type="checkbox"/> M_ALL_FEATURES		Sets whether to calculate all edge features. (summarize)
<input type="checkbox"/> M_DEFAULT		Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE		Specifies that all edge features will not be calculated.
<input type="checkbox"/> M_ENABLE		Specifies that all edge features will be calculated.
<input type="checkbox"/> M_BOX		Sets whether to calculate the four extreme edgel coordinates of each edge (M_BOX_X_MIN , M_BOX_X_MAX , M_BOX_Y_MIN , and M_BOX_Y_MAX). (summarize)
<input type="checkbox"/> M_DEFAULT		Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE		Specifies that all the extreme edgel coordinates will not be calculated.
<input type="checkbox"/> M_ENABLE		Specifies that all the extreme edgel coordinates will be calculated.
<input type="checkbox"/> M_CENTER_OF_GRAVITY		Sets whether to calculate the X- and Y-coordinates of the center of gravity of each edge (M_CENTER_OF_GRAVITY_X and M_CENTER_OF_GRAVITY_Y). (summarize)
<input type="checkbox"/> M_DEFAULT		Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE		Specifies that both the X- and Y-coordinates of the center of gravity will not be calculated.
<input type="checkbox"/> M_ENABLE		Specifies that both the X- and Y-coordinates of the center of gravity will be calculated.
<input type="checkbox"/> M_CIRCLE_FIT		Sets whether to calculate the circle fit values of each edge (M_CIRCLE_FIT_CENTER_X , M_CIRCLE_FIT_CENTER_Y , M_CIRCLE_FIT_RADIUS , M_CIRCLE_FIT_ERROR , and M_CIRCLE_FIT_COVERAGE). (summarize)
<input type="checkbox"/> M_DEFAULT		Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE		Specifies that all the circle fit values will not be calculated.
<input type="checkbox"/> M_ENABLE		Specifies that all the circle fit values will be calculated.
<input type="checkbox"/> M_CONTACT_POINTS		Sets whether to calculate the minimum/maximum X-coordinate at the minimum/maximum Y-coordinate of each edge (M_X_MIN_AT_Y_MIN , M_X_MAX_AT_Y_MAX , M_Y_MAX_AT_X_MIN , and M_Y_MIN_AT_X_MAX). Together with its corresponding extreme edgel coordinate, these points represent four contact points on the convex perimeter of the edge. For example, M_X_MIN_AT_Y_MIN and M_BOX_Y_MIN is one of the four contact points. (summarize)
<input type="checkbox"/> M_DEFAULT		Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE		Specifies that the four contact points will not be calculated.
<input type="checkbox"/> M_ENABLE		Specifies that the four contact points will be calculated.
<input type="checkbox"/> M_ELLIPSE_FIT		Sets whether to calculate the ellipse fit values of each edge (M_ELLIPSE_FIT_ANGLE , M_ELLIPSE_FIT_CENTER_X , M_ELLIPSE_FIT_CENTER_Y , M_ELLIPSE_FIT_COVERAGE , M_ELLIPSE_FIT_ERROR , M_ELLIPSE_FIT_MINOR_AXIS , and M_ELLIPSE_FIT_MAJOR_AXIS). (summarize)
<input type="checkbox"/> M_DEFAULT		Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE		Specifies that all the ellipse fit values will not be calculated.
<input type="checkbox"/> M_ENABLE		Specifies that all the ellipse fit values will be calculated.
<input type="checkbox"/> M_FERET_BOX		Sets whether to calculate the X- and Y-Feret values of each edge (M_FERET_X and M_FERET_Y). (summarize)
<input type="checkbox"/> M_DEFAULT		Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE		Specifies that the X- and Y-Feret values will not be calculated.
<input type="checkbox"/> M_ENABLE		Specifies that the X- and Y-Feret values will be calculated.

<input type="checkbox"/> M_FIRST_POINT	Sets whether to calculate the X- and Y-coordinate of each edge's first point (M_FIRST_POINT_X , and M_FIRST_POINT_Y). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the X- and Y-coordinate of each edge's first point will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that the X- and Y-coordinate of each edge's first point will be calculated.
<input type="checkbox"/> M_LINE_FIT	Sets whether to calculate the line fit values of each edge (M_LINE_FIT_A , M_LINE_FIT_B , M_LINE_FIT_C , and M_LINE_FIT_ERROR). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that all the line fit values will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that all the line fit values will be calculated.
<input type="checkbox"/> M_POSITION	Sets whether to calculate both the X- and Y-position of each edge (M_POSITION_X and M_POSITION_Y). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that both the X- and Y-position will not be calculated.
<input type="checkbox"/> M_ENABLE	Specifies that both the X- and Y-position will be calculated.

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are used when performing post-calculations on extracted edges and can typically be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts.

For performing post-calculations on extracted edges	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_APPROXIMATION_TOLERANCE	Sets the resolution of edge approximation. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 50.0. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the resolution. When set to 0.0, a very fine edge approximation is performed. When set to 100.0, a coarse edge approximation is performed. (summarize)
<input type="checkbox"/> M_CHAIN_APPROXIMATION	Sets the simple geometric feature used when performing the edge approximation. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that the edge approximation will not be performed. In this case, only the edge map is calculated. (summarize)
<input type="checkbox"/> M_LINE	Specifies that the edge approximation will be performed using a polygonal segmentation of each edge. In this case, both the edge map and the edge approximation is calculated. Note that for an M_LINE edge approximation, each vertex has a null bulge value. (summarize)
<input type="checkbox"/> M_FILL_GAP_ANGLE	Sets the aperture angle where Edge Finder searches for edge extremity candidates when filling edge gaps. That is, M_FILL_GAP_ANGLE , along with M_FILL_GAP_DISTANCE , define a region where two chain extremities can be linked. Note that two chain extremities can only be linked if both are included in the search region of the other. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 360.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the aperture angle, in degrees.
<input type="checkbox"/> M_FILL_GAP_CANDIDATE	Sets whether to join an edge extremity with the other extremity of the same edge, or with an extremity of any edge. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Specifies that the extremity of an edge can be connected with the extremity of any edge.

<input type="checkbox"/> M_SAME	Specifies that the extremity of an edge can only be connected with the other extremity of the same edge.
<input type="checkbox"/> M_FILL_GAP_CONTINUITY	Sets the continuity constraint used when performing edge gap filling, when more than one edge extremity candidate is present. Among the whole set of extremity candidate edgels found, the one which fulfills the continuity criteria best is chosen to fill the gap. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 50.0. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the continuity constraint. When set to 0.0, the closest candidate is chosen to link edge segments together. When set to 100.0, the candidate that gives the most continuous result (minimum curvature) is chosen. (summarize)
<input type="checkbox"/> M_FILL_GAP_DISTANCE	Sets the maximum distance radius where Edge Finder searches for edge extremity candidates when filling edge gaps. That is, M_FILL_GAP_DISTANCE , along with M_FILL_GAP_ANGLE , define a region where two chain extremities can be linked. Note that two chain extremities can only be linked if both are included in the search region of the other. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> M_INFINITE	Specifies an infinite maximum distance radius.
<input type="checkbox"/> Value	Specifies the maximum distance radius, in pixels. Note that generally, large values should not be set. Typically, you would set M_FILL_GAP_DISTANCE to a value in the range of 2 to 5 pixels. (summarize)
<input type="checkbox"/> M_FILL_GAP_POLARITY	Sets the use of the edge polarity to perform the filling of edge gaps. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Specifies that gaps between edges will be filled, regardless of their polarity.
<input type="checkbox"/> M_REVERSE	Specifies that gaps between edges that have reverse polarity will be filled. Note that M_REVERSE is not available for M_CREST Edge Finder contexts. (summarize)
<input type="checkbox"/> M_SAME	Specifies that gaps between edges that have the same polarity will be filled. Note that M_SAME is not available for M_CREST Edge Finder contexts. (summarize)

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings can be specified for an Edge Finder result buffer.

● For a result buffer	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_DRAW_CROSS_SIZE	Sets the size of the cross used to identify certain features when drawing with MedgeDraw() . This cross is used to draw the edge's edgels (M_DRAW_EDGELS), mid-point (M_DRAW_POSITION), and center of gravity (M_DRAW_CENTER_OF_GRAVITY). For more information, see MedgeDraw() . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 4.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the size of the cross, in pixels.
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_X	Sets the X-coordinate of the top left corner of the region in the Edge Finder result buffer to use when drawing in the destination image buffer. For more information, see MedgeDraw() . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels.

	(summarize)
<input type="checkbox"/> Value	Specifies the relative X-offset, in pixels.
<input checked="" type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_Y	Sets the Y-coordinate of the top left corner of the region in the Edge Finder result buffer to use when drawing in the destination image buffer. For more information, see MedgeDraw() . (summarize)
<input checked="" type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the relative Y-offset, in pixels.
<input checked="" type="checkbox"/> M_DRAW_SCALE_X	Sets the scale factor in the X-direction. This value is used when performing zoomed drawings of results. For more information, see MedgeDraw() . (summarize)
<input checked="" type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the scale factor in the X-direction.
<input checked="" type="checkbox"/> M_DRAW_SCALE_Y	Sets the scale factor in the Y-direction. This value is used when performing zoomed drawings of results. For more information, see MedgeDraw() . (summarize)
<input checked="" type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the scale factor in the Y-direction.
<input checked="" type="checkbox"/> M_MODEL_FINDER_COMPATIBLE	Sets whether the Edge Finder result buffer can be used with a Model Finder context. When used together, you can define models from the result of an edge extraction, or you can find model occurrences in the result of an edge extraction. (summarize)
<input checked="" type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input checked="" type="checkbox"/> M_DISABLE	Specifies that the Edge Finder result buffer cannot be used with a Model Finder context.
<input checked="" type="checkbox"/> M_ENABLE	Specifies that the Edge Finder result buffer can be used with a Model Finder context.
<input checked="" type="checkbox"/> M_NEAREST_NEIGHBOR_RADIUS	Sets the radius distance used to select the closest edge or edges from a point. For more information, see the MedgeSelect() Condition parameter values M_NEAREST_NEIGHBOR and M_ALL_NEAREST_NEIGHBORS . (summarize)
<input checked="" type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 2.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the radius distance, in pixels.

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are used to set closest-edgel constraints for [MedgeGetNeighbors\(\)](#). This function retrieves the coordinates of edgels, from an Edge Finder result buffer, that meet the constraints set below, and correspond to the closest neighbors from a list of user-specified source points.

The following values can only be specified for an Edge Finder result buffer.

● For setting closest-edgel constraints	
<input checked="" type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input checked="" type="checkbox"/> M_NEIGHBOR_ANGLE	Sets the gradient angle that an edgel must have, before being considered a candidate, when finding the closest edgels to a list of points. M_NEIGHBOR_ANGLE is based on a source angle, which you must provide for each source point, with MedgeGetNeighbors() . Note that you can set a tolerance for this angle, using MedgeControl() with M_NEIGHBOR_ANGLE_TOLERANCE .

	M_NEIGHBOR_ANGLE can only be used if the internal angle buffer (MedgeControl() with M_SAVE_ANGLE) was initially saved. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Specifies that the edgel candidate can have any gradient angle. Note that M_ANY is equivalent to setting M_NEIGHBOR_ANGLE_TOLERANCE to 360.0°. In this case, you need not provide the source angle(s) in MedgeGetNeighbors() . (summarize)
<input type="checkbox"/> M_REVERSE	Specifies that the gradient angle of the edgel candidate must have the reverse angle (+ 180°) of the source point.
<input type="checkbox"/> M_SAME	Specifies that the gradient angle of the edgel candidate must have the same angle as the source point.
<input type="checkbox"/> M_SAME_OR_REVERSE	Specifies that the gradient angle of the edgel candidate must either have the same, or the reverse angle of the source point.
<input type="checkbox"/> M_NEIGHBOR_ANGLE_TOLERANCE	Sets the angular tolerance to use for the angle constraint (M_NEIGHBOR_ANGLE), when finding the closest edgels to a list of points. Only edgels that have a gradient angle that falls within this angular range can be returned. Note that M_NEIGHBOR_ANGLE_TOLERANCE is ignored if M_NEIGHBOR_ANGLE is set to M_ANY . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 180.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angular tolerance, in degrees. This value is applied to the angle constraint. Note that setting M_NEIGHBOR_ANGLE_TOLERANCE to 360.0° is equivalent to setting M_NEIGHBOR_ANGLE to M_ANY . (summarize)
<input type="checkbox"/> M_NEIGHBOR_MAXIMUM_NUMBER	Sets the maximum number of edgels that can be returned (for each point), when finding the closest edgels to a list of points. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1. (summarize)
<input type="checkbox"/> Value	Specifies the maximum number of edgels.
<input type="checkbox"/> M_NEIGHBOR_MINIMUM_SPACING	Sets the minimum distance separating closest edgel candidates within the same edge, when finding the closest edgels to a list of points. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_INFINITE .
<input type="checkbox"/> M_INFINITE	Specifies no minimum distance separating two edgel candidates.
<input type="checkbox"/> Value > = 1	Specifies the minimum distance, in edgels.
<input type="checkbox"/> M_SEARCH_ANGLE	Sets the search angle constraint (applied the Edge Finder result buffer), when finding the closest edgels to a list of points. Only edgels that are located along this angle can be returned. M_SEARCH_ANGLE is relative to the source angle set with MedgeGetNeighbors() . Note that you can set the angular tolerance and the angular orientation for M_SEARCH_ANGLE , using MedgeControl() with M_SEARCH_ANGLE_TOLERANCE and M_SEARCH_ANGLE_SIGN , respectively. M_SEARCH_ANGLE , M_SEARCH_ANGLE_TOLERANCE , and M_SEARCH_ANGLE_SIGN limit the search region for potential edgel candidates to an angular sector within the Edge Finder result buffer. Only edgels that are located within this region can be returned. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_SEARCH_ANGLE_SIGN	Sets the orientation to use for the search angle constraint, when finding the closest edgels to a list of points. The orientation is relative to the source angle set with MedgeGetNeighbors() . For example, if the source angle is 45°, then the reverse orientation would be 225°. Note that you can set the search angle and the angular tolerance for M_SEARCH_ANGLE_SIGN , using MedgeControl() with M_SEARCH_ANGLE and M_SEARCH_ANGLE_TOLERANCE , respectively. M_SEARCH_ANGLE , M_SEARCH_ANGLE_TOLERANCE , and M_SEARCH_ANGLE_SIGN limit the search region for potential edgel candidates to an angular sector within the Edge Finder result buffer. Only edgels that are located within this region can be returned. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_SAME .

<input type="checkbox"/> M_REVERSE	Specifies that the orientation of the angle must be the reverse (+ 180°) of the source angle. That is, the angle used will look flipped when compared to the source angle. (summarize)
<input type="checkbox"/> M_SAME	Specifies that the orientation of the angle must be the same as the source angle.
<input type="checkbox"/> M_SAME_OR_REVERSE	Specifies that the orientation of the angle can be the same as, or the reverse of, the source angle.
<input type="checkbox"/> M_SEARCH_ANGLE_TOLERANCE	<p>Sets the angular tolerance to use for the search angle constraint, when finding the closest edgels to a list of points. Only edgels that are located within this angular range (in the Edge Finder result buffer) can be returned.</p> <p>Note that you can set the search angle and the angular orientation for M_SEARCH_ANGLE_TOLERANCE, using MedgeControl() with M_SEARCH_ANGLE and M_SEARCH_ANGLE_SIGN, respectively.</p> <p>M_SEARCH_ANGLE, M_SEARCH_ANGLE_TOLERANCE, and M_SEARCH_ANGLE_SIGN limit the search region for potential edgel candidates to an angular sector within the Edge Finder result buffer. Only edgels that are located within this region can be returned.</p> (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 360.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_SEARCH_RADIUS_MAX	<p>Sets the maximum radius distance in the Edge Finder result buffer that will be searched for the closest edgels. You must use MedgeGetNeighbors() to specify the source point from which the radius is measured.</p> <p>Used together, M_SEARCH_RADIUS_MAX and M_SEARCH_RADIUS_MIN define a ring-like region, in the Edge Finder result buffer, that will be used to find the closest edgels. All edgels that fall outside this ring will be ignored.</p> (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_INFINITE .
<input type="checkbox"/> M_INFINITE	Specifies a maximum distance radius that spans all edgels in the Edge Finder result buffer.
<input type="checkbox"/> Value	Specifies the maximum distance radius, in pixels.
<input type="checkbox"/> M_SEARCH_RADIUS_MIN	<p>Sets the minimum radius distance in the Edge Finder result buffer that will be searched for the closest edgels. You must use MedgeGetNeighbors() to specify the source point from which the radius is measured.</p> <p>Used together, M_SEARCH_RADIUS_MAX and M_SEARCH_RADIUS_MIN define a ring-like region, in the Edge Finder result buffer, that will be used to find the closest edgels. All edgels that fall outside this ring will be ignored.</p> (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the minimum distance radius, in pixels.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeDraw

Synopsis

Draw specific edge results in the destination image buffer.

Syntax

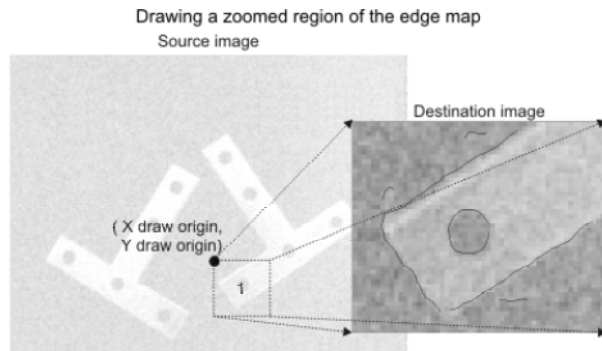
```
void MedgeDraw(
    MIL_ID GraphContId,
    MIL_ID EdgeResultId,
    MIL_ID DestImageId,
    MIL_INT Operation,
    MIL_INT IndexOrLabel,
    MIL_INT ControlFlag
)
```

Description

This function draws specific edge results in the destination buffer.

To draw edge results, the required information must be available for the specified edge or group of edges, in the Edge Finder result buffer. To verify availability, call [MedgeGetResult\(\)](#) with the required edge result combined with **M_AVAILABLE**.

You can also draw from a zoomed region of the source image that was used to calculate results by specifying the appropriate values for [MedgeControl\(\)](#) with [M_DRAW_RELATIVE_ORIGIN_X](#), [M_DRAW_RELATIVE_ORIGIN_Y](#), [M_DRAW_SCALE_X](#), and [M_DRAW_SCALE_Y](#). The relative origin values specify, in pixels, the coordinates of the top-left corner of the region in the source image, while the scale values specify the X- and Y-scaling factors used to fill the destination buffer. Note that these control types are applied to an Edge Finder result buffer.



1. The size of the region that is drawn in the destination image is determined by the scale factor and the size of the destination image.

Note that if the edges are calculated using a calibrated source image, Edge Finder takes the calibration into account; that is, drawings might be distorted, according to the calibration. For example, a straight line (in the world) might be drawn as a curve.

Parameters

GraphContId

Specifies the graphics context to use when drawing. This parameter must be set to one of the following values:

• For specifying the graphics context

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL_graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

EdgeResultId

Specifies the identifier of the Edge Finder result buffer from which to extract the results to draw. The Edge Finder result buffer must have been previously allocated on the required system using [MedgeAllocResult\(\)](#).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD; Matrox Solios GigE; Matrox Solios eA/XA; Matrox Solios eCL/XCL]

The Edge Finder result buffer ([EdgeResultId](#)) must be allocated on the same system as the graphics context buffer ([GraphContId](#)). If it is not, an error will occur.

DestImageId

Specifies the identifier of the destination image buffer in which to draw. This destination image can be any supported MIL image. You can also choose to annotate an image non-destructively, by drawing into its display's overlay buffer. When using **MedgeDraw()** to draw results extracted from a calibrated source image, the destination drawing buffer need not be calibrated to the same world. **MedgeDraw()** draws all the requested results according to the calibrated source image.

Operation

Specifies the type of operation to perform. The possible [Operation](#) parameter values in the table below can be added together to draw multiple feature results at once.

Note that only one internal drawing buffer can be drawn at a time; for example, you cannot combine [M_DRAW_ANGLE](#) and [M_DRAW_MAGNITUDE](#) in the same operation.

● For specifying the type of operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_ANGLE	Draws the internal angle buffer. Note that to perform this operation, the angle buffer must have been previously saved in the result buffer by enabling M_SAVE_ANGLE in MedgeControl() . (summarize)
<input type="checkbox"/> M_DRAW_BOX	Draws a bounding box around each edge. Note that to perform this operation, the M_BOX feature in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_CENTER_OF_GRAVITY +	Draws a cross at the center of gravity of each edge. You can modify the cross size by setting MedgeControl() with M_DRAW_CROSS_SIZE to an appropriate value. Note that to perform this operation, the M_CENTER_OF_GRAVITY feature in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_CIRCLE_FIT	Draws the circle fit of each edge. Note that to perform this operation, the M_CIRCLE_FIT feature in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_CROSS_DERIVATIVE	Draws the internal cross-derivative buffer of the source image. M_DRAW_CROSS_DERIVATIVE is only relevant for M_CREST Edge Finder contexts. Note that to perform this operation, the derivatives must have been previously saved in the result buffer by enabling M_SAVE_DERIVATIVES in MedgeControl() . (summarize)
<input type="checkbox"/> M_DRAW_EDGE	Draws each edge.
<input type="checkbox"/> M_DRAW_EDGELS	Draws a cross at each edgel. You can modify the cross size by setting MedgeControl() with M_DRAW_CROSS_SIZE to an appropriate value. (summarize)
<input type="checkbox"/> M_DRAW_ELLIPSE_FIT	Draws the ellipse fit of each edge. Note that to perform this operation, the M_ELLIPSE_FIT feature in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_FERET_MAX +	Draws the edges' maximum Feret, using an H-type line (-). This line is drawn on contact with the edge's two extrema edgels, at the maximum Feret diameter's angle.

	Note that to perform this operation, the M_FERET_MAX_DIAMETER and M_FERET_MAX_ANGLE features in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_FERET_MIN +	Draws the edges' minimum Feret, using an H-type line (-). This line is drawn on contact with the edge's two extrema edgels, at the minimum Feret diameter's angle. Note that to perform this operation, the M_FERET_MIN_DIAMETER and M_FERET_MIN_ANGLE features in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_FIRST_DERIVATIVE_X	Draws the internal first derivative buffer of the source image, in the X-direction. M_DRAW_FIRST_DERIVATIVE_X is only relevant for M_CONTOUR Edge Finder contexts. Note that to perform this operation, the derivatives must have been previously saved in the result buffer by enabling M_SAVE_DERIVATIVES in MedgeControl() . (summarize)
<input type="checkbox"/> M_DRAW_FIRST_DERIVATIVE_Y	Draws the internal first derivative buffer of the source image, in the Y-direction. M_DRAW_FIRST_DERIVATIVE_Y is only relevant for M_CONTOUR Edge Finder contexts. Note that to perform this operation, the derivatives must have been previously saved in the result buffer by enabling M_SAVE_DERIVATIVES in MedgeControl() . (summarize)
<input type="checkbox"/> M_DRAW_GENERAL_FERET +	Draws the edges' general Feret, using an H-type line (-). This line is drawn on contact with the edge's two extrema edgels, at the general Feret diameter's angle. Note that to perform this operation, the M_GENERAL_FERET feature in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_IMAGE	Draws the source image. Note that to perform this operation, the source image must have been previously saved in the result buffer by enabling M_SAVE_IMAGE in MedgeControl() . (summarize)
<input type="checkbox"/> M_DRAW_INDEX	Draws each edge's index value.
<input type="checkbox"/> M_DRAW_LABEL	Draws each edge's label value. Note that to perform this operation, the M_LABEL_VALUE feature in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_LINE_FIT	Draws the line fit of each edge. Note that to perform this operation, the M_LINE_FIT feature in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_MAGNITUDE	Draws the internal magnitude buffer. Note that to perform this operation, the magnitude buffer must have been previously saved in the result buffer by enabling M_SAVE_MAGNITUDE in MedgeControl() . (summarize)
<input type="checkbox"/> M_DRAW_MASK	Draws the mask buffer. Note that to perform this operation, the mask buffer must have been previously saved in the result buffer by enabling M_SAVE_MASK in MedgeControl() . (summarize)
<input type="checkbox"/> M_DRAW_POSITION +	Draws a cross at the center of each edge. You can modify the cross size by setting MedgeControl() with M_DRAW_CROSS_SIZE to an appropriate value. Note that to perform this operation, the M_POSITION feature in MedgeControl() must be calculated. (summarize)
<input type="checkbox"/> M_DRAW_SECOND_DERIVATIVE_X	Draws the internal second derivative buffer of the source image, in the X-direction. M_DRAW_SECOND_DERIVATIVE_X can only be used with M_CREST Edge Finder contexts. Note that to perform this operation, the derivatives must have been previously saved in the result buffer by enabling M_SAVE_DERIVATIVES in MedgeControl() . (summarize)
<input type="checkbox"/> M_DRAW_SECOND_DERIVATIVE_Y	Draws the internal second derivative buffer of the source image, in the Y-direction. M_DRAW_SECOND_DERIVATIVE_Y can only be used with M_CREST Edge Finder contexts. Note that to perform this operation, the derivatives must have been previously saved in the result buffer by enabling M_SAVE_DERIVATIVES in MedgeControl() . (summarize)
<input type="checkbox"/> M_DRAW_SEGMENTS	Draws the segments of the edge approximation. Note that to perform this operation, edge approximations must have been previously calculated by setting MedgeControl() with M_CHAIN_APPROXIMATION to M_LINE . (summarize)
<input type="checkbox"/> M_DRAW_VERTICES	Draws the segment intersections (or vertices) of the edge approximation.

Note that to perform this operation, edge approximations must have been previously calculated by setting `MedgeControl()` with `M_CHAIN_APPROXIMATION` to `M_LINE`.
([summarize](#))

Combination constant for `M_DRAW_CENTER_OF_GRAVITY`; `M_DRAW_FERET_MAX`; `M_DRAW_FERET_MIN`; `M_DRAW_GENERAL_FERET`; `M_DRAW_POSITION`;

You can add the following value to the above-mentioned values to specify that the operation's numeric value is drawn.

● For <code>M_DRAW_CENTER_OF_GRAVITY</code> , <code>M_DRAW_POSITION</code> , <code>M_DRAW_FERET_MIN</code> , <code>M_DRAW_FERET_MAX</code> , and <code>M_DRAW_GENERAL_FERET</code>	
Value	Description
<code>M_DRAW_VALUE</code>	Specifies that the operation's numerical value is drawn. For example, if you add <code>M_DRAW_VALUE</code> to <code>M_DRAW_CENTER_OF_GRAVITY</code> , the center of gravity's coordinates are drawn within parenthesis, and centered above the drawing cross. This is the default value. (summarize)

IndexOrLabel

Specifies the edge(s) to draw. This parameter must be set to one of the values below.

● For specifying the edge(s) to draw	
Value	Description
<code>M_DEFAULT +</code>	Same as <code>M_INCLUDED_EDGES</code> .
<code>M_ALL_EDGES +</code>	Specifies all edges.
<code>M_EXCLUDED_EDGES +</code>	Specifies all currently excluded edges.
<code>M_INCLUDED_EDGES +</code>	Specifies all currently included edges.
<code>Value +</code>	Specifies either the edge's index or label. For more information, see combination values below. Valid index values must fall within the following range: 0 to the number of included edges in the Edge Finder result buffer - 1. You can retrieve valid label values using <code>MedgeGetResult()</code> with <code>M_LABEL_VALUE</code> . (summarize)

Combination constants for any of the possible values of the `IndexOrLabel` parameter

You can add one of the following values to the above-mentioned values to specify whether you are providing an index or a label value.

● For an index or a label value	
Value	Description
<code>M_TYPE_INDEX</code>	Specifies an edge using its index value. Note that only included edges (<code>M_INCLUDED_EDGES</code>) can be drawn. This is the default value. (summarize)
<code>M_TYPE_LABEL</code>	Specifies an edge using its label value.

ControlFlag

Reserved for future expansion and must be set to `M_DEFAULT`.

Compilation information

Header	Include mil.h.
--------	----------------

Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeFree

Synopsis

Free an Edge Finder context or an Edge Finder result buffer.

Syntax

```
void MedgeFree(
    MIL_ID ObjectId
)
```

Description

This function deletes the specified Edge Finder context or Edge Finder result buffer identifier, and releases any memory associated with it.

All Edge Finder contexts and all Edge Finder result buffers allocated on a particular system must be freed before the system can be freed.

Parameter

ObjectId

Specifies the identifier of the Edge Finder context or Edge Finder result buffer to free. These must have been successfully allocated (with [MedgeAlloc\(\)](#) or [MedgeAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeGetNeighbors

Synopsis

Get edgels from an Edge Finder result buffer that are the closest neighbors to a list of user-specified point coordinates.

Syntax

```
void MedgeGetNeighbors(
    MIL_ID EdgeResultId,
    MIL_INT SizeOfArray,
    const MIL_DOUBLE *SrcArrayXPtr,
    const MIL_DOUBLE *SrcArrayYPtr,
    const MIL_DOUBLE *SrcArrayAnglePtr,
    MIL_DOUBLE *DstArrayXPtr,
    MIL_DOUBLE *DstArrayYPtr,
    MIL_INT *DstArrayIndexPtr,
    MIL_INT *DstArrayLabelPtr,
    MIL_INT ControlFlag
)
```

Description

This function retrieves the coordinates of edgels, from an Edge Finder result buffer, that correspond to the closest neighbors from a list of user-specified source point coordinates. You can also set a series of constraints that potential results must adhere to before being returned. To do so, use the appropriate settings in [MedgeControl\(\)](#).

User-specified coordinates and constraints must be specified in the same world that edgels have been calculated in. For example, if edgels have been calculated using a calibrated source image, values must be specified in the same real world units. Note that you can change the output units using [McalControl\(\)](#) with [M_OUTPUT_UNITS](#) set to [M_PIXEL](#) or [M_WORLD](#).

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [DstArrayXPtr](#), [DstArrayYPtr](#), [DstArrayIndexPtr](#) and [DstArrayLabelPtr](#) parameters to [M_NULL](#). When these parameters are set to [M_NULL](#), the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [MedgeGetNeighbors\(\)](#) again and you pass an array to at least one of the result parameters. You must ensure that the array(s) is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

Parameters

EdgeResultId

Specifies the identifier of the Edge Finder result buffer from which to search for edgel candidates.

SizeOfArray

Specifies the number of points provided. This value represents how many user-specified point coordinates will be searched for in the Edge Finder result buffer.

SrcArrayXPtr

Specifies the address of the array containing the X-coordinate(s) of the source point(s).

SrcArrayYPtr

Specifies the address of the array containing the Y-coordinate(s) of the source point(s).

SrcArrayAnglePtr

Specifies the address of the array containing the constraint angle(s) used to refine the search for edgel candidates. Note that angle values (0° to 360°) are measured counter-clockwise and must be mapped between either 0 to 255 (for contour contexts) or 0 to 127 (for crest contexts).

If no value is required, set this parameter to [M_NULL](#).

DstArrayXPtr

Specifies the address of the variable in which the X-coordinate(s) of the edgel(s) found in the Edge Finder result buffer is to be written. If no edgels were found, ignore the information written. Note that if no edgels were found, the index ([DstArrayIndexPtr](#)) and label ([DstArrayLabelPtr](#)) of the edgels is **M_NULL**.

If no value is required, set this parameter to **M_NULL**.

DstArrayYPtr

Specifies the address of the variable in which the Y-coordinate(s) of the edgel(s) found in the Edge Finder result buffer is to be written. If no edgels were found, ignore the information written. Note that if no edgels were found, the index ([DstArrayIndexPtr](#)) and label ([DstArrayLabelPtr](#)) of the edgels is **M_NULL**.

If no value is required, set this parameter to **M_NULL**.

DstArrayIndexPtr

Specifies the address of the variable in which the index of the edgel(s) found in the Edge Finder result buffer is to be written. If no edgels were found, **M_NULL** is written.

If no value is required, set this parameter to **M_NULL**.

DstArrayLabelPtr

Specifies the address of the variable in which the label of the edgel's edge found in the Edge Finder result buffer is to be written. If no edgels were found, **M_NULL** is written.

If no value is required, set this parameter to **M_NULL**.

ControlFlag

Specifies the accuracy with which to return edgels.

For specifying the accuracy	
Value	Description
M_DEFAULT	Same as M_GET_EDGELS .
M_GET_EDGELS	Specifies that edgels are returned with normal accuracy. In this case, only edgels explicitly located in the Edge Finder result buffer can be returned. (summarize)
M_GET_SUBEDGELS	Specifies that edgels are returned with high accuracy. In this case, a point between two edgels can be returned. When using M_GET_SUBEDGELS , the M_NEIGHBOR_MAXIMUM_NUMBER constraint in MedgeControl() must be set to 1. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeGetResult

Synopsis

Get results of the included edges from an Edge Finder result buffer.

Syntax

```
void MedgeGetResult(
    MIL_ID EdgeResultId,
    MIL_INT EdgeIndexOrLabelValue,
    MIL_INT ResultType,
    void *FirstResultArrayPtr,
    void *SecondResultArrayPtr
)
```

Description

This function retrieves the results of a specified type from an Edge Finder result buffer, after an [MedgeCalculate\(\)](#) call.

The following result types can be retrieved from the Edge Finder result buffer:

- General results related to the edge extraction and edge processing.
- Edge chain results.
- Edge feature results.

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image.

Note that if your image is calibrated, results are calculated in the real world; therefore, retrieving results in pixel units instead of world units will typically be slower.

Note that in the presence of distortion, some results are meaningless when converted from real-world to pixel units (for example, the Feret angles). For example, if an edge appears warped in the source image, but the calibration object of the source image compensates for this during the extraction, the resulting Feret angles are meaningful in the real-world coordinate system, and meaningless in the image coordinate system.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set **FirstResultArrayPtr** and **SecondResultArrayPtr** to **M_NULL**. When these parameters are set to **M_NULL**, the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call **MedgeGetResult()** again and you pass an array to at least on of the result parameters. You must ensure that the array(s) is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

Note, if the request queued does not return results in the second result array, the second array pointer is not padded for that particular request when the queue is executed.

By setting the **ResultType** parameter to [M_INTERACTIVE](#), you can view the results interactively.

Parameters

EdgeResultId

Specifies the identifier of the Edge Finder result buffer from which to get results.

EdgeIndexOrLabelValue

Specifies the edge(s) from which to get results. This parameter can be set to one of the following values.

● For edge results	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_DEFAULT +	Same as M_ALL .
<input type="checkbox"/> M_ALL +	Specifies all edges. That is, the target array(s) will be filled with the specified type of result for all the included edges. Note that target array values are ordered by edge indices. (summarize)
<input type="checkbox"/> Value +	Specifies either the edge's index or label. Index values must fall within the following range: 0 to the number of included edges in the Edge Finder result buffer - 1. Label values can be returned with the result type M_LABEL_VALUE . (summarize)

Combination constants for the values listed in [For edge results](#)

You can add one of the following values to the above-mentioned values to specify whether you are providing an index or a label value.

● For specifying an index or label value	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_INDEX	Specifies an edge using its index value. This is the default value. (summarize)
<input type="checkbox"/> M_TYPE_LABEL	Specifies an edge using its label value.

ResultType

Specifies the type of result to retrieve or opens an interactive dialog box that displays the results stored in the result buffer.

To display the results currently stored in the result buffer in an interactive dialog box, select the following.

Unless otherwise specified, the following values require that you pass the [FirstResultArrayPtr](#) parameter and the [SecondResultArrayPtr](#) parameter [M_NULL](#).

● For displaying results in an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box containing a spreadsheet that displays the types of results stored in the result buffer. The results are displayed in separate columns. If there are queued requests, only those results are displayed. (summarize)

To retrieve general results related to the edge extraction and edge processing, the **ResultType** parameter must be set to one of the values below. Note that, unless otherwise specified, results are only returned to **FirstResultArrayPtr**; when this is the case, **SecondResultArrayPtr** must be set to **M_NULL**.

Unless otherwise specified, the following values require that you pass the address of an array of type `MIL_DOUBLE` with a size equal to 1 to the [FirstResultArrayPtr](#) parameter. In addition, you must pass [M_NULL](#) to the [SecondResultArrayPtr](#) parameter.

● For retrieving results related to edge extraction and edge processing	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CONTEXT_TYPE +	Returns the type of the Edge Finder context that was used to extract edges. (summarize) <i>FirstResultArrayPtr info</i> Return values: M_CONTOUR; M_CREST; (details)
<input type="checkbox"/> M_SIZE_X +	Returns the width of the source image and internal processing buffers, in pixels. Note that if you have used MedgePut() to add edge chains to the Edge Finder result buffer, the width returned can be bigger than the width of the source image buffer if the edge chains exceed the boundaries of the source image buffer.

	(summarize)				
<input type="checkbox"/> M_SIZE_Y +	<p>Returns the height of the source image and internal processing buffers, in pixels.</p> <p>Note that if you have used MedgePut() to add edge chains to the Edge Finder result buffer, the height returned can be bigger than the height of the source image buffer if the edge chains exceed the boundaries of the source image buffer.</p> <p>(summarize)</p>				
<input type="checkbox"/> M_THRESHOLD_HIGH +	<p>Returns the upper bound value, used by the hysteresis threshold to extract edges.</p> <p>(summarize)</p> <div> <i>FirstResultArrayPtr info</i> Return values: M_DEFAULT; Value; (details) </div>				
<input type="checkbox"/> M_THRESHOLD_LOW +	<p>Returns the lower bound value, used by the hysteresis threshold to extract edges.</p> <p>(summarize)</p> <div> <i>FirstResultArrayPtr info</i> Return values: M_DEFAULT; Value; (details) </div>				
<input type="checkbox"/> M_THRESHOLDS +	<p>Returns both the lower and upper bound values, used by the hysteresis threshold to extract edges.</p> <p>(summarize)</p> <div> <i>FirstResultArrayPtr info</i> Return values: M_DEFAULT; Value; (details) </div> <div> <i>SecondResultArrayPtr info</i> Return values: M_DEFAULT; Value; (details) </div>				
<input type="checkbox"/> M_TIMEOUT_END +	<p>Returns whether the timeout limit was reached. You can set the timeout limit using MedgeControl() with M_TIMEOUT. By default, there is no limit.</p> <p>(summarize)</p> <div> <i>FirstResultArrayPtr info</i> Return values: <table> <tr> <td>M_FALSE</td><td>Specifies that the timeout limit has not been reached.</td></tr> <tr> <td>M_TRUE</td><td>Specifies that the timeout limit has been reached.</td></tr> </table> </div>	M_FALSE	Specifies that the timeout limit has not been reached.	M_TRUE	Specifies that the timeout limit has been reached.
M_FALSE	Specifies that the timeout limit has not been reached.				
M_TRUE	Specifies that the timeout limit has been reached.				

To retrieve information about the internal processing buffers, you must specify the internal buffer about which to retrieve the information and the type of information to retrieve. To do so, set **ResultType** to a combination of two values, one from each of the following two tables. The information retrieved can be used to allocate a buffer with the same properties as the internal processing buffer. This newly allocated buffer can then be used, for example, as the destination buffer for [MedgeDraw\(\)](#). Note that, unless otherwise specified, results are only returned to **FirstResultArrayPtr**; when this is the case, **SecondResultArrayPtr** must be set to **M_NULL**.

The internal Edge Finder buffers cannot be accessed directly; however, they can be accessed by creating a buffer with the same characteristics as the internal buffer and using [MedgeDraw\(\)](#) to draw the contents of the internal Edge Finder buffer into your new buffer.

Unless otherwise specified, the following values require that you pass the address of an array of type `MIL_DOUBLE` with a size equal to 1 to the [FirstResultArrayPtr](#) parameter. In addition, you must pass `M_NULL` to the [SecondResultArrayPtr](#) parameter.

● For retrieving information about the internal processing buffers	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ANGLE_ID +	<p>Returns information about the internal angle buffer.</p> <p>Note that information about the internal angle buffer is only available if it was saved using MedgeControl() with M_SAVE_ANGLE set to M_ENABLE. You must specify a combination value from the table below.</p> <p>(summarize)</p>
<input type="checkbox"/> M_CROSS_DERIVATIVE_ID +	<p>Returns information about the internal cross derivative buffer.</p> <p>Note that information about the internal cross derivative buffer is only available if it was saved using MedgeControl() with M_SAVE_DERIVATIVES set to M_ENABLE. You must specify a combination value from the table below.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FIRST_DERIVATIVE_X_ID +	<p>Returns information about the internal first derivative buffer in the X-direction.</p> <p>Note that information about the internal first derivative in the X-direction buffer is only available if it was saved using MedgeControl() with M_SAVE_DERIVATIVES set to M_ENABLE.</p>

	<p>You must specify a combination value from the table below. (summarize)</p>
<input type="checkbox"/> M_FIRST_DERIVATIVE_Y_ID +	<p>Returns information about the internal first derivative buffer in the Y-direction.</p> <p>Note that information about the internal first derivative buffer in the Y-direction is only available if it was saved using <code>MedgeControl()</code> with <code>M_SAVE_DERIVATIVES</code> set to <code>M_ENABLE</code>.</p> <p>You must specify a combination value from the table below. (summarize)</p>
<input type="checkbox"/> M_FIRST_DERIVATIVES_ID +	<p>Returns information about the internal first derivative buffers in both the X- and Y-directions.</p> <p>Note that information about the internal first derivative buffers in both the X- and Y-directions is only available if it was saved using <code>MedgeControl()</code> with <code>M_SAVE_DERIVATIVES</code> set to <code>M_ENABLE</code>.</p> <p>You must specify a combination value from the table below. (summarize)</p> <div> <div><i>FirstResultArrayPtr info</i></div> <div> <p>Return values:</p> <p>X-direction first derivative buffer information Information about the internal first derivative buffer in the X-direction.</p> </div> </div> <div> <div><i>SecondResultArrayPtr info</i></div> <div> <p>Return values:</p> <p>Y-direction first derivative buffer information Information about the internal first derivative buffer in the Y-direction.</p> </div> </div>
<input type="checkbox"/> M_IMAGE_ID +	<p>Returns information about the source buffer.</p> <p>Note that information about the source buffer is only available if it was saved using <code>MedgeControl()</code> with <code>M_SAVE_IMAGE</code> set to <code>M_ENABLE</code>.</p> <p>You must specify a combination value from the table below. (summarize)</p>
<input type="checkbox"/> M_MAGNITUDE_ID +	<p>Returns information about the internal magnitude buffer.</p> <p>Note that information about the internal magnitude buffer is only available if it was saved using <code>MedgeControl()</code> with <code>M_SAVE_MAGNITUDE</code> set to <code>M_ENABLE</code>.</p> <p>The magnitude buffer is calculated according to <code>MedgeControl()</code> with <code>M_MAGNITUDE_TYPE</code>. If <code>M_MAGNITUDE_TYPE</code> is set to <code>M_SQR_NORM</code>, magnitude values are truncated on 15.0 bits. If <code>M_MAGNITUDE_TYPE</code> is set to <code>M_NORM</code>, magnitude values are truncated on 8.7 bits, with fixed-point precision. 32-bit float buffers use the same conventions, but with floating-point precision.</p> <p>You must specify a combination value from the table below. (summarize)</p>
<input type="checkbox"/> M_MASK_ID +	<p>Returns information about the internal mask buffer.</p> <p>Note that information about the internal mask buffer is only available if it was saved using <code>MedgeControl()</code> with <code>M_SAVE_MASK</code> set to <code>M_ENABLE</code>.</p> <p>You must specify a combination value from the table below. (summarize)</p>
<input type="checkbox"/> M_SECOND_DERIVATIVE_X_ID +	<p>Returns information about the internal second derivative buffer in the X-direction.</p> <p>Note that information about the internal second derivative buffer in the X-direction is only available if it was saved using <code>MedgeControl()</code> with <code>M_SAVE_DERIVATIVES</code> set to <code>M_ENABLE</code>.</p> <p>You must specify a combination value from the table below. (summarize)</p>
<input type="checkbox"/> M_SECOND_DERIVATIVE_Y_ID +	<p>Returns information about the internal second derivative buffer in the Y-direction.</p> <p>Note that information about the internal second derivative buffer in the Y-direction is only available if it was saved using <code>MedgeControl()</code> with <code>M_SAVE_DERIVATIVES</code> set to <code>M_ENABLE</code>.</p> <p>You must specify a combination value from the table below. (summarize)</p>
<input type="checkbox"/> M_SECOND_DERIVATIVES_ID +	<p>Returns information about the internal second derivative buffers in both the X- and Y-directions.</p> <p>Note that information about the internal second derivative buffers in both the X- and Y-directions is only available if it was saved using <code>MedgeControl()</code> with <code>M_SAVE_DERIVATIVES</code> set to <code>M_ENABLE</code>.</p> <p>You must specify a combination value from the table below. (summarize)</p> <div> <div><i>FirstResultArrayPtr info</i></div> <div> <p>Return values:</p> <p>X-direction second derivative buffer information Information about the internal second derivative buffer in the X-direction.</p> </div> </div> <div> <div><i>SecondResultArrayPtr info</i></div> </div>

Return values:

Y-direction second derivative Information about the internal second derivative buffer in the Y-direction.
buffer information

Combination constants for the values listed in [For retrieving information about the internal processing buffers](#)

You must add one of the following values to the above-mentioned values to specify the type of information to retrieve from the specified internal processing buffer.


Note that to inquire the width or height of the internal processing buffers, use [M_SIZE_X](#) or [M_SIZE_Y](#).

● For specifying the type of information to retrieve from an internal processing buffer																	
Value	Description																
M_SIGN	<p>Returns the buffer range. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values:</p> <table> <tr> <td>M_SIGNED</td><td>Buffer range is signed.</td></tr> <tr> <td>M_UNSIGNED</td><td>Buffer range is unsigned.</td></tr> </table>	M_SIGNED	Buffer range is signed.	M_UNSIGNED	Buffer range is unsigned.												
M_SIGNED	Buffer range is signed.																
M_UNSIGNED	Buffer range is unsigned.																
M_SIZE_BAND	Returns the number of buffer color bands.																
M_SIZE_BIT	Returns the depth per band, in bits.																
M_TYPE	<p>Returns the buffer data type and depth. Depth is returned in bits. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values:</p> <table> <tr> <td>1 + M_UNSIGNED</td><td>1-bit unsigned data.</td></tr> <tr> <td>8 + M_SIGNED</td><td>8-bit signed data.</td></tr> <tr> <td>8 + M_UNSIGNED</td><td>8-bit unsigned data.</td></tr> <tr> <td>16 + M_SIGNED</td><td>16-bit signed data.</td></tr> <tr> <td>16 + M_UNSIGNED</td><td>16-bit unsigned data.</td></tr> <tr> <td>32 + M_FLOAT</td><td>32-bit float data.</td></tr> <tr> <td>32 + M_SIGNED</td><td>32-bit signed data.</td></tr> <tr> <td>32 + M_UNSIGNED</td><td>32-bit unsigned data.</td></tr> </table>	1 + M_UNSIGNED	1-bit unsigned data.	8 + M_SIGNED	8-bit signed data.	8 + M_UNSIGNED	8-bit unsigned data.	16 + M_SIGNED	16-bit signed data.	16 + M_UNSIGNED	16-bit unsigned data.	32 + M_FLOAT	32-bit float data.	32 + M_SIGNED	32-bit signed data.	32 + M_UNSIGNED	32-bit unsigned data.
1 + M_UNSIGNED	1-bit unsigned data.																
8 + M_SIGNED	8-bit signed data.																
8 + M_UNSIGNED	8-bit unsigned data.																
16 + M_SIGNED	16-bit signed data.																
16 + M_UNSIGNED	16-bit unsigned data.																
32 + M_FLOAT	32-bit float data.																
32 + M_SIGNED	32-bit signed data.																
32 + M_UNSIGNED	32-bit unsigned data.																

To retrieve edge chain results, the **ResultType** parameter can be set to one of the values below. If **EdgeIndexOrLabelValue** is set to an index value or a label value, the edge chain result for the specified edge is returned. If **EdgeIndexOrLabelValue** is set to [M_ALL](#), the edge chain result for all the included edges in the Edge Finder result buffer is returned. Note that, unless otherwise specified, results are only returned to **FirstResultArrayPtr**; when this is the case, **SecondResultArrayPtr** must be set to **M_NULL**.

Unless otherwise specified, the following values require that you pass the [FirstResultArrayPtr](#) parameter and the [SecondResultArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to the number of chained edges in the edge. This number can be obtained using [MedgeGetResult\(\)](#) with [M_NUMBER_OF_CHAINED_EDGELS](#).

● For retrieving edge chain results	
Value	Description
M_BULGES +	<p>Returns the bulge values between vertices. Note that to retrieve bulge values, the chain approximation (M_CHAIN_APPROXIMATION) must have been previously performed (see MedgeControl()).</p> <p>The bulge determines the circular arc that lies between the current vertex and the next vertex; it is equal to the tangent of 1/4 the angle between consecutive vertices, and is negative for a clockwise arc. For example, the bulge of a straight line is 0 while the bulge of a semicircle is 1. (summarize)</p>

	<i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_VERTICES to determine the size of the array.
<input type="checkbox"/> M_CHAIN +	Returns the coordinates of the edge(s)'s edgels. (summarize)
	<i>FirstResultArrayPtr info</i> Return values: X-coordinate value The edgels' X-coordinates.
	<i>SecondResultArrayPtr info</i> Return values: Y-coordinate value The edgels' Y-coordinates.
<input type="checkbox"/> M_CHAIN_ANGLE +	Returns the angle values of the edge(s)'s edgels. Angles are returned, counter-clockwise, from 0 degrees to 360 degrees and mapped in the range of 0 to 255 (that is, an 8-bit range). (summarize)
<input type="checkbox"/> M_CHAIN_CODE +	Returns the edge(s)'s chain code. The chain code describes how edgels are connected using the following neighbor code: <div style="display: flex; align-items: center; justify-content: center;">  <div style="margin: 0 10px;"> $\begin{bmatrix} 3 & 2 & 1 \\ 4 & 0 & \\ 5 & 6 & 7 \end{bmatrix}$ </div> </div>
<input type="checkbox"/> M_CHAIN_INDEX +	Returns the index of each edgel's edge.
<input type="checkbox"/> M_CHAIN_MAGNITUDE +	Returns the magnitude values of the edge(s)'s edgels.
<input type="checkbox"/> M_CHAIN_MAGNITUDE + M_CHAIN_ANGLE +	Returns the magnitude values and the angle values of the edge(s)'s edgels. Angles are returned, counter-clockwise, from 0 degrees to 360 degrees and mapped in the range of 0 to 255 (that is, an 8-bit range). (summarize)
	<i>FirstResultArrayPtr info</i> Return values: Magnitude value The magnitude values of the edge(s)'s edgels.
	<i>SecondResultArrayPtr info</i> Return values: Angle value The angle values of the edge(s)'s edgels.
<input type="checkbox"/> M_CHAIN_X +	Returns the X-coordinates of the edge(s)'s edgels.
<input type="checkbox"/> M_CHAIN_Y +	Returns the Y-coordinates of the edge(s)'s edgels.
<input type="checkbox"/> M_NUMBER_OF_CHAINED_EDGELS +	Returns the number of edgels in the edge(s). (summarize)
	<i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: This array must be of size 1.
<input type="checkbox"/> M_NUMBER_OF_CHAINS +	Returns the number of included edges. (summarize)
	<i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: This array must be of size 1.
<input type="checkbox"/> M_NUMBER_OF_VERTICES +	Returns the number of vertices in the chain approximation. This value is not valid if MedgeControl() with M_CHAIN_APPROXIMATION is disabled. (summarize)
	<i>FirstResultArrayPtr info</i>

	Data type: array of type MIL_DOUBLE Array size: This array must be of size 1.
<input type="checkbox"/> M_VERTICES +	<p>Returns the coordinates of the vertices in the chain approximation. This value is not valid if MedgeControl() with M_CHAIN_APPROXIMATION is disabled. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_VERTICES to determine the size of the array. Return values: X-coordinate value The X-coordinates of the vertices in the chain approximation.</p> <p><i>SecondResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_VERTICES to determine the size of the array. Return values: Y-coordinate value The Y-coordinates of the vertices in the chain approximation.</p>
<input type="checkbox"/> M_VERTICES_CHAIN_INDEX +	<p>Returns the index of the vertices' corresponding edge, in a chain approximation. This value is not valid if MedgeControl() with M_CHAIN_APPROXIMATION is disabled. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_VERTICES to determine the size of the array.</p>
<input type="checkbox"/> M_VERTICES_INDEX +	<p>Returns the index of the vertices' corresponding edgels, in a chain approximation. This value is not valid if MedgeControl() with M_CHAIN_APPROXIMATION is disabled. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_VERTICES to determine the size of the array.</p>
<input type="checkbox"/> M_VERTICES_X +	<p>Returns the X-coordinates of the vertices in the chain approximation. This value is not valid if MedgeControl() with M_CHAIN_APPROXIMATION is disabled. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_VERTICES to determine the size of the array.</p>
<input type="checkbox"/> M_VERTICES_Y +	<p>Returns the Y-coordinates of the vertices in the chain approximation. This value is not valid if MedgeControl() with M_CHAIN_APPROXIMATION is disabled. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_VERTICES to determine the size of the array.</p>

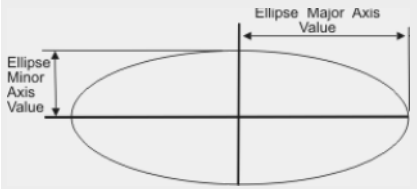
To retrieve edge feature results, the **ResultType** parameter must be set to one of the values below. Note that the specified feature must have already been calculated with [MedgeCalculate\(\)](#).

If **EdgeIndexOrLabelValue** is set to an index value or a label value, the edge feature for the specified edge is returned. If **EdgeIndexOrLabelValue** is set to [M_ALL](#), the edge feature for all the included edges in the Edge Finder result buffer is returned. Note that, unless otherwise specified, results are only returned to **FirstResultArrayPtr**; when this is the case, **SecondResultArrayPtr** must be set to **M_NULL**.

Unless otherwise specified, the following values require that you pass the [FirstResultArrayPtr](#) parameter and the [SecondResultArrayPtr](#) parameter the address of an array of type MIL_DOUBLE with a size equal to the number of edges. This number can be obtained using [MedgeGetResult\(\)](#) with [M_NUMBER_OF_CHAINS](#).

● For retrieving edge features results	
<input type="checkbox"/> Value	Description

M_AVERAGE_STRENGTH +	Returns the average strength value of each edge.				
M_BOX_X_MAX +	Returns the X-coordinate of each edge's extreme right edgel.				
M_BOX_X_MIN +	Returns the X-coordinate of each edge's extreme left edgel.				
M_BOX_Y_MAX +	Returns the Y-coordinate of each edge's extreme bottom edgel.				
M_BOX_Y_MIN +	Returns the Y-coordinate of each edge's extreme top edgel.				
M_CENTER_OF_GRAVITY +	<p>Returns the coordinates of each edge's center of gravity. (summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Return values:</p> <table> <tr> <td>X-coordinate value</td><td>The X-coordinate of each edge's center of gravity.</td></tr> </table> </div> <div> <p><i>SecondResultArrayPtr info</i></p> <p>Return values:</p> <table> <tr> <td>Y-coordinate value</td><td>The Y-coordinate of each edge's center of gravity.</td></tr> </table> </div>	X-coordinate value	The X-coordinate of each edge's center of gravity.	Y-coordinate value	The Y-coordinate of each edge's center of gravity.
X-coordinate value	The X-coordinate of each edge's center of gravity.				
Y-coordinate value	The Y-coordinate of each edge's center of gravity.				
M_CENTER_OF_GRAVITY_X +	Returns the X-coordinate of each edge's center of gravity.				
M_CENTER_OF_GRAVITY_Y +	Returns the Y-coordinate of each edge's center of gravity.				
M_CIRCLE_FIT_CENTER_X +	Returns the X-coordinate of the center of the circle that is the best fit for each edge.				
M_CIRCLE_FIT_CENTER_Y +	Returns the Y-coordinate of the center of the circle that is the best fit for each edge.				
M_CIRCLE_FIT_COVERAGE +	<p>Returns the coverage of the circle that is the best fit for each edge. The coverage describes the portion of the circle covered by the edge. (summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Return values:</p> <table> <tr> <td>0.0 to 1.0</td><td>The lower the value, the lower the coverage. For example, 0 equals no coverage, 0.5 equals 50 percent coverage, and 1.0 equals 100 percent coverage.</td></tr> </table> </div>	0.0 to 1.0	The lower the value, the lower the coverage. For example, 0 equals no coverage, 0.5 equals 50 percent coverage, and 1.0 equals 100 percent coverage.		
0.0 to 1.0	The lower the value, the lower the coverage. For example, 0 equals no coverage, 0.5 equals 50 percent coverage, and 1.0 equals 100 percent coverage.				
M_CIRCLE_FIT_ERROR +	<p>Returns the fit error of the circle that is the best fit for each edge. This is calculated as the average quadratic error. (summarize)</p>				
M_CIRCLE_FIT_RADIUS +	Returns the radius of the circle that is the best fit for each edge.				
M_CLOSURE +	<p>Returns the closure status of each edge. (summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Return values:</p> <table> <tr> <td>M_FALSE</td><td>Specifies an open edge.</td></tr> <tr> <td>M_TRUE</td><td>Specifies a closed edge.</td></tr> </table> </div>	M_FALSE	Specifies an open edge.	M_TRUE	Specifies a closed edge.
M_FALSE	Specifies an open edge.				
M_TRUE	Specifies a closed edge.				
M_CONVEX_PERIMETER +	Returns the convex elongation of each edge.				
M_ELLIPSE_FIT_ANGLE +	Returns the angle of the ellipse that is the best fit for each edge.				
M_ELLIPSE_FIT_CENTER_X +	Returns the X-coordinate of the center of the ellipse that is the best fit for each edge.				
M_ELLIPSE_FIT_CENTER_Y +	Returns the Y-coordinate of the center of the ellipse that is the best fit for each edge.				
M_ELLIPSE_FIT_COVERAGE +	<p>Returns the coverage of the ellipse that is the best fit for each edge. The coverage describes the portion of the ellipse covered by the edge. (summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Return values:</p> <table> <tr> <td>0.0 to 1.0</td><td>The lower the value, the lower the coverage. For example, 0.0 equals no coverage, 0.5 equals 50 percent coverage, and 1.0 equals 100 percent coverage.</td></tr> </table> </div>	0.0 to 1.0	The lower the value, the lower the coverage. For example, 0.0 equals no coverage, 0.5 equals 50 percent coverage, and 1.0 equals 100 percent coverage.		
0.0 to 1.0	The lower the value, the lower the coverage. For example, 0.0 equals no coverage, 0.5 equals 50 percent coverage, and 1.0 equals 100 percent coverage.				
M_ELLIPSE_FIT_ERROR +	Returns the fit error of the ellipse that is the best fit for each edge.				
M_ELLIPSE_FIT_MAJOR_AXIS +	Returns the major axis of the ellipse that is the best fit for each edge. The major axis divides the ellipse across its long dimension into two equal halves.				

	 <p>(summarize)</p>
<input type="checkbox"/> M_ELLIPSE_FIT_MINOR_AXIS +	<p>Returns the minor axis of the ellipse that is the best fit for each edge. The minor axis divides the ellipse across its short dimension into two equal halves, perpendicular to the major axis.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FAST_LENGTH +	<p>Returns the fast length of each edge.</p>
<input type="checkbox"/> M_FERET_BOX +	<p>Returns the X- and Y-Feret values of each edge. The X-Feret is the dimension of the minimum bounding box of an edge in the horizontal direction. The Y-Feret is the dimension of the minimum bounding box of an edge in the vertical direction.</p> <p>(summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Return values:</p> <p>X-Feret value The X-Feret value of each edge.</p> </div> <div> <p><i>SecondResultArrayPtr info</i></p> <p>Return values:</p> <p>Y-Feret value The Y-Feret value of each edge.</p> </div>
<input type="checkbox"/> M_FERET_ELONGATION +	<p>Returns the Feret elongation of each edge. It is accurate for reasonably compact objects, but becomes less accurate for very elongated objects.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_MAX_ANGLE +	<p>Returns the maximum Feret angle of each edge, in degrees. This is the angle at which the maximum Feret diameter is found. Positive values indicate a counter-clockwise displacement from the positive X-axis.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_MAX_DIAMETER +	<p>Returns the maximum Feret diameter of each edge.</p>
<input type="checkbox"/> M_FERET_MEAN_DIAMETER +	<p>Returns the average Feret diameter at all the angles checked (see M_NUMBER_OF_FERETS).</p>
<input type="checkbox"/> M_FERET_MIN_ANGLE +	<p>Returns the minimum Feret angle of each edge.</p>
<input type="checkbox"/> M_FERET_MIN_DIAMETER +	<p>Returns the minimum Feret diameter of each edge.</p>
<input type="checkbox"/> M_FERET_X +	<p>Returns the X-Feret value of each edge. This is the dimension of the minimum bounding box of an edge in the horizontal direction.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FERET_Y +	<p>Returns the Y-Feret value of each edge. This is the dimension of the minimum bounding box of an edge in the vertical direction.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FIRST_POINT +	<p>Returns the coordinates of each edge's first point.</p> <p>(summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Return values:</p> <p>X-coordinate value The first point X-coordinate of each edge.</p> </div> <div> <p><i>SecondResultArrayPtr info</i></p> <p>Return values:</p> <p>Y-coordinate value The first point Y-coordinate of each edge.</p> </div>
<input type="checkbox"/> M_FIRST_POINT_X +	<p>Returns the X-coordinate of each edge's first point.</p>
<input type="checkbox"/> M_FIRST_POINT_Y +	<p>Returns the Y-coordinate of each edge's first point.</p>
<input type="checkbox"/> M_GENERAL_FERET +	<p>Returns the general Feret of each edge. This is the Feret diameter calculated at M_GENERAL_FERET_ANGLE, which is set in MedgeControl().</p> <p>(summarize)</p>

M_LABEL_VALUE +	Returns the label value of each edge in an image. The label value is a positive integer greater or equal to one; each edge in an image has a unique label value. (summarize)
M_LENGTH +	Returns the length of each edge. M_LENGTH gives a more accurate but slower approximation of the edge's length than M_FAST_LENGTH . (summarize)
M_LINE_FIT_A +	Returns the <i>A</i> variable of the line that is the best fit for each edge.
M_LINE_FIT_B +	Returns the <i>B</i> variable of the line that is the best fit for each edge.
M_LINE_FIT_C +	Returns the <i>C</i> variable of the line that is the best fit for each edge.
M_LINE_FIT_ERROR +	Returns the fit error of the line that is the best fit for each edge. This is calculated as the average quadratic error. (summarize)
M_MOMENT_ELONGATION +	<p>Returns the moment elongation of each edge. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values: 0.0 to 1.0 For example, a straight edge has a null elongation value, while a circular edge has an elongation of 1.0.</p>
M_MOMENT_ELONGATION_ANGLE +	Returns the angle of the principal axis along each edge's moment elongation.
M_POSITION +	<p>Returns the X- and Y-position of each edge. The position of the edge is defined by the middle edgel of the edge. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values: X-position value The X-position of each edge.</p> <p><i>SecondResultArrayPtr info</i> Return values: Y-position value The Y-position of each edge.</p>
M_POSITION_X +	Returns the X-position of each edge. The position of the edge is defined by the middle edgel of the edge. (summarize)
M_POSITION_Y +	Returns the Y-position of each edge. The position of the edge is defined by the middle edgel of the edge. (summarize)
M_SIZE +	Returns the number of edgels in each edge.
M_STRENGTH +	Returns the strength value of each edge.
M_TORTUOSITY +	Returns the tortuosity measure of each edge.
M_X_MAX_AT_Y_MAX +	Returns the maximum X-coordinate at the maximum Y-coordinate of each edge. Together with M_BOX_Y_MAX , this is one of four contact points on the convex perimeter of the edge. (summarize)
M_X_MIN_AT_Y_MIN +	Returns the minimum X-coordinate at the minimum Y-coordinate of each edge. Together with M_BOX_Y_MIN , this is one of four contact points on the convex perimeter of the edge. (summarize)
M_Y_MAX_AT_X_MIN +	Returns the maximum Y-coordinate at the minimum X-coordinate of each edge. Together with M_BOX_X_MIN , this is one of four contact points on the convex perimeter of the edge. (summarize)
M_Y_MIN_AT_X_MAX +	Returns the minimum Y-coordinate at the maximum X-coordinate of each edge. Together with M_BOX_X_MAX , this is one of four contact points on the convex perimeter of the edge. (summarize)

Combination constant for the values listed in [For retrieving edge features results](#)

You can add the following value to the above-mentioned values to determine if the result is available in the result buffer.



● For edge feature results					
▢ Value	Description				
▢ M_AVAILABLE	<p>Returns whether an edge feature result type is available to be retrieved. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values:</p> <table> <tr> <td>M_NULL</td><td>The result is not available to be retrieved.</td></tr> <tr> <td>Value</td><td>The result is available to be retrieved.</td></tr> </table> <p><i>SecondResultArrayPtr info</i> Return values: Must be set to M_NULL.</p>	M_NULL	The result is not available to be retrieved.	Value	The result is available to be retrieved.
M_NULL	The result is not available to be retrieved.				
Value	The result is available to be retrieved.				

Combination constants for [M_FERET_MAX_DIAMETER](#); [M_FERET_MIN_DIAMETER](#); [M_GENERAL_FERET](#);

You can add one of the following values to the above-mentioned values to get the indices of the edgels from which the specified Feret was calculated.

● For M_FERET_MAX_DIAMETER, M_FERET_MIN_DIAMETER, or M_GENERAL_FERET					
▢ Value	Description				
▢ M_FIRST_FERET_INDEX	<p>Returns the first index of the edgel from which the specified Feret was calculated. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINS to determine the size of the array.</p> <p><i>SecondResultArrayPtr info</i> Return values: Must be set to M_NULL.</p>				
▢ M_FIRST_FERET_INDEX + M_SECOND_FERET_INDEX +	<p>Returns the first and second index of the edgel from which the specified Feret was calculated. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINS to determine the size of the array. Return values:</p> <table> <tr> <td>First index value</td><td>First edgel index.</td></tr> </table> <p><i>SecondResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINS to determine the size of the array. Return values:</p> <table> <tr> <td>Second index value</td><td>Second edgel index.</td></tr> </table>	First index value	First edgel index.	Second index value	Second edgel index.
First index value	First edgel index.				
Second index value	Second edgel index.				
▢ M_SECOND_FERET_INDEX	<p>Returns the second index of the edgel from which the specified Feret was calculated. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINS to determine the size of the array.</p> <p><i>SecondResultArrayPtr info</i> Return values: Must be set to M_NULL.</p>				

Combination constant for [M_CHAIN](#); [M_CHAIN_MAGNITUDE + M_CHAIN_ANGLE](#); [M_VERTICES](#); [M_CENTER_OF_GRAVITY](#); [M_FERET_BOX](#); [M_FIRST_POINT](#); [M_POSITION](#); [M_FIRST_FERET_INDEX + M_SECOND_FERET_INDEX](#);

You can add the following value to the above-mentioned values to get the results in a packed format.

If you pack values, result types that return values in both **FirstResultArrayPtr** and **SecondResultArrayPtr** will be returned together, interlaced in **FirstResultArrayPtr**. For example, if you decide to pack **MedgeGetResult()** with **M_CENTER_OF_GRAVITY**, **FirstResultArrayPtr** will contain both the X- and Y-coordinates of the edge's center of gravity (XY XY XY...).

● For packing values	
▢ Value	Description
▢ M_PACKED	Returns the specified values in a packed format. Note that only result types that return values in both FirstResultArrayPtr and SecondResultArrayPtr can be packed. (summarize)

Combination constants for the values listed in [For retrieving edge features results](#)

You can add one of the following values to the above-mentioned values to get statistical result information.

Note that results are only returned to **FirstResultArrayPtr**; therefore, **SecondResultArrayPtr** must be set to **M_NULL**.

● For retrieving statistics about results	
▢ Value	Description
▢ M_MAX	Returns the maximum value of the requested result type. (summarize) <i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: This array must be of size 1.
▢ M_MAX_ABS	Returns the maximum absolute value of the requested result type. (summarize) <i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: This array must be of size 1.
▢ M_MEAN	Returns the mean value of the requested result type. (summarize) <i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: This array must be of size 1.
▢ M_MIN	Returns the minimum value of the requested result type. (summarize) <i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: This array must be of size 1.
▢ M_MIN_ABS	Returns the minimum absolute value of the requested result type. (summarize) <i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: This array must be of size 1.
▢ M_STANDARD_DEVIATION	Returns the standard deviation of the requested result type. (summarize) <i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: This array must be of size 1.

Combination constants for the values listed in all tables except For displaying results in an interactive dialog box

You can add one of the following values to the above-mentioned values to cast the requested results to a required data type.

For specifying the data type	
Value	Description
M_TYPE_DOUBLE	<p>Casts the requested results to a <i>double</i>. This is the default value. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type double Array size: Array size depends on the result being cast. Note: When multiple results. • Data type: double Note: When a single result.
M_TYPE_LONG	<p>Casts the requested results to a <i>long</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type long Array size: Array size depends on the result being cast. Note: When multiple results. • Data type: long Note: When a single result.
M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: Array size depends on the result being cast. Note: When multiple results. • Data type: MIL_DOUBLE Note: When a single result.
M_TYPE_MIL_ID	<p>Casts the requested results to a <i>MIL_ID</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_ID Array size: Array size depends on the result being cast. Note: When multiple results. • Data type: MIL_ID Note: When a single result.
M_TYPE_MIL_INT	<p>Casts the requested results to a <i>MIL_INT</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: Array size depends on the result being cast. Note: When multiple results. • Data type: MIL_INT Note: When a single result.
M_TYPE_MIL_INT32	<p>Casts the requested results to a <i>MIL_INT32</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: Array size depends on the result being cast. Note: When multiple results. • Data type: MIL_INT32 Note: When a single result.
M_TYPE_MIL_INT64	<p>Casts the requested results to a <i>MIL_INT64</i>.</p>

[\(summarize\)](#)

FirstResultArrayPtr and SecondResultArrayPtr info

- Data type: array of type MIL_INT64
Array size: Array size depends on the result being cast.
Note: When multiple results.
- Data type: MIL_INT64
Note: When a single result.

FirstResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type double
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_ID
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- double
- long
- M_NULL
- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address of the first array in which to write the requested information.

You must set this parameter to **M_NULL** if you are setting the **ResultType** parameter to **M_INTERACTIVE**.

SecondResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type double
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_ID
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- double
- long
- M_NULL
- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address of the second array in which to write the requested information. If nothing is to be written, set **SecondResultArrayPtr** to **M_NULL**.

You must set this parameter to **M_NULL** if you are setting the **ResultType** parameter to **M_INTERACTIVE**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.

DLL Requires mil.dll; miledge.dll.

MedgeInquire

Synopsis

Inquire about an Edge Finder context or an Edge Finder result buffer.

Syntax

```
MIL_INT MedgeInquire(  
    MIL_ID ContextOrResultId,  
    MIL_INT InquireType,  
    void *UserVarPtr  
)
```

Description

This function inquires information about the specified Edge Finder context or Edge Finder result buffer.

Note that for an Edge Finder result buffer, this function only retrieves information about result buffer settings (set with [MedgeControl\(\)](#), for example). To retrieve results from the Edge Finder result buffer, use [MedgeGetResult\(\)](#).

By setting the [InquireType](#) parameter to [M_INTERACTIVE](#), you can view the setting of inquire types interactively.

Parameters

ContextOrResultId

Specifies either the Edge Finder context or the Edge Finder result buffer about which to inquire information. Both the Edge Finder context and the Edge Finder result buffer must have been previously allocated on the system using [MedgeAlloc\(\)](#) or [MedgeAllocResult\(\)](#), respectively.

InquireType

Specifies the context or result setting about which to inquire.

For an Edge Finder context, the [InquireType](#) parameter can be set to one of the following.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter [M_NULL](#).

● For a context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] For an Edge Finder context, opens a read-only dialog box that displays the settings of the inquire types of the specified Edge Finder context.

To inquire about the operation settings for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts, set this parameter to one of the following values:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For M_CONTOUR and M_CREST contexts (operation settings)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ACCURACY +	Returns the edgel accuracy of the edge extraction. (summarize)
	<i>UserVarPtr info</i>

	Return values: M_DEFAULT; M_DISABLE; M_HIGH; M_VERY_HIGH; (details)
<input type="checkbox"/> M_CHAIN_ALL_NEIGHBORS +	<p>Returns whether edge chains are built using all the available neighboring edgels. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_EXTRACTION_SCALE +	<p>Returns the image scale with which to extract the edges. Returns the scale of the image at which to do the edge extraction. Once the extraction is complete, the results are scaled to the original scale of the image. A lower extraction scale speeds up the search but can result in a less reliable result, including, the loss of important details and/or a reduction in the accuracy of the search results. M_EXTRACTION_SCALE is for advanced users of the Edge Finder module. The default setting usually provides the most accurate search results. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)</p>
<input type="checkbox"/> M_FILTER_MODE +	<p>Returns the mode in which to perform the edge extraction filter. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_KERNEL; M_RECURSIVE; (details)</p>
<input type="checkbox"/> M_FILTER_SMOOTHNESS +	<p>Returns the degree of smoothness (strength of denoising) of the edge extraction filter. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>
<input type="checkbox"/> M_FILTER_TYPE +	<p>Returns the type of filter used when performing the edge extraction. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DERICHE; M_FREI_CHEN; M_PREWITT; M_SHEN; M_SOBEL; (details)</p>
<input type="checkbox"/> M_FLOAT_MODE +	<p>Returns whether the entire edge extraction process is forced to be performed using floating-point precision calculations. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_KERNEL_DEPTH +	<p>Returns the depth of the convolution kernel used for kernel filtering mode. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 8; 16; M_DEFAULT; (details)</p>
<input type="checkbox"/> M_KERNEL_SIZE +	<p>Returns the actual size (same in X and Y) of the convolution kernel used for kernel filtering mode. Note that the kernel size can change according to the filter type and smoothness factor, the kernel's width and depth, and whether you are using floating-point calculations. (summarize)</p>
<input type="checkbox"/> M_KERNEL_WIDTH +	<p>Returns the maximum size (same in X and Y) of the convolution kernel set for kernel filtering mode. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_AUTO; M_DEFAULT; Value >= 3; (details)</p>
<input type="checkbox"/> M_MAGNITUDE_TYPE +	<p>Returns the type of magnitude value used to calculate the magnitude of the edge at each edgel position. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_NORM; M_SQR_NORM; (details)</p>
<input type="checkbox"/> M_MODIFICATION_COUNT +	<p>Returns the current value of the modification counter. The modification counter is increased by one each time settings for the context are modified. Although you cannot identify the modification counter's contents, you can compare them throughout your application to know if the context has been altered. If the modification counter has changed you can, for example, prompt the user to save before closing the application. (summarize)</p>

<input type="checkbox"/> M_OVERSCAN +	<p>Returns the type of overscan used by the convolution filters when extracting edges for kernel filtering mode. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_MIRROR; M_REPLACE; M_TRANSPARENT; (details)</p>
<input type="checkbox"/> M_OVERSCAN_REPLACE_VALUE +	<p>Returns the replacement value for the overscan pixel values when using replacement type overscan. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_REPLACE_MAX; M_REPLACE_MIN; Value; (details)</p>
<input type="checkbox"/> M_THRESHOLD_HIGH +	<p>Returns the user-defined upper bound of the hysteresis threshold. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)</p>
<input type="checkbox"/> M_THRESHOLD_LOW +	<p>Returns the user-defined lower bound of the hysteresis threshold. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)</p>
<input type="checkbox"/> M_THRESHOLD_MODE +	<p>Returns the threshold mode of the edge extraction. Note that lower threshold values result in a more sensitive edgel detection. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_HIGH; M_LOW; M_MEDIUM; M_USER_DEFINED; M_VERY_HIGH; (details)</p>
<input type="checkbox"/> M_THRESHOLD_TYPE +	<p>Returns the type of the hysteresis threshold used when performing the edge extraction. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_FULL_HYSTERESIS; M_HYSTERESIS; M_NO_HYSTERESIS; (details)</p>
<input type="checkbox"/> M_TIMEOUT +	<p>Returns the maximum edge extraction and calculation time for MedgeCalculate(). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; Value > 0; (details)</p>

To inquire about the operation settings for [M_CREST](#) Edge Finder contexts, set this parameter to the value below.

The following values require that you pass the value listed in the data-type area of the specified parameter.

● For M_CREST contexts (operation settings)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FOREGROUND_VALUE +	<p>Returns the color used to extract line crests from the image. (summarize)</p> <p><i>UserVarPtr info</i> Data type: MIL_DOUBLE Return values: M_ANY; M_DEFAULT; M_FOREGROUND_BLACK; M_FOREGROUND_WHITE; (details)</p>

To inquire whether internal buffers have been saved in the Edge Finder result buffer, set this parameter to one of the values below. These values can be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For M_CONTOUR and M_CREST contexts (internal buffers)	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_SAVE_ANGLE +	Returns whether the internal angle buffer used to extract edges is saved in the Edge Finder result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SAVE_CHAIN_ANGLE +	Returns whether the angle value at each edgel position is saved in the Edge Finder result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SAVE_CHAIN_MAGNITUDE +	Returns whether the magnitude value at each edgel position is saved in the Edge Finder result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SAVE_DERIVATIVES +	Returns whether the internal derivative buffers used to extract edges are saved in the Edge Finder result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SAVE_IMAGE +	Returns whether the source image is saved in the Edge Finder result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SAVE_MAGNITUDE +	Returns whether the internal magnitude buffer used to extract edges is saved in the Edge Finder result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SAVE_MASK +	Returns whether the mask buffer is saved in the Edge Finder result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)

To inquire about which edge features will be calculated for each edge, set this parameter to one of the values below. These values can be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts. Note that, when an edge feature has been calculated, **M_ENABLE** is returned.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a *MIL_DOUBLE*.

● For M_CONTOUR and M_CREST contexts (edge features)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_AVERAGE_STRENGTH +	Returns whether the average strength value of each edge will be calculated. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_BOX_X_MAX +	Returns whether the extreme right edgel coordinate of each edge will be calculated. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_BOX_X_MIN +	Returns whether the extreme left edgel coordinate of each edge will be calculated. (summarize)
	<i>UserVarPtr info</i>

	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_BOX_Y_MAX +	Returns whether the extreme bottom edgel coordinate of each edge will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_BOX_Y_MIN +	Returns whether the extreme top edgel coordinate of each edge will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_CENTER_OF_GRAVITY_X +	Returns whether the X-position of each edge's center of gravity will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_CENTER_OF_GRAVITY_Y +	Returns whether the Y-position of each edge's center of gravity will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_CIRCLE_FIT_CENTER_X +	Returns whether the X-coordinate of the center of the circle that is the best fit for each edge will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_CIRCLE_FIT_CENTER_Y +	Returns whether the Y-coordinate of the center of the circle that is the best fit for each edge will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_CIRCLE_FIT_COVERAGE +	Returns whether the coverage of the circle that is the best fit for each edge will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_CIRCLE_FIT_ERROR +	Returns whether the fit error of the circle that is the best fit for each edge will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_CIRCLE_FIT_RADIUS +	Returns whether the radius of the circle that is the best fit for each edge will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_CLOSURE +	Returns whether the closure of each edge will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_CONVEX_PERIMETER +	Returns whether the convex elongation of each edge will be calculated. (summarize)
	UserVarPtr info Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_ELLIPSE_FIT_ANGLE +	Returns whether the angle of the ellipse that is the best fit for each edge will be calculated.

	(summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_ELLIPSE_FIT_CENTER_X +	Returns whether the X-coordinate of the center of the ellipse that is the best fit for each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_ELLIPSE_FIT_CENTER_Y +	Returns whether the Y-coordinate of the center of the ellipse that is the best fit for each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_ELLIPSE_FIT_COVERAGE +	Returns whether the coverage of the ellipse that is the best fit for each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_ELLIPSE_FIT_ERROR +	Returns whether the fit error of the ellipse that is the best fit for each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_ELLIPSE_FIT_MAJOR_AXIS +	Returns whether the major axis of the ellipse that is the best fit for each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_ELLIPSE_FIT_MINOR_AXIS +	Returns whether the minor axis the ellipse that is the best fit for each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FAST_LENGTH +	Returns whether the length of each edge will be quickly calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FERET_ELONGATION +	Returns whether the Feret elongation of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FERET_MAX_ANGLE +	Returns whether the maximum Feret angle of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FERET_MAX_DIAMETER +	Returns whether the maximum Feret diameter of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FERET_MEAN_DIAMETER +	Returns whether the average Feret diameter of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div>

		Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FERET_MIN_ANGLE +		Returns whether the minimum Feret angle of each edge will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FERET_MIN_DIAMETER +		Returns whether the minimum Feret diameter of each edge will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FERET_X +		Returns whether the X-Feret value of each edge will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FERET_Y +		Returns whether the Y-Feret value of each edge will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FIRST_POINT_X +		Returns whether the X-coordinate of each edge's first point (starting point) will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_FIRST_POINT_Y +		Returns whether the Y-coordinate of each edge's first point (starting point) will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_GENERAL_FERET +		Returns whether the general Feret of each edge will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_LABEL_VALUE +		Returns whether the label value of each edge in an image will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_LENGTH +		Returns whether the length of each edge will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_LINE_FIT_A +		Returns whether the <i>A</i> variable of the line that is the best fit for each edge will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_LINE_FIT_B +		Returns whether the <i>B</i> variable of the line that is the best fit for each edge will be calculated. (summarize)
	UserVarPtr info	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_LINE_FIT_C +		Returns whether the <i>C</i> variable of the line that is the best fit for each edge will be calculated.

	(summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_LINE_FIT_ERROR +	Returns whether the fit error of the line that is the best fit for each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_MOMENT_ELONGATION +	Returns whether the moment elongation of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_MOMENT_ELONGATION_ANGLE +	Returns whether the angle of the principal axis along each edge's moment elongation will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_POSITION_X +	Returns whether the X-position of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_POSITION_Y +	Returns whether the Y-position of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SIZE +	Returns whether the number of edgels in each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_STRENGTH +	Returns whether the strength value of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_TORTUOSITY +	Returns whether the tortuosity measure of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_X_MAX_AT_Y_MAX +	Returns whether the X-maximum at Y-maximum contact point of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_X_MIN_AT_Y_MIN +	Returns whether the X-minimum at Y-minimum contact point of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_Y_MAX_AT_X_MIN +	Returns whether the Y-maximum at X-minimum contact point of each edge will be calculated. (summarize)
	<div>UserVarPtr info</div>

	Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_Y_MIN_AT_X_MAX +	Returns whether the Y-minimum at X-maximum contact point of each edge will be calculated. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)

To inquire about the corresponding edge feature associated with the specified sorting key, set this parameter to one of the values below. These values can be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For M_CONTOUR and M_CREST contexts (sorting key)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SORTn_DOWN +	Returns the feature associated with the <i>n</i> th sorting key (in descending order), where <i>n</i> stands for an integer between 1 and 3. If no feature has been associated, M_NULL is returned. (summarize)
<input type="checkbox"/> M_SORTn_UP +	Returns the feature associated with the <i>n</i> th sorting key (in ascending order), where <i>n</i> stands for an integer between 1 and 3. If no feature has been associated, M_NULL is returned. (summarize)

To inquire about general Edge Finder context settings, set this parameter to one of the values below. These values can be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For M_CONTOUR and M_CREST contexts (general settings)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CONTEXT_TYPE +	Returns the type of the Edge Finder context. (summarize)
	<i>UserVarPtr info</i> Return values: M_CONTOUR; M_CREST; (details)
<input type="checkbox"/> M_FILTER_POWER +	Returns the power of the filter used to extract edges. This indicates the decreasing factor of the noise variance. As the following image illustrates, <i>h</i> represents the filter values, and the sum is computed through the filter support (which can be either finite or infinite): $Power = \frac{\sum h^2}{(\sum h)^2}$ (summarize)
<input type="checkbox"/> M_MASK_SIZE_X +	Returns the X-size of the mask, in pixels. If no mask has been set (MedgeMask()), M_NULL is returned. (summarize)
<input type="checkbox"/> M_MASK_SIZE_Y +	Returns the Y-size of the mask, in pixels. If no mask has been set (MedgeMask()), M_NULL is returned. (summarize)

To inquire about the values used to calculate Feret settings, set this parameter to one of the values below. These values can be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts.

The following values require that you pass the value listed in the data-type area of the specified parameter.

● For M_CONTOUR and M_CREST contexts (Feret settings)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FERET_ANGLE_SEARCH_MAX +	Returns the end of the angular search region used for M_FERET_MAX_DIAMETER , M_FERET_MIN_DIAMETER , M_FERET_MAX_ANGLE , and M_FERET_MIN_ANGLE

	calculations. (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> <div>Return values: 0.0 to 360.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_FERET_ANGLE_SEARCH_MIN +	Returns the start of the angular search region used for M_FERET_MAX_DIAMETER , M_FERET_MIN_DIAMETER , M_FERET_MAX_ANGLE , and M_FERET_MIN_ANGLE calculations. (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> <div>Return values: 0.0 to 360.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_GENERAL_FERET_ANGLE +	Returns the angle at which to calculate M_GENERAL_FERET . (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> <div>Return values: 0.0 to 360.0; M_DEFAULT; M_MOMENT_ELONGATION_ANGLE; (details)</div> </div>
<input type="checkbox"/> M_NUMBER_OF_FERETS +	Returns the number of Ferets used to calculate M_FERET_MAX_DIAMETER , M_FERET_MIN_DIAMETER , M_FERET_MAX_ANGLE , M_FERET_MIN_ANGLE , M_FERET_ELONGATION , and M_CONVEX_PERIMETER . (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> <div>Return values: M_DEFAULT; Value; (details)</div> </div>

To inquire about the settings used when performing post-calculations on extracted edges, set this parameter to one of the values below. These values can be specified for both [M_CONTOUR](#) and [M_CREST](#) Edge Finder contexts.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

● For M_CONTOUR and M_CREST contexts (post-calculations)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_APPROXIMATION_TOLERANCE +	Returns the resolution of the edge approximation. (summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 100.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_CHAIN_APPROXIMATION +	Returns the simple geometric features used to approximate edges. (summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: M_DEFAULT; M_DISABLE; M_LINE; (details)</div> </div>
<input type="checkbox"/> M_FILL_GAP_ANGLE +	Returns the aperture angle where Edge Finder searches for edge extremity candidates when filling edge gaps. (summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 360.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_FILL_GAP_CANDIDATE +	Returns whether an edge extremity is filled with the other extremity of the same edge, or with an extremity of any edge. (summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: M_ANY; M_DEFAULT; M_SAME; (details)</div> </div>
<input type="checkbox"/> M_FILL_GAP_CONTINUITY +	Returns the continuity constraint used when performing edge gap filling, when more than one edge extremity candidate is present. (summarize)
	<div> <div>UserVarPtr info</div> </div>

	Return values: 0.0 to 100.0; M_DEFAULT; (details)
<input type="checkbox"/> M_FILL_GAP_DISTANCE +	Returns the maximum distance radius where Edge Finder searches for edge extremity candidates when filling edge gaps. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_INFINITE; Value; (details)
<input type="checkbox"/> M_FILL_GAP_POLARITY +	Returns the polarity constraint used when performing the filling of edge gaps. (summarize)
	<i>UserVarPtr info</i> Return values: M_ANY; M_DEFAULT; M_REVERSE; M_SAME; (details)

To inquire about Edge Finder result buffer settings, set this parameter to one of the following values:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For result buffers	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_CROSS_SIZE +	Returns the size of the cross used for certain drawing operations. For example, M_DRAW_EDGELS , M_DRAW_POSITION , and M_DRAW_CENTER_OF_GRAVITY are drawn using this cross. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_X +	Returns the X-coordinate of the drawing region's top-left corner in the source image. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_Y +	Returns the Y-coordinate of the drawing region's top-left corner in the source image. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_DRAW_SCALE_X +	Returns the scale factor in the X-direction to perform zoomed drawings of results. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)
<input type="checkbox"/> M_DRAW_SCALE_Y +	Returns the scale factor in the Y-direction to perform zoomed drawings of results. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)
<input type="checkbox"/> M_MODEL_FINDER_COMPATIBLE +	Returns whether the Edge Finder result buffer is ready to be used with a Model Finder context. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_NEAREST_NEIGHBOR_RADIUS +	Returns the radius distance used to select (MedgeSelect()) the closest edge or edges from a point. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)

To inquire about Edge Finder result buffer settings used when performing closest-edgel operations with [MedgeGetNeighbors\(\)](#), set this parameter to one of the values below.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

● For result buffers (closest-edgel operations)	
☐ Value	Description
☐ <code>M_NEIGHBOR_ANGLE +</code>	<p>Returns the gradient angle that an edgel must have, before being considered a candidate, when finding the closest edgels to a list of points. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>M_ANY</code>; <code>M_DEFAULT</code>; <code>M_REVERSE</code>; <code>M_SAME</code>; <code>M_SAME_OR_REVERSE</code>; (details)</p>
☐ <code>M_NEIGHBOR_ANGLE_TOLERANCE +</code>	<p>Returns the angular tolerance used for the angle constraint (M_NEIGHBOR_ANGLE), when finding the closest edgels to a list of points. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>0.0</code> to <code>360.0</code>; <code>M_DEFAULT</code>; (details)</p>
☐ <code>M_NEIGHBOR_MAXIMUM_NUMBER +</code>	<p>Returns the maximum number of closest edgel candidates that can be returned (for each source point), when finding the closest edgels to a list of points. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>M_DEFAULT</code>; Value; (details)</p>
☐ <code>M_NEIGHBOR_MINIMUM_SPACING +</code>	<p>Returns the minimum distance separating edgels within the same edge, in order for each to be considered potential candidates, when finding the closest edgels to a list of points. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>M_DEFAULT</code>; <code>M_INFINITE</code>; Value > = 1; (details)</p>
☐ <code>M_SEARCH_ANGLE +</code>	<p>Returns the search angle constraint (applied the Edge Finder result buffer), when finding the closest edgels to a list of points. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>0.0</code> to <code>360.0</code>; <code>M_DEFAULT</code>; (details)</p>
☐ <code>M_SEARCH_ANGLE_SIGN +</code>	<p>Returns the orientation to use for the search angle constraint, when finding the closest edgels to a list of points. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>M_DEFAULT</code>; <code>M_REVERSE</code>; <code>M_SAME</code>; <code>M_SAME_OR_REVERSE</code>; (details)</p>
☐ <code>M_SEARCH_ANGLE_TOLERANCE +</code>	<p>Returns the angular tolerance used for the search angle constraint, when finding the closest edgels to a list of points. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>0.0</code> to <code>360.0</code>; <code>M_DEFAULT</code>; (details)</p>
☐ <code>M_SEARCH_RADIUS_MAX +</code>	<p>Returns the maximum radius distance used to search for the closest edgel candidates that match source edgels. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>M_DEFAULT</code>; <code>M_INFINITE</code>; Value; (details)</p>
☐ <code>M_SEARCH_RADIUS_MIN +</code>	<p>Returns the minimum radius distance used to search for the closest edgel candidates that match source edgels. (summarize)</p> <p><i>UserVarPtr info</i> Return values: <code>M_DEFAULT</code>; Value; (details)</p>

To inquire about the system on which either the Edge Finder context or Edge Finder result buffer has been allocated, set this parameter to the value below.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

● For inquiring about the system	

Value	Description
M_OWNER_SYSTEM +	Returns the identifier of the system on which either the Edge Finder context or Edge Finder result buffer has been allocated. (summarize)
	<i>UserVarPtr info</i> Return values: MIL system identifier; M_DEFAULT_HOST; (details)

Combination constant for [the values listed in](#) all tables **except For a context**

You can add the following value to the above-mentioned values to get the default value of an inquire type.

For inquiring the default value	
Value	Description
M_DEFAULT	Returns the default value of the specified inquire type. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE

Combination constant for [the values listed in](#) all tables **except For a context**

You can add the following value to the above-mentioned values to determine whether an inquire type is sortable or supported for the Edge Finder context currently being inquired.

Note that to inquire if an inquire type is sortable, you must add [M_SUPPORTED](#) to the appropriate sort value ([M_SORTn_UP](#) or [M_SORTn_DOWN](#)) and the specified inquire type (for example, [M_SIZE](#) + [M_SORTn_UP](#) + [M_SUPPORTED](#)). If the inquire type is not supported for the Edge Finder context, or sortable, [M_NULL](#) is returned.

For inquiring if an inquire type is sortable or is supported	
Value	Description
M_SUPPORTED	Returns whether the specified inquire type is either sortable, or supported for the Edge Finder context. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE Return values: M_NULL Inquire type is not sortable, or is not supported. Value != 0 Inquire type is sortable, or is supported.

Combination constants for [the values listed in](#) all tables **except For a context**

You can add one of the following values to the above-mentioned values to cast the requested information to a required data type.

For specifying the data type	
Value	Description
M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . This is the default value. (summarize)
	<i>UserVarPtr info</i> Data type: double

<div><div></div><div>M_TYPE_LONG</div></div>	Casts the requested information to a <i>long</i> . (summarize)
	<div>UserVarPtr info</div> <div>Data type: long</div>
<div><div></div><div>M_TYPE_MIL_DOUBLE</div></div>	Casts the requested information to a <i>MIL_DOUBLE</i> . (summarize)
	<div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div>
<div><div></div><div>M_TYPE_MIL_ID</div></div>	Casts the requested information to a <i>MIL_ID</i> . Note that M_TYPE_MIL_ID should only be used with M_OWNER_SYSTEM . (summarize)
	<div>UserVarPtr info</div> <div>Data type: MIL_ID</div>
<div><div></div><div>M_TYPE_MIL_INT</div></div>	Casts the requested information to a <i>MIL_INT</i> . (summarize)
	<div>UserVarPtr info</div> <div>Data type: MIL_INT</div>
<div><div></div><div>M_TYPE_MIL_INT32</div></div>	Casts the requested information to a <i>MIL_INT32</i> . (summarize)
	<div>UserVarPtr info</div> <div>Data type: MIL_INT32</div>
<div><div></div><div>M_TYPE_MIL_INT64</div></div>	Casts the requested information to a <i>MIL_INT64</i> . (summarize)
	<div>UserVarPtr info</div> <div>Data type: MIL_INT64</div>

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• double• long• M_NULL• MIL_DOUBLE• MIL_ID• MIL_INT• MIL_INT32• MIL_INT64

Specifies the address in which the inquiry result will be written.

Return value

The return value is the required information, cast to a *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeMask

Synopsis

Mask regions of the Edge Finder context.

Syntax

```
void MedgeMask(
    MIL_ID ContextId,
    MIL_ID MaskImageId,
    MIL_INT ControlFlag
)
```

Description

This function allows you to set a mask for an Edge Finder context. The mask is applied to the image that uses the context. Subsequent calls to [MedgeCalculate\(\)](#) will therefore extract edges only in the context's unmasked regions.

You can also mask edges during [post-calculation](#). In this case, masked edges are excluded from the result buffer and subsequent calculations. Note that partially masked edges are cropped.

By setting the [ControlFlag](#) parameter to **M_INTERACTIVE**, you can create or modify the mask interactively.

Parameters

ContextId

Specifies the Edge Finder context in which to set the mask. The Edge Finder context must have been previously allocated on the required system using [MedgeAlloc\(\)](#).

MaskImageId

Specifies the identifier of the image buffer used to identify the masked pixels in the Edge Finder context.

For the identifier of the image buffer	
Value	Description
M_NULL	Specifies to ignore this parameter. This parameter must be set to M_NULL when removing the mask or creating/modifying the mask interactively. (summarize)
MIL image buffer identifier	Specifies the image buffer to use as a mask. If the size of the mask image is greater than the source image, the mask will be clipped. A masked pixel corresponds to a non-zero value in the mask buffer. Note that this buffer can be calibrated; however, this is not necessary, since the mask is applied on a pixel basis. (summarize)

ControlFlag

Specifies the function's control flag. This parameter must be set to one of the following values.

For specifying the function's control flag	
Value	Description
M_DEFAULT	Sets the image specified in MaskImageId as the mask.
M_INTERACTIVE	<i>[This is only applicable to Windows]</i> Opens a dialog box that allows you to create or modify the mask interactively. To draw the mask, place the cursor over the required area, click your mouse and drag.

You must set the [MaskImageId](#) parameter to `M_NULL`.
[\(summarize\)](#)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgePut

Synopsis

Put edge chains from user-supplied arrays into an Edge Finder result buffer.

Syntax

```
void MedgePut (
    MIL_ID EdgeResultId,
    MIL_INT NbEdgels,
    const MIL_INT *ArrayIndexPtr,
    const MIL_DOUBLE *ArrayXPtr,
    const MIL_DOUBLE *ArrayYPtr,
    const MIL_DOUBLE *ArrayAnglePtr,
    const MIL_DOUBLE *ArrayMagnitudePtr,
    MIL_INT ControlFlag
)
```

Description

This function copies edge chains from user-supplied arrays to a specified Edge Finder result buffer. This can be useful if you want to construct an Edge Finder result buffer that cannot be obtained from one image. You can therefore combine results from multiple Edge Finder result buffers to, for example, build a complete model to add to a Model Finder context.

You must only add edge chains that respect the following constraints, whereby pixels are considered connected based on an 8-connected lattice:

- Consecutive edgels must occupy separate connected pixels.
- No branches are allowed (they start or end a separate chain).
- There must not be an edge in the edge map that is more than 1 pixel wide. This means that the first and fourth edgel in the edge chain must not be in neighboring pixels.

All edge chains must be at the same scale and this scale must be the scale at which they were originally extracted. If edges were extracted at a different scale ([MedgeControl\(\)](#) with [M_EXTRACTION_SCALE](#)), you must convert them to their original extraction scale.

If edgels are calculated with pixel accuracy, you must provide edgels with coordinates that are integer values; in this case, the distance between consecutive edgels should be one pixel. If edgels are calculated with subpixel accuracy, you can provide edgels with coordinates that are double values; in this case, the distance between consecutive edgels is such that, each pixel touched by the edge corresponds to one edgel. To change the accuracy edgels are calculated with, use [MedgeControl\(\)](#) with [M_ACCURACY](#).

To use an Edge Finder result buffer that contains edges set using [MedgePut\(\)](#) with the Model Finder module, you must respect the following limitation. First, edges in the source image buffer must have been previously extracted and saved in the Edge Finder result buffer, using [MedgeCalculate\(\)](#). Second, the coordinates of the edge chains in the user-supplied arrays must not exceed the boundaries of the source image buffer. You can inquire the width and height of the source image buffer using [MedgeGetResult\(\)](#) with [M_SIZE_X](#) and [M_SIZE_Y](#).

The X- and Y-coordinates of the edgels you are adding to the Edge Finder result buffer must be provided in pixel units (as opposed to real-world values). If your current results have been calibrated, the coordinates added to those results will be automatically transformed to their appropriate real-world values. For more information, see the [Getting results in real-world units](#) section in [Chapter 5: Camera calibration](#).

Parameters

EdgeResultId

Specifies the identifier of the Edge Finder result buffer in which to put user-supplied data.

NbEdgels

Specifies the number of edgels to add to the Edge Finder result buffer.

ArrayIndexPtr

Specifies the address of the array containing the indices of the edgels' edges to add to the Edge Finder result buffer. If you do not want to provide this information, set this parameter to [M_NULL](#). In this case, only one edge will be

added.

ArrayXPtr

Specifies the address of the array containing the X-coordinate(s) of the edgel(s) to add to the Edge Finder result buffer.

ArrayYPtr

Specifies the address of the array containing the Y-coordinate(s) of the edgel(s) to add to the Edge Finder result buffer.

ArrayAnglePtr

Specifies the address of the array containing the orientation of the edgel(s) to add to the Edge Finder result buffer. If you do not want to provide this information, set this parameter to **M_NULL**.

Angle values must be mapped in the range of 0 to 255. That is, 0° corresponds to 0, and 360° corresponds to 256.

ArrayMagnitudePtr

Specifies the address of the array containing the magnitude of the edgel(s) to add to the Edge Finder result buffer. If you do not want to provide this information, set this parameter to **M_NULL**.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeRestore

Synopsis

Restore an Edge Finder context from disk.

Syntax

```
MIL_ID MedgeRestore(  
    MIL_CONST_TEXT_PTR Filename,  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *ContextIdPtr  
)
```

Description

This function restores an Edge Finder context that was previously saved to a file, using [MedgeSave\(\)](#) or [MedgeStream\(\)](#). This function restores all the Edge Finder context's settings that were in effect when the Edge Finder context was saved.

Parameters

Filename

Specifies the name and path of the file from which to restore the Edge Finder context. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

● For specifying the file name and path	
☐ Value	Description
<div>MIL_TEXT(MIL_TEXT_PTR FileName)</div>	<div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)</div> <div><div>Parameters</div><div>FileName</div><div>Specifies the drive, directory, and name of the file.</div></div>
<div>☐ M_INTERACTIVE</div>	<div>[This is only applicable to Windows] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.</div>

SystemId

Specifies the system on which to restore the Edge Finder context.

This parameter should be set to one of the following values:

● For specifying the system identifier	
☐ Value	Description
<div>☐ M_DEFAULT_HOST</div>	<div>Specifies the default Host system of the current MIL application.</div>
<div>☐ MIL system identifier</div>	<div>Specifies a valid system identifier, previously allocated using MsysAlloc().</div>

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the Edge Finder context identifier. Since the **MedgeRestore()** function also returns the Edge Finder context identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the Edge Finder context identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeSave

Synopsis

Save an Edge Finder context to a file, or save edge chains and/or edge approximations from an Edge Finder result buffer to a CAD (Computer-Aided Design) file.

Syntax

```
void MedgeSave(
    MIL_CONST_TEXT_PTR FileName,
    MIL_ID ContextOrResultId,
    MIL_INT ControlFlag
)
```

Description

This function saves all the information about the previously allocated Edge Finder context to disk. This information can be reloaded, using [MedgeRestore\(\)](#) or [MedgeStream\(\)](#). However, any associated calibration objects are not saved.

This function also saves calculated edges (edge chains and/or edge approximations) included in a previously calculated Edge Finder result buffer to disk, in a standard DXF format CAD file. Note that edge chains and edge approximations are saved in the DXF file in separate layers.

Note that the Edge Finder context or the calculated edges are saved in real-world units if there is a calibration object associated to them. Otherwise, they are saved in pixel units.

Parameters

FileName

Specifies the name and path of the file in which to save the Edge Finder context, or the destination CAD file. For easier use with other Matrox Imaging software products, when saving an Edge Finder context to a file, use the MEF file extension, and when saving calculated edges from an Edge Result buffer to a CAD file, use the DXF file extension. The function internally handles the opening and closing of this file. If this file already exists, it will be overwritten.

This parameter can be set to one of the following values:

● For specifying the file name and path	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile". (summarize)
	Parameters
	FileName Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.

ContextOrResultId

Specifies either the identifier of the Edge Finder context to save, or the identifier of the Edge Finder result buffer from which to save to a CAD file.

ControlFlag

Specifies whether to save edge chains or edge approximations to the DXF CAD file for Edge Finder result buffers. For Edge Finder contexts, this parameter must be set to **M_DEFAULT**.

For Edge Finder result buffers, this parameter must be set to one of the following values. Note that to save both edge chains and edge approximations, [ControlFlag](#) values can be combined (**M_CHAIN** + **M_CHAIN_APPROXIMATION**).

● For Edge Finder result buffers	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CHAIN	Specifies that edge chains will be saved.
<input type="checkbox"/> M_CHAIN_APPROXIMATION	Specifies that edge approximations will be saved.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeSelect

Synopsis

Select edges for calculations and result retrieval.

Syntax

```
void MedgeSelect(
    MIL_ID EdgeResultId,
    MIL_INT Operation,
    MIL_INT SelectionCriterion,
    MIL_INT Condition,
    MIL_DOUBLE Param1,
    MIL_DOUBLE Param2
)
```

Description

This function selects edges that meet a specified criterion. These edges will be included in or excluded from future operations (calculations and result retrieval), or deleted entirely from the Edge Finder result buffer. To call this function, [MedgeCalculate\(\)](#) must have been called at least once.

If this function is not called at least once, all edges are included by default. If there is more than one call to this function, the effect of the calls is cumulative, unless [M_INCLUDE_ONLY](#) or [M_EXCLUDE_ONLY](#) is specified as the operation to perform.

Once an edge has been excluded, it can typically be re-included by specifying [M_INCLUDE](#) or [M_INCLUDE_ONLY](#) in a future call to this function (with the correct criterion). However, if you use the result buffer with different images (in a call to [MedgeCalculate\(\)](#)), all results in the result buffer are discarded and all new edges are re-included.

Unlike most other MIL functions, if the edges have been calculated using a calibrated source image, you must specify relevant values in the real world. Note that you can change the output units using [McalControl\(\)](#) with [M_OUTPUT_UNITS](#) set to [M_PIXEL](#) or [M_WORLD](#).

You can use this function to select edges based on one of the following:

- A calculated edge feature (see [MedgeControl\(\)](#)), where the edge selection depends on whether the specified edge feature meets the specified condition.
- The inter-relationship of edges, where the edge selection depends on whether edges meet the specified box or chain condition of the specified edge or group of edges.
- The current status of edges, where the edge selection depends on a specific edge, all edges, all included edges, or all excluded edges. These will be included, included only, excluded, excluded only, or deleted. For example, you can include only ([M_INCLUDE_ONLY](#)) the excluded edges ([M_EXCLUDED_EDGES](#)), which will essentially swap the previously included and excluded edges.
- The proximity of an edge or edges to a specified point, where the edge selection depends on the specified radius, and the nearest neighbor condition.

You can also use this function to crop and select a portion of a specified edge.

For more information on the different uses of this function, see the [Calculating and retrieving results](#) section in [Chapter 9: Edge Finder](#) and the [Advanced edge extraction](#) section in [Chapter 9: Edge Finder](#).

By setting the [Operation](#) parameter to [M_INTERACTIVE](#), you can select the edges interactively.

Parameters

EdgeResultId

Specifies the identifier of the Edge Finder result buffer to be used in the edge selection process.

Operation

Specifies the operation to perform on the specified edges. Set this parameter to one of the following values.

• For specifying the operation

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DELETE	Deletes edges that meet the specified condition. M_DELETE affects only included edges, unless otherwise stated by the SelectionCriterion parameter. M_DELETE removes edges permanently from the Edge Finder result buffer and, consequently, prevents these edges from being re-included. (summarize)
<input type="checkbox"/> M_EXCLUDE	Excludes all edges that meet the specified condition. M_EXCLUDE affects only the status of currently included edges. (summarize)
<input type="checkbox"/> M_EXCLUDE_ONLY	Excludes only those edges that meet the specified condition and includes all others. The exclusion does not consider the present status of edges (whether they are excluded), except for edges that have been deleted (M_DELETE), unless otherwise stated by the SelectionCriterion parameter. (summarize)
<input type="checkbox"/> M_INCLUDE	Includes all edges that meet the specified condition. M_INCLUDE affects only the status of currently excluded edges. (summarize)
<input type="checkbox"/> M_INCLUDE_ONLY	Includes only those edges that meet the specified condition and excludes all others. The inclusion does not consider the present status of edges (whether they are included), except for edges that have been deleted (M_DELETE), unless otherwise stated by the SelectionCriterion parameter. (summarize)
<input type="checkbox"/> M_INTERACTIVE	Opens a dialog box that allows you to select the edges interactively. You must set the SelectionCriterion , Condition , SelectionCriterion Param1 , and Param2 parameters to M_NULL . (summarize)

SelectionCriterion

Specifies on what the selection criterion will be based. Set this parameter to **M_NULL** if you are setting the [Operation](#) parameter to **M_INTERACTIVE**.

To specify an edge selection based on edge features, the [SelectionCriterion](#) parameter must be set to one of the values below. The specified feature will be used as part of the selection criterion. Note that the specified Edge Finder result buffer must already contain the results for the specified feature (for more information on each edge feature, see [MedgeControl\(\)](#)).

● For specifying the selection criterion	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_AVERAGE_STRENGTH	Uses the average strength.
<input type="checkbox"/> M_BOX_X_MAX	Uses the extreme right edgel coordinate.
<input type="checkbox"/> M_BOX_X_MIN	Uses the extreme left edgel coordinate.
<input type="checkbox"/> M_BOX_Y_MAX	Uses the extreme bottom edgel coordinate.
<input type="checkbox"/> M_BOX_Y_MIN	Uses the extreme top edgel coordinate.
<input type="checkbox"/> M_CENTER_OF_GRAVITY_X	Uses the center of gravity's X-position.
<input type="checkbox"/> M_CENTER_OF_GRAVITY_Y	Uses the center of gravity's Y-position.
<input type="checkbox"/> M_CIRCLE_FIT_CENTER_X	Uses the X-coordinate of the center of the circle that is the best fit for each edge.
<input type="checkbox"/> M_CIRCLE_FIT_CENTER_Y	Uses the Y-coordinate of the center of the circle that is the best fit for each edge.
<input type="checkbox"/> M_CIRCLE_FIT_COVERAGE	Uses the coverage of the circle that is the best fit for each edge.
<input type="checkbox"/> M_CIRCLE_FIT_ERROR	Uses the fit error of the circle that is the best fit for each edge.
<input type="checkbox"/> M_CIRCLE_FIT_RADIUS	Uses the radius of the circle that is the best fit for each edge.
<input type="checkbox"/> M_CLOSURE	Uses the closure state.
<input type="checkbox"/> M_CONVEX_PERIMETER	Uses the convex elongation.
<input type="checkbox"/> M_FAST_LENGTH	Uses the quickly calculated length.

<input type="checkbox"/> M_FERET_ELONGATION	Uses the Feret elongation.
<input type="checkbox"/> M_FERET_MAX_ANGLE	Uses the maximum Feret angle.
<input type="checkbox"/> M_FERET_MAX_DIAMETER	Uses the maximum Feret diameter.
<input type="checkbox"/> M_FERET_MEAN_DIAMETER	Uses the average Feret diameter at all the angles checked (see M_NUMBER_OF_FERETS in MedgeControl()).
<input type="checkbox"/> M_FERET_MIN_ANGLE	Uses the minimum Feret angle.
<input type="checkbox"/> M_FERET_MIN_DIAMETER	Uses the minimum Feret diameter.
<input type="checkbox"/> M_FERET_X	Uses the X-Feret value.
<input type="checkbox"/> M_FERET_Y	Uses the Y-Feret value.
<input type="checkbox"/> M_FIRST_POINT_X	Uses the first point's X-coordinate.
<input type="checkbox"/> M_FIRST_POINT_Y	Uses the first point's Y-coordinate.
<input type="checkbox"/> M_GENERAL_FERET	Uses the general Feret.
<input type="checkbox"/> M_LABEL_VALUE	Uses the label value.
<input type="checkbox"/> M_LENGTH	Uses the length.
<input type="checkbox"/> M_LINE_FIT_A	Uses the <i>A</i> variable of the line that is the best fit for each edge.
<input type="checkbox"/> M_LINE_FIT_B	Uses the <i>B</i> variable of the line that is the best fit for each edge.
<input type="checkbox"/> M_LINE_FIT_C	Uses the <i>C</i> variable of the line that is the best fit for each edge.
<input type="checkbox"/> M_LINE_FIT_ERROR	Uses the fit error of the line that is the best fit for each edge.
<input type="checkbox"/> M_MOMENT_ELONGATION	Uses the moment elongation.
<input type="checkbox"/> M_POSITION_X	Uses the X-position.
<input type="checkbox"/> M_POSITION_Y	Uses the Y-position.
<input type="checkbox"/> M_SIZE	Uses the number of edgels.
<input type="checkbox"/> M_STRENGTH	Uses the strength.
<input type="checkbox"/> M_TORTUOSITY	Uses the tortuosity measure.
<input type="checkbox"/> M_X_MAX_AT_Y_MAX	Uses the maximum X-coordinate at the maximum Y-coordinate.
<input type="checkbox"/> M_X_MIN_AT_Y_MIN	Uses the minimum X-coordinate at the minimum Y-coordinate.
<input type="checkbox"/> M_Y_MAX_AT_X_MIN	Uses the maximum Y-coordinate at the minimum X-coordinate.
<input type="checkbox"/> M_Y_MIN_AT_X_MAX	Uses the minimum Y-coordinate at the maximum X-coordinate.

To specify an edge selection based on the inter-relationship of edges, you must set the [SelectionCriterion](#) parameter to either a specific edge, or to a group of edges. In this case, the [SelectionCriterion](#) parameter defines the subset of edges to look inside (**M_INSIDE_...**) or outside (**M_OUTSIDE_...**); that is, the edges to operate on are those relative to the edge(s) specified by the [SelectionCriterion](#) parameter. Note that some [Operation](#) and [SelectionCriterion](#) parameter values, when used together, do not have any effect, such as excluding ([M_EXCLUDE](#)) all the excluded edges ([M_EXCLUDED_EDGES](#)).

To specify the edge(s), set [SelectionCriterion](#) to one of the following values:

● For specifying the edge(s)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL_EDGES	Specifies all edges regardless of their present status, except for edges that have been deleted.
<input type="checkbox"/> M_EXCLUDED_EDGES	Specifies all currently excluded edges, except for edges that have been deleted.
<input type="checkbox"/> M_INCLUDED_EDGES	Specifies all currently included edges, except for edges that have been deleted.
<input type="checkbox"/> M_SPECIFIC_EDGE	Specifies a specific edge. Use Param1 to set the label value of the edge. (summarize)

To specify an edge selection based on the current status of edges, the [SelectionCriterion](#) parameter must specify the edge or the status of the edges on which to operate. Note that in this case, the [SelectionCriterion](#) parameter should

be set to one of the same values as an edge selection based on the inter-relationship of edges (see above).

To specify an edge selection based on the proximity of the edges to a point, the [SelectionCriterion](#) parameter must be set to **M_NULL**.

To crop and select a portion of a specified edge, the [SelectionCriterion](#) parameter must be set to the following value:

● For cropping and selecting a portion of a specified edge	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CROP_CHAIN	Specifies a chain cropping. This is done by specifying the chain to crop (Param1), the side of the chain to crop (Condition), and the edgel index position where you want the chain to be cropped (Param2). The newly created edge receives a new label and can be excluded, included, or deleted from the result buffer. The excluded part of the edge keeps its label value. (summarize)

Condition

Specifies the condition for the edge selection. Set this parameter to **M_NULL** if you are setting the [Operation](#) parameter to **M_INTERACTIVE**.

To specify an edge selection based on edge features, the [Condition](#) parameter must either be set to a condition that uses one limit, or a condition that uses two limits.

To crop and select a portion of a specified edge, the [Condition](#) parameter must be set to a condition that uses one limit.

For conditions that use one limit, set the [Condition](#) parameter to one of the values below. Note that, unless otherwise specified, these values are valid for both feature-based and crop-based edge selections.

● For conditions that use one limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EQUAL	Specifies that edges with values for the specified feature equal to Param1 are included, excluded, or deleted from future operations on the specified Edge Finder result buffer. This value is not valid for crop-based edge selections. (summarize)
<input type="checkbox"/> M_GREATER	Specifies that edges with values for the specified feature greater than the specified limit value are included, excluded, or deleted from future operations on the specified Edge Finder result buffer. For feature-based edge selections, the limit value is set with Param1 . For crop-based edge selections, the limit value is set with Param2 . (summarize)
<input type="checkbox"/> M_GREATER_OR_EQUAL	Specifies that edges with values for the specified feature greater than or equal to the specified limit value are included, excluded, or deleted from future operations on the specified Edge Finder result buffer. For feature-based edge selections, the limit value is set with Param1 . For crop-based edge selections, the limit value is set with Param2 . (summarize)
<input type="checkbox"/> M_LESS	Specifies that edges with values for the specified feature less than the specified limit value are included, excluded, or deleted from future operations on the specified Edge Finder result buffer. For feature-based edge selections, the limit value is set with Param1 . For crop-based edge selections, the limit value is set with Param2 . (summarize)
<input type="checkbox"/> M_LESS_OR_EQUAL	Specifies that edges with values for the specified feature less than or equal to the specified limit value are included, excluded, or deleted from future operations on the specified Edge Finder result buffer. For feature-based edge selections, the limit value is set with Param1 . For crop-based edge selections, the limit value is set with Param2 . (summarize)
<input type="checkbox"/> M_NOT_EQUAL	Specifies that edges with values for the specified feature not equal to Param1 are included, excluded, or deleted from future operations on the specified Edge Finder result buffer. This value is not valid for crop-based edge selections. (summarize)

For conditions that use two limits, set the [Condition](#) parameter to one of the values below. Note that these values are only valid for feature-based edge selections.

● For conditions that use two limits	

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN_RANGE	Specifies that edges with values for the specified feature in the range Param1 to Param2 , inclusive, are included, excluded, or deleted from future operations on the specified Edge Finder result buffer.
<input type="checkbox"/> M_OUT_RANGE	Specifies that edges with values for the specified feature less than Param1 , or greater than Param2 , are included, excluded, or deleted from future operations on the specified Edge Finder result buffer.

To specify an edge selection based on the inter-relationship of edges, set the [Condition](#) parameter to one of the following box or chain values:

● For specifying an edge selection based on the inter-relationship of edges	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INSIDE_BOX	Specifies that edges inside the bounding box of the specified edge, or inside the bounding box of the edges of the specified group, are included, excluded, or deleted from future operations on the specified Edge Finder result buffer.
<input type="checkbox"/> M_INSIDE_CHAIN	Specifies that edges inside the specified edge, or inside edges of the specified group, are included, excluded, or deleted from future operations on the specified Edge Finder result buffer. Note that if an edge is not closed, this value has no effect. For more information, see M_CLOSURE in MedgeControl() . Also, if an edge is not completely included by closed edge, it is considered as an outside or equal edge. (summarize)
<input type="checkbox"/> M_INSIDE_OR_EQUAL_BOX	Specifies that edges inside or equal to the bounding box of the specified edge, or inside or equal to the bounding box of the edges of the specified group, are included, excluded, or deleted from future operations on the specified Edge Finder result buffer.
<input type="checkbox"/> M_INSIDE_OR_EQUAL_CHAIN	Specifies that edges inside or equal to the specified edge, or inside or equal to the edges of the specified group, are included, excluded, or deleted from future operations on the specified Edge Finder result buffer. Note that if an edge is not closed, this value has no effect. For more information, see M_CLOSURE in MedgeControl() . Also, if an edge is not completely included by closed edge, it is considered as an outside or equal edge. (summarize)
<input type="checkbox"/> M_OUTSIDE_BOX	Specifies that edges outside the bounding box of the specified edge, or outside the bounding box of the edges of the specified group, are included, excluded, or deleted from future operations on the specified Edge Finder result buffer.
<input type="checkbox"/> M_OUTSIDE_CHAIN	Specifies that edges outside the specified edge, or outside edges of the specified group, are included, excluded, or deleted from future operations on the specified Edge Finder result buffer. Note that if an edge is not closed, this value has no effect. For more information, see M_CLOSURE in MedgeControl() . Also, if an edge is not completely included by closed edge, it is considered as an outside or equal edge. (summarize)
<input type="checkbox"/> M_OUTSIDE_OR_EQUAL_BOX	Specifies that edges outside or equal to the bounding box of the specified edge, or outside or equal to the bounding box of the edges of the specified group, are included, excluded, or deleted from future operations on the specified Edge Finder result buffer.
<input type="checkbox"/> M_OUTSIDE_OR_EQUAL_CHAIN	Specifies that edges outside or equal to the specified edge, or outside or equal to edges of the specified group, are included, excluded, or deleted from future operations on the specified Edge Finder result buffer.

To specify an edge selection based on the current status of edges, set [Condition](#) to [M_NULL](#).

To specify an edge selection based on the proximity of the edges to a point, set the [Condition](#) parameter to one of the following neighbor values:

● For specifying an edge selection based on the current status of edges	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL_NEAREST_NEIGHBORS	Specifies that the closest edges within the specified radius and location is included, excluded, or deleted from future operations on the specified Edge Finder result buffer. Define the radius using MedgeControl() with M_NEAREST_NEIGHBOR_RADIUS . (summarize)
<input type="checkbox"/> M_NEAREST_NEIGHBOR	Specifies that the closest edge within the specified radius and location is included, excluded, or deleted from future operations on the specified Edge Finder result buffer. Define the radius using MedgeControl() with M_NEAREST_NEIGHBOR_RADIUS . (summarize)

Param1

Specifies a value that is dependent on the feature and condition chosen. If the edges have been calculated using a calibrated source image, you must specify the relevant values in the real world. Set this parameter to **M_NULL** if you are setting the **Operation** parameter to **M_INTERACTIVE**.

To specify an edge selection based on edge features, where the condition uses one limit, set **Param1** to the required condition limit. To specify an edge selection based on edge features, where the condition uses two limits, set **Param1** to the low condition limit.

To specify an edge selection based on the inter-relationship of edges, where **SelectionCriterion** is set to a specific edge (**M_SPECIFIC_EDGE**), set **Param1** to the label value of the edge. To specify an edge selection based on the inter-relationship of edges, where **SelectionCriterion** is set to a group of edges (**M_ALL_EDGES**, **M_EXCLUDED_EDGES**, **M_INCLUDED_EDGES**), set **Param1** to **M_NULL**.

To specify an edge selection based on the current status of edges, set **Param1** to **M_NULL**.

To specify an edge selection based on the proximity of the edges to a point, set **Param1** to the X-coordinate of this point.

To crop and select a portion of a specified edge (**M_CROP_CHAIN**), set **Param1** to the label value of the edge to crop.

Param2

Specifies a value that is dependent on the feature and condition chosen. If the edges have been calculated using a calibrated source image, you must specify the relevant values in the real world. Set this parameter to **M_NULL** if you are setting the **Operation** parameter to **M_INTERACTIVE**.

To specify an edge selection based on edge features, where the condition uses one limit, set **Param2** to **M_NULL**. To specify an edge selection based on edge features, where the condition uses two limits, set **Param2** to the high condition limit.

To specify an edge selection based on the inter-relationship of edges, where **SelectionCriterion** is either set to a specific edge (**M_SPECIFIC_EDGE**), or to a group of edges (**M_ALL_EDGES**, **M_EXCLUDED_EDGES**, **M_INCLUDED_EDGES**), set **Param2** to **M_NULL**.

To specify an edge selection based on the current status of edges, set **Param2** to **M_NULL**.

To specify an edge selection based on the proximity of the edges to a point, set **Param2** to the Y-coordinate of this point.

To crop and select a portion of a specified edge (**M_CROP_CHAIN**), set **Param2** to the edgel index where to split the edge.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

MedgeStream

Synopsis

Load, restore, or save an Edge Finder context from/to a file or memory stream, or save calculated edges from an Edge Finder result buffer to a file or memory stream in CAD DXF format.

Syntax

```
void MedgeStream(  
    MIL_TEXT_PTR MemPtrOrFileName,  
    MIL_ID SystemId,  
    MIL_INT Operation,  
    MIL_INT StreamType,  
    MIL_DOUBLE Version,  
    MIL_INT ControlFlag,  
    MIL_ID *ContextOrResultIdPtr,  
    MIL_INT *SizeByteVarPtr  
)
```

Description

This function can load, restore, or save an Edge Finder context from/to a file or memory stream. Moreover, this function can save previously calculated edges (edge chains and/or edge approximations), stored in an Edge Finder result buffer, to a file or memory stream in a standard CAD DXF format. Edge chains and edge approximations are saved in the DXF file or memory stream in separate layers.

Note that you cannot load or restore any CAD DXF file or memory stream using this function.

To inquire the number of bytes necessary to save an Edge Finder context or calculated edges to memory stream, you should call this function (**MedgeStream()**) first with **M_INQUIRE_SIZE_BYTE**.

Note that the content saved to memory stream is equivalent to the content saved to file. In addition, any file saved with **MedgeSave()** is equivalent to a file saved using this function.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using **MedgeStream()**, you can choose to save a backwards-compatible version of the Edge Finder context, which will work using a version of MIL that is up to one major release older than the current version (depending on which version is specified). For example, if you allocate a Edge Finder context using MIL 9.0 and save it to version 8.0, you can restore this context on a computer where MIL 8.0 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore Edge Finder contexts saved using MIL version 7.5 or above. Settings that do not exist in the lower version will be filled with default values when the Edge Finder context is loaded or restored.

Parameters

MemPtrOrFileName

Specifies the file or memory stream.

● For the file or memory stream	
☐ Value	Description
☐ MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving an Edge Finder context to a file, use the MEF file extension, and when saving calculated edges from an Edge Finder result buffer to a CAD file, use the DXF file extension. The function internally handles the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p> <p>To open or close the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p>
	Parameters

		<i>FileName</i> Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE .	
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. The parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)	
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (MedgeStream()) first with M_INQUIRE_SIZE_BYTE . (summarize)	

SystemId

Specifies the system on which to restore the Edge Finder context. For **M_INQUIRE_SIZE_BYTE**, **M_LOAD**, and **M_SAVE**, **SystemId** is ignored and should be set to **M_NULL**. For an **M_RESTORE** operation, this parameter should be set to one of the following values:

● For M_RESTORE		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.	
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .	

Operation

Specifies the operation to perform. This parameter must be set to one of the following values:

● For specifying the operation to perform		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save an Edge Finder context or calculated edges to a memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)	
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated Edge Finder context.	
<input type="checkbox"/> M_RESTORE	Restores an Edge Finder context from a file or memory stream and assigns it an identifier.	
<input type="checkbox"/> M_SAVE	Saves an Edge Finder context to a specified file or memory stream, or saves calculated edges from an Edge Finder result buffer to a file or memory stream in the standard CAD DXF format.	

StreamType









Specifies the type of stream in which to store/from which to restore the Edge Finder context or calculated edges. This parameter must be set to one of the following values:

● For the type of stream		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_FILE	Specifies a file stream.	
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)	

Version

Specifies the MIL version of the Edge Finder context. This parameter must be set to one of the following values.





● For specifying the MIL version		

 Value	Description
 M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL. For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
 M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
 M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
 M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
 M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
 M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
 M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Specifies the function's control flag. For any operation on Edge Finder contexts, this parameter must be set to **M_DEFAULT**.

For calculated edges from an Edge Finder result buffer, this parameter stipulates whether to save edge chains and/or edge approximations to the CAD DXF file or memory stream. In this case, this parameter must be set to one or a combination ([M_CHAIN](#) + [M_CHAIN_APPROXIMATION](#)) of the following values:

● For calculated edges from an Edge Finder result buffer	
 Value	Description
 M_DEFAULT	Same as M_CHAIN .
 M_CHAIN	Specifies that edge chains will be saved.
 M_CHAIN_APPROXIMATION	Specifies that edge approximations will be saved.

ContextOrResultIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the Edge Finder context or the Edge Finder result buffer.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, **ContextOrResultIdPtr** specifies the address of the variable from which to read the Edge Finder context identifier or Edge Finder result buffer identifier.

For an [M_LOAD](#) operation, **ContextOrResultIdPtr** specifies the address of the variable from which to read the identifier of the Edge Finder context where the file or memory stream content will be loaded.

For an [M_RESTORE](#) operation, **ContextOrResultIdPtr** specifies the address of the variable in which to return the identifier of the restored Edge Finder context. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the Edge Finder context or calculated edges, in bytes. If the size is not required, you can set this parameter to **M_NULL**.

Note that the size of Edge Finder context will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; miledge.lib.
DLL	Requires mil.dll; miledge.dll.

Mfpga functions

Synopsis

The functions prefixed with Mfpga make up the FPGA module. The FPGA module allows you to perform custom processing operations using an on-board Processing FPGA. Performing processing operations on-board frees up the Host for other tasks. You can allocate one or more command contexts to carry out a required operation using a specific processing unit (PU) on a target Processing FPGA. You can configure, link, and queue multiple commands, as well as store necessary information in PU user-specific registers. This module assumes that the appropriate FPGA configuration with the required PUs has been loaded into the Processing FPGA. For information on configuring a Processing FPGA and using this module, see [Chapter 28: Using MIL with a Processing FPGA](#).

Functions

- [MfpgaCommandAlloc](#)
- [MfpgaCommandControl](#)
- [MfpgaCommandFree](#)
- [MfpgaCommandInquire](#)
- [MfpgaCommandQueue](#)
- [MfpgaControl](#)
- [MfpgaGetHookInfo](#)
- [MfpgaGetRegister](#)
- [MfpgaHookFunction](#)
- [MfpgaInquire](#)
- [MfpgaLoad](#)
- [MfpgaSetDestination](#)
- [MfpgaSetLink](#)
- [MfpgaSetRegister](#)
- [MfpgaSetSource](#)

MfpgaCommandAlloc

Synopsis

Allocate an FPGA command context for a PU in the FPGA configuration loaded in a Processing FPGA on a target system.

Syntax

```
MIL_INT MfpgaCommandAlloc(
    MIL_ID MILSystemId,
    MIL_INT DeviceNum,
    MIL_INT FunctionId,
    MIL_INT SubfunctionId,
    MIL_INT FunctionNum,
    MIL_INT ExecutionMode,
    MIL_INT ControlFlag,
    MIL_FPGA_CONTEXT *FPGACommandContextPtr
)
```

Description

This function allocates an FPGA command context. An FPGA command context is used to contain the necessary command information to perform the required operation using the specified PU in a Processing FPGA on a target system, without writing it immediately to the target hardware.

Note that the FPGA command context is valid only for the thread on which the current command context is allocated. It cannot be referenced by any other thread.

After calling this function, you should ensure that the context was successfully allocated by verifying that the context handle is not **M_NULL**. For example, the context will not be allocated if the specified Processing FPGA does not contain the specified PU.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

In the Matrox Odyssey family of boards, this function is only supported on the Matrox Odyssey Xpro+ board.

Parameters

MILSystemId

Specifies the identifier of the system that has the required Processing FPGA.

DeviceNum

Specifies the Processing FPGA for which to allocate the command context. This parameter must be set to the following value:

For specifying the rank of the Processing FPGA																							
Value	Description	corona-II (a)	gigse vision (c)	gpus processing (d)	hellos ea/Xa (e)	hellos ec/Xd (f)	hellos ed/Xd (g)	leee 1.394 iildc (h)	lris (i)	met-II /dl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ec/Xcl (r)	odyssey ed/Xd (s)	solios ea/Xa (t)	solios ec/Xcl (u)	solios gige (v)	via (w)
M_DEVn	Specifies the rank of the Processing FPGA on the board, where <i>n</i> can be a value between 0 and the total number of Processing FPGAs-1.																q	r	s	t	u	v	

FunctionId

Specifies the function identifier of the required PU. The function identifier is specified in the header of the required PU's FPGA register file. Instead of directly using the function identifiers for Matrox PUs, you should use their provided equivalent FPGA constant, specified in the PU's reference description in the Matrox FPGA Component Reference. Note that the range of custom PU function identifiers is between 0xFC00 and 0xFFFF, inclusive.

ControlFlag

This parameter is reserved for future use. Set this parameter to **M_DEFAULT**.

FPGACommandContextPtr

Specifies the address of the variable in which to write the handle of the FPGA command context. The command context is valid only for the thread on which the command context is allocated. It cannot be referenced by any other thread.

Return value

The returned value is **M_VALID** if allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

Control a specified FPGA command context setting.

```
void MfpgaCommandControl(
    MIL_FPGA_CONTEXT FpgaCommandContext,
    MIL_INT ControlType,
    void *ControlValuePtr
)
```

This function controls the various settings of the specified FPGA command context. To inquire information about an FPGA command context setting, see [MfpgaCommandInquire\(\)](#). To control or inquire about a general Processing FPGA setting, refer to [MfpgaControl\(\)](#) or [MfpgaInquire\(\)](#), respectively.

FpgaCommandContext

Specifies the handle of the FPGA command context associated with the PU.

ControlType

Specifies the FPGA command context setting to control.

See the [Parameter associations](#) section for possible values.

ControlValuePtr

Specifies the address of the variable which contains the value to assign to the command context setting.

See the [Parameter associations](#) section for possible values.

Possible values for the **ControlType** and **ControlValuePtr** parameters are described in the table below.

- For controlling FPGA Command Settings

For controlling FPGA Command Settings						
ControlType		Description	vio (w) solos gige (v) solos ed/xcl (u) solos ea/xa (t) odyssey ed/xcl (s) odyssey ea/xa (q) nexus (p) morphis qxt (o) morphis (m) met-II/std (m) met-II/mrc (l) met-II/dlg (k) met-II/ccl (i) iris (i) leece 1394/liic (h) hellios ed/xcl (g) hellios eci/xcl (f) hellios ea/xa (e) gpu processing (d) gige vision (c) cronoplus (b) corona-II (a)			
ControlValuePtr						
M_COMPLETION_MODE		Specifies how the processing operation should be issued on the system command queue. Note that this parameter overrides the setting specified using MfpgaCommandAlloc() with the ExecutionMode parameter. (summarize)			t	u v
M_DEFAULT		Specifies that the command is queued according to the thread synchronization mode. See			t	u v

MfpgaCommandFree

Synopsis

Free an FPGA command context.

Syntax

```
MIL_INT MfpgaCommandFree(  
    MIL_FPGA_CONTEXT FpgaCommandContext,  
    MIL_INT ControlFlag  
)
```

Description

This function deallocates a previously allocated FPGA command context. The FPGA command context should be freed after an [MfpgaCommandQueue\(\)](#) is issued. The processing operation that is associated with the command context, will complete its operation even if **MfpgaCommandFree()** is executed before processing is finished.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
In the Matrox Odyssey family of boards, this function is only supported on the Odyssey Xpro+ board.

Parameters

- FpgaCommandContext
- Specifies the handle of the FPGA command context to deallocate. The command context must have been previously allocated on the system using [MfpgaCommandAlloc\(\)](#).
- ControlFlag
- This parameter is reserved for future use. Set this parameter to **M_DEFAULT**.

Return value

The returned value is **M_VALID** if deallocation is successful. If deallocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

MfpgaCommandInquire

Synopsis

Inquire about a specified FPGA command context setting.

Syntax

```
void MfpgaCommandInquire(
    MIL_FPGA_CONTEXT FpgaCommandContext,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquire information about a specified FPGA command context setting.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

In the Matrox Odyssey family of boards, this function is only supported on the Matrox Odyssey Xpro+ board.

Parameters

FpgaCommandContext

Specifies the handle of the FPGA command context associated with the PU.

InquireType

Specifies the type of setting about which to inquire. These inquire types correspond to register fields of the PU associated with the specified FPGA command context. This parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a MIL_INT.

For specifying the type of setting																							
Value	Description	corona-II (a)	gige vision (c)	gige vision (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ieee 1394 i/dc (h)	iris (i)	met-II /d (j)	met-II /dig (k)	met-II /rnc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nextis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	vio (w)
M_FUNCTION_ID	Returns the PU's function identifier.																q	r	s	t	u	v	
M_INSTANCE_ID	Returns the PU's instance number.																				t	u	v
M_MAJOR_VERSION	Returns the major version number of the PU.																q	r	s	t	u	v	
M_MINOR_VERSION	Returns the minor version number of the PU.																q	r	s	t	u	v	
M_NATIVE_ID	Returns the native function identifier of the command context. This identifier can be used when mixing board-specific code (from the native library function set) with MIL code. (summarize)																				t	u	v
M_NUMBER_OF_EVENTS	Returns the total number of interrupts that the PU can generate.																q	r	s	t	u	v	
M_REG_IOCTL_OFFSET	Returns the position of the I/O control register space, relative to the start of the PU's register space, in quadwords.																q	r	s	t	u	v	
M_REG_USER_FULL_SIZE	Returns the size of the PU's entire register space, which includes the header, I/O control, and user-specific register sections, in quadwords.																				t	u	v

[illegible]

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- MIL_INT

Specifies the address in which to write the requested information.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

MfpgaCommandQueue

Synopsis

Put an FPGA command on the system command queue of the current thread.

Syntax

```
void MfpgaCommandQueue(
    MIL_FPGA_CONTEXT FpgaCommandContext,
    MIL_INT CompletionMode,
    MIL_INT QueueType
)
```

Description

This function sends the command, defined by the specified FPGA command context, to the system command queue of the current thread. The command will be executed when the target Processing FPGA hardware resources are available.

If you link multiple command contexts, their commands should be gathered in a complex command, using `MfpgaCommandQueue()` with `M_WAIT`. Add the last command to the complex command using `M_DISPATCH`. If multiple commands are being dispatched at the same time (as in cascaded or parallel scenarios), then MIL uses the synchronous/asynchronous setting and the completion mode of the last command (that is, the one with the call to `MfpgaCommandQueue()` with `M_DISPATCH`). Set the completion mode for all other commands in the complex command to `M_DEFAULT`.

Note that two commands can typically run at the same time if they do not reference the same buffer and use different FPGA components that can access their buffers using different paths. When a command is sent asynchronously, you can use `MthrWait()` with `M_THREAD_WAIT` to force the current thread to wait for the completion of all commands in the thread's system command queue. Alternatively, you can use `MbufHookFunction()` with `M_MODIFIED_BUFFER` to notify your MIL application when the operation has finished processing the image.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

In the Matrox Odyssey family of boards, this function is only supported on the Matrox Odyssey Xpro+ board.

Parameters

FpgaCommandContext

Specifies the handle of the FPGA command context associated with the PU, which defines the command. The command context must have been previously allocated on the system using [MfpgaCommandAlloc\(\)](#).

CompletionMode

Specifies when the processing operation will be tagged as completed. This parameter can be set to one of the following values.

● For specifying when the command is completed																												
Value	Description	corona-II (a)	gige vision (c)	coronaplus (b)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ea/xa (g)	helios ec/xd (h)	helios ea/xa (i)	helios ec/xd (j)	helios ea/xa (k)	met-II /cl (l)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qct (o)	nextis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	odyssey ea/xa (t)	sollos ec/xd (u)	sollos gige (v)	sollos gige (w)	sollos ec/xd (x)	sollos ec/xd (y)	
M_DEFAULT	Specifies the default value. If any of the processing operations have destination buffers, the default value will be M_DESTINATION_WRITTEN . If none of the processing operations have a destination buffer but at least support interrupts, then the default value will be M_PROCESSING_COMPLETED . If none of the processing operations have a destination buffer and none of the processing operations support interrupts, then the default value will be M_SOURCE_READ . (summarize)																	q	r	s	t	u	v					
M_DESTINATION_WRITTEN	Specifies that the command is complete when all destination buffers are written. You should select this completion mode rather than M_PROCESSING_COMPLETED if you want to ensure that all Processing FPGA operations have completed and that the results are available to the Host.																						t	u	v			

MfpgaControl

Synopsis

Controls a global setting of a specified Processing FPGA.

Syntax

```
MIL_INT MfpgaControl(  
    MIL_ID MilSystemId,  
    MIL_INT FpgaDeviceNumber,  
    MIL_INT ControlType,  
    MIL_INT *ControlValuePtr  
)
```

Description

This function controls global settings of a specified Processing FPGA. See [MfpgaCommandControl\(\)](#) to retrieve information about a specific command context.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
In the Matrox Odyssey family of boards, this function is only supported on the Odyssey Xpro+ board.

Parameters

MilSystemId

Specifies the identifier of the system that has the required Processing FPGA.

FpgaDeviceNumber

Specifies the Processing FPGA on the system to control. This parameter must be set to the following value:

For specifying the rank of the Processing FPGA															
Value	Description														
M_DEVn	Specifies the rank of the Processing FPGA to control, where <i>n</i> can be a value between 0 and the total number of Processing FPGAs-1.														
	corona-II (a)	gige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ec/Xcl (f)	helios ed/Xd (g)	ilee 1394 i/dc (h)	iris (i)	met-II /cl (j)	met-II /dlq (k)	met-II /std (m)	met-II /mc (l)	morphis (n)	morphis qxt (o)	nexis (p)
															odyssey ea/Xa (q)
															odyssey ec/Xcl (r)
															odyssey ed/Xd (s)
															solos ea/Xa (t)
															solos ec/Xcl (u)
															vio (w)
															solos gige (v)

ControlType

Specifies the Processing FPGA setting to control.

See the [Parameter associations](#) section for possible values.

ControlValuePtr

Specifies the value to assign to the Processing FPGA setting.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValuePtr** parameters are described in the table below.

- [For controlling Processing FPGA settings](#)

For controlling Processing FPGA settings																									
ControlType	Description																								
ControlValuePtr																									
<input type="checkbox"/> M_ERROR	Sets whether basic parameter checking occurs. Note that, if enabled, this will also report errors when attempting to associate a command context to a PU not in the FPGA configuration, and when attempting to use an invalid interrupt. (summarize)																								
<input type="checkbox"/> M_DEFAULT	Same as M_PRINT_ENABLE .																								
<input type="checkbox"/> M_PRINT_DISABLE	Disables printing of error messages.																								
<input type="checkbox"/> M_PRINT_ENABLE	Enables printing of error messages.																								

Return value

The returned value is **M_VALID** if successful. If the operation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

MfpgaGetHookInfo

Synopsis

Get information about a PU hook event.

Syntax

```
MIL_INT MfpgaGetHookInfo(
    MIL_ID EventId,
    MIL_INT InfoType,
    void *UserVarPtr
)
```

Description

This function allows you to get information about the event that caused the hook-handler function to be called. The `MfpgaGetHookInfo()` function should only be called within the scope of a PU hook-handler function (see `MfpgaHookFunction()`).

Parameters

EventId

Specifies the PU event identifier received by the hook-handler function (see [MfpgaHookFunction\(\)](#)).

InfoType

Specifies the type of information to get. This parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

[illegible]

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- MIL_DOUBLE
- MIL_INT

Specifies the address in which to write the requested information.

Return value

The returned value is **M_NULL** if successful. If the operation fails, a non-null (!**M_NULL**) value is returned.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

MfpgaGetRegister

Synopsis

Set up a request to read registers of a PU.

Syntax

```
void MfpgaGetRegister(
    MIL_FPGA_CONTEXT FpgaCommandContext,
    MIL_INT RegisterSection,
    MIL_INT Offset,
    MIL_INT Length,
    void *ValuePtr,
    MIL_INT ReadAccessFlag
)
```

Description

This function sets up a request to read registers of the PU associated with the specified command context. A maximum of four **MfpgaGetRegister()** calls can be made with any FPGA command context, although you can return the contents of multiple registers with each call. For a Matrox PU, you should consult the register file of the PU in the Matrox FPGA Components Reference. You can specify whether read accesses will be collected and made before processing or after the PU issues its end-of-processing interrupt.

For Matrox PUs and custom PUs created using the Matrox Processing Unit Designer, a C structure is created to represent each register of the PU. The structures are supplied in header files (`fpga_*.h`), located in the `Matrox Imaging\ML\Examples\SoliosFDK\Include` directory for Matrox Solios and in the `Matrox Imaging\odyssey\src\headers\local\fpga_*.h` directory for Matrox Odyssey. You must include the appropriate header file to use the structures. See the [Setting and retrieving results from PU registers](#) section in [Chapter 28: Using MIL with a Processing FPGA](#) for more information on the Matrox register structure.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

In the Matrox Odyssey family of boards, this function is only supported on the Matrox Odyssey Xpro+ board.

Parameters

FpgaCommandContext

Specifies the handle of the FPGA command context associated with the PU. The command context must have been previously allocated on the system using [MfpgaCommandAlloc\(\)](#).

RegisterSection

Specifies the section of the PU's register space to access. You must set this parameter to the following value:

For register space access					
<input checked="" type="checkbox"/> Value	Description				
		vld (w)	solios gige (v)	solios ecd/xcl (u)	solios ea/Xa (t)
				odyssey ecd/xcl (r)	odyssey ecd/xcl (s)
				odyssey ea/Xa (q)	nexis (p)
				morphis (n)	met-II /std (m)
				met-II /mc (l)	met-II /dig (k)
				met-II /ci (j)	irfs (i)
				ieee_1394 iildc (h)	helios ed/Xd (g)
					helios ecd/xcl (f)
					helios ea/Xa (e)
					gpu processing (d)
					gige vision (c)
					cronoplus (b)
					corona-II (a)
<input checked="" type="checkbox"/> M_USER	Specifies to access the user-specific section of the PU's register space.				

Offset

Specifies the offset from the start of the specified register section, from which to begin reading, in bytes. The offset must be a multiple of 4 bytes.

Length

Specifies how many bytes of the register to read. The length must be a multiple of 4 bytes.

ValuePtr

Specifies the address of the variable in which to write the value read from the register. This address must remain valid for the duration of the operation, otherwise a memory corruption will occur.

ReadAccessFlag

Specifies when the register read access must take place. This parameter must be set to one of the following values:

For specifying when to read the register															
<input type="checkbox"/> Value	Description														
<input type="checkbox"/> M_WHEN_COMPLETED	Accesses the registers after the PU finishes processing (according to MfpgaCommandQueue() completion mode).														
<input type="checkbox"/> M_WHEN_DISPATCHED	Accesses the registers during PU setup, prior to the start of processing.														
	corona-II (a)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ecl/xcl (f)	helios ed/xd (g)	leee 1394 ildc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)
															odyssey ea/xa (q)
															odyssey ed/xd (r)
															odyssey ed/xd (s)
															solios ea/xa (t)
															solios ecl/xd (u)
															solios gige (v)
															vio (w)

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

MfpgaHookFunction

Synopsis

Hook a function to an event generated by a PU on a target system.

Syntax

```
void MfpgaHookFunction(
    MIL_ID MILSystemId,
    MIL_INT DeviceNum,
    MIL_INT FunctionId,
    MIL_INT SubfunctionId,
    MIL_INT FunctionNum,
    MIL_INT HookType,
    MFPGAHOOKFCTPTR HookHandlerPtr,
    void *UserDataPtr
)
```

Description

This function allows you to attach or detach a user-defined function to an event generated by a specified PU in the FPGA configuration loaded in a Processing FPGA on a target system. Once a hook-handler function is defined and hooked to an event, it is automatically called when the event occurs. You can only hook one function to an event; additional hooks that are attached to the same event will not be handled.

Parameters

MILSystemId

Specifies the identifier of the system that has the required Processing FPGA.

DeviceNum

Specifies the Processing FPGA on the system which has the required PU. This parameter must be set to the following value:

For specifying the rank of the Processing FPGA					
Value	Description	solos wio (w)	solos gige (v)	solos eci/xci (u)	solos ea/xa (t)
M_DEVn	Specifies the rank of the Processing FPGA on the board, where n can be a value between 0 and the total number of Processing FPGAs-1.	odyssey ed/xd (s) odyssey eci/xci (r) odyssey ea/xa (q)	t	u	v
		next (p)	morphis qxt (o)	morphis (n)	met-II/std (m)
		met-II/mc (l)	met-II/dig (k)	met-II/ccl (j)	iris (i)
		leee 1.394 ilic (h)	hellios ed/xd (g)	hellios eci/xci (f)	hellios ea/xa (e)
		gpu processing (d)	gige vision (c)	cronoplus (b)	corona-II (a)

FunctionId

Specifies the function identifier of the PU. The function identifier is specified in the header of the target PU's FPGA register file. For the function identifiers of Matrox PUs, see the Matrox FPGA Component Reference. Note that the range of custom PU function identifiers is between 0xFC00 and 0xFDFD, inclusive. This parameter can be set to one of the following values:

For specifying the function identifier	
Value	Description
mo (w)	
solios gige (v)	
solios eci/xci (u)	
solios ea/xi (t)	
odyssey eci/xci	
odyssey ea/xi (t)	
odyssey ea/xi (t)	
nexts (p)	
morphis qxt (o)	
morphis (n)	
met-II/std (m)	
met-II/mic (l)	
met-II/dig (k)	
met-II/pl (i)	
iris (i)	
leee 1394 i/cd (t)	
hellas ed/xci (g)	
hellas eci/xci (f)	
hellas ea/xi (e)	
gpu processing	
gige vision (c)	
conosplus (b)	
corona-II (a)	

Upon successful completion, the hook-handler function should return **M_NULL**. Note, MFTYPE and MFPGAHOOKFCTPTR are reserved MIL predefined types for functions and data pointers, respectively.

UserDataPtr

Specifies the address of the user data that you want to make available to the hook-handler function. This address is passed to the hook-handler function, through its [UserDataPtr](#) parameter, when the specified event occurs. Set this parameter to **M_NULL** if not used.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

[illegible]

ValuePtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type MIL_INT
- array of type MIL_TEXT_CHAR
- MIL_INT

Specifies the address at which to write the requested information.

Return value

The returned value is **M_VALID** if successful. If the operation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

MfpgaLoad

Synopsis

Load a new FPGA configuration into a Processing FPGA on a target system.

Syntax

```
MIL_INT MfpgaLoad(
    MIL_ID MILSystemId,
    MIL_INT FpgaDeviceNum,
    MIL_TEXT_PTR FirmwareFile,
    MIL_INT ControlFlag
)
```

Description

This function loads new configuration data into a Processing PGA on the target system. This function allows you to avoid rebooting the Host and then using MilConfig to specify and **MsysAlloc()** to load a different FPGA configuration into a Processing FPGA. Loading a new FPGA configuration can take longer to complete than other commands. It is recommended that **MfpgaLoad()** not be called during a time-critical section of your application. You must wait for all processing to finish executing before loading the new configuration into the Processing FPGA, otherwise data corruption might occur.

Parameters

MILSystemId

Specifies the identifier of the system that has the required Processing FPGA.

FpgaDeviceNum

Specifies the rank of the Processing FPGA into which to load the new FPGA configuration. This parameter must be set to the following value:

For specifying the rank of the Processing FPGA			
Value	Description		
M_DEVn	Specifies the rank of the Processing FPGA on the board, where n can be a value between 0 and the total number of Processing FPGAs-1.	mo (w)	solos gige (v)
		solos eci/xd (u)	solos ea/xa (t)
		odyssey eci/xd (s)	odyssey ea/xa (q)
		odyssey eci/xd (r)	odyssey ea/xa (q)
		nexts (p)	nexts qdt (o)
		morphis (n)	morphis qdt (o)
		met-II/std (m)	met-II/std (m)
		met-II/mc (l)	met-II/mc (l)
		met-II/dig (k)	met-II/dig (k)
		met-II/cl (i)	met-II/cl (i)
		iris (j)	iris (j)
		ilee 1394 ilic (h)	ilee 1394 ilic (h)
		hellos ed/xd (g)	hellos ed/xd (g)
		hellos eci/xd (f)	hellos eci/xd (f)
		hellos ea/xa (e)	hellos ea/xa (e)
		gpu processing (d)	gpu processing (d)
		gige vision (c)	gige vision (c)
		chromoplus (b)	chromoplus (b)
		corona-II (a)	corona-II (a)

FirmwareFile

Specifies the name and path of the file from which to load the FPGA configuration.

For specifying the file name and path																							
Value	Description																						
<div>MIL_TEXT(MIL_TEXT_PTR <i>FileName</i></div>	Specifies the name and path of the file from which to load the FPGA configuration. (summarize)																						
		corona-II (a)	gige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ed/Xd (f)	helios ed/Xd (g)	ieee 1394 iIIdc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ed/Xd (r)	odyssey ed/Xd (s)	solios ea/Xa (t)	solios ed/Xcl (u)	solios ed/Xcl (v)	vio (w)	

)
	<i>Parameters</i>
	<i>FileName</i> Specifies the name and path of the file.

ControlFlag

This parameter is reserved for future use. Set this parameter to **M_DEFAULT**.

Return value

The returned value is **M_VALID** if successful. If the operation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

MfpgaSetDestination

Synopsis

Specify a destination buffer for the FPGA processing operation.

Syntax

```
void MfpgaSetDestination(
    MIL_FPGA_CONTEXT FpgaCommandContext,
    MIL_BUFFER_INFO DestBuf,
    MIL_INT StreamOutputNum,
    MIL_INT ControlFlag
)
```

Description

This function specifies a destination buffer for the FPGA processing operation. This buffer is used to save the data from the specified stream output port of the PU associated with the specified FPGA command context. To route the data to another PU, refer to [MfpgaSetLink\(\)](#).

All PUs have documented limitations on the types of buffers to which their stream output port(s) can output data. You must keep these limitations in mind when allocating your destination MIL buffer(s). This function will not automatically convert the buffers so that they are appropriate for the operation. For more information, refer to the PU documentation available in the Matrox FPGA Component Reference help file.

Note that in general, child buffers are not supported; only color-band child buffers are supported.

[Matrox Solios GigE; Matrox Solios eA/XA; Matrox Solios eCL/XCL]
When allocating destination buffers on Matrox Solios, special care should be taken regarding the buffer width. The width of buffers allocated in acquisition memory must be a multiple of 64 bytes. The width of destination buffers on the Host must be a multiple of 8 bytes.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
In the Matrox Odyssey family of boards, this function is only supported on the Matrox Odyssey Xpro+ board.

Parameters

FpgaCommandContext

Specifies the handle of the FPGA command context associated with the PU.

DestBuf


Specifies the handle of the destination buffer. Use [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#) to get the handle. The destination buffer can reside in Host memory or in on-board memory, if supported by the FPGA configuration. If allocated in Host memory, the destination buffer must be in non-paged, Processing FPGA accessible memory (using [MbufAlloc...\(\)](#) with [M_FPGA_ACCESSIBLE](#) + [M_HOST_MEMORY](#)). If allocated on-board, the destination buffer must be allocated in on-board, Processing FPGA accessible memory (using [MbufAlloc...\(\)](#) with [M_FPGA_ACCESSIBLE](#)).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
Note that Matrox Odyssey does not support destination buffers in Host memory.

StreamOutputNum

Specifies the stream output port whose data will be routed to the destination buffer. You must set this parameter to the following value:

For specifying the rank of the stream output port		
<input type="checkbox"/> Value	Description	
		vio (w)
		solios gige (v)
		solios ec/xcl (
		solios ea/xa (t
		odyssey ec/xc
		odyssey ea/xa
		nexis (n)
		morphis qxt (c
		morphis (n)
		met-II/std (tr
		met-II/mc (l)
		met-II/dig (k
		met-II/d (l)
		iris (l)
		leece 1394 i/c
		helios ed/xd (
		helios ec/xcl (
		helios ea/xa (t
		gpu processin
		gige vision (c)
		coronaplus (b)
		corona-II (a)

 **M_OUTPUTn** Specifies the rank of the stream output port. You can set *n* to a value between 0 and 9, inclusive.

ControlFlag

This is reserved for future use. Set this parameter to **M_DEFAULT**.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

MfpgaSetLink

Synopsis

Cascade the PUs of two commands, such that a stream output of one PU is routed to a stream input port of the other PU.

Syntax

```
void MfpgaSetLink(
    MIL_FPGA_CONTEXT SrcFpgaCommandContext,
    MIL_INT SrcStreamPort,
    MIL_FPGA_CONTEXT DestFpgaCommandContext,
    MIL_INT DestStreamPort,
    MIL_INT ControlFlag
)
```

Description

This function cascades (links) the ports of two PUs that are interconnected in the FPGA configuration and loaded in a Processing FPGA. This function allows you to route data from one PU's stream output port to the specified stream input port of another PU.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

In the Matrox Odyssey family of boards, this function is only support on the Odyssey Xpro+ board.

Parameters

SrcFpgaCommandContext

Specifies the handle of the FPGA command context to use in the link.

SrcStreamPort

Specifies the stream output port of the PU associated with the source command context. You must set this parameter to the following value:

For specifying the rank of the stream output port	
<div>Value</div>	<div>Description</div>
<div>M_OUTPUTn</div>	<div>Specifies the rank of the PU's stream output port to use in the link. You must set <i>n</i> to a value between 0 and 9, inclusive.</div> <div>(summarize)</div>

DestFpgaCommandContext

Specifies the handle of the destination command context to use in the link.

DestStreamPort

Specifies the stream input port of the PU associated with the destination command context. You must set this parameter to the following value:

For specifying the rank of the stream input port	
Value	Description
	coron
	gigen
	gpu
	hellc
	hellf
	hellr
	lrs
	met
	met
	met
	mor
	mor
	nexi
	odys
	odys
	odys
	solid
	solid
	wio

[illegible]

ControlFlag

This parameter is reserved for future use. Set this parameter to **M_DEFAULT**.

Compilation information

Header	Include mil.h; milpga.h.
Library	Use mil.lib; milpga.lib.
DLL	Requires mil.dll; milpga.dll.

MfpgaSetRegister

Synopsis

Set the contents of one or more FPGA registers.

Syntax

```
void MfpgaSetRegister(
    MIL_FPGA_CONTEXT FpgaCommandContext,
    MIL_INT RegisterSection,
    MIL_INT Offset,
    MIL_INT Length,
    void *ValuePtr,
    MIL_INT WriteAccessFlag
)
```

Description

This function writes values to FPGA registers of the PU associated with the specified command context. A maximum of four **MfpgaSetRegister()** calls can be made with any FPGA command context, although you can set the contents of multiple registers in each call. For a Matrox PU, you should consult the register file of the PU in the Matrox FPGA Components Reference. You can specify whether write accesses will be collected and made before processing. Note that for those PUs that have interrupts you can specify that write accesses will be collected after the PU issues its end-of-processing interrupt.

For Matrox PUs and custom PUs created using the Matrox Processing Unit Designer, a C structure is created to represent each register of the PU. The structures are supplied in header files (fpga_*.h), located in the *Matrox Imaging\MIL\Examples\SoliosFDK\Include* directory for Matrox Solios and in the *Matrox Imaging\odyssey\src\headers\local\mfpga_*.h* directory for Matrox Odyssey. You must include the appropriate header file to use the structures. See the [Setting and retrieving results from PU registers](#) section in [Chapter 28: Using MIL with a Processing FPGA](#) for more information on the Matrox register structure.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
In the Matrox Odyssey family of boards, this function is only supported on the Odyssey Xpro+ board.

Parameters

FpgaCommandContext

Specifies the handle of the FPGA command context associated with the PU.

RegisterSection

Specifies the section of the PU's register space to access. You must set this parameter to the following value:

For register space access															
<input checked="" type="checkbox"/> Value	Description														
															vio (w)
															solios gige (v)
															solios ecl/xcl (u)
															solios ea/xa (t)
															odyssey ed/xd (s)
															odyssey ecl/xcl (r)
															odyssey ea/xa (q)
															nexis (p)
															morphis qxt (o)
															morphis (n)
															met-II/std (m)
															met-II/mc (l)
															met-II/dig (k)
															met-II/cl (j)
															iris (i)
															leee 1394 i/dc (h)
															helios ed/xd (g)
															helios ed/xcl (f)
															helios ea/xa (e)
															gpu processing (d)
															gige vision (c)
															cronosplus (b)
															corona-II (a)
<input checked="" type="checkbox"/> M_USER	Specifies to access the user-specific section of the PU's register space.														
															q
															r
															s
															t
															u
															v

Offset

Specifies the offset from the start of the specified register section, from which to begin writing, in bytes. The offset must be a multiple of 4 bytes.

Length

Specifies the number of bytes to write in the register. The length must be a multiple of 4 bytes.

ValuePtr

Specifies the address of the variable in which to write the value to be sent to the register.

WriteAccessFlag

Specifies when the register write access must take place. This parameter must be set to one of the following values:

For specifying when to access the register															
<input type="checkbox"/> Value	Description														
<input type="checkbox"/> M_WHEN_COMPLETED	Accesses the registers after the PU issues its end-of-processing interrupt.														
<input type="checkbox"/> M_WHEN_DISPATCHED	Accesses the registers during PU setup, prior to the start of processing.														
	corona-II (a)														
	gpi vision (c)														
	helios ea/xa (e)														
	helios ec/xci (f)														
	helios ed/xd (g)														
	iris (i)														
	met-II /ci (j)														
	met-II /dlg (k)														
	met-II /mc (l)														
	met-II /std (m)														
	morphe (n)														
	morphe qxt (o)														
	nexis (p)														
	odyssey ea/xa (q)														
	odyssey ec/xd (r)														
	odyssey ed/xd (s)														
	solos ea/xa (t)														
	solos ed/xd (u)														
	solos glge (v)														
	vio (w)														

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

MfpgaSetSource

Synopsis

Specify a source buffer for the FPGA processing operation.

Syntax

```
void MfpgaSetSource(
    MIL_FPGA_CONTEXT FpgaCommandContext,
    MIL_BUFFER_INFO SrcBuf,
    MIL_INT StreamInputNum,
    MIL_INT ControlFlag
)
```

Description

This function specifies a source buffer for the FPGA processing operation. Data from the specified buffer is routed to the specified stream input port of the PU associated with the specified FPGA command context. To obtain the data from another PU, refer to [MfpgaSetLink\(\)](#).

All PUs have documented limitations on the types of buffers from which their stream input port(s) can receive data. You must keep these limitations in mind when allocating your source MIL buffer(s). This function will not automatically convert the buffers so that they are appropriate for the operation. For more information, refer to the PU documentation available in the Matrox FPGA Component Reference help file.

Note that in general, child buffers are not supported; only color-band child buffers are supported.

[Matrox Solios eA/XA; Matrox Solios eCL/XCL]

When allocating source buffers on Matrox Solios, special care should be taken regarding the buffer width. The width of buffers allocated in acquisition memory must be a multiple of 64 bytes.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

In the Matrox Odyssey family of boards, this function is only supported on the Matrox Odyssey Xpro+ board.

Parameters

FpgaCommandContext

Specifies the handle of the FPGA command context associated with the PU.

SrcBuf	DestBuf	CopyLen	CopyType	CopyStatus
0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4
5	5	5	5	5
6	6	6	6	6
7	7	7	7	7
8	8	8	8	8
9	9	9	9	9
10	10	10	10	10
11	11	11	11	11
12	12	12	12	12
13	13	13	13	13
14	14	14	14	14
15	15	15	15	15
16	16	16	16	16
17	17	17	17	17
18	18	18	18	18
19	19	19	19	19
20	20	20	20	20
21	21	21	21	21
22	22	22	22	22
23	23	23	23	23
24	24	24	24	24
25	25	25	25	25
26	26	26	26	26
27	27	27	27	27
28	28	28	28	28
29	29	29	29	29
30	30	30	30	30
31	31	31	31	31
32	32	32	32	32
33	33	33	33	33
34	34	34	34	34
35	35	35	35	35
36	36	36	36	36
37	37	37	37	37
38	38	38	38	38
39	39	39	39	39
40	40	40	40	40
41	41	41	41	41
42	42	42	42	42
43	43	43	43	43
44	44	44	44	44
45	45	45	45	45
46	46	46	46	46
47	47	47	47	47
48	48	48	48	48
49	49	49	49	49
50	50	50	50	50
51	51	51	51	51
52	52	52	52	52
53	53	53	53	53
54	54	54	54	54
55	55	55	55	55
56	56	56	56	56
57	57	57	57	57
58	58	58	58	58
59	59	59	59	59
60	60	60	60	60
61	61	61	61	61
62	62	62	62	62
63	63	63	63	63
64	64	64	64	64
65	65	65	65	65
66	66	66	66	66
67	67	67	67	67
68	68	68	68	68
69	69	69	69	69
70	70	70	70	70
71	71	71	71	71
72	72	72	72	72
73	73	73	73	73
74	74	74	74	74
75	75	75	75	75
76	76	76	76	76
77	77	77	77	77
78	78	78	78	78
79	79	79	79	79
80	80	80	80	80
81	81	81	81	81
82	82	82	82	82
83	83	83	83	83
84	84	84	84	84
85	85	85	85	85
86	86	86	86	86

Specifies the handle of the source buffer. Use `MfuncInquire()` with `M_BUFFER_INFO` to get the handle. The source buffer can only reside in on-board memory. If allocated on-board, the source buffer must be allocated in on-board, Processing FPGA accessible memory (using `MbufAlloc...()` with `M_FPGA_ACCESSIBLE`).

StreamInputNum

Specifies the stream input port to which to route data from the source buffer. You must set this parameter to the following value:

For specifying the rank of the stream input port	
Value	<div> <div>solos ec/xd (u)</div> <div>solos ea/xa (t)</div> <div>odyssey ed/xd (s)</div> <div>odyssey ec/xd (r)</div> <div>odyssey ea/xa (q)</div> <div>nexis (p)</div> <div>morphis qxt (o)</div> <div>morphis (n)</div> <div>met-II/std (m)</div> <div>met-II/mc (l)</div> <div>met-II/dig (k)</div> <div>met-II/cl (j)</div> <div>iris (i)</div> <div>leee 1394 ilic (h)</div> <div>hellos ed/xd (g)</div> <div>hellos ec/xd (f)</div> <div>hellos ea/xa (e)</div> <div>gpu processing (d)</div> <div>gige vision (c)</div> <div>chromopus (b)</div> <div>corona-II (a)</div> </div>
M_INPUTn	<div> <div>Specifies the rank of the stream input port. You can set n to a value between 0 and 9, inclusive.</div> <div>(summarize)</div> </div>

ControlFlag

This is reserved for future use. Set this parameter to **M_DEFAULT**.

Compilation information

Header	Include mil.h; milfpga.h.
Library	Use mil.lib; milfpga.lib.
DLL	Requires mil.dll; milfpga.dll.

Mfunc functions

Synopsis

The functions prefixed with Mfunc make up the Function Development module. The Function Development module allows programmers to define functions to extend MIL's functionality. Using this toolkit, you can implement functions and integrate them directly into the MIL library, where they behave like standard MIL functions (for example, respecting error handling and tracing).

Functions

- MfuncAlloc
- MfuncAllocId
- MfuncBufAncestorId
- MfuncBufAncestorOffsetBand
- MfuncBufAncestorOffsetBit
- MfuncBufAncestorOffsetX
- MfuncBufAncestorOffsetY
- MfuncBufAttribute
- MfuncBufFormat
- MfuncBufHostAddress
- MfuncBufHostAddressBand
- MfuncBufId
- MfuncBufMaxValue
- MfuncBufMinValue
- MfuncBufNativeId
- MfuncBufOwnerSystemId
- MfuncBufOwnerSystemType
- MfuncBufParentId
- MfuncBufParentOffsetBand
- MfuncBufParentOffsetX
- MfuncBufParentOffsetY
- MfuncBufPhysicalAddress
- MfuncBufPhysicalAddressBand
- MfuncBufPitch
- MfuncBufPitchByte
- MfuncBufSizeBand
- MfuncBufSizeBit
- MfuncBufSizeX
- MfuncBufSizeY
- MfuncBufType
- MfuncCall
- MfuncErrorReport
- MfuncFree
- MfuncFreeId
- MfuncInquire
- MfuncParamCheck
- MfuncParamDouble
- MfuncParamId
- MfuncParamIdPointer
- MfuncParamLong
- MfuncParamMilDouble
- MfuncParamMilInt
- MfuncParamMilInt32
- MfuncParamMilInt64
- MfuncParamPointer
- MfuncParamString
- MfuncParamValue

MfuncAlloc

Synopsis

Allocate a MIL function context for your user-defined function.

Syntax

```
MIL_ID MfuncAlloc(
    MIL_CONST_TEXT_PTR FunctionName,
    MIL_INT ParameterNumber,
    MFUNCFCTPTR SlaveFunctionPtr,
    MIL_CONST_TEXT_PTR SlaveFunctionDLLName,
    MIL_TEXT_PTR SlaveFunctionName,
    MIL_INT SlaveFunctionOpcode,
    MIL_INT InitFlag,
    MIL_ID *FuncIdPtr
)
```

Description

This function allows you to allocate a MIL function context for the current user-defined function. Call **MfuncAlloc()** in the master function of the user-defined function. **MfuncAlloc()** signals the creation of a user-defined MIL function context, and should be the first MIL function called in the master function. Once the function context is allocated, your function is known as a user-defined MIL function. A user-defined MIL function is considered a standard MIL function, respecting all MIL environment controls, such as tracing and error handling.

Parameters

FunctionName

Specifies the name of the user-defined function.

For specifying the name of the user-defined function		
Value	Description	
MIL_TEXT(MIL_TEXT_PTR FunctionName)	Specifies the name of the user-defined function. This is the same as the name of the current master function. This is also the name that you will use to call the user-defined MIL function in your application. The name of the function must be a null-terminated string. (summarize)	
	Parameters	
	FunctionName	Specifies the name of the user-defined function.

ParameterNumber

Specifies the number of parameters passed to the current user-defined function. Note that the number of parameters should not exceed 16.

SlaveFunctionPtr

Specifies the address of the slave function.

The slave function must be declared as follows:

```
void MFTYPE SlaveFunction(
    MIL_ID FunctionId
```

)

Parameters:

FunctionId

*Specifies the identifier of the user-defined MIL function that references this slave function. This is the identifier returned by the **MfuncAlloc()** function.*

If your slave function will be executed by a system with an on-board processor or if the your slave function's code will be compiled into a library file, you can set this parameter to **M_NULL**.

SlaveFunctionDLLName

Specifies the name of the library file which contains the slave function's code. The library file must be visible to the system executing the slave function.

When developing a user-defined MIL function which will be used in a Distributed MIL application, it is recommended that a copy of the library file be placed on all the slave nodes in the cluster, in a directory which is part of the node's path environment variable.

If the library file is not located in a directory which is part of the path environment variable, you can alternatively provide the full path to the file.

If your slave function will be executed by a system with an on-board processor or if the your slave function's code will be included in your application, you can set this parameter to **M_NULL**.

SlaveFunctionName

Specifies the name of the slave function to export from the library file.

This parameter must be set if a library file is provided to the [SlaveFunctionDLLName](#) parameter.

If your slave function will be executed by a system with an on-board processor or if the your slave function's code will be included in your application, you can set this parameter to **M_NULL**.

SlaveFunctionOpcode

Allows you to assign an opcode to the slave function. The opcode will be used to locate the slave function when an application executes the user-defined MIL function. To specify an opcode, you must select a user module (see the table below), and combine it with an offset. For example, **M_USER_MODULE_3** + 12.

• For assigning an opcode to the slave function

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_USER_MODULE_n +	Specifies the user module. Set <i>n</i> to a value between 1 and 7, inclusive. (summarize)

Combination constant for any of the possible values of the [SlaveFunctionOpcode](#) parameter

You can add the following value to the above-mentioned values to specify the offset.

• For specifying the offset

<input type="checkbox"/> Value	Description
<input type="checkbox"/> 1 to 32	Specifies the offset of the slave function opcode.

InitFlag

Allows you to provide more information about the user-defined MIL function.

The values below allow you to set whether the slave function is executed asynchronously or synchronously.

• For specifying whether the slave function is executed asynchronously or synchronously

Value	Description
M_DEFAULT	Specifies the default value. If the system on which the function context is allocated has a remote processor, the default is the same as M_ASYNCHRONOUS_FUNCTION + M_REMOTE , if the system does not have a remote processor, the default is the same as M_SYNCHRONOUS_FUNCTION + M_LOCAL . (summarize)
M_ASYNCHRONOUS_FUNCTION +	Specifies that the user-defined function will not wait for the slave function to finish executing before executing the next function (typically MfuncFree()) in the master function. You must specify a combination value from the table below . (summarize)
M_SYNCHRONOUS_FUNCTION +	Specifies that the user-defined function will wait for the slave function to finish executing before executing the next function in the master function. You must specify a combination value from the table below . (summarize)

Combination constants for [M_ASYNCHRONOUS_FUNCTION](#); [M_SYNCHRONOUS_FUNCTION](#);

You must add one of the following values to the above-mentioned values to set whether the slave function can be executed remotely (on a system with an on-board processor).

For specifying whether the slave function can be executed remotely	
Value	Description
M_LOCAL	Specifies that the slave function must be executed by the Host processor. Note that MIL functions called from the slave function will still be executed on their target system. This value cannot be added to M_ASYNCHRONOUS_FUNCTION . (summarize)
M_REMOTE	Specifies that the slave function will be executed remotely, if possible. If you select M_REMOTE , but do not compile your slave function with a cross-compiler specific to the on-board processor, an error will be generated. For more information, see the Steps to create a user-defined MIL function section in Chapter 27: The MIL function development module . (summarize)

Combination constants for [M_SYNCHRONOUS_FUNCTION](#);

You can add one of the following values to the above-mentioned value to set the type of the user-defined MIL function.

Note that both [M_ALLOC](#) and [M_FREE](#) imply that the user-defined function is [M_SYNCHRONOUS_FUNCTION](#), and cannot be added to [M_ASYNCHRONOUS_FUNCTION](#).

For setting the type of user-defined MIL function	
Value	Description
M_ALLOC	Specifies that the user-defined function is an allocation function, used to allocate a user-defined MIL object on a required system.
M_FREE	Specifies that the user-defined function frees an object allocated using a user-defined MIL allocation function. Note that when this value is used, the user-defined MIL function must accept the MIL identifier of the object to free as its only parameter. (summarize)

FuncIdPtr

Specifies the address of the variable in which to store the MIL identifier provided for the user-defined function. Since the [MfuncAlloc\(\)](#) function also returns the MIL identifier, you can set this parameter to [M_NULL](#).

Return value

The returned value is the function context identifier if the allocation is successful. If allocation fails, [M_NULL](#) is returned.

Compilation information

Header	Include mil.h.
--------	----------------

Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncAllocId

Synopsis

Associate a MIL identifier with a user-defined object.

Syntax

```
MIL_ID MfuncAllocId(  
    MIL_ID FunctionId,  
    MIL_UINT64 ObjectType,  
    void *ObjectPtr  
)
```

Description

This function allows you to associate a MIL identifier with a user-defined object (such as, a structure or an array). You can associate a MIL identifier with a user object so that it is recognized by MIL and treated as a standard MIL object during tracing or error handling. Once the object is associated with an identifier, the object is known as a user-defined MIL object.

The parameters passed to **MfuncAllocId()** do not allow you to specify whether the user-defined object is allocated on the Host or on a system with an on-board processor. For information on creating user-defined objects on systems with on-board processors and associating these object with MIL identifiers, see the [Associating a MIL identifier with a user-defined object](#) section in [Chapter 27: The MIL function development module](#).

You must also specify a MIL type for your object (**ObjectType** parameter). MIL allows up to 32 different user-defined MIL object types.

Use the MIL identifier to refer to the user-defined MIL object. To refer to the actual data grouped into this user-defined object, use the pointer to the object (**ObjectPtr** parameter).

To inquire about the type of this object, or the pointer to it, use the [MfuncInquire\(\)](#) function.

Parameters

FunctionId

Specifies a MIL function identifier. This parameter can be set to the following values:

● For a MIL function identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default MIL function identifier. Use this value to create user-defined MIL objects outside of a user-defined MIL function. (summarize)
<input type="checkbox"/> MIL function identifier	Specifies the identifier of a user-defined MIL function.

ObjectType

Specifies the MIL type of the user-defined object to be associated with a MIL identifier. To specify the type, you must specify one of the groups listed in the table below and combine it with an offset. The two object groups allow you to distinguish between custom created objects, especially when the created objects are to be used in different MIL modules.

● For specifying the MIL type of user-defined object	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_USER_OBJECT_1 +	Specifies that the user object falls under the first group of user object types. You must specify a combination value from the table below . (summarize)
<input type="checkbox"/> M_USER_OBJECT_2 +	Specifies that the user object falls under the second group of user object types.

You must specify a combination value from the table [below](#).
([summarize](#))

Combination constant for any of the possible values of the [ObjectType](#) parameter

You must add the following value to the above-mentioned values to set the offset.

The offset allows you to distinguish between the different object types of the same group (for example, [M_USER_OBJECT_1](#) + 0x0001).

● For distinguishing between the different object types	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> Value	Specifies the offset within the selected object type group. The value must have only one of its 16 least significant bits set. (summarize)

ObjectPtr

Specifies the address of the user-defined object that is to be associated with a MIL identifier. Note that user-defined objects stored in the memory of an on-board processor are not accessible from Host, and vice versa.

This object can be a structure, an array, or any other data type.

Return value

The returned value is the MIL identifier associated with the object if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufAncestorId

Synopsis

Return the MIL identifier of the ancestor buffer.

Syntax

```
MIL_ID MfuncBufAncestorId(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function retrieves the MIL identifier of the ancestor buffer.

Only child buffers have an ancestor buffer. The ancestor buffer is the buffer from which the specified buffer ([BufferInfoHandle](#)) ultimately originated. It is the root buffer; it does not have a parent buffer (it is not a child buffer of another buffer).

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the MIL identifier of the specified MIL buffer's ancestor buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_ANCESTOR_ID](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufAncestorOffsetBand

Synopsis

Return the band offset relative to the ancestor buffer.

Syntax

```
MIL_INT MfuncBufAncestorOffsetBand(
    MIL_BUFFER_INFO BufferInfoHandle
)
```

Description

This function returns the band offset of the specified buffer relative to its ancestor buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the band offset of the specified MIL buffer relative to its ancestor buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_ANCESTOR_OFFSET_BAND](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufAncestorOffsetBit

Synopsis

Return the bit offset relative to the ancestor buffer.

Syntax

```
MIL_INT MfuncBufAncestorOffsetBit(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function returns the bit offset of the specified MIL buffer relative its ancestor buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the bit offset of the specified MIL buffer relative its ancestor buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_ANCESTOR_OFFSET_BIT](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufAncestorOffsetX

Synopsis

Return the X-offset relative to the ancestor buffer.

Syntax

```
MIL_INT MfuncBufAncestorOffsetX(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function returns the X-offset of the specified MIL buffer relative its ancestor buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the X-offset of the specified MIL buffer relative its ancestor buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_ANCESTOR_OFFSET_X](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufAncestorOffsetY

Synopsis

Return the Y-offset relative to the ancestor buffer.

Syntax

```
MIL_INT MfuncBufAncestorOffsetY(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function returns the Y-offset of the specified MIL buffer relative its ancestor buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the Y-offset of the specified MIL buffer relative its ancestor buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_ANCESTOR_OFFSET_Y](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufAttribute

Synopsis

Return the requested format of the specified buffer as set using **MbufAlloc...**() with **Attribute**.

Syntax

```
MIL_INT64 MfuncBufAttribute(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function returns the requested format of the specified buffer as set using **MbufAlloc...**() with **Attribute**.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using **MfuncInquire()** with **M_BUFFER_INFO**.

Return value

The return value is a bit-encoded value that specifies the buffer's attributes, set at allocation.

Remark

- This function is equivalent to using **MbufInquire()** with **M_EXTENDED_ATTRIBUTE**, but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufFormat

Synopsis

Return the actual format of the specified buffer.

Syntax

```
MIL_INT64 MfuncBufFormat(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function returns the requested format of the specified buffer as set using **MbufAlloc...()** with Attribute.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using **MfuncInquire()** with **M_BUFFER_INFO**.

Return value

The return value is a bit-encoded value that specifies the buffer's current formatting.

Remark

- This function is equivalent to using **MbufInquire()** with **M_EXTENDED_FORMAT**, but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufHostAddress

Synopsis

Returns the Host address of the buffer.

Syntax

```
void MfuncBufHostAddress(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function returns the Host address of the specified MIL buffer, if the buffer is visible from the Host address space and is not a planar 3-band buffer.

For a planar 3-band buffer, you can determine its Host address using [MfuncBufHostAddressBand\(\)](#).

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the Host address of the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_HOST_ADDRESS](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufHostAddressBand

Synopsis

Returns the Host address of a band of the buffer.

Syntax

```
void MfuncBufHostAddressBand(
    MIL_BUFFER_INFO BufferInfoHandle,
    MIL_INT Band
)
```

Description

This function returns the Host address of a band of the specified MIL buffer, if the buffer is a planar 3-band buffer.

Parameters

- BufferInfoHandle
- Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).
- Band
- Specifies which band's address to return.

Return value

The return value is the Host address of a band of the specified MIL buffer.

Remark

- This function provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufId

Synopsis

Return the MIL identifier of the buffer.

Syntax

```
MIL_ID MfuncBufId(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function retrieves the MIL identifier of the specified buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the MIL identifier of the specified MIL buffer.

Remark

- This function provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufMaxValue

Synopsis

Return the maximum pixel value possible in the buffer.

Syntax

```
MIL_DOUBLE MfuncBufMaxValue(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function returns the maximum pixel value possible in the specified MIL buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the maximum pixel value possible in the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_MAX](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufMinValue

Synopsis

Return the minimum pixel value possible in the buffer.

Syntax

```
MIL_DOUBLE MfuncBufMinValue(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function returns the minimum pixel value possible in the specified buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the minimum pixel value possible in the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_MIN](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufNativeId

Synopsis

Return the native identifier of the buffer.

Syntax

```
MIL_INT MfuncBufNativeId(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function returns the native identifier of the specified buffer.

This identifier can be used when mixing board-specific code (from the native library function set) with MIL code.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the native identifier of the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_NATIVE_ID](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufOwnerSystemId

Synopsis

Return the MIL identifier of the system on which the buffer has been allocated.

Syntax

```
MIL_ID MfuncBufOwnerSystemId(
    MIL_BUFFER_INFO BufferInfoHandle
)
```

Description

This function retrieves the MIL identifier of the system on which the specified MIL buffer has been allocated.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the MIL identifier of the system on which the specified MIL buffer has been allocated.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_OWNER_SYSTEM](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufOwnerSystemType

Synopsis

Return the type of system on which the buffer has been allocated.

Syntax

```
MIL_INT MfuncBufOwnerSystemType(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function retrieves the type of system on which the specified MIL buffer has been allocated.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the type of system on which the specified MIL buffer has been allocated.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_OWNER_SYSTEM_TYPE](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufParentId

Synopsis

Return the MIL identifier of the parent buffer.

Syntax

```
MIL_ID MfuncBufParentId(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function retrieves the MIL identifier of the specified MIL buffers' parent buffer.

Only child buffers have a parent buffer. The parent buffer is the buffer from which the specified buffer was defined. The parent buffer can itself have a parent buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the MIL identifier of the specified MIL buffers' parent buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_PARENT_ID](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufParentOffsetBand

Synopsis

Return the band offset relative to the parent buffer.

Syntax

```
MIL_INT MfuncBufParentOffsetBand(
    MIL_BUFFER_INFO BufferInfoHandle
)
```

Description

This function returns the band offset of the specified buffer relative to its parent buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the band offset of the specified MIL buffer relative to its parent buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_PARENT_OFFSET_BAND](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufParentOffsetX

Synopsis

Return the X-offset relative to the parent buffer.

Syntax

```
MIL_INT MfuncBufParentOffsetX(
    MIL_BUFFER_INFO BufferInfoHandle
)
```

Description

This function returns the X-offset of the specified MIL buffer relative its parent buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the X-offset of the specified MIL buffer relative its parent buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_PARENT_OFFSET_X](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufParentOffsetY

Synopsis

Return the Y-offset relative to the parent buffer.

Syntax

```
MIL_INT MfuncBufParentOffsetY(
    MIL_BUFFER_INFO BufferInfoHandle
)
```

Description

This function returns the Y-offset of the specified MIL buffer relative its parent buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the Y-offset of the specified MIL buffer relative its ancestor buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_PARENT_OFFSET_Y](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufPhysicalAddress

Synopsis

Return the physical address of the buffer.

Syntax

```
void MfuncBufPhysicalAddress(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function retrieves the physical address of the specified MIL buffer, if it is not a planar 3-band buffer.

For a planar 3-band buffer, you can determine its physical address using [MfuncBufPhysicalAddressBand\(\)](#).

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the physical address of the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_PHYSICAL_ADDRESS](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufPhysicalAddressBand

Synopsis

Returns the physical address of a band of the buffer.

Syntax

```
void MfuncBufPhysicalAddressBand(
    MIL_BUFFER_INFO BufferInfoHandle,
    MIL_INT Band
)
```

Description

This function returns the physical address of a band of the specified MIL buffer, if the buffer is a planar 3-band buffer.

Parameters

- BufferInfoHandle
- Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).
- Band
- Specifies which band's address to return.

Return value

The return value is the physical address of a band of the specified MIL buffer.

Remark

- This function provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufPitch

Synopsis

Return the number of pixels between the beginnings of any two adjacent lines of the buffer.

Syntax

```
MIL_INT MfuncBufPitch(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function retrieves the number of pixels between the beginnings of any two adjacent lines of the specified MIL buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the number of pixels between the beginnings of any two adjacent lines of the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_PITCH](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufPitchByte

Synopsis

Return the number of bytes between the beginnings of any two adjacent lines of the buffer.

Syntax

```
MIL_INT MfuncBufPitchByte(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function retrieves the number of bytes between the beginnings of any two adjacent lines of the specified MIL buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the number of bytes between the beginnings of any two adjacent lines of the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_PITCH_BYTE](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufSizeBand

Synopsis

Return the number of buffer color bands.

Syntax

```
MIL_INT MfuncBufSizeBand(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function retrieves the number of color bands in the specified MIL buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the number of color bands in the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_SIZE_BAND](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufSizeBit

Synopsis

Return the depth per band, in bits.

Syntax

```
MIL_INT MfuncBufSizeBit(
    MIL_BUFFER_INFO BufferInfoHandle
)
```

Description

This function retrieves the depth per band, in bits, of the specified MIL buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the depth per band, in bits, of the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_SIZE_BIT](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufSizeX

Synopsis

Return the width of the buffer, in pixels.

Syntax

```
MIL_INT MfuncBufSizeX(
    MIL_BUFFER_INFO BufferInfoHandle
)
```

Description

This function retrieves the width, in pixels, of the specified MIL buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the width, in pixels, of the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_SIZE_X](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufSizeY

Synopsis

Return the height of the buffer, in pixels.

Syntax

```
MIL_INT MfuncBufSizeY(
    MIL_BUFFER_INFO BufferInfoHandle
)
```

Description

This function retrieves the height, in pixels, of the specified MIL buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the height, in pixels, of the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_SIZE_Y](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncBufType

Synopsis

Return the buffer data type and depth.

Syntax

```
MIL_INT MfuncBufType(  
    MIL_BUFFER_INFO BufferInfoHandle  
)
```

Description

This function retrieves the data type and depth, in bits, of the specified MIL buffer.

Parameter

BufferInfoHandle

Specifies the handle of the MIL buffer. The buffer handle must be obtained using [MfuncInquire\(\)](#) with [M_BUFFER_INFO](#).

Return value

The return value is the buffer data type and depth, in bits, of the specified MIL buffer.

Remark

- This function is equivalent to using [MbufInquire\(\)](#) with [M_TYPE](#), but provides no parameter checking or error reporting.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncCall

Synopsis

Execute the slave function.

Syntax

```
MIL_INT MfuncCall(  
    MIL_ID FunctionId  
)
```

Description

This function executes the slave function of the user-defined MIL function on the target system. The target system is determined by the values of the user-defined MIL function parameters, registered in the master function. (For more information, see the [Master/slave dynamics on a remote system](#) section in [Chapter 27: The MIL function development module](#).) Call **MfuncCall()** from the master function of the user-defined MIL function. Note that you must call **MfuncCall()** from the same thread as **MfuncAlloc()** and **MfuncFree()**. If tracing and error reporting are enabled, the tracing and error messages will be reported to screen. You can control the error handling and tracing behavior as you would with other MIL functions, using [MappControl\(\)](#).

Note that if a MIL_ID parameter was registered in the master function with [MfuncParamId\(\)](#), the validity of that identifier will be checked during the execution of **MfuncCall()**. If the identifier is not valid, the slave function is not executed.

Parameter

FunctionId

Specifies the identifier of the user-defined MIL function.

The slave function executed using **MfuncCall()** must be declared as follows:

```
void MFTYPE SlaveFunction(  
    MIL_ID FunctionId  
)
```

Parameters:

FunctionId

Specifies the identifier of the user-defined MIL function that references this slave function. This is the identifier returned by the [MfuncAlloc\(\)](#) function.

Return value

The returned value is **M_NULL** if an error occurred; otherwise, not null.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncErrorReport

Synopsis

Report an error message.

Syntax

```
MIL_INT MfuncErrorReport (
    MIL_ID FunctionId,
    MIL_INT ErrorCode,
    MIL_CONST_TEXT_PTR ErrorMessage,
    MIL_CONST_TEXT_PTR ErrorSubMessage1,
    MIL_CONST_TEXT_PTR ErrorSubMessage2,
    MIL_CONST_TEXT_PTR ErrorSubMessage3
)
```

Description

This function allows you to log an error message in a user-defined MIL function using the MIL error handling mechanism. You can call the **MfuncErrorReport()** function from both the slave and the master functions of your user-defined MIL function, but only after a call to **MfuncAlloc()** and before a call to **MfuncFree()**. When **MfuncErrorReport()** is called, MIL will treat your error as a normal MIL error. To report the logged messages to screen, you must enable error reporting using the **MappControl()** function. These errors can also be read using the standard MIL error functions (**MappGetError()**), which is especially useful when you want to keep error reporting disabled.

If you report an error with an error code set to **M_NULL**, you will reset any pending internal error that a MIL function call might have generated inside the user-defined MIL function. You must reset the pending error if you don't want the MIL error message to be reported. If you don't reset the pending error at the end of the slave function, and you don't report your own error, MIL will detect any pending error and report the error message, prefixed with the name of the user-defined MIL function.

Parameters

FunctionId

Specifies the MIL function identifier. This parameter can be set to the following values:

● For the MIL function identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default MIL function identifier. Use this value when MfuncErrorReport() is called from within a slave function. (summarize)
<input type="checkbox"/> MIL function identifier	Specifies the identifier of a user-defined function.

ErrorCode

Assigns a numeric code to the user-defined MIL function's group of error messages. To specify the error code, combine the value in the table below with an offset. For example, **M_FUNC_ERROR** + 65.

● For assigning a numeric code	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FUNC_ERROR +	Specifies a numeric code to a group of error messages. By specifying this value, you ensure that the error messages of the user-defined MIL function do not conflict with MIL specific errors. (summarize)
<input type="checkbox"/> M_NULL +	Specifies that no error message will be returned. Any pending internal error generated by a MIL function call inside the user-defined MIL function will be reset. (summarize)

Combination constant for any of the possible values of the [ErrorCode](#) parameter

You can add the following value to the above-mentioned values to specify the offset.

● For specifying the offset	
☐ Value	Description
☐ 0 to 99	Specifies the value of the offset.

ErrorMessage

Specifies the text of your error message to report

● For specifying the text of the error message	
☐ Value	Description
☐ <code>MIL_TEXT(MIL_TEXT_PTR ErrorMessage)</code>	Specifies the text of your error message; this is a null terminated string. The error message, including the terminating null character, must not be longer than M_ERROR_MESSAGE_SIZE (128) characters. (summarize)
	Parameters
	ErrorMessage Specifies the text of your error message.

ErrorSubMessage1

Specifies the text of your first error sub-message to report.

● For specifying the text of the first error sub-message	
☐ Value	Description
☐ <code>MIL_TEXT(MIL_TEXT_PTR ErrorMessage)</code>	Specifies the text of your first error sub-message. The error sub-message must be a null terminated string. The error message, including the terminating null character, must not be longer than M_ERROR_MESSAGE_SIZE (128) characters. If you do not want to use this error sub-message, pass M_NULL . (summarize)
	Parameters
	ErrorMessage Specifies the text of your first error sub-message.

ErrorSubMessage2

Specifies the text of your second error sub-message to report.

● For specifying the text of the second error sub-message	
☐ Value	Description
☐ <code>MIL_TEXT(MIL_TEXT_PTR ErrorMessage)</code>	Specifies the text of your second error sub-message. For details on creating appropriate error sub-messages, see the description of the ErrorSubMessage1 parameter. (summarize)
	Parameters
	ErrorMessage Specifies the text of your second error sub-message.

ErrorSubMessage3

Specifies the text of your third error sub-message to report.

● For specifying the text of the third error sub-message		
❏ Value		Description
❏ MIL_TEXT(MIL_TEXT_PTR ErrorMessage)	Specifies the text of your third error sub-message. For details on creating appropriate error sub-messages, see the description of the ErrorSubMessage1 parameter. (summarize)	
		Parameters
	ErrorMessage	Specifies the text of your third error sub-message.

Return value

The returned value is **M_NULL** if an error occurred during the error log operation; otherwise, not null.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncFree

Synopsis

Free a MIL function context.

Syntax

```
void MfuncFree(
    MIL_ID FunctionId
)
```

Description

This function frees the function context allocated for a user-defined MIL function. **MfuncFree()** must be the last MIL function called in the master function. Note that you must call **MfuncFree()** from the same thread as [MfuncAlloc\(\)](#) and [MfuncCall\(\)](#).

Parameter

FunctionId

Specifies the identifier of the user-defined MIL function whose function context you want to free.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncFreeId

Synopsis

Free the MIL identifier associated with a user-defined MIL object.

Syntax

```
void MfuncFreeId(
    MIL_ID FunctionId,
    MIL_ID ObjectId
)
```

Description

This function frees a MIL identifier associated with a user-defined object using the [MfuncAllocId\(\)](#) function.

Parameters

FunctionId

Specifies a MIL function identifier. This parameter can be set to the following values:

For the MIL function identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default MIL function identifier. Use M_DEFAULT to free user-defined MIL objects created outside of any particular user-defined MIL function. (summarize)
<input type="checkbox"/> MIL function identifier	Specifies the identifier of a user-defined MIL function.

ObjectId

Specifies the MIL identifier of the user-defined MIL object to free.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncInquire

Synopsis

Retrieve information on a user-defined MIL object.

Syntax

```
MIL_INT MfuncInquire(  
    MIL_ID ObjectId,  
    MIL_INT InquireType,  
    void *UserVarPtr  
)
```

Description

This function retrieves information on the specified user-defined MIL object. This object must have been associated with a MIL identifier using the [MfuncAllocId\(\)](#) function.

Parameters

ObjectId

Specifies the identifier of the user-defined MIL object.

InquireType

Specifies the user-defined MIL object feature about which to inquire.

This parameter can be set to one of the following:

The following values require that you pass the value listed in the data-type area of the specified parameter.

For the user-defined MIL object feature	
Value	Description
<input type="checkbox"/> M_BUFFER_INFO	Returns the handle of the MIL buffer. (summarize) <div><i>UserVarPtr info</i> Data type: MIL_BUFFER_INFO</div>
<input type="checkbox"/> M_OBJECT_PTR	Returns the address of the user-defined MIL object. (summarize) <div><i>UserVarPtr info</i> Data type: address to a pointer</div>
<input type="checkbox"/> M_OBJECT_TYPE_EXTENDED	Returns the type of the specified MIL object. (summarize) <div><i>UserVarPtr info</i> Data type: MIL_INT64 Return values: <div><div>M_APPLICATION</div><div>MIL application.</div></div><div><div>M_ARRAY</div><div>MIL array.</div></div><div><div>M_BLOB_FEATURE_LIST</div><div>MIL blob feature list.</div></div></div>

M_BLOB_RESULT	MIL blob result.
M_CAL_PARENT	MIL calibration.
M_CODE_OBJECT	MIL code read.
M_COL_MATCH_CONTEXT	MIL Color context.
M_COL_MATCH_RESULT	MIL Color result.
M_COUNT_LIST	MIL count list.
M_DIGITIZER	MIL digitizer.
M_DISPLAY	MIL display.
M_3DMAP_LASER_CONTEXT	MIL 3D reconstruction laser context.
M_3DMAP_LASER_DATA	MIL 3D reconstruction laser result.
M_EDGE_CONTOUR	MIL edge contour.
M_EDGE_CREST	MIL edge crest.
M_EDGE_RESULT	MIL edge result.
M_EVENT_LIST	MIL event list.
M_EXTREME_LIST	MIL extreme list.
M_GRAPHIC_CONTEXT	MIL graphics context.
M_HIST_LIST	MIL history list.
M_IMAGE	MIL image.
M_KERNEL	MIL kernel.
M_LUT	MIL LUT.
M_MEAS_CONTEXT	MIL measurement context.
M_MEAS_MARKER	MIL measurement marker.
M_MEAS_RESULT	MIL measurement result.
M_MET_CONTEXT	MIL metrology context.
M_MET_RESULT	MIL metrology result.
M_MOD_GEOMETRIC	MIL Model Finder context.
M_MOD_GEOMETRIC_CONTROLLED	MIL Model Finder context.
M_MOD_RESULT	MIL Model Finder result.
M_OCR_FONT	MIL OCR font.
M_OCR_RESULT	MIL OCR result.
M_PAT_MODEL	MIL pattern matching model.
M_PAT_RESULT	MIL pattern matching result.
M_PROJ_LIST	MIL project list.
M_REG_CONTEXT	MIL registration context.
M_REG_RESULT	MIL registration result.
M_STRUCT_ELEMENT	MIL structural element.
M_STR_FEATURE_BASED_CONTEXT	MIL String Reader context.
M_STR_RESULT	MIL String Reader result.
M_SYSTEM	MIL system object.
M_USER_OBJECT_1	First user-defined object.
M_USER_OBJECT_2	Second user-defined object.

Accepts the address of one of the following (see above for specifics on which is expected):

- address to a pointer
- MIL_BUFFER_INFO
- MIL_INT64

Specifies the address in which to write the requested information.

Return value

Returns the setting of the object about which the inquiry was made.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamCheck

Synopsis

Verify whether parameter checking is required.

Syntax

```
MIL_INT MfuncParamCheck(  
    MIL_ID FunctionId  
)
```

Description

This function allows you to verify, from within your user-defined MIL function, whether parameter checking is enabled or disabled. To enable or disable the checking of parameters, use the [MappControl\(\)](#) [M_PARAMETER](#) control type.

Call the **MfuncParamCheck()** function prior to checking the parameters of the specified user-defined MIL function. The return value of the **MfuncParamCheck()** function should dictate whether to execute the functions that perform the parameter checking, or not. Then, to save the parameter checking time for a time-critical user-defined MIL function, it is sufficient to disable parameter checking using [MappControl\(\)](#) with [M_CHECK_DISABLE](#).

Parameter

FunctionId

Specifies the identifier of the user-defined MIL function.

Return value

The returned value is **M_NULL** if no parameter checking is required; otherwise, checking is required.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamDouble

Synopsis

Register a parameter of type *double*.

Syntax

```
void MfuncParamDouble(  
    MIL_ID FunctionId,  
    MIL_INT ParamIndex,  
    MIL_DOUBLE ParamValue  
)
```

Description

This function allows you to register a type *double* parameter of the current user-defined MIL function. The **MfuncParamDouble()** function should be called in the master function after a call to [MfuncAlloc\(\)](#) and before a call to [MfuncCall\(\)](#).

Parameters

- FunctionId
- Specifies the identifier of the user-defined MIL function.
- ParamIndex
- Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.
- ParamValue
- Specifies the value of the type *double* parameter.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamId

Synopsis

Register a MIL_ID parameter.

Syntax

```
void MfuncParamId(
    MIL_ID FunctionId,
    MIL_INT ParamIndex,
    MIL_ID ParamValue,
    MIL_UINT64 ParamIs,
    MIL_INT Attribute
)
```

Description

This function allows you to register a MIL_ID parameter of the specified user-defined MIL function. The **MfuncParamId()** function should be called in the master function after a call to [MfuncAlloc\(\)](#) and before a call to [MfuncCall\(\)](#).

Parameters

FunctionId

Specifies the identifier of the user-defined MIL function.

ParamIndex

Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.

ParamValue

Specifies the MIL_ID of the MIL object passed to the MIL_ID parameter.

ParamIs

Specifies the type of the MIL object. The available object types are divided into two groups. When the specified object is of a type listed in the first group, you can specify a single or multiple object type(s) from this group. For example, if the object is expected to be of type [M_ARRAY](#) or [M_COUNT_LIST](#), set the [ParamIs](#) parameter to [M_ARRAY](#) + [M_COUNT_LIST](#). The object types of the second group cannot be used in combination with each other.

The first group of object types includes the following:

● For specifying the type of MIL object (one or multiple)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_APPLICATION	Specifies an application object.
<input type="checkbox"/> M_ARRAY	Specifies an array object.
<input type="checkbox"/> M_COUNT_LIST	Specifies a count list object.
<input type="checkbox"/> M_DIGITIZER	Specifies a digitizer object.
<input type="checkbox"/> M_DISPLAY	Specifies a display object.
<input type="checkbox"/> M_EVENT	Specifies an event object.
<input type="checkbox"/> M_EVENT_LIST	Specifies an event list object.
<input type="checkbox"/> M_EXTREME_LIST	Specifies an extreme list object.

<input type="checkbox"/> M_GRAPHIC_CONTEXT	Specifies a graphics context object.
<input type="checkbox"/> M_HIST_LIST	Specifies a histogram list object.
<input type="checkbox"/> M_IMAGE	Specifies an image object.
<input type="checkbox"/> M_KERNEL	Specifies a kernel object.
<input type="checkbox"/> M_LUT	Specifies a LUT object.
<input type="checkbox"/> M_PROJ_LIST	Specifies a project list object.
<input type="checkbox"/> M_STRUCT_ELEMENT	Specifies a structuring element object.
<input type="checkbox"/> M_SYSTEM	Specifies a system object.

The second group of object types includes the following:

● For specifying the second group of MIL object types (only one)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_3DMAP_LASER_CONTEXT	Specifies a 3D reconstruction context object.
<input type="checkbox"/> M_3DMAP_LASER_DATA	Specifies a 3D reconstruction result object.
<input type="checkbox"/> M_BLOB_FEATURE_LIST	Specifies a blob feature list object.
<input type="checkbox"/> M_BLOB_RESULT	Specifies a blob result object.
<input type="checkbox"/> M_CAL_PARENT	Specifies a calibration object.
<input type="checkbox"/> M_CODE_OBJECT	Specifies a code object.
<input type="checkbox"/> M_COL_MATCH_CONTEXT	Specifies a color analysis context object (for matching).
<input type="checkbox"/> M_COL_MATCH_RESULT	Specifies a color analysis result object (for matching).
<input type="checkbox"/> M_EDGE_CONTOUR	Specifies an edge contour object.
<input type="checkbox"/> M_EDGE_CREST	Specifies an edge crest object.
<input type="checkbox"/> M_EDGE_RESULT	Specifies an edge result object.
<input type="checkbox"/> M_MEAS_CONTEXT	Specifies a measurement context object.
<input type="checkbox"/> M_MEAS_MARKER	Specifies a measurement marker object.
<input type="checkbox"/> M_MEAS_RESULT	Specifies a measurement result object.
<input type="checkbox"/> M_MOD_GEOMETRIC	Specifies a Model Finder context object.
<input type="checkbox"/> M_MOD_RESULT	Specifies a Model Finder result object.
<input type="checkbox"/> M_OCR_FONT	Specifies a character recognition font object.
<input type="checkbox"/> M_OCR_RESULT	Specifies a character recognition result object.
<input type="checkbox"/> M_PAT_MODEL	Specifies a pattern matching model object.
<input type="checkbox"/> M_PAT_RESULT	Specifies a pattern matching result object.
<input type="checkbox"/> M_USER_OBJECT_1	Specifies an object from group one of the user-defined object types (MfuncAllocId()).
<input type="checkbox"/> M_USER_OBJECT_2	Specifies an object from group two of the user-defined object types (MfuncAllocId()).

Attribute

Specifies whether the MIL object will be used for input or output.

This parameter should be set to one or both ([M_IN](#) + [M_OUT](#)) of the following values:

● For specifying the type of attributes

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN +	Specifies that the MIL_ID passed to ParamValue must be that of an input object.
<input type="checkbox"/> M_OUT +	Specifies that the MIL_ID passed to ParamValue must be that of an output object.

Combination constants for any of the possible values of the [Attribute](#) parameter

You can add one or more of the following values to the above-mentioned values to specify the buffer type, if [ParamIs](#) is set to [M_IMAGE](#).

If you specify an [M_IMAGE](#) object ([ParamValue](#)) that does not correspond to any of the type(s) selected from the values listed below, an error is returned. If none of the values specified below are added, any [M_IMAGE](#) object will be accepted. Note that you cannot specify any of the following if the [ParamIs](#) parameter is set to a combination of values.

● For M_IN and M_OUT	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DISP	Specifies that the buffer can be displayed.
<input type="checkbox"/> M_GRAB	Specifies that the buffer can have data grabbed into it.
<input type="checkbox"/> M_PROC	Specifies that the buffer can be processed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamIdPointer

Synopsis

Register a MIL_ID pointer parameter.

Syntax

```
void MfuncParamIdPointer(
    MIL_ID FunctionId,
    MIL_INT ParamIndex,
    MIL_ID *ParamValue,
    MIL_UINT64 ParamIs,
    MIL_INT Attribute
)
```

Description

This function allows you to register a parameter that accepts a pointer to a MIL_ID object, as a parameter of the specified user-defined MIL function. The **MfuncParamIdPointer()** function should be called in the master function after a call to **MfuncAlloc()** and before a call to **MfuncCall()**.

Parameters

- FunctionId**
- Specifies the identifier of the user-defined MIL function.
- ParamIndex**
- Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.
- ParamValue**
- Specifies the address of the MIL_ID object passed to the MIL_ID pointer parameter.
- ParamIs**

Specifies the type of the MIL_ID object passed to the MIL_ID pointer parameter. The available object types are divided into two groups. When the specified object is of a type listed in the first group, you can specify a single or multiple object type(s) from this group. For example, if the object is expected to be of type **M_ARRAY** or **M_COUNT_LIST**, set the **ParamIs** parameter to **M_ARRAY + M_COUNT_LIST**. The object types of the second group cannot be used in combination with each other.

The first group of object types includes the following:

● For specifying the type of the MIL_ID object (one or multiple)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_APPLICATION	Specifies an application object.
<input type="checkbox"/> M_ARRAY	Specifies an array object.
<input type="checkbox"/> M_COUNT_LIST	Specifies a count list object.
<input type="checkbox"/> M_DIGITIZER	Specifies a digitizer object.
<input type="checkbox"/> M_DISPLAY	Specifies a display object.
<input type="checkbox"/> M_EVENT_LIST	Specifies an event list object.
<input type="checkbox"/> M_EXTREME_LIST	Specifies an extreme list object.

<input type="checkbox"/> M_GRAPHIC_CONTEXT	Specifies a graphics context object.
<input type="checkbox"/> M_HIST_LIST	Specifies a histogram list object.
<input type="checkbox"/> M_IMAGE	Specifies an image object.
<input type="checkbox"/> M_KERNEL	Specifies a kernel object.
<input type="checkbox"/> M_LUT	Specifies a LUT object.
<input type="checkbox"/> M_PROJ_LIST	Specifies a project list object.
<input type="checkbox"/> M_STRUCT_ELEMENT	Specifies a structuring element object.
<input type="checkbox"/> M_SYSTEM	Specifies a system object.

The second group of object types includes the following:

● For specifying the second group of MIL_ID object types (only one)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_3DMAP_LASER_CONTEXT	Specifies a 3D reconstruction context object.
<input type="checkbox"/> M_3DMAP_LASER_DATA	Specifies a 3D reconstruction result object.
<input type="checkbox"/> M_BLOB_FEATURE_LIST	Specifies a blob feature list object.
<input type="checkbox"/> M_BLOB_RESULT	Specifies a blob result object.
<input type="checkbox"/> M_CAL_PARENT	Specifies a calibration object.
<input type="checkbox"/> M_CODE_OBJECT	Specifies a code object.
<input type="checkbox"/> M_COL_MATCH_CONTEXT	Specifies a color analysis context object (for matching).
<input type="checkbox"/> M_COL_MATCH_RESULT	Specifies a color analysis result object (for matching).
<input type="checkbox"/> M_EDGE_CONTOUR	Specifies an edge contour object.
<input type="checkbox"/> M_EDGE_CREST	Specifies an edge crest object.
<input type="checkbox"/> M_EDGE_RESULT	Specifies an edge result object.
<input type="checkbox"/> M_MEAS_CONTEXT	Specifies a measurement context object.
<input type="checkbox"/> M_MEAS_MARKER	Specifies a measurement marker object.
<input type="checkbox"/> M_MEAS_RESULT	Specifies a measurement result object.
<input type="checkbox"/> M_MOD_GEOMETRIC	Specifies a Model Finder context object.
<input type="checkbox"/> M_MOD_RESULT	Specifies a Model Finder result object.
<input type="checkbox"/> M_OCR_FONT	Specifies a character recognition font object.
<input type="checkbox"/> M_OCR_RESULT	Specifies a character recognition result object.
<input type="checkbox"/> M_PAT_MODEL	Specifies a pattern matching model object.
<input type="checkbox"/> M_PAT_RESULT	Specifies a pattern matching result object.
<input type="checkbox"/> M_USER_OBJECT_1	Specifies an object from group one of the user-defined object types (MfuncAllocId()).
<input type="checkbox"/> M_USER_OBJECT_2	Specifies an object from group two of the user-defined object types (MfuncAllocId()).

Attribute

Specifies whether the pointer will be used for input or output.

This parameter should be set to one or both ([M_IN](#) + [M_OUT](#)) of the following values:

● For specifying the attribute of the pointer parameter

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN	Specifies that the pointer parameter is pointing to a MIL_ID object that can be used for input.
<input type="checkbox"/> M_OUT	Specifies that the pointer parameter is pointing to a MIL_ID object that can be used for output.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamLong

Synopsis

Register a parameter of type *long*.

Syntax

```
void MfuncParamLong(
    MIL_ID FunctionId,
    MIL_INT ParamIndex,
    MIL_INT ParamValue
)
```

Description

This function allows you to register a type *long* parameter of the current user-defined MIL function. The **MfuncParamLong()** function should be called in the master function after a call to [MfuncAlloc\(\)](#) and before a call to [MfuncCall\(\)](#).

Parameters

- FunctionId
- Specifies the identifier of the user-defined MIL function.
- ParamIndex
- Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.
- ParamValue
- Specifies the value of the type *long* parameter.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamMilDouble

Synopsis

Register a parameter of type *MIL_DOUBLE*.

Syntax

```
void MfuncParamMilDouble(  
    MIL_ID FunctionId,  
    MIL_INT ParamIndex,  
    MIL_DOUBLE ParamValue  
)
```

Description

This function allows you to register a type *MIL_DOUBLE* parameter of the current user-defined MIL function. The **MfuncParamMilDouble()** function should be called in the master function after a call to [MfuncAlloc\(\)](#) and before a call to [MfuncCall\(\)](#).

Parameters

- FunctionId
- Specifies the identifier of the user-defined MIL function.
- ParamIndex
- Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.
- ParamValue
- Specifies the value of the type *MIL_DOUBLE* parameter.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamMilInt32

Synopsis

Register a parameter of type MIL_INT32.

Syntax

```
void MfuncParamMilInt32(  
    MIL_ID FunctionId,  
    MIL_INT ParamIndex,  
    MIL_INT32 ParamValue  
)
```

Description

This function allows you to register a type *MIL_INT32* parameter of the current user-defined MIL function. The **MfuncParamMilInt32()** function should be called in the master function after a call to [MfuncAlloc\(\)](#) and before a call to [MfuncCall\(\)](#).

Parameters

- FunctionId
- Specifies the identifier of the user-defined MIL function.
- ParamIndex
- Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.
- ParamValue
- Specifies the value of the type *MIL_INT32* parameter.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamMilInt64

Synopsis

Register a parameter of type MIL_INT64.

Syntax

```
void MfuncParamMilInt64(
    MIL_ID FunctionId,
    MIL_INT ParamIndex,
    MIL_INT64 ParamValue
)
```

Description

This function allows you to register a type *MIL_INT64* parameter of the current user-defined MIL function. The **MfuncParamMilInt64()** function should be called in the master function after a call to [MfuncAlloc\(\)](#) and before a call to [MfuncCall\(\)](#).

Parameters

- FunctionId
- Specifies the identifier of the user-defined MIL function.
- ParamIndex
- Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.
- ParamValue
- Specifies the value of the type *MIL_INT64* parameter.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamMilInt

Synopsis

Register a parameter of type MIL_INT.

Syntax

```
void MfuncParamMilInt(  
    MIL_ID FunctionId,  
    MIL_INT ParamIndex,  
    MIL_INT ParamValue  
)
```

Description

This function allows you to register a type *MIL_INT* parameter of the current user-defined MIL function. The **MfuncParamMilInt()** function should be called in the master function after a call to **MfuncAlloc()** and before a call to **MfuncCall()**.

Parameters

- FunctionId
- Specifies the identifier of the user-defined MIL function.
- ParamIndex
- Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.
- ParamValue
- Specifies the value of the type *MIL_INT* parameter.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamPointer

Synopsis

Register a pointer parameter.

Syntax

```
void MfuncParamPointer(
    MIL_ID FunctionId,
    MIL_INT ParamIndex,
    void *ParamValue,
    MIL_INT Size,
    MIL_INT Attribute
)
```

Description

This function allows you to register a pointer parameter of the current user-defined MIL function. The **MfuncParamPointer()** function should be called in the master function after a call to **MfuncAlloc()** and before a call to **MfuncCall()**.

Note that to register a pointer parameter that specifies the address of a MIL_ID object, you must use the **MfuncParamIdPointer()** function.

Parameters

FunctionId

Specifies the identifier of the user-defined MIL function.

ParamIndex

Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.

ParamValue

Specifies the address passed to the pointer parameter. Note that you should not use this parameter to point to data types that can be handled using the other provided **MfuncParam...()** functions.

Size

Specifies the size of the data, in bytes, pointed to by the **ParamValue** parameter.

Attribute

Specifies whether the pointer will be used for input or output.

This parameter should be set to one or both (**M_IN** + **M_OUT**) of the following values:

For specifying the attribute of the pointer parameter	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN	Specifies that the parameter is pointing to a variable or a data structure that will be used for input.
<input type="checkbox"/> M_OUT	Specifies that the parameter is pointing to a variable or a data structure that will be used for output.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.

DLL	Requires mil.dll.
-----	-------------------

MfuncParamString

Synopsis

Register a null-terminated string parameter.

Syntax

```
void MfuncParamString(
    MIL_ID FunctionId,
    MIL_INT ParamIndex,
    MIL_CONST_TEXT_PTR ParamValue,
    MIL_INT Size,
    MIL_INT Attribute
)
```

Description

This function allows you to register a null-terminated string parameter of the current user-defined MIL function. The **MfuncParamString()** function should be called in the master function after a call to **MfuncAlloc()** and before a call to **MfuncCall()**.

Parameters

- FunctionId
- Specifies the identifier of the user-defined MIL function.
- ParamIndex
- Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.
- ParamValue
- Specifies the string to be registered.

● For specifying the string		
☐ Value	Description	
☐ MIL_TEXT(MIL_TEXT_PTR ParamValue)	Specifies the address passed to the string parameter. (summarize)	
	Parameters	
	ParamValue	Specifies the address of the string.

Size

Specifies the size of the data pointed to by the **ParamValue** parameter. When the string parameter of the user-defined function is used for output (**M_OUT**), you must first inquire about the maximum size of the string that can be stored at the specified memory location, and then set the **Size** parameter to this value.

● For specifying the size of the data	
☐ Value	Description
☐ M_DEFAULT	Calculates the size automatically. You can use M_DEFAULT only when the Attribute parameter is set to M_IN .

	(summarize)
<input type="checkbox"/> Value	Specifies the size of the data, in bytes.

Attribute

Specifies whether the string will be used for input or output.

This parameter should be set to one or both ([M_IN](#) + [M_OUT](#)) of the following values:

● For specifying the attribute of the string parameter	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN	Specifies that the string parameter will be used for input.
<input type="checkbox"/> M_OUT	Specifies that the string parameter will be used for output.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MfuncParamValue

Synopsis

Read the value of the specified MIL function parameter.

Syntax

```
void MFTYPE MfuncParamValue(  
    MIL_ID FunctionId,  
    MIL_INT ParamIndex,  
    void *ParamValuePtr  
)
```

Description

This function allows you to read the value of a parameter registered in the master function of your user-defined MIL function, from the slave function. The requested value is written to the address specified by the [ParamValuePtr](#) parameter.

Parameters

FunctionId

Specifies the identifier of the user-defined MIL function.

ParamIndex

Specifies the index of the parameter within the user-defined MIL function's parameter list. The index of the first parameter is considered to be one.

ParamValuePtr

Specifies the address of the variable in which to write the value of the specified parameter. The type of the variable depends on the type of the parameter registered in the master function.

Return value

Returns the value of the specified parameter.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

Mgen functions

Synopsis

The functions prefixed with Mgen make up the Data Generation module. The basic Data Generation module provides a limited set of data generation tools that can be used to automatically generate predefined data in a data buffer (for example, generating ramp in a LUT buffer).

Functions

[MgenLutFunction](#)

[MgenLutRamp](#)

[MgenWarpParameter](#)

MgenLutFunction

Synopsis

Generate data into a LUT buffer using a specified standard mathematical function.

Syntax

```
void MgenLutFunction(
    MIL_ID LutBufId,
    MIL_INT Func,
    MIL_DOUBLE a,
    MIL_DOUBLE b,
    MIL_DOUBLE c,
    MIL_INT StartIndex,
    MIL_DOUBLE StartXValue,
    MIL_INT EndIndex
)
```

Description

This function generates a value for each LUT index within the specified index range (StartIndex to EndIndex inclusive), according to the specified mathematical function. Each function takes a value X. The StartXValue parameter specifies the initial X value. The remaining entries of the index range are generated by incrementing the value of X by 1 for each index.

Parameters

- LutBufId
- Specifies the identifier of the LUT in which to generate values. This parameter must be given a valid LUT buffer identifier. Allocate a LUT buffer, using MbufAlloc1d() or MbufAllocColor(). If the LUT is a multi-band LUT (allocated with MbufAllocColor()), the same data is written to all bands.
- Func
- Specifies the mathematical function to use for calculations. For M_SIN, M_COS, and M_TAN, X is considered to be in degrees. All results are converted to integer by truncation, except when using a floating-point LUT buffer. Note, if the given parameters cause an overflow or underflow, indeterminate results will be written in the destination LUT.

This parameter can be set to one of the following:

● For specifying the mathematical function	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_COS	Uses the following formula for calculation: $a \cos (b\ x) + c$.
<input type="checkbox"/> M_EXP	Uses the following formula for calculation: $a\ b^x + c$.
<input type="checkbox"/> M_LOG	Uses the following formula for calculation: $a \log_b (x) + c$.
<input type="checkbox"/> M_QUAD	Uses the following formula for calculation: $a\ x^2 + b\ x + c$.
<input type="checkbox"/> M_SIN	Uses the following formula for calculation: $a \sin (b\ x) + c$.
<input type="checkbox"/> M_TAN	Uses the following formula for calculation: $a \tan (b\ x) + c$.

- a
- Specifies a function constant for the Func parameter.
- b

Specifies a function constant for the [Func](#) parameter.

c

Specifies a function constant for the [Func](#) parameter.

StartIndex

Specifies the first LUT index entry for which to generate values.

StartXValue

Specifies the initial value of X in the function.

EndIndex

Specifies the last LUT index entry for which to generate values. This index entry must be greater than or equal to the [StartIndex](#) value.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgenLutRamp

Synopsis

Generate ramp data into a LUT buffer.

Syntax

```
void MgenLutRamp(
    MIL_ID LutId,
    MIL_INT StartIndex,
    MIL_DOUBLE StartValue,
    MIL_INT EndIndex,
    MIL_DOUBLE EndValue
)
```

Description

This function generates a ramp, inverse ramp, or a constant in the specified LUT buffer region (StartIndex to EndIndex). The increment between LUT entries is the difference between StartValue and EndValue, divided by the number of entries.

If you need to generate a more complex LUT, use MgenLutFunction() or generate the values with your Host system and load them into a MIL LUT buffer, using MbufPut1d() or MbufPutColor().

Parameters

- LutId
- Specifies the identifier of the LUT in which to generate values. This parameter must be given a valid LUT buffer identifier. Allocate a LUT buffer, using MbufAlloc1d() or MbufAllocColor().
- StartIndex
- Specifies the first LUT index entry for which to generate values. StartIndex must be less than or equal to EndIndex.
- StartValue
- Specifies the extreme value from which the increment is calculated; that is, the first LUT entry.
- EndIndex
- Specifies the last LUT index entry for which to generate values. EndIndex must be greater than or equal to StartIndex.
- EndValue
- Specifies the extreme value from which the increment is calculated; that is, the last LUT entry. If both StartValue and EndValue are the same, the entire LUT range is filled with this value. If EndValue is smaller than StartValue, an inverse ramp is generated. These parameters accept only integer values, except when using a floating-point LUT buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgenWarpParameter

Synopsis

Generate coefficients or LUTs used to warp an image.

Syntax

```
void MgenWarpParameter(
    MIL_ID InWarpParameter,
    MIL_ID OutXLutOrCoefId,
    MIL_ID OutYLutId,
    MIL_INT OperationMode,
    MIL_INT Transform,
    MIL_DOUBLE Val1,
    MIL_DOUBLE Val2
)
```

Description

This function generates coefficients or LUTs for use with [MimWarp\(\)](#). Specifically, this function can generate:

- Coefficients for a first-order polynomial warping ([M_WARP_POLYNOMIAL](#)).
- Coefficients for a perspective polynomial warping that maps an arbitrary quadrilateral onto a rectangle ([M_WARP_4_CORNER](#)) or that maps a rectangle onto an arbitrary quadrilateral ([M_WARP_4_CORNER_REVERSE](#)).
- LUTs for a first-order polynomial warping or for a perspective polynomial warping ([M_WARP_LUT](#)).

For a first-order polynomial warping, coefficients can be generated for a rotation, a scaling, a shearing, or a translation. To combine coefficients (for example, to generate coefficients for a rotation and translation), you need to make separate calls to this function, using the previous output buffer as your current input buffer. After all the coefficients are generated, pass the coefficient buffer to [MimWarp\(\)](#).

For a perspective polynomial warping, coefficients can be generated to map an arbitrary quadrilateral onto a rectangle or vice versa. After all the coefficients are generated, pass the coefficient buffer to [MimWarp\(\)](#).

Note that you can also perform a first-order polynomial warping or perspective polynomial warping using LUTs. To generate the required LUTs, use LUT buffers for your output rather than a coefficient buffer while generating the last set of coefficients. In this case, the coefficients will not be saved. If you need to save the coefficients, use [MgenWarpParameter\(\)](#) in [M_WARP_LUT](#) mode after you have generated all coefficients. After the LUTs are generated, pass the LUT buffers to [MimWarp\(\)](#).

First-order polynomial warpings and perspective polynomial warpings are performed by associating each pixel position of the destination buffer, (x_d, y_d) , with a specific point in the source buffer, (x_s, y_s) , according to the following equation:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ c_0 & c_1 & c_2 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix}$$

Where x_s and y_s are defined as follows:

$$x_s = \frac{x}{w} = \frac{a_0 x_d + a_1 y_d + a_2}{c_0 x_d + c_1 y_d + c_2}$$

$$y_s = \frac{y}{w} = \frac{b_0 x_d + b_1 y_d + b_2}{c_0 x_d + c_1 y_d + c_2}$$

Note that in the case of a first-order polynomial warping, the c_0, c_1, c_2 coefficients are 0, 0, and 1, respectively. The equation then reduces to:

$$x_s = a_0 x_d + a_1 y_d + a_2$$

$$y_s = b_0 x_d + b_1 y_d + b_2$$

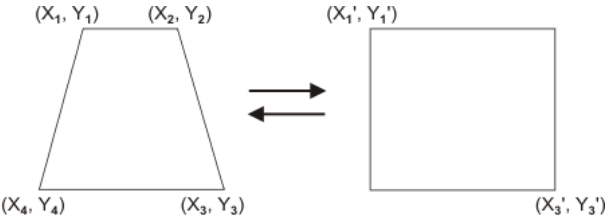
Parameters

InWarpParameter

Specifies the input buffer from which to generate coefficients or LUT entries.

When generating coefficients for a first-order polynomial warping, set [InWarpParameter](#) to `M_NULL` if this is your first call to `MgenWarpParameter()`. If you are combining coefficients and this is a second (or later) call, set [InWarpParameter](#) to the identifier of the previous output buffer.

When generating coefficients for perspective warpings that map an arbitrary quadrilateral onto a rectangle or that map a rectangle onto an arbitrary quadrilateral, [InWarpParameter](#) must be set to the identifier of a one-dimensional floating-point buffer with an `M_ARRAY` attribute.



The buffer must contain 12 entries; each entry corresponds to one of the following points.

Entry and its corresponding point	Entry and its corresponding point
entry [0] = X1	entry [6] = X4
entry [1] = Y1	entry [7] = Y4
entry [2] = X2	entry [8] = X1'
entry [3] = Y2	entry [9] = Y1'
entry [4] = X3	entry [10] = X3'
entry [5] = Y3	entry [11] = Y3'

When generating LUTs for a first-order polynomial warping or perspective polynomial warping, [InWarpParameter](#) must be set to the identifier of a 3x3 floating-point buffer that has an `M_ARRAY` attribute. The first row specifies the a_n coefficients, the second row specifies the b_n coefficients, and the third row specifies the c_n coefficients.

OutXLutOrCoefId

Specifies the buffer in which to place generated coefficients or X-LUT entries.

For coefficients (either those for a first-order polynomial warping or for a perspective polynomial warping), the buffer must be a 3x3 floating-point buffer with an `M_ARRAY` attribute.

For X-LUT entries, the buffer must be signed 16- or 32-bit integer, have the same X and Y size as the destination buffer that you will eventually pass to `MimWarp()`, and have an `M_LUT` attribute.

OutYLutId

Specifies the buffer in which to place Y-LUT entries. This buffer must be signed 16- or 32-bit integer, have the same X and Y size as the destination buffer that you will eventually pass to `MimWarp()`, and have an `M_LUT` attribute.

If you are not generating LUTs, set the [OutYLutId](#) parameter to `M_NULL`.

OperationMode

Specifies the mode of operation. It can be set to the following:

● For specifying the mode of operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_WARP_4_CORNER	Generates coefficients for a perspective polynomial warping that maps an arbitrary quadrilateral onto a rectangle.

<input type="checkbox"/> M_WARP_4_CORNER_REVERSE	Generates coefficients for a perspective polynomial warping that maps a rectangle onto an arbitrary quadrilateral.
<input type="checkbox"/> M_WARP_LUT +	Generates LUTs for a first-order polynomial warping or a perspective polynomial warping.
<input type="checkbox"/> M_WARP_POLYNOMIAL	Generates coefficients for a first-order polynomial warping.

Combination constant for [M_WARP_LUT](#);

You can add the following value to the above-mentioned value to specify the number of fractional bits for the source address.

● For the output LUT buffers	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FIXED_POINT+n	Specifies the number of fractional bits for the source address (x_s , y_s). The default value is 0. (summarize)

Transform

Specifies the type of first-order polynomial warping for which to generate coefficients. If you are not generating coefficients for a first-order polynomial warping, set the this parameter to **M_DEFAULT**.

● For specifying the type of first-order polynomial warping	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ROTATE	Generates coefficients for a counter-clockwise rotation around (0,0) by Val1 °.
<input type="checkbox"/> M_SCALE	Generates coefficients for an image scaling, by a factor of Val1 in the X-direction and by a factor of Val2 in the Y-direction.
<input type="checkbox"/> M_SHEAR_X	Generates coefficients for a shearing in the X-direction, by a factor of Val1 .
<input type="checkbox"/> M_SHEAR_Y	Generates coefficients for a shearing in the Y-direction, by a factor of Val1 .
<input type="checkbox"/> M_TRANSLATE	Generates coefficients for a translation by Val1 pixels in the X-direction and by Val2 pixels in the Y-direction.

Val1

Specifies the first transform constant. If this parameter is not being used, set it to **M_NULL**.

Val2

Specifies the second transform constant. If this parameter is not being used, set it to **M_NULL**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

Mgra functions

Synopsis

The functions prefixed with Mgra make up the Graphics module. The basic Graphics module provides a limited set of graphic primitives that can be used to create drawings and text annotations in an image.

Functions

- MgraAlloc
- MgraArc
- MgraArcFill
- MgraBackColor
- MgraClear
- MgraColor
- MgraControl
- MgraDot
- MgraDots
- MgraFill
- MgraFont
- MgraFontScale
- MgraFree
- MgraInquire
- MgraLine
- MgraLines
- MgraRect
- MgraRectFill
- MgraText

MgraAlloc

Synopsis

Allocate a graphics context.

Syntax

```
MIL_ID MgraAlloc(  
    MIL_ID SystemId,  
    MIL_ID *GraphContIdPtr  
)
```

Description

This function allocates a graphics context, which specifies drawing and text parameters for use in subsequent MIL graphic functions.

Upon allocation of a graphics context, the drawing and text parameters are set to the following default values:

Name	Value
Foreground color	0xFFFFFFFF
Background color	0x00000000
Font	M_FONT_DEFAULT_SMALL
Font scale	X = 1.0, Y = 1.0

You can modify these values, using [MgraColor\(\)](#), [MgraBackColor\(\)](#), [MgraFont\(\)](#), and [MgraFontScale\(\)](#), or inquire about the current values, using [MgraInquire\(\)](#).

You can set the attributes of the graphics context (for example, background transparency), using [MgraControl\(\)](#).

When a graphics context is no longer required, release it, using [MgraFree\(\)](#).

Note, upon allocation of an application, a default graphics context is automatically allocated. Rather than using **MgraAlloc()** to allocate a graphics context, you can use this default graphics context, by specifying **M_DEFAULT** wherever a graphics context identifier is required.

Parameters

SystemId

Specifies the system on which to allocate the graphics context.

This parameter should be set to one of the following values:

For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

GraphContIdPtr

Specifies the address of the variable in which the graphics context identifier is to be written. Since the **MgraAlloc()** function also returns the buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the graphics context identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraArc

Synopsis

Draw an arc.

Syntax

```
void MgraArc(
    MIL_ID GraphContId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE XCenter,
    MIL_DOUBLE YCenter,
    MIL_DOUBLE XRad,
    MIL_DOUBLE YRad,
    MIL_DOUBLE StartAngle,
    MIL_DOUBLE EndAngle
)
```

Description

This function draws an elliptic arc based on an ellipse centered at (XCenter, YCenter) with radii XRad and YRad. The arc is defined by the start angle StartAngle and the end angle EndAngle. The arc is drawn with the foreground color of the specified graphics context.

This function uses the image coordinate system (pixel coordinates). To draw in a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using McaTransformCoordinate() or McaTransformCoordinateList().

If part of the arc falls outside of the specified target buffer, that part is clipped off.

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc().

DestImageBufId

Specifies the identifier of the image buffer in which to draw.

XCenter

Specifies the X-coordinate of the ellipse center, relative to the top-left corner of the specified target buffer.

YCenter

Specifies the Y-coordinate of the ellipse center, relative to the top-left corner of the specified target buffer.

XRad

Specifies the radius of the ellipse along the X-axis. The radius should be given in pixels and must be greater than 0.

YRad

Specifies the radius of the ellipse along the Y-axis. The radius should be given in pixels and must be greater than 0.

StartAngle

Specifies the angle at which to start drawing the arc, moving in a counter-clockwise direction. Express the angle in degrees relative to the positive X-axis.

EndAngle

Specifies the angle at which to end drawing the arc, moving in a counter-clockwise direction. Express the angle in degrees relative to the positive X-axis.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraArcFill

Synopsis

Draw a filled elliptic arc.

Syntax

```
void MgraArcFill(
    MIL_ID GraphContId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE XCenter,
    MIL_DOUBLE YCenter,
    MIL_DOUBLE XRad,
    MIL_DOUBLE YRad,
    MIL_DOUBLE StartAngle,
    MIL_DOUBLE EndAngle
)
```

Description

This function draws a filled elliptic arc based on an ellipse centered at (XCenter, YCenter) with radii XRad and YRad. The arc is defined by the start angle StartAngle and end angle EndAngle. The arc is drawn and filled with the foreground color of the specified graphics context.

This function uses the image coordinate system (pixel coordinates). To draw in a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using McaTransformCoordinate() or McaTransformCoordinateList().

If part of the arc falls outside of the specified target buffer, that part is clipped off.

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc().

DestImageBufId

Specifies the identifier of the image buffer in which to draw.

XCenter

Specifies the X-coordinate of the ellipse center, relative to the top-left corner of the specified target buffer.

YCenter

Specifies the Y-coordinate of the ellipse center, relative to the top-left corner of the specified target buffer.

XRad

Specifies the radius of the ellipse along the X-axis. The radius should be given in pixels and must be greater than 0.

YRad

Specifies the radius of the ellipse along the Y-axis. The radius should be given in pixels and must be greater than 0.

StartAngle

Specifies the angle at which to start drawing the arc, moving in a counter-clockwise direction. Express the angle in degrees relative to the positive X-axis.

EndAngle

Specifies the angle at which to end drawing the arc, moving in a counter-clockwise direction. Express the angle in degrees relative to the positive X-axis.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraBackColor

Synopsis

Sets the background color of a graphics context.

Syntax

```
void MgraBackColor(  
    MIL_ID GraphContId,  
    MIL_DOUBLE BackgroundColor  
)
```

Description

This function sets the background color of a specified graphics context.

Parameters

GraphContId

Specifies the graphics context with which to associate the background color. This parameter must be set to one of the following values:

For specifying the graphics context	
Value	Description
M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

BackgroundColor

Sets the background color of a specified graphics context. This parameter must be set to one of the following values:

For the background color									
Value	Description								
M_RGB888(MIL_INT Red, MIL_INT Green, MIL_INT Blue)	<div>Specifies an RGB value when using the graphics context to draw in an 8-bit, 3-band buffer. The red, green, and blue values must be values between 0 and 255, inclusive.</div> <div>When drawing in a 16-bit or 32-bit multi-band buffer, the components of the RGB value are cast to the type of the destination buffer's bands.</div> <div>To specify a 16-bit or 32-bit color component, use MgraControl() with M_BACKCOLOR combined with the appropriate constant (M_RED, M_GREEN, or M_BLUE). (summarize)</div> <table><tr><td colspan="2">Parameters</td></tr><tr><td>Red</td><td>Specifies the red component, as a value between 0 and 255.</td></tr><tr><td>Green</td><td>Specifies the green component, as a value between 0 and 255.</td></tr><tr><td>Blue</td><td>Specifies the blue component, as a value between 0 and 255.</td></tr></table>	Parameters		Red	Specifies the red component, as a value between 0 and 255.	Green	Specifies the green component, as a value between 0 and 255.	Blue	Specifies the blue component, as a value between 0 and 255.
Parameters									
Red	Specifies the red component, as a value between 0 and 255.								
Green	Specifies the green component, as a value between 0 and 255.								
Blue	Specifies the blue component, as a value between 0 and 255.								
M_COLOR_BLACK	Specifies the color black.								
M_COLOR_BLUE	Specifies the color blue.								

<input type="checkbox"/> M_COLOR_BRIGHT_GRAY	Specifies the color bright gray.
<input type="checkbox"/> M_COLOR_CYAN	Specifies the color cyan.
<input type="checkbox"/> M_COLOR_DARK_BLUE	Specifies the color dark blue.
<input type="checkbox"/> M_COLOR_DARK_CYAN	Specifies the color dark cyan.
<input type="checkbox"/> M_COLOR_DARK_GREEN	Specifies the color dark green.
<input type="checkbox"/> M_COLOR_DARK_MAGENTA	Specifies the color dark magenta.
<input type="checkbox"/> M_COLOR_DARK_RED	Specifies the color dark red.
<input type="checkbox"/> M_COLOR_DARK_YELLOW	Specifies the color dark yellow.
<input type="checkbox"/> M_COLOR_GRAY	Specifies the color gray.
<input type="checkbox"/> M_COLOR_GREEN	Specifies the color green.
<input type="checkbox"/> M_COLOR_LIGHT_BLUE	Specifies the color light blue.
<input type="checkbox"/> M_COLOR_LIGHT_GRAY	Specifies the color light gray.
<input type="checkbox"/> M_COLOR_LIGHT_GREEN	Specifies the color light green.
<input type="checkbox"/> M_COLOR_LIGHT_WHITE	Specifies the color light white.
<input type="checkbox"/> M_COLOR_MAGENTA	Specifies the color magenta.
<input type="checkbox"/> M_COLOR_RED	Specifies the color red.
<input type="checkbox"/> M_COLOR_WHITE	Specifies the color white.
<input type="checkbox"/> M_COLOR_YELLOW	Specifies the color yellow.
<input type="checkbox"/> Value	<p>Specifies a grayscale value.</p> <p>When drawing in a 1-band buffer, the grayscale value is cast to the type of the destination buffer.</p> <p>When drawing in a multi-band buffer, the grayscale value is cast to the type of the destination buffer's bands. This value will be replicated in each band.</p> <p>(summarize)</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraClear

Synopsis

Clear an image buffer to a specified foreground color.

Syntax

```
void MgraClear(  
    MIL_ID GraphContId,  
    MIL_ID DestImageBufId  
)
```

Description

This function clears the entire specified buffer to the foreground color of the specified graphics context.

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

● For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

DestImageBufId

Specifies the identifier of the buffer to clear. This parameter must be given a valid image buffer identifier.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraColor

Synopsis

Sets the foreground color of a graphics context.

Syntax

```
void MgraColor(
    MIL_ID GraphContId,
    MIL_DOUBLE ForegroundColor
)
```

Description

This function sets the foreground color of a specified graphics context.

Parameters

GraphContId

Specifies the graphics context with which to associate the foreground color. This parameter must be set to one of the following values:

For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

ForegroundColor

Sets the foreground color of a specified graphics context. This parameter must be set to one of the following values:

For the foreground color	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_RGB888(MIL_INT Red, MIL_INT Green, MIL_INT Blue)	Specifies an RGB value when using the graphics context to draw in an 8-bit, 3-band buffer. The red, green, and blue values must be values between 0 and 255, inclusive.
	When drawing in a 16-bit or 32-bit multi-band buffer, the components of the RGB value are cast to the type of the destination buffer's bands.
	To specify a 16-bit or 32-bit color component, use MgraControl() with M_COLOR combined with the appropriate constant (M_RED , M_GREEN , or M_BLUE). (summarize)
	<div>Parameters</div> <div>Red</div> <div>Specifies the red component, as a value between 0 and 255.</div>
	<div>Green</div> <div>Specifies the green component, as a value between 0 and 255.</div>
<input type="checkbox"/> M_COLOR_BLACK	<div>Blue</div> <div>Specifies the blue component, as a value between 0 and 255.</div>
	Specifies the color black.
<input type="checkbox"/> M_COLOR_BLUE	Specifies the color blue.

<input type="checkbox"/> M_COLOR_BRIGHT_GRAY	Specifies the color bright gray.
<input type="checkbox"/> M_COLOR_CYAN	Specifies the color cyan.
<input type="checkbox"/> M_COLOR_DARK_BLUE	Specifies the color dark blue.
<input type="checkbox"/> M_COLOR_DARK_CYAN	Specifies the color dark cyan.
<input type="checkbox"/> M_COLOR_DARK_GREEN	Specifies the color dark green.
<input type="checkbox"/> M_COLOR_DARK_MAGENTA	Specifies the color dark magenta.
<input type="checkbox"/> M_COLOR_DARK_RED	Specifies the color dark red.
<input type="checkbox"/> M_COLOR_DARK_YELLOW	Specifies the color dark yellow.
<input type="checkbox"/> M_COLOR_GRAY	Specifies the color gray.
<input type="checkbox"/> M_COLOR_GREEN	Specifies the color green.
<input type="checkbox"/> M_COLOR_LIGHT_BLUE	Specifies the color light blue.
<input type="checkbox"/> M_COLOR_LIGHT_GRAY	Specifies the color light gray.
<input type="checkbox"/> M_COLOR_LIGHT_GREEN	Specifies the color light green.
<input type="checkbox"/> M_COLOR_LIGHT_WHITE	Specifies the color light white.
<input type="checkbox"/> M_COLOR_MAGENTA	Specifies the color magenta.
<input type="checkbox"/> M_COLOR_RED	Specifies the color red.
<input type="checkbox"/> M_COLOR_WHITE	Specifies the color white.
<input type="checkbox"/> M_COLOR_YELLOW	Specifies the color yellow.
<input type="checkbox"/> Value	<p>Specifies a grayscale value.</p> <p>When drawing in a 1-band buffer, the grayscale value is cast to the type of the destination buffer.</p> <p>When drawing in a multi-band buffer, the grayscale value is cast to the type of the destination buffer's bands. This value will be replicated in each band.</p> <p>(summarize)</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraControl

Synopsis

Control a specified graphics context setting.

Syntax

```
void MgraControl(  
    MIL_ID GraphContId,  
    MIL_INT ControlType,  
    MIL_DOUBLE ControlValue  
)
```

Description

This function allows you to set the attributes of a graphics context.

Parameters

GraphContId

Specifies the graphics context to control. This parameter must be set to one of the following:

Note that there is a different default graphics context for each thread.

For specifying the graphics context

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

ControlType

Specifies the graphic setting to control.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the values needed for the control.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the table below.

- [For the graphics settings](#)

For the graphics settings	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	

<div><div></div><div>M_BACKCOLOR</div><div>+</div></div>	<div>Sets the background color of a specified graphics context.</div> <div>(summarize)</div>								
<div><div></div><div>M_RGB888(<div>MIL_INT <i>Red</i>, MIL_INT <i>Green</i>, MIL_INT <i>Blue</i></div>)</div></div>	<div>Specifies an RGB value when using the graphics context to draw in an 8-bit, 3-band buffer. The red, green, and blue values must be values between 0 and 255, inclusive.</div> <div>When drawing in a 16-bit or 32-bit multi-band buffer, the components of the RGB value are cast to the type of the destination buffer's bands.</div> <div>To specify a 16-bit or 32-bit color component, use MgraControl() with M_BACKCOLOR combined with the appropriate constant (M_RED, M_GREEN, or M_BLUE).</div> <div>(summarize)</div> <div><table><tr><td></td><td><i>Parameters</i></td></tr><tr><td></td><td><i>Red</i><div>Specifies the red component, as a value between 0 and 255.</div></td></tr><tr><td></td><td><i>Green</i><div>Specifies the green component, as a value between 0 and 255.</div></td></tr><tr><td></td><td><i>Blue</i><div>Specifies the blue component, as a value between 0 and 255.</div></td></tr></table></div>		<i>Parameters</i>		<i>Red</i> <div>Specifies the red component, as a value between 0 and 255.</div>		<i>Green</i> <div>Specifies the green component, as a value between 0 and 255.</div>		<i>Blue</i> <div>Specifies the blue component, as a value between 0 and 255.</div>
	<i>Parameters</i>								
	<i>Red</i> <div>Specifies the red component, as a value between 0 and 255.</div>								
	<i>Green</i> <div>Specifies the green component, as a value between 0 and 255.</div>								
	<i>Blue</i> <div>Specifies the blue component, as a value between 0 and 255.</div>								
<div><div></div><div>M_COLOR_BLACK</div></div>	<div>Specifies the color black.</div>								
<div><div></div><div>M_COLOR_BLUE</div></div>	<div>Specifies the color blue.</div>								
<div><div></div><div>M_COLOR_BRIGHT_GRAY</div></div>	<div>Specifies the color bright gray.</div>								
<div><div></div><div>M_COLOR_CYAN</div></div>	<div>Specifies the color cyan.</div>								
<div><div></div><div>M_COLOR_DARK_BLUE</div></div>	<div>Specifies the color dark blue.</div>								
<div><div></div><div>M_COLOR_DARK_CYAN</div></div>	<div>Specifies the color dark cyan.</div>								
<div><div></div><div>M_COLOR_DARK_GREEN</div></div>	<div>Specifies the color dark green.</div>								
<div><div></div><div>M_COLOR_DARK_MAGENTA</div></div>	<div>Specifies the color dark magenta.</div>								
<div><div></div><div>M_COLOR_DARK_RED</div></div>	<div>Specifies the color dark red.</div>								
<div><div></div><div>M_COLOR_DARK_YELLOW</div></div>	<div>Specifies the color dark yellow.</div>								
<div><div></div><div>M_COLOR_GRAY</div></div>	<div>Specifies the color gray.</div>								
<div><div></div><div>M_COLOR_GREEN</div></div>	<div>Specifies the color green.</div>								
<div><div></div><div>M_COLOR_LIGHT_BLUE</div></div>	<div>Specifies the color light blue.</div>								
<div><div></div><div>M_COLOR_LIGHT_GRAY</div></div>	<div>Specifies the color light gray.</div>								
<div><div></div><div>M_COLOR_LIGHT_GREEN</div></div>	<div>Specifies the color light green.</div>								
<div><div></div><div>M_COLOR_LIGHT_WHITE</div></div>	<div>Specifies the color light white.</div>								
<div><div></div><div>M_COLOR_MAGENTA</div></div>	<div>Specifies the color magenta.</div>								
<div><div></div><div>M_COLOR_RED</div></div>	<div>Specifies the color red.</div>								
<div><div></div><div>M_COLOR_WHITE</div></div>	<div>Specifies the color white.</div>								
<div><div></div><div>M_COLOR_YELLOW</div></div>	<div>Specifies the color yellow.</div>								
<div><div></div><div>Value</div></div>	<div>Specifies a grayscale value. The buffer can be a 1-band or multi-band buffer. The specified value is cast to the buffer type and depth.</div> <div>The default value is 0.</div> <div>(summarize)</div>								
<div><div></div><div>M_BACKGROUND_MODE</div></div>	<div>Controls the setting of the background color on the drawing surface.</div> <div>(summarize)</div>								
<div><div></div><div>M_OPAQUE</div></div>	<div>Fills background with the current background color before drawing text.</div>								

	This is the default value. (summarize)								
<div><div></div>M_TRANSPARENT</div>	Does not change background before drawing text. This creates a transparent background for printed characters. (summarize)								
<div><div></div>M_COLOR +</div>	Sets the foreground color of a specified graphics context. (summarize)								
<div><div><div><div></div>M_RGB888(MIL_INT <i>Red</i>, MIL_INT <i>Green</i>, MIL_INT <i>Blue</i>)</div></div></div>	<p>Specifies an RGB value when using the graphics context to draw in an 8-bit, 3-band buffer. The red, green, and blue values must be values between 0 and 255, inclusive.</p> <p>When drawing in a 16-bit or 32-bit multi-band buffer, the components of the RGB value are cast to the type of the destination buffer's bands.</p> <p>To specify a 16-bit or 32-bit color compenent, use MgraControl() with M_COLOR combined with the appropriate constant (M_RED, M_GREEN, or M_BLUE). (summarize)</p> <table><tr><td></td><td><i>Parameters</i></td></tr><tr><td></td><td><i>Red</i> Specifies the red value of an RGB value.</td></tr><tr><td></td><td><i>Green</i> Specifies the green value of an RGB value.</td></tr><tr><td></td><td><i>Blue</i> Specifies the blue value of an RGB value.</td></tr></table>		<i>Parameters</i>		<i>Red</i> Specifies the red value of an RGB value.		<i>Green</i> Specifies the green value of an RGB value.		<i>Blue</i> Specifies the blue value of an RGB value.
	<i>Parameters</i>								
	<i>Red</i> Specifies the red value of an RGB value.								
	<i>Green</i> Specifies the green value of an RGB value.								
	<i>Blue</i> Specifies the blue value of an RGB value.								
<div><div></div>M_COLOR_BLACK</div>	Specifies the color black.								
<div><div></div>M_COLOR_BLUE</div>	Specifies the color blue.								
<div><div></div>M_COLOR_BRIGHT_GRAY</div>	Specifies the color bright gray.								
<div><div></div>M_COLOR_CYAN</div>	Specifies the color cyan.								
<div><div></div>M_COLOR_DARK_BLUE</div>	Specifies the color dark blue.								
<div><div></div>M_COLOR_DARK_CYAN</div>	Specifies the color dark cyan.								
<div><div></div>M_COLOR_DARK_GREEN</div>	Specifies the color dark green.								
<div><div></div>M_COLOR_DARK_MAGENTA</div>	Specifies the color dark magenta.								
<div><div></div>M_COLOR_DARK_RED</div>	Specifies the color dark red.								
<div><div></div>M_COLOR_DARK_YELLOW</div>	Specifies the color dark yellow.								
<div><div></div>M_COLOR_GRAY</div>	Specifies the color gray.								
<div><div></div>M_COLOR_GREEN</div>	Specifies the color green.								
<div><div></div>M_COLOR_LIGHT_BLUE</div>	Specifies the color light blue.								
<div><div></div>M_COLOR_LIGHT_GRAY</div>	Specifies the color light gray.								
<div><div></div>M_COLOR_LIGHT_GREEN</div>	Specifies the color light green.								
<div><div></div>M_COLOR_LIGHT_WHITE</div>	Specifies the color light white.								
<div><div></div>M_COLOR_MAGENTA</div>	Specifies the color magenta.								
<div><div></div>M_COLOR_RED</div>	Specifies the color red.								
<div><div></div>M_COLOR_WHITE</div>	Specifies the color white.								
<div><div></div>M_COLOR_YELLOW</div>	Specifies the color yellow.								
<div><div></div>Value</div>	<p>Specifies a grayscale value. The buffer can be a 1-band or multi-band buffer. The specified value is cast to the buffer type and depth.</p> <p>The default value is -1. For example, this value is equivalent to 0xFF for an 8-bit buffer. (summarize)</p>								

Combination constants for [M_BACKCOLOR](#); [M_COLOR](#);

You can add one of the following values to the above-mentioned values to specify the color component.

For M_COLOR or M_BACKCOLOR	
Value	Description
M_BLUE	Specifies the blue color component.
M_GREEN	Specifies the green color component.
M_RED	Specifies the red color component.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraDot

Synopsis

Draw a dot.

Syntax

```
void MgraDot(
    MIL_ID GraphContId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE XPos,
    MIL_DOUBLE YPos
)
```

Description

This function draws a dot at the specified drawing position, using the foreground color of the specified graphics context.

This function uses the image coordinate system (pixel coordinates). To draw in a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using [McalTransformCoordinate\(\)](#) or [McalTransformCoordinateList\(\)](#).

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

For specifying the graphics context	
Value	Description
M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

DestImageBufId

Specifies the identifier of the buffer in which to draw. This parameter must be given a valid image buffer identifier.

XPos

Specifies the X-coordinate of the drawing position. The given coordinate is relative to the top-left corner of the specified target buffer. It should be valid in the specified image buffer; otherwise, nothing will be drawn.

YPos

Specifies the Y-coordinate of the drawing position. The given coordinate is relative to the top-left corner of the specified target buffer. It should be valid in the specified image buffer; otherwise, nothing will be drawn.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraDots

Synopsis

Draw one or more dots.

Syntax

```
void MgraDots(
    MIL_ID GraphContId,
    MIL_ID DestImageBufId,
    MIL_INT NumberOfDots,
    const MIL_DOUBLE *XPosArray,
    const MIL_DOUBLE *YPosArray,
    MIL_INT ControlFlag
)
```

Description

This function draws dots at the specified drawing positions, using the foreground color of the specified graphics context.

This function uses the image coordinate system (pixel coordinates). To draw in a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using [McalTransformCoordinate\(\)](#) or [McalTransformCoordinateList\(\)](#).

Note that prior to MIL 9.0, this function only supported integer positions; the [XPosArray](#) and [YPosArray](#) parameters only accepted arrays of type *long*. As of MIL 9.0, this function includes support for positions with floating-point precision. To implement this change and maintain backwards compatibility when using a C++ compiler (*.cpp), **MgraDots()** is available as an inline function which automatically calls **MgraDotsDouble()**, **MgraDotsInt64()**, and **MgraDotsInt32()**, depending on whether [XPosArray](#) and [YPosArray](#) receive arrays of type *MIL_INT32*, *MIL_INT64*, or *MIL_DOUBLE*, respectively. To maintain backwards compatibility when using a C compiler (*.c), **MgraDots()** maps to **MgraDotsInt32()**; you must explicitly call **MgraDotsDouble()** or **MgraDotsInt64()** to pass [XPosArray](#) and [YPosArray](#) arrays of type *MIL_INT64* or *MIL_DOUBLE*, respectively. If you are an advanced user and want to retrieve a pointer to **MgraDots()**, you must use the Double, Int64, or Int32 version of this function, since **MgraDots()** is actually a macro or an overloaded function.

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

For the identifier of the graphics context	
Value	Description
M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
MIL graphics context identifier	Specifies the identifier of the graphics context, which you have allocated using the MgraAlloc() .

DestImageBufId

Specifies the identifier of the buffer in which to draw. This parameter must be given a valid image buffer identifier.

NumberOfDots

Specifies the number of dots to draw.

XPosArray

Specifies the address of the array containing the X-coordinate(s) of the dots to be drawn.

Each given coordinate is relative to the top-left corner of the specified target buffer. It should be valid in the specified image buffer; otherwise, nothing will be drawn.

YPosArray

Specifies the address of the array containing the Y-coordinate(s) of the dots to be drawn.

Each given coordinate is relative to the top-left corner of the specified target buffer. It should be valid in the specified image buffer; otherwise, nothing will be drawn.

ControlFlag

Specifies the function's control flag and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraFill

Synopsis

Perform a boundary-type seed fill.

Syntax

```
void MgraFill(
    MIL_ID GraphContId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE XStart,
    MIL_DOUBLE YStart
)
```

Description

This function performs a boundary-type seed fill. It fills in an area of the target buffer, with the foreground color of the specified graphics context, starting from the specified seed position. Filling occurs on adjacent pixels (vertically and horizontally to original seed pixel) that have the same value as the original seed pixel.

If the source buffer is a multi-band buffer, this function will process each band separately. This means that each band of the adjacent pixels will be compared with the corresponding band of the seed pixel. This can produce strange results if, for example, you try to fill the inside of a red circle with blue. The blue will spread to the whole image since the red circle does not exist in the blue band.

This function accepts only pixel coordinates. If you want to fill an area of a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using [McalTransformCoordinate\(\)](#) or [McalTransformCoordinateList\(\)](#).

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

For specifying the graphics context	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

DestImageBufId

Specifies the identifier of the buffer in which to draw. This parameter must be given a valid image buffer identifier.

XStart

Specifies the X-coordinate of the seed position. If the specified point is not within an enclosed area, filling occurs until the boundaries of the buffer are encountered. The given coordinate is relative to the top-left corner of the specified target buffer. It should be valid in the specified image buffer; otherwise, the operation is not performed.

YStart

Specifies the Y-coordinate of the seed position. If the specified point is not within an enclosed area, filling occurs until the boundaries of the buffer are encountered. The given coordinate is relative to the top-left corner of the specified target buffer. It should be valid in the specified image buffer; otherwise, the operation is not performed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.

DLL	Requires mil.dll.
-----	-------------------

MgraFont

Synopsis

Associate a text font with a graphics context.

Syntax

```
void MgraFont(  
    MIL_ID GraphContId,  
    MIL_INT FontName  
)
```

Description

This function associates a character font with the specified graphics context for use with subsequent [MgraText\(\)](#) function calls.

Parameters

GraphContId

Specifies the identifier of the graphics context with which to associate the character font. This parameter must be set to one of the following values:

● For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

FontName

Specifies the font with which to write text. This parameter can be set to one of the following:

● For specifying the font	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FONT_DEFAULT	Same as M_FONT_DEFAULT_SMALL .
<input type="checkbox"/> M_FONT_DEFAULT_LARGE	Sets the parameter to the default font with 16x32 pixel wide characters.
<input type="checkbox"/> M_FONT_DEFAULT_MEDIUM	Sets the parameter to the default font with 12x24 pixel wide characters.
<input type="checkbox"/> M_FONT_DEFAULT_SMALL	Sets the parameter to the default font with 8x16 pixel wide characters.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraFontScale

Synopsis

Set the font scale of a graphics context.

Syntax

```
void MgraFontScale(
    MIL_ID GraphContId,
    MIL_DOUBLE XFontScale,
    MIL_DOUBLE YFontScale
)
```

Description

This function sets the font scale of the specified graphics context for use with subsequent [MgraText\(\)](#) function calls.

Parameters

GraphContId

Specifies the identifier of the graphics context for which to set the font scale. This parameter must be set to one of the following values:

For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

XFontScale

Specifies the X-scale-factor by which to multiply the width of the font characters. This parameter can be independently set to any positive floating-point value. The default scale factor is 1.0.

Note, using a font with a scale of 1.0 accelerates text drawing.

YFontScale

Specifies the Y-scale-factor by which to multiply the height of the font characters. This parameter can be independently set to any positive floating-point value. The default scale factor is 1.0.

Note, using a font with a scale of 1.0 accelerates text drawing.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraFree

Synopsis

Free a graphics context.

Syntax

```
void MgraFree(
    MIL_ID GraphContId
)
```

Description

This function deallocates a graphics context previously allocated with [MgraAlloc\(\)](#).

Parameter

GraphContId

Specifies the identifier of the graphics context to deallocate. If **M_DEFAULT** is specified, an error will occur.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraInquire

Synopsis

Inquire about the graphics parameters.

Syntax

```
MIL_INT MgraInquire(
    MIL_ID GraphContId,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires about a graphic parameter in the specified graphics context.

Parameters

GraphContId

Specifies the identifier of the graphics context on which to perform the inquiry. This parameter must be set to one of the following values:

For specifying the graphics context	
Value	Description
M_DEFAULT	Specifies that the default graphics context of the current MIL application is used. Note that there is a different default graphics context for each thread. (summarize)
Value	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc().

InquireType

Specifies the graphic parameter about which to inquire. This parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the UserVarPtr parameter the address of a MIL_DOUBLE.

For specifying graphic parameters	
Value	Description
M_BACKCOLOR +	Returns the background color. (summarize) <div>UserVarPtr info Return values: M_COLOR_BLACK; M_COLOR_BLUE; M_COLOR_BRIGHT_GRAY; M_COLOR_CYAN; M_COLOR_DARK_BLUE; M_COLOR_DARK_CYAN; M_COLOR_DARK_GREEN; M_COLOR_DARK_MAGENTA; M_COLOR_DARK_RED; M_COLOR_DARK_YELLOW; M_COLOR_GRAY; M_COLOR_GREEN; M_COLOR_LIGHT_BLUE; M_COLOR_LIGHT_GRAY; M_COLOR_LIGHT_GREEN; M_COLOR_LIGHT_WHITE; M_COLOR_MAGENTA; M_COLOR_RED; M_COLOR_WHITE; M_COLOR_YELLOW; Value; (details) Byte-encoded RGB value An encoded RGB value. To retrieve the R, G, and B components, use the M_RGB888_R, M_RGB888_G, and M_RGB888_B macros.</div>
M_BACKGROUND_MODE	Returns the background mode. (summarize) <div>UserVarPtr info Data type: MIL_INT</div>

	Return values: M_OPAQUE; M_TRANSPARENT; (details)
<input type="checkbox"/> M_COLOR +	<p>Returns the foreground color. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_COLOR_BLACK; M_COLOR_BLUE; M_COLOR_BRIGHT_GRAY; M_COLOR_CYAN; M_COLOR_DARK_BLUE; M_COLOR_DARK_CYAN; M_COLOR_DARK_GREEN; M_COLOR_DARK_MAGENTA; M_COLOR_DARK_RED; M_COLOR_DARK_YELLOW; M_COLOR_GRAY; M_COLOR_GREEN; M_COLOR_LIGHT_BLUE; M_COLOR_LIGHT_GRAY; M_COLOR_LIGHT_GREEN; M_COLOR_LIGHT_WHITE; M_COLOR_MAGENTA; M_COLOR_RED; M_COLOR_WHITE; M_COLOR_YELLOW; Value; (details)</p> <p>Byte-encoded RGB value An encoded RGB value. To retrieve the R, G, and B components, use the M_RGB888_R, M_RGB888_G, and M_RGB888_B macros.</p>
<input type="checkbox"/> M_FONT	<p>Returns the character font. (summarize)</p> <p><i>UserVarPtr info</i> Data type: MIL_INT Return values: M_FONT_DEFAULT; M_FONT_DEFAULT_LARGE; M_FONT_DEFAULT_MEDIUM; M_FONT_DEFAULT_SMALL; (details)</p>
<input type="checkbox"/> M_FONT_X_SCALE	<p>Returns the font scaling factor in X. (summarize)</p> <p><i>UserVarPtr info</i> Return values: Please see the XFontScale parameter of MgraFontScale(). (details)</p>
<input type="checkbox"/> M_FONT_Y_SCALE	<p>Returns the font scaling factor in Y. (summarize)</p> <p><i>UserVarPtr info</i> Return values: Please see the YFontScale parameter of MgraFontScale(). (details)</p>
<input type="checkbox"/> M_OWNER_SYSTEM	<p>Returns the MIL identifier (MIL_ID) of the system on which the graphics context has been allocated. (summarize)</p> <p><i>UserVarPtr info</i> Data type: MIL_ID Return values: MIL system identifier; M_DEFAULT_HOST; (details)</p>

Combination constants for [M_BACKCOLOR](#); [M_COLOR](#);

You can add one of the following values to the above-mentioned values to get the color value used in the graphics context for a 16-bit or 32-bit multi-band buffer.

You must inquire each color component (R,G, and B) separately.

For example, you would make the following call to inquire the red color component: `MgraInquire(M_DEFAULT, M_COLOR + M_RED, &red color component)`.

● For M_COLOR or M_BACKCOLOR	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BLUE	Returns the blue component.
<input type="checkbox"/> M_GREEN	Returns the green component.
<input type="checkbox"/> M_RED	Returns the red component.

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- MIL_DOUBLE
- MIL_ID

- MIL_INT

Specifies the address in which the requested information is to be written. Since this function also returns the requested information, you can set this parameter to M_NULL.

Return value

The returned value is the requested system information cast to *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraLine

Synopsis

Draw a line.

Syntax

```
void MgraLine(
    MIL_ID GraphContId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE XStart,
    MIL_DOUBLE YStart,
    MIL_DOUBLE XEnd,
    MIL_DOUBLE YEnd
)
```

Description

This function draws a line starting and ending at the specified coordinates, using the foreground color of the specified graphics context.

Note that the given coordinates ([XStart](#), [YStart](#), [XEnd](#), [YEnd](#)) are relative to the top-left corner of the specified target buffer. They should be valid in the specified buffer; otherwise, the line is clipped outside the buffer boundaries.

This function uses the image coordinate system (pixel coordinates). To draw in a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using [McalTransformCoordinate\(\)](#) or [McalTransformCoordinateList\(\)](#).

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

● For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> Value	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

DestImageBufId

Specifies the identifier of the buffer in which to draw. This parameter must be given a valid image buffer identifier.

XStart

Specifies the X-coordinate for the start of the line position.

YStart

Specifies the Y-coordinate for the start of the line position.

XEnd

Specifies the X-coordinate for the end of the line position.

YEnd

Specifies the Y-coordinate for the end of the line position.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraLines

Synopsis

Draw one or more lines.

Syntax

```
void MgraLines(
    MIL_ID GraphContId,
    MIL_ID DestImageBufId,
    MIL_INT NumberOfLines,
    const MIL_DOUBLE *XStartArray,
    const MIL_DOUBLE *YStartArray,
    const MIL_DOUBLE *XEndArray,
    const MIL_DOUBLE *YEndArray,
    MIL_INT ControlFlag
)
```

Description

This function draws lines starting and ending at the specified coordinates, using the foreground color of the specified graphics context.

The given coordinates ([XStartArray](#), [YStartArray](#), [XEndArray](#), [YEndArray](#)) are relative to the top-left corner of the specified target buffer. They should be valid in the specified buffer; otherwise, the line is clipped outside the buffer boundaries.

This function uses the image coordinate system (pixel coordinates). To draw in a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using [McalTransformCoordinate\(\)](#) or [McalTransformCoordinateList\(\)](#).

Note that prior to MIL 9.0, this function only supported integer positions; the [XStartArray](#), [YStartArray](#), [XEndArray](#), and [YEndArray](#) parameters only accepted arrays of type *long*. As of MIL 9.0, this function includes support for positions with floating-point precision. To implement this change and maintain backwards compatibility when using a C++ compiler (*.cpp), **MgraLines()** is available as an inline function which automatically calls **MgraLinesDoubleQ()**, **MgraLinesInt64Q()**, and **MgraLinesInt32Q()**, depending on whether [XStartArray](#), [YStartArray](#), [XEndArray](#), and [YEndArray](#) receive arrays of type *MIL_INT32*, *MIL_INT64*, or *MIL_DOUBLE*, respectively. To maintain backwards compatibility when using a C compiler (*.c), **MgraLines()** maps to **MgraLinesInt32Q()**; you must explicitly call **MgraLinesDoubleQ()** or **MgraLinesInt64Q()** to pass [XStartArray](#), [YStartArray](#), [XEndArray](#), and [YEndArray](#) arrays of type *MIL_INT64* or *MIL_DOUBLE*, respectively. If you are an advanced user and want to retrieve a pointer to **MgraLines()**, you must use the Double, Int64, or Int32 version of this function, since **MgraLines()** is actually a macro or an overloaded function.

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies the identifier of the graphics context, which you have allocated using the MgraAlloc() .

DestImageBufId

Specifies the identifier of the buffer in which to draw. This parameter must be given a valid image buffer identifier.

NumberOfLines

Specifies the number of lines to draw.

XStartArray

Specifies the address of the array containing the X-coordinate(s) of the start of the lines to be drawn.

YStartArray

Specifies the address of the array containing the Y-coordinate(s) of the start of the lines to be drawn.

XEndArray

Specifies the address of the array containing the X-coordinate(s) of the end of the lines to be drawn.

YEndArray

Specifies the address of the array containing the Y-coordinate(s) of the end of the lines to be drawn.

ControlFlag

Specifies the function's control flag and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraRect

Synopsis

Draw a rectangle.

Syntax

```
void MgraRect(  
    MIL_ID GraphContId,  
    MIL_ID DestImageBufId,  
    MIL_DOUBLE XStart,  
    MIL_DOUBLE YStart,  
    MIL_DOUBLE XEnd,  
    MIL_DOUBLE YEnd  
)
```

Description

This function draws a rectangle starting from the specified top-left coordinate to the specified bottom-right corner. The rectangle is drawn in the foreground color of the specified graphics context.

Note that the given coordinates ([XStart](#), [YStart](#), [XEnd](#), [YEnd](#)) are relative to the top-left corner of the specified target buffer. They should be valid in the specified buffer; otherwise, the line is clipped outside the buffer boundaries.

This function uses the image coordinate system (pixel coordinates). To draw in a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using [McalTransformCoordinate\(\)](#) or [McalTransformCoordinateList\(\)](#).

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

For specifying the graphics context	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> Value	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

DestImageBufId

Specifies the identifier of the buffer in which to draw. This parameter must be given a valid image buffer identifier.

XStart

Specifies the X-coordinate of the top-left corner of the rectangle.

YStart

Specifies the Y-coordinate of the top-left corner of the rectangle.

XEnd

Specifies the X-coordinate of the bottom-right corner of the rectangle.

YEnd

Specifies the Y-coordinate of the bottom-right corner of the rectangle.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraRectFill

Synopsis

Draw a filled rectangle.

Syntax

```
void MgraRectFill(
    MIL_ID GraphContId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE XStart,
    MIL_DOUBLE YStart,
    MIL_DOUBLE XEnd,
    MIL_DOUBLE YEnd
)
```

Description

This function draws a filled rectangle starting from the specified top-left coordinate to the specified bottom-right corner. The rectangle is drawn and filled in the foreground color of the specified graphics context.

Note that the given coordinates ([XStart](#), [YStart](#), [XEnd](#), [YEnd](#)) are relative to the top-left corner of the specified target buffer. They should be valid in the specified buffer; otherwise, the line is clipped outside the buffer boundaries.

This function uses the image coordinate system (pixel coordinates). To draw in a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using [McalTransformCoordinate\(\)](#) or [McalTransformCoordinateList\(\)](#).

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

● For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> Value	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

DestImageBufId

Specifies the identifier of the buffer in which to draw. This parameter must be given a valid image buffer identifier.

XStart

Specifies the X-coordinate of the top-left corner of the rectangle.

YStart

Specifies the Y-coordinate of the top-left corner of the rectangle.

XEnd

Specifies the X-coordinate of the bottom-right corner of the rectangle.

YEnd

Specifies the Y-coordinate of the bottom-right corner of the rectangle.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MgraText

Synopsis

Write text.

Syntax

```
void MgraText(  
    MIL_ID GraphContId,  
    MIL_ID DestImageBufId,  
    MIL_DOUBLE XStart,  
    MIL_DOUBLE YStart,  
    MIL_CONST_TEXT_PTR String  
)
```

Description

This function writes the specified string to the specified buffer starting at the specified writing position, using the parameters (colors, font, and size) defined in the graphics context. Use [MgraFont\(\)](#) and [MgraFontScale\(\)](#) to modify the font and size. Use [MgraControl\(\)](#) to obtain a transparent background for printed characters.

Note that the coordinates at which to start writing the string ([XStart](#), [YStart](#)) are relative to the top-left corner of the specified target buffer. They should be valid in the specified buffer; otherwise, the line is clipped outside the buffer boundaries.

This function uses the image coordinate system (pixel coordinates). To write in a calibrated image using real-world coordinates, you must first convert the coordinates from their real-world value to their pixel value using [McalTransformCoordinate\(\)](#) or [McalTransformCoordinateList\(\)](#).

Parameters

GraphContId

Specifies the identifier of the graphics context. This parameter must be set to one of the following values:

For specifying the graphics context	
Value	Description
M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
Value	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

DestImageBufId

Specifies the identifier of the buffer in which to write. This parameter must be given a valid image buffer identifier.

XStart

Specifies the X-coordinate of the position at which to start writing the top-left corner of the first character.

YStart

Specifies the Y-coordinate of the position at which to start writing the top-left corner of the first character.

String

Specifies the string that must be written in the destination buffer.



● For specifying the string		
☐ Value	Description	
☐ MIL_TEXT(MIL_TEXT_PTR String)	Specifies the address of the string that must be written in the destination buffer. There is no restriction on the length of the string, except that the string must be null (\0) terminated. (summarize)	
		Parameters
	String	Specifies the address of the string.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

Mim functions

Synopsis

The functions prefixed with Mim make up the Image Processing module. The basic Image Processing module gives the user the possibility to do point-to-point, statistical, spatial filtering, morphological, and geometric transformation operations. These operations allow you to perform image enhancements, distortion corrections, and analyses.

Functions

- MimAlloc
- MimAllocResult
- MimArith
- MimArithMultiple
- MimBinarize
- MimClip
- MimClose
- MimConnectMap
- MimControl
- MimConvert
- MimConvolve
- MimCountDifference
- MimDeinterlace
- MimDilate
- MimDistance
- MimDraw
- MimEdgeDetect
- MimErode
- MimFindExtreme
- MimFlip
- MimFree
- MimGetResult
- MimGetResult1d
- MimHistogram
- MimHistogramEqualize
- MimInquire
- MimLabel
- MimLocateEvent
- MimLocatePeak1d
- MimLutMap
- MimMorphic
- MimOpen
- MimPolarTransform
- MimProject
- MimRank
- MimResize
- MimRestore
- MimRotate
- MimSave
- MimShift
- MimStat
- MimStream
- MimThick
- MimThin
- MimTransform
- MimTranslate
- MimWarp
- MimWatershed
- MimZoneOfInfluence

MimAlloc

Synopsis

Allocate an image processing context.

Syntax

```
MIL_ID MimAlloc(  
    MIL_ID SystemId,  
    MIL_INT ContextType,  
    MIL_INT ControlFlag,  
    MIL_ID *IdVarPtr  
)
```

Description

This function allocates an image processing context on the specified system.

When the image processing context is no longer required, you should release its memory, using [MimFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the context. This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ContextType

Specifies the type of image processing context. This parameter should be set to the value below:

● For specifying the type of context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEINTERLACE_CONTEXT	Specifies an image processing context that can be used with MimDeinterlace() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

IdVarPtr

Specifies the address of the variable in which to write the image processing context identifier. Since the **MimAlloc()** function also returns the image processing context identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the image processing context identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimAllocResult

Synopsis

Allocate an image processing result buffer.

Syntax

```
MIL_ID MimAllocResult(  
    MIL_ID SystemId,  
    MIL_INT NbEntries,  
    MIL_INT ResultType,  
    MIL_ID *ImResultIdPtr  
)
```

Description

This function allocates a result buffer with the specified number of entries, for use with the Image Processing module's statistical functions.

When the result buffer is no longer required, you should release its memory, using [MimFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the result buffer. This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> Value	Specifies a valid system identifier, previously allocated using MsysAlloc() .

NbEntries

Specifies the number of buffer entries of the specified result buffer type ([ResultType](#)). This parameter should be set to one of the following values:

● For specifying the number of buffer entries	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that MIL automatically decides the size of the result buffer. This applies only when the ResultType parameter is set to M_COUNT_LIST or M_STAT_LIST . (summarize)
<input type="checkbox"/> Value	Specifies a valid number of buffer entries.

ResultType

Specifies the type of data that will be stored in this result buffer. This parameter must be set to one of the following values:

● For specifying the type of data	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_COUNT_LIST	Specifies MimCountDifference() results.

<input type="checkbox"/> M_EVENT_LIST +	Specifies MimLocateEvent() results.
<input type="checkbox"/> M_EXTREME_LIST +	Specifies MimFindExtreme() results.
<input type="checkbox"/> M_HIST_LIST	Specifies MimHistogram() results.
<input type="checkbox"/> M_PROJ_LIST +	Specifies MimProject() results.
<input type="checkbox"/> M_STAT_LIST	Specifies MimStat() results.

Combination constant for [M_EVENT_LIST](#); [M_EXTREME_LIST](#); [M_PROJ_LIST](#);

You can add the following value to the above-mentioned values to set the type of data stored in the result buffer.

● For [M_PROJ_LIST](#), [M_EXTREME_LIST](#), or [M_EVENT_LIST](#) to change the type of data to store

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FLOAT	Allocates a result buffer of type <i>float</i> . You should allocate a <i>float</i> result type buffer when the source buffer specified by MimProject() , MimFindExtreme() , or MimLocateEvent() is of type <i>float</i> . If the specified source buffer is of type <i>float</i> and the result buffer is not, the functions can produce inaccurate results. Note that allocating a result buffer of type <i>float</i> when the source buffer is not, will slow down the operations performed by the MimProject() , MimFindExtreme() , or MimLocateEvent() function. (summarize)

ImResultIdPtr

Specifies the address of the variable in which to write the image processing result buffer identifier. Since **MimAllocResult()** also returns the image processing result buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimArith

Synopsis

Perform a point-to-point arithmetic operation.

Syntax

```
void MimArith(
    MIL_DOUBLE Src1ImageBufIdOrConst,
    MIL_DOUBLE Src2ImageBufIdOrConst,
    MIL_ID DestImageBufId,
    MIL_INT Operation
)
```

Description

This function performs the specified point-to-point operation on two images, an image and a constant, an image, or a constant, storing results in the specified destination image buffer.

Parameters

Src1ImageBufIdOrConst

Specifies the data source of the first operand. This parameter can be given an image buffer identifier or a constant. When using a constant, it will be considered to have the same type as the destination buffer.

Src2ImageBufIdOrConst

Specifies the data source of the second operand. This parameter can be given an image buffer identifier or a constant. If the selected operation uses only one operand, set this parameter to **M_NULL**. When using a constant, it will be considered to have the same type as the destination buffer.

DestImageBufId

Specifies the identifier of the destination of the results. This parameter must be given an image buffer identifier.

Operation

Specifies the operation to perform. This parameter should be set in accordance to the operands. Operations using two image buffer operands are the following:

● For specifying the type of operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ADD +	Adds the values of the first image to the values of the second image.
<input type="checkbox"/> M_AND	Performs a bitwise AND operation on the two operands.
<input type="checkbox"/> M_DIV +	Divides the values of the first image by the values of the second image. Note that dividing a value by 0 will generate a destination pixel with an unpredictable value. This will not generate an error. (summarize)
<input type="checkbox"/> M_MAX	Compares the values of the first image with the values of the second image, and takes the maximum of the two.
<input type="checkbox"/> M_MIN	Compares the values of the first image with the values of the second image, and takes the minimum of the two.
<input type="checkbox"/> M_MULT +	Multiplies the values of the first image with the values of the second image.
<input type="checkbox"/> M_NAND	Performs a bitwise 'AND-NOT' operation on the two operands.
<input type="checkbox"/> M_NOR	Performs a bitwise NOR operation on the two operands.
<input type="checkbox"/> M_OR	Performs a bitwise OR operation on the two operands.

<input type="checkbox"/> M_SUB +	Subtracts the values of the second image from the first image (for example, image 1 - image 2).
<input type="checkbox"/> M_SUB_ABS +	Takes the absolute value of the result of subtracting the two operands.
<input type="checkbox"/> M_XNOR	Performs a bitwise exclusive NOR operation on the operands.
<input type="checkbox"/> M_XOR	Performs a bitwise exclusive OR operation on the operands.

The following are operations that use an image buffer operand and a constant. If the operation is not commutative, you must give the constant as either the first or second operand, as can be determined from the predefined operation name.

● For operations using one image buffer and a constant	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ADD_CONST +	Adds a constant to the values of an image.
<input type="checkbox"/> M_AND_CONST	Performs a bitwise AND operation on the image and a constant.
<input type="checkbox"/> M_CONST_DIV +	Divides a constant by the values of an image (for example, constant / image). Note that dividing the constant by 0 will generate a destination pixel with an unpredictable value. This will not generate an error. (summarize)
<input type="checkbox"/> M_CONST_PASS	Copies the constant to each destination pixel.
<input type="checkbox"/> M_CONST_SUB +	Subtracts the image from the constant (for example, constant - image).
<input type="checkbox"/> M_DIV_CONST +	Divides the values of an image by a constant (for example, image / constant). Note that dividing a value by 0 will generate an error. (summarize)
<input type="checkbox"/> M_MAX_CONST	Compares the values of one image with a constant, and takes the maximum of the two.
<input type="checkbox"/> M_MIN_CONST	Compares the values of one image with a constant, and takes the minimum of the two.
<input type="checkbox"/> M_MULT_CONST +	Multiplies the values of the image by the constant.
<input type="checkbox"/> M_NAND_CONST	Performs a bitwise AND-NOT operation with an image and a constant.
<input type="checkbox"/> M_NOR_CONST	Performs a bitwise NOR operation with an image and a constant.
<input type="checkbox"/> M_OR_CONST	Performs a bitwise OR operation with an image and a constant.
<input type="checkbox"/> M_SUB_CONST +	Subtracts an constant from the values of an image (for example, image - constant).
<input type="checkbox"/> M_XNOR_CONST	Performs a bitwise exclusive NOR operation with an image and a constant.
<input type="checkbox"/> M_XOR_CONST	Performs a bitwise exclusive OR operation with an image and a constant.

Combination constant for [M_ADD](#); [M_MULT](#); [M_SUB](#); [M_SUB_ABS](#); [M_ADD_CONST](#); [M_CONST_SUB](#); [M_MULT_CONST](#); [M_SUB_CONST](#);

You can add the following value to the above-mentioned values to specify that the operation result should be saturated if necessary.

● For M_ADD , M_ADD_CONST , M_SUB , M_SUB_ABS , M_SUB_CONST , M_CONST_SUB , M_MULT , and M_MULT_CONST	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SATURATION	Forces the operation to saturate any resulting pixel values that overflow or underflow the possible range of the destination buffer. The pixel values are clipped to fit within the buffer's range, rather than wrapped around the range. If you do not specify M_SATURATION and there is an overflow or underflow, the resulting pixel values are either clipped (saturated) or wrapped around the buffer's range (not saturated), depending on which is faster for your hardware. (summarize)

Combination constant for [M_DIV](#); [M_CONST_DIV](#); [M_DIV_CONST](#);

You can add the following value to the above-mentioned values to specify that the result is altered to a fixed point format.

● For M_DIV, M_CONST_DIV, and M_DIV_CONST	
☐ Value	Description
☐ M_FIXED_POINT	<p>Forces the operation to result in a fixed point format.</p> <p>Note that operations using this attribute result in an 0.8 fixed-point format for an 8-bit destination. That is, the 8 bits comprise only the fractional portion of the value, and no integer portion. When the destination image buffer is 16-bit, operations that use the M_FIXED_POINT attribute result in an 8.8 fixed-point format (where the eight most-significant bits are the integer portion and the eight least-significant bits are the fractional portion). For a 32-bit destination, the result is in a 16.16 fixed-point format.</p> <p>Note that this attribute cannot be used when using a floating-point destination buffer.</p> <p>(summarize)</p>

Operations using only one image buffer operand are the following:

● For operations using one image buffer	
☐ Value	Description
☐ M_ABS	Takes the absolute value of the image values.
☐ M_NEG	Performs a negation operation on an image.
☐ M_NOT	Performs a bitwise NOT operation on an image.
☐ M_PASS	Copies the source buffer to the destination image buffer.

Remarks

- This function is optimized for packed binary buffers.
- When performing a division operation (for example, using [M_DIV](#) or [M_DIV + M_FIXED_POINT](#)), dividing a value by 0 will generate a destination pixel with an unpredictable value. However, an error is not generated.
- Logical operations ([M_OR](#), [M_NOR](#), [M_OR_CONST](#), [M_NOR_CONST](#), [M_NOT](#), [M_AND](#), [M_NAND](#), [M_AND_CONST](#), [M_NAND_CONST](#), [M_XOR](#), [M_NOR](#), [M_XOR_CONST](#), [M_XNOR_CONST](#)) cannot be performed on a floating-point buffer.
- In-place processing is supported (source equals destination), but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimArithMultiple

Synopsis

Perform a point-to-point arithmetic operation using multiple source images.

Syntax

```
void MimArithMultiple(
    MIL_DOUBLE Src1ImageBufId,
    MIL_DOUBLE Src2ImageBufIdOrConst,
    MIL_DOUBLE Src3ImageBufIdOrConst,
    MIL_DOUBLE Src4Const,
    MIL_DOUBLE Src5Const,
    MIL_ID DestImageBufId,
    MIL_INT Operation,
    MIL_INT OperationFlag
)
```

Description

This function performs the specified point-to-point operation using multiple images, images and constants, or constants, storing results in the specified destination image buffer. Note that this function does not take 1-bit buffers in any of its parameters.

Parameters

Src1ImageBufId

Specifies the data source of the first operand; this parameter must be given an image buffer identifier.

See the [Parameter associations](#) section for possible values.

Src2ImageBufIdOrConst

Specifies the data source of the second operand; this parameter can be given an image buffer identifier or a constant.

See the [Parameter associations](#) section for possible values.

Src3ImageBufIdOrConst

Specifies the data source of the third operand; this parameter can be given an image buffer identifier or a constant.

See the [Parameter associations](#) section for possible values.

Src4Const

Specifies the data source of the fourth operand; this parameter must be given a constant.

See the [Parameter associations](#) section for possible values.

Src5Const

Specifies the data source of the fifth operand; this parameter must be given a constant.

See the [Parameter associations](#) section for possible values.

DestImageBufId

Specifies the identifier of the destination of the results; this parameter must be given an image buffer identifier.

Operation

Specifies the operation to perform.

See the [Parameter associations](#) section for possible values.

OperationFlag

Must be set to **M_DEFAULT**.

Parameter associations

Possible values for the **Src1ImageBufId**, **Src2ImageBufIdOrConst**, **Src3ImageBufIdOrConst**, **Src4Const**, **Src5Const**, and **Operation** parameters are described in the table below.

- [For the Operation, Src1ImageBufId, Src2ImageBufId, Src3ImageBufId, Src4ImageBufId, and Src5ImageBufId parameters](#)

Note that any unused parameters should be set to **M_NULL**.

For the Operation, Src1ImageBufId, Src2ImageBufId, Src3ImageBufId, Src4ImageBufId, and Src5ImageBufId parameters

Operation	Description
<div>Src1ImageBufId</div> <div>Src2ImageBufIdOrConst</div> <div>Src3ImageBufIdOrConst</div> <div>Src4Const</div> <div>Src5Const</div>	
M_MULTIPLY_ACCUMULATE_1 +	<p>Performs a point-to-point multiply and accumulate 1 operation using the following equation:</p> $\text{DestImage} = \frac{(\text{Src1Image} * \text{Src2Const}) + \text{Src3Const}}{\text{Src4Const}}$ <p>(summarize)</p>
<div>Src1ImageBufId</div>	<p>Specifies the image buffer identifier required for the first operand.</p> <p>(summarize)</p>
<div>Src2ImageBufIdOrConst</div>	<p>Specifies the constant required for the second operand.</p> <p>Note that the constant is cast to the same type as the buffer specified for first operand.</p> <p>(summarize)</p>
<div>Src3ImageBufIdOrConst</div>	<p>Specifies the constant required for the third operand.</p> <p>Note that the constant is cast to the same type as the buffer specified for first operand.</p> <p>(summarize)</p>
<div>Src4Const</div>	<p>Specifies the constant required for the fourth operand; this constant must be a power of 2.</p> <p>(summarize)</p>
M_MULTIPLY_ACCUMULATE_2 +	<p>Performs a point-to-point multiply and accumulate 2 operation using the following equation:</p> $\text{DestImage} = \frac{(\text{Src1Image} * \text{Src2Const}) + (\text{Src3Image} * \text{Src4Const})}{\text{Src5Const}}$ <p>(summarize)</p>
<div>Src1ImageBufId</div>	<p>Specifies the image buffer identifier required for the first operand.</p> <p>(summarize)</p>
<div>Src2ImageBufIdOrConst</div>	<p>Specifies the constant required for the second operand.</p>

		Note that the constant is cast to the same type as the buffer specified for first operand. (summarize)
<input type="checkbox"/> Src3ImageBufIdOrConst		Specifies the image buffer identifier required for the third operand. Note that the image buffer's type must either match that of the first operand, or it can be float. (summarize)
<input type="checkbox"/> Src4Const		Specifies the constant required for the fourth operand. Note that the constant is cast to the same type as the buffer specified for first operand. (summarize)
<input type="checkbox"/> Src5Const		Specifies the constant required for the fifth operand; this constant must be a power of 2. (summarize)
<input type="checkbox"/> M_OFFSET_GAIN +		Performs a per-pixel gain and offset correction operation using the following equation: $\text{DestImage} = \frac{(\text{Src1Image} - \text{Src2Image}) * \text{Src3Image}}{\text{Src4Const}}$ (summarize)
<input type="checkbox"/> Src1ImageBufId		Specifies the image buffer identifier required for the first operand. (summarize)
<input type="checkbox"/> Src2ImageBufIdOrConst		Specifies the image buffer identifier required for the second operand. Note that the image buffer's type must either match that of the first operand, or it can be float. (summarize)
<input type="checkbox"/> Src3ImageBufIdOrConst		Specifies the image buffer identifier required for the third operand. Note that the image buffer's type must either match that of the first operand, or it can be float. (summarize)
<input type="checkbox"/> Src4Const		Specifies the constant required for the fourth operand; this constant must be a power of 2. (summarize)
<input type="checkbox"/> M_WEIGHTED_AVERAGE +		Performs a weighted average operation using the following equation: $\text{DestImage} = \left(\frac{1}{\text{Src2Const}} \right) * \text{Src1Image} + \left(1 - \frac{1}{\text{Src2Const}} \right) * \text{Src3Image}$ (summarize)
<input type="checkbox"/> Src1ImageBufId		Specifies the image buffer identifier required for the first operand. (summarize)
<input type="checkbox"/> Src2ImageBufIdOrConst		Specifies the constant required for the second operand; this constant must be a power of 2. (summarize)
<input type="checkbox"/> Src3ImageBufIdOrConst		Specifies the image buffer identifier required for the third operand. Note that the image buffer's type must either match that of the first operand, or it can be float. (summarize)

Combination constant for the values listed in [For the Operation, Src1ImageBufId, Src2ImageBufId, Src3ImageBufId, Src4ImageBufId, and Src5ImageBufId parameters](#)

You can add the following value to the above-mentioned value to specify that the operation result should be saturated if necessary.

For the Operation parameter	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SATURATION	Forces the operation to saturate any resulting pixel values that overflow or underflow the possible range of the destination buffer. The pixel values are clipped to fit within the buffer's range, rather than wrapped around the range.

	If you do not specify M_SATURATION and there is an overflow or underflow, the resulting pixel values are either clipped (saturated) or wrapped around the buffer's range (not saturated), depending on which is faster for your hardware. (summarize)
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Remarks

- In-place processing is supported (source equals destination), but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).
- Note that if one of the buffers is float, the operation will be performed in float.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimBinarize

Synopsis

Perform a point-to-point binary thresholding operation.

Syntax

```
MIL_INT MimBinarize(  
    MIL_ID SrcImageBufId,  
    MIL_ID DestImageBufId,  
    MIL_INT Condition,  
    MIL_DOUBLE CondLow,  
    MIL_DOUBLE CondHigh  
)
```

Description

This function performs binary thresholding on the specified image. Each pixel that meets the specified condition is set to the highest unsigned destination buffer value, while other pixels are set to 0. If a floating-point destination buffer is specified, pixels that meet the condition are set to 1. For example, the highest buffer value for an 8-bit buffer is 0xff (regardless if the source buffer is signed).

MIL can automatically determine only one threshold value from the source image's histogram. In this case, if the histogram contains only two peaks, the threshold value is set to the minimum value between these peaks. If the histogram contains more than two peaks, then the threshold value will typically be set between the two principal peaks, though exceptions exist for pure black and full saturation (0, 255).

When the condition has two limits, the automatically calculated threshold value will be assigned to the parameter ([CondHigh](#) or [CondLow](#)) that is set to [M_DEFAULT](#). However, if the calculated value causes [CondHigh](#) to be less than [CondLow](#), then the user-specified value will be assigned to both parameters. If the [CondLow](#) and [CondHigh](#) parameters are both set to [M_DEFAULT](#), the automatically calculated threshold value will be assigned to both parameters. When the condition has only one limit, [CondHigh](#) is ignored.

If you want this function to return the automatically calculated threshold value, without binarizing the source image, set the [DestImageBufId](#) parameter to [M_NULL](#) and set the [CondLow](#) or [CondHigh](#) parameters to [M_DEFAULT](#).

Parameters

- SrcImageBufId
- Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.
- DestImageBufId
- Specifies the identifier of the destination of the results. In general, this parameter must be given an image buffer identifier.
- Condition
- Specifies the thresholding condition. This parameter must be set to one of the values below.
- The following are conditions that use two limits ([CondLow](#) and [CondHigh](#)):

● For specifying the threshold condition	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN_RANGE	Sets pixels with values between CondLow and CondHigh , inclusive, to the highest buffer value. Other pixels are set to 0. If a floating-point destination buffer is specified, pixels that meet the condition are set to 1. (summarize)
<input type="checkbox"/> M_OUT_RANGE	Sets pixels with values less than CondLow , or greater than CondHigh , to the highest buffer value. Other pixels are set to 0. If a floating-point destination buffer is specified, pixels that meet the condition are set to 1. (summarize)

The following are conditions that use one limit ([CondLow](#)):

● For conditions that use one limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_GREATER_OR_EQUAL .
<input type="checkbox"/> M_EQUAL	Sets pixel values equal to CondLow to the highest buffer value possible, while other pixels are set to zero. If a floating-point destination buffer is specified, pixels that meet the condition are set to 1. (summarize)
<input type="checkbox"/> M_GREATER	Sets pixel values greater than CondLow to the highest buffer value possible, while other pixels are set to zero. If a floating-point destination buffer is specified, pixels that meet the condition are set to 1. (summarize)
<input type="checkbox"/> M_GREATER_OR_EQUAL	Sets pixel values greater than or equal to CondLow to the highest buffer value possible, while other pixels are set to zero. If a floating-point destination buffer is specified, pixels that meet the condition are set to 1. (summarize)
<input type="checkbox"/> M_LESS	Sets pixel values less than CondLow to the highest buffer value possible, while other pixels are set to zero. If a floating-point destination buffer is specified, pixels that meet the condition are set to 1. (summarize)
<input type="checkbox"/> M_LESS_OR_EQUAL	Sets pixel values less than or equal to CondLow to the highest buffer value possible, while other pixels are set to zero. If a floating-point destination buffer is specified, pixels that meet the condition are set to 1. (summarize)
<input type="checkbox"/> M_NOT_EQUAL	Sets pixel values not equal to CondLow to the highest buffer value possible, while other pixels are set to zero. If a floating-point destination buffer is specified, pixels that meet the condition are set to 1. (summarize)

CondLow

Specifies the lower limit of the selected condition. This parameter must be set to one of the following values:

● For specifying the lower limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the parameter will be determined from the image's histogram.
<input type="checkbox"/> Value	Specifies the lower limit. Note that the value for CondLow is cast to the source buffer's data type and depth. If the source buffer is binary, the value must be equal to 0 or 1. (summarize)

CondHigh

Specifies the upper limit of the selected condition. This parameter must be set to one of the following values:

● For specifying the upper limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the parameter will be determined from the image's histogram. When the condition has one limit, only CondLow is used and this setting does not apply. (summarize)
<input type="checkbox"/> M_NULL	Specifies that this setting does not apply. This setting can be used when the condition uses only one limit. (summarize)
<input type="checkbox"/> Value	Specifies the upper limit.

Note that the value for `CondHigh` is cast to the source buffer's data type and depth. If the source buffer is binary, the value must be equal to 0 or 1.
[\(summarize\)](#)

Return value

When `DestImageBufId` is given an image buffer identifier, this function returns the parameter setting of `CondLow` (`M_DEFAULT` or the specified value). When `DestImageBufId` is set to `M_NULL`, this function returns the low threshold value (automatically calculated or specified).

Remarks

- This function is optimized for packed binary buffers.
- In-place processing is supported (source equals destination), but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include <code>mil.h</code> .
Library	Use <code>mil.lib</code> ; <code>milim.lib</code> .
DLL	Requires <code>mil.dll</code> ; <code>milim.dll</code> .

MimClip

Synopsis

Perform a point-to-point clipping operation.

Syntax

```
void MimClip(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT Condition,
    MIL_DOUBLE CondLow,
    MIL_DOUBLE CondHigh,
    MIL_DOUBLE WriteLow,
    MIL_DOUBLE WriteHigh
)
```

Description

This function clips each image pixel that meets the specified condition. If the condition has one clipping point, each pixel that satisfies this condition is replaced with the specified WriteLow value. If it has two clipping points, they are either replaced with the WriteLow or WriteHigh value depending on the condition. Pixels that do not satisfy the condition are not affected.

Parameters

SrcImageBufId

Specifies the identifier of the image data source.

DestImageBufId

Specifies the identifier of the destination image buffer.

Condition

Specifies the clipping condition. This parameter must be set to one of the values below.

The following are conditions that use two clipping points:

For clipping	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN_RANGE	Replaces pixel values in the range of CondLow and CondHigh, inclusive, with WriteLow.
<input type="checkbox"/> M_OUT_RANGE	Replaces pixel values less than CondLow with WriteLow and those greater than CondHigh with WriteHigh.

The following are conditions that use one clipping point:

For specifying conditions that use one clipping point	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EQUAL	Replaces pixel values equal to CondLow with WriteLow.
<input type="checkbox"/> M_GREATER	Replaces pixel values greater than CondLow with WriteLow.
<input type="checkbox"/> M_GREATER_OR_EQUAL	Replaces pixel values greater than or equal to CondLow with WriteLow.

<input type="checkbox"/> M_LESS	Replaces pixel values less than CondLow with WriteLow .
<input type="checkbox"/> M_LESS_OR_EQUAL	Replaces pixel values less than or equal to CondLow with WriteLow .
<input type="checkbox"/> M_NOT_EQUAL	Replaces pixel values not equal to CondLow with WriteLow .

CondLow

Specifies the lower clipping point of the selected condition. If the condition uses only one limit, set the [CondLow](#) parameter to the required limit and set [CondHigh](#) to [M_NULL](#).

This parameter is cast to the source buffer's data type and interpreted accordingly.

● For specifying the lower clipping point	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> Value	Specifies the lower limit. Note, if the source buffer is binary, CondLow must be equal to 0 or 1. (summarize)

CondHigh

Specifies the upper clipping point of the selected condition.

This parameter is cast to the source buffer's data type and interpreted accordingly.

● For specifying the upper clipping point	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> Value	Specifies the upper limit. Note, if the source buffer is binary, CondLow and CondHigh must be equal to 0 or 1. (summarize)

WriteLow

Specifies the values to write to the destination buffer when a pixel satisfies the specified low clipping condition. The condition determines whether [WriteLow](#) is written. This parameter is cast to the destination buffer's data type.

● For specifying the value to write to the destination buffer when a pixel satisfies the specified low clipping point	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> Value	Specifies the values to write when the pixel satisfies the low clipping condition. Note, if the destination buffer is binary, WriteLow must be equal to 0 or 1. (summarize)

WriteHigh

Specifies the values to write to the destination buffer when a pixel satisfies the specified high clipping condition. The condition determines whether [WriteHigh](#) is written. This parameter is cast to the destination buffer's data type.

● For specifying the value to write to the destination buffer when a pixel satisfies the specified high clipping point	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NULL	Must be set to this value when the condition uses only one limit.
<input type="checkbox"/> Value	Specifies the values to write when the pixel satisfies the high clipping condition. Note, if the destination buffer is binary, WriteHigh must be equal to 0 or 1. (summarize)

Remarks

- This function is optimized for packed binary buffers.
- In-place processing is supported (source equals destination), but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimClose

Synopsis

Perform a binary or grayscale closing-type morphological operation.

Syntax

```
void MimClose(  
    MIL_ID SrcImageBufId,  
    MIL_ID DestImageBufId,  
    MIL_INT NbIteration,  
    MIL_INT ProcMode  
)
```

Description

This function performs a binary or grayscale closing operation on the given source image for the specified number of iterations. A closing is a dilation followed by an erosion.

In binary mode, this function uses a 3 x 3 full rectangular structuring element; in grayscale mode, a 3 x 3 empty one.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the resulting image. This parameter must be given an image buffer identifier.

NbIteration

Specifies the number of times to iterate the operation.

When this parameter is set to 0 and [ProcMode](#) is set to [M_BINARY](#), the source image is binarized and the result is copied into the destination image buffer.

When this parameter is set to 0 and [ProcMode](#) is set to [M_GRAYSCALE](#), the source image is copied into the destination image buffer.

ProcMode

Specifies the processing mode to use. This parameter can be set to the following:

● For specifying the processing mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BINARY	Treats non-zero pixels as ones (1) during processing. The resulting non-zero pixels will have all bits set to one. (summarize)
<input type="checkbox"/> M_GRAYSCALE	Uses the source image's gray values for processing. The resulting buffer will also contain gray values. (summarize)

Remarks

- This function is optimized for packed binary buffers.
- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimConnectMap

Synopsis

Perform a 3 by 3 binary connectivity mapping.

Syntax

```
void MimConnectMap(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_ID LutBufId
)
```

Description

This function performs a 3 x 3 binary connectivity mapping. It calculates a connectivity code for each pixel in the source image, then maps the codes through the specified LUT buffer.

Parameters

SrcImageBufId

Specifies the identifier of the source of the operation. This parameter must be given an image buffer identifier. The source pixels are treated as binary (that is, all non-zero pixels are treated as 1). Pixel connectivity codes are determined in the following order:

$$\begin{bmatrix} n_3 & n_2 & n_1 \\ n_4 & n_8 & n_0 \\ n_5 & n_6 & n_7 \end{bmatrix} \text{ where } n_i \text{ is either 0 or 1}$$

$$Connectivity\ code = \sum_{i=0}^8 2^i n_i.$$

$$Result = LUTMAP[Connectivity\ code] .$$

DestImageBufId

Specifies the identifier of the destination of the results. This parameter must be given an image buffer identifier.

LutBufId

Specifies the identifier of the LUT buffer. As each connectivity code has 9 bits, you should supply a LUT buffer with at least 512 (2⁹) entries; otherwise unpredictable results can occur.

Remarks

- This function is optimized for packed binary buffers.
- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimControl

Synopsis

Control an image processing context setting.

Syntax

```
void MimControl(
    MIL_ID ContextId,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function sets the specified control for an image processing context. All the control type settings can be inquired using [MimInquire\(\)](#).

Parameters

- ContextId
- Specifies the identifier of the image processing context. The image processing context must have been previously allocated on the system using [MimAlloc\(\)](#).
- ControlType
- Specifies the processing feature to control.
- See the [Parameter associations](#) section for possible values.
- ControlValue
- Specifies the value needed for the control.
- See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the table below.

- [For an M_DEINTERLACE_CONTEXT image processing context](#)

The following [ControlType](#) and [ControlValue](#) settings can be specified for an M_DEINTERLACE_CONTEXT image processing context, used in [MimDeinterlace\(\)](#) operations.

For an M_DEINTERLACE_CONTEXT image processing context	
ControlType	Description
ControlValue	
M_DEINTERLACE_TYPE	<p>Sets the deinterlacing algorithm to use. The chosen algorithm can either be applied to all the pixels in the source image or to the pixels that are part of an object in motion (adaptive version of the algorithm).</p> <p>To determine if a pixel is part of a moving object, the adaptive algorithm compares it with the pixel at the same location in neighboring frames. If the difference between the maximum and minimum pixel intensity exceeds a set threshold (M_MOTION_DETECT_THRESHOLD), then the pixel is considered to be part of a moving object. Otherwise, the pixel is considered to be part of the background. The deinterlacing algorithm is not applied to the background pixels. Instead, the background pixels in the output image will be formed by the corresponding pixels in the even or odd field.</p> <p>(summarize)</p>

<input type="checkbox"/> M_DEFAULT	Same as M_DISCARD .
<input type="checkbox"/> M_ADAPTIVE_AVERAGE	Applies the M_AVERAGE algorithm to the pixels that are considered to be part of a moving object and leaves the background pixels unchanged.
<input type="checkbox"/> M_ADAPTIVE_BOB	Applies the M_BOB algorithm to the pixels that are considered to be part of a moving object and leaves the background pixels unchanged.
<input type="checkbox"/> M_ADAPTIVE_DISCARD	Applies the M_DISCARD algorithm to the pixels that are considered to be part of a moving object and leaves the background pixels unchanged.
<input type="checkbox"/> M_AVERAGE	Performs the averaging algorithm. This algorithm is equivalent to performing the M_DISCARD algorithm twice, once using the first field in the frame and once using the second. The resulting two frames will then be averaged to form one deinterlaced output frame. (summarize)
<input type="checkbox"/> M_BOB	Performs the bob algorithm. This algorithm performs the M_DISCARD algorithm twice, once using the first field in the frame and once using the second. The result is two output frames. Therefore, the output frame rate is twice as high as the input frame rate. (summarize)
<input type="checkbox"/> M_DISCARD	Performs the discard algorithm. This algorithm takes one field from the source image and discards the other. The second field is then calculated from this field. Each row of the second field is obtained by averaging the two corresponding neighboring rows in the first field. For example, the first row of the second field is calculated from the average of the first and second rows of the first field. (summarize)
<input type="checkbox"/> M_DISCARD_FIELD	Sets the field to discard when using the M_DISCARD or M_ADAPTIVE_DISCARD algorithm. Note, in the averaging and bob algorithms, the discard algorithm is called twice; the first field is discarded on the first call and the second field is discarded on the second call. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_EVEN_FIELD .
<input type="checkbox"/> M_EVEN_FIELD	Discards the even field.
<input type="checkbox"/> M_ODD_FIELD	Discards the odd field.
<input type="checkbox"/> M_FIRST_FIELD	Sets the first field to be processed for each input frame and consequently sets the order of the output frames when using the M_BOB or M_ADAPTIVE_BOB algorithm. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_EVEN_FIELD .
<input type="checkbox"/> M_EVEN_FIELD	Specifies that the even field will be processed first.
<input type="checkbox"/> M_ODD_FIELD	Specifies that the odd field will be processed first.
<input type="checkbox"/> M_MOTION_DETECT_NUM_FRAMES	Sets the number of frames to use for comparison purposes to determine if a pixel is part of an object in motion. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 2. (summarize)
<input type="checkbox"/> Value > 1	Specifies the number of frames.
<input type="checkbox"/> M_MOTION_DETECT_OUTPUT	Sets whether the output images of MimDeinterlace() are deinterlaced images or images indicating which pixels are considered to be part of an object in motion (the internal motion detection mask). In the latter case, the pixel values are either 0, if they are part of a background object, or the maximum unsigned value (0xFF for an 8 bit image), if they are part of an object in motion. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	The output images are the deinterlaced images.
<input type="checkbox"/> M_ENABLE	The output images indicate the background pixels and the pixels in motion.
<input type="checkbox"/> M_MOTION_DETECT_REFERENCE_FRAME	Sets the index of the frame to process within the group of frames that are used for motion detection. This frame is used as the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_CENTER_FRAME .
<input type="checkbox"/> 0<=Value<M_MOTION_DETECT_NUM_FRAMES	Specifies the index of the frame relative to the first frame of the group. The first frame of the group has index 0. (summarize)
<input type="checkbox"/> M_CENTER_FRAME	Specifies that the center frame in the group is used as the reference frame.
<input type="checkbox"/> M_FIRST_FRAME	Specifies that the first frame in the group is used as the reference frame.
<input type="checkbox"/> M_LAST_FRAME	Specifies that the last frame in the group is used as the reference frame.

<input type="checkbox"/> M_MOTION_DETECT_THRESHOLD	Specifies the threshold value used to differentiate between pixels that are part of objects in motion and background pixels. Each pixel is compared with the pixel at the same location in neighboring frames. If the difference between the maximum and minimum pixel intensity exceeds the specified threshold, the pixel is considered part of a moving object. Otherwise, it is considered part of the background. (summarize)
<input type="checkbox"/> Value >= 0	Specifies the threshold.
<input type="checkbox"/> M_SOURCE_FIRST_IMAGE	Specifies the index of the input image used to generate the first deinterlaced image. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> Value >= 0	Specifies the index of the image in the source image array.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimConvert

Synopsis

Perform a color conversion.

Syntax

```
void MimConvert(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_ID ConversionType
)
```

Description

This function performs a color conversion on the source image and places the result in the destination buffer.

The values generated will be normalized to the possible range of values in the destination buffer. For example, the hue values from 0 to 360° will generate the values from 0 to 255 in an 8-bit destination buffer.

Note that the values in a signed source buffer must be positive. For a floating-point destination buffer, the values generated will be normalized between 0 and 1.

If necessary, you can perform a mathematically optimal color-to-grayscale conversion using `McolProject()` with `M_PRINCIPAL_COMPONENT_PROJECTION`.

Parameters

SrcImageBufId

Specifies the identifier of the source image buffer. Source buffer values must be positive. If specifying a floating-point buffer, the values of the buffer must be normalized between 0 and 1 before calling this function.

DestImageBufId

Specifies the identifier of the destination image buffer. If specifying a floating-point buffer, the values generated will be normalized between 0 and 1. This normalization does not apply to the `MIL array buffer identifier` conversion type.

ConversionType

Specifies the type of conversion to perform. The conversion can be specified with either a predefined or custom transformation matrix. This parameter can be set to one of the following.

● For specifying the type of conversion to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_HSL_TO_RGB	Specifies a HSL to RGB conversion. Both the source and destination image buffers must be 3-band image buffers. (summarize)
<input type="checkbox"/> M_L_TO_RGB	Specifies an L (luminance) to RGB conversion. The luminance (intensity) is repeated in each color band of the destination buffer, creating a monochromatic (gray) RGB buffer. The source buffer must be a 1-band image buffer and the destination buffer must be a 3-band image buffer. (summarize)
<input type="checkbox"/> M_RGB_TO_H	Specifies an RGB to H conversion, where H represents the hue of the HSL color space. The source image must be a 3-band image buffer and the destination buffer must be a 1-band image buffer. (summarize)
<input type="checkbox"/> M_RGB_TO_HSL	Specifies an RGB to HSL (hue, luminance (intensity), and saturation) conversion. Both the source and destination image buffers must be 3-band image buffers. The hue component (H) is stored in band 0, the luminance component (L) is stored in band 2, and the saturation component (S) is stored in band 1. (summarize)
<input type="checkbox"/> M_RGB_TO_L	Specifies an RGB to L conversion, where L represents the luminance (intensity) of the HSL color space. The source image must be a 3-band image buffer and the destination buffer must be a 1-band image buffer.

	Note that M_RGB_TO_L and M_RGB_TO_Y produce similar results but not identical results because they are calculated to different color spaces. (summarize)
<input type="checkbox"/> M_RGB_TO_Y	Specifies an RGB to Y conversion, where Y represents the luminance of the YUV color space. The source buffer must have 3 bands. The destination buffer can have 1 or 3 bands. If it has 3 bands, only the first band is overwritten. Note that M_RGB_TO_L and M_RGB_TO_Y produce similar results but not identical results because they are calculated to different color spaces. (summarize)
<input type="checkbox"/> MIL array buffer identifier	Specifies a general matrix multiplication transform will be used to modify the source image's color information. The matrix multiplication is performed using the provided 1-band, 32-bit floating-point buffer allocated using MbufAlloc2d() with M_ARRAY . You should use MbufPut() to define the values of the matrix. The source image buffer must be a 3-band buffer. The computation is shown below. M_ARRAY buffer Source pixel Destination pixel $\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \cdot \begin{bmatrix} Src_{band0} \\ Src_{band1} \\ Src_{band2} \end{bmatrix} = \begin{bmatrix} Dest_{band0} \\ Dest_{band1} \\ Dest_{band2} \end{bmatrix}$ $\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \cdot \begin{bmatrix} Src_{band0} \\ Src_{band1} \\ Src_{band2} \\ 1 \end{bmatrix} = \begin{bmatrix} Dest_{band0} \\ Dest_{band1} \\ Dest_{band2} \end{bmatrix}$ $\begin{bmatrix} a & b & c \end{bmatrix} \cdot \begin{bmatrix} Src_{band0} \\ Src_{band1} \\ Src_{band2} \end{bmatrix} = \begin{bmatrix} Dest_{band0} \end{bmatrix}$ $\begin{bmatrix} a & b & c & d \end{bmatrix} \cdot \begin{bmatrix} Src_{band0} \\ Src_{band1} \\ Src_{band2} \\ 1 \end{bmatrix} = \begin{bmatrix} Dest_{band0} \end{bmatrix}$ When the provided array buffer is a 3X3 matrix or a 4X3 matrix (with optional offset values), the destination should be a 3-band buffer. When the provided array buffer is a 3X1 matrix or a 4X1 matrix (with an optional offset value), the destination should be a 1-band buffer. The final values are clipped (saturated) according to the destination buffer's bit depth. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimConvolve

Synopsis

Perform a general convolution operation.

Syntax

```
void MimConvolve(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_ID KernelBufId
)
```

Description

This function performs a general convolution operation on the source buffer using the specified filter, storing results in the specified destination buffer. This function supports two types of filters: Finite Impulse Response (FIR) filters and Infinite Impulse Response (IIR) filters. FIR filters operate on a finite neighborhood, while IIR filters take in account all values in an image. You can specify either a predefined or a custom FIR or IIR filter.

This function provides a number of predefined FIR. For predefined FIR filters, the overscan is set to transparent overscan pixel values, saturation is enabled for results, and the kernel's default center pixel is set as the top-left pixel of the central elements in a neighborhood. When using predefined FIR filters, you cannot change operation control settings except for the overscan.

This function provides two types of predefined IIR filters. Predefined IIR filters, [M_SHEN_FILTER\(\)](#) and [M_DERICHE_FILTER\(\)](#), use a recursive implementation, and the overscan pixel values are ignored. The normalization factor is automatically set so that it is possible to use the full data range of the destination buffer without overflows, and the result is given in the number of bits of the destination buffer. When using predefined IIR filters, you cannot change operation control settings except for the smoothness value. To use an IIR filter in kernel mode, you must use a custom IIR filter.

This function also accepts a custom FIR or IIR filter with which you can customize operation control settings. To define a custom filter, you must allocate a kernel buffer using [MbufAlloc1d\(\)](#) or [MbufAlloc2d\(\)](#). Note that the kernel buffer size can be constrained by the available resources.

To define a custom FIR filter, you must load the kernel buffer with values, using [MbufPut\(\)](#). To define a custom IIR filter, use [MbufControlNeighborhood\(\)](#) with [M_FILTER_TYPE](#), [M_FILTER_OPERATION](#), and [M_FILTER_MODE](#); the kernel values are automatically calculated based on these settings.

When using custom FIR or IIR filters, you can change default operation control settings using [MbufControlNeighborhood\(\)](#).

For predefined and custom FIR filters, when using 32-bit integer buffers and saturation is disabled, the accumulator buffer is a 32-bit integer buffer (MIL_INT); whereas, when saturation is enabled, the accumulator buffer is a 64-bit floating-point internal buffer (double). For IIR filters, when using 32-bit floating-point source or destination images, operations are performed using floating-point precision; whereas, for other types of images, operations are performed using fixed-point precision.

For custom FIR filters, this function will internally separate a large kernel into two 1-dimensional kernels, if possible, to increase the speed of the convolution operation. For more information on separating kernels, see the [Finite Impulse Response \(FIR\) filters](#) subsection in the [Custom spatial filters](#) section in [Chapter 4: Advanced image processing](#).

If the source and destination are multi-band buffers, the same filter is applied to every band of the source buffer.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.

Note that for a FIR filter or custom IIR filter which uses a kernel mode implementation, the size of the image buffer must be greater or equal to the size of the kernel.

DestImageBufId

Specifies the identifier of the destination of the results. This parameter must be given an image buffer identifier.

Note that for a FIR filter or custom IIR filter which uses a kernel mode implementation, the size of the image buffer must be greater or equal to the size of the kernel.

KernelBufId

Specifies the filter to use. Set this parameter to a predefined filter or the identifier of the kernel buffer which defines the custom filter.

To specify a predefined FIR filter, set this parameter to one of the following:

● For specifying a predefined FIR filter	
☐ Value	Description
☐ M_EDGE_DETECT +	<p>Computes the gradient of the image using the following operation, which is based on a Sobel filter.</p> $\left(\left[\begin{array}{ccc} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{array} \right] + \left[\begin{array}{ccc} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{array} \right] \right) / 2$ <p>(summarize)</p>
☐ M_EDGE_DETECT2 +	<p>Computes the gradient of the image using the following operation, which is based on a Prewitt filter.</p> $\left(\left[\begin{array}{ccc} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{array} \right] + \left[\begin{array}{ccc} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{array} \right] \right) / 2$ <p>(summarize)</p>
☐ M_HORIZ_EDGE +	<p>Computes the absolute value of the horizontal derivative of the image using the following horizontal edge detection filter.</p> $\left[\begin{array}{ccc} 2 & 2 & 2 \\ 0 & 0 & 0 \\ -2 & -2 & -2 \end{array} \right]$ <p>(summarize)</p>
☐ M_LAPLACIAN_EDGE +	<p>Computes the Laplacian values of the image using the following 4-connected Laplacian edge detection filter.</p> $\left[\begin{array}{ccc} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{array} \right]$ <p>(summarize)</p>
☐ M_LAPLACIAN_EDGE2 +	<p>Computes the Laplacian values of the image using the following 8-connected Laplacian edge detection filter.</p> $\left[\begin{array}{ccc} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{array} \right]$ <p>(summarize)</p>
☐ M_SHARPEN +	<p>Performs a sharpening operation on the image using the following 8-connected Laplacian filter combined with the original image. Note that this operation results in a stronger sharpening effect than M_SHARPEN2.</p> $\left[\begin{array}{ccc} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{array} \right]$ <p>(summarize)</p>
☐ M_SHARPEN2 +	<p>Performs a sharpening operation on the image using the following 4-connected Laplacian filter combined with the original image. Note that this operation results in a sharpening effect that is not as strong as the one achieved with M_SHARPEN.</p>

	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ <p>(summarize)</p>
<input type="checkbox"/> M_SMOOTH +	<p>Performs a smoothing operation on the image using the following smoothing filter.</p> $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} / 16$ <p>(summarize)</p>
<input type="checkbox"/> M_VERT_EDGE +	<p>Computes the absolute value of the vertical derivative of the image using the following vertical edge detection filter.</p> $\begin{bmatrix} -2 & 0 & 2 \\ -2 & 0 & 2 \\ -2 & 0 & 2 \end{bmatrix}$ <p>(summarize)</p>

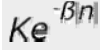
Combination constants for the values listed in [For specifying a predefined FIR filter](#)

You can add one of the following values to the above-mentioned values to specify how to determine the value of a destination pixel when its associated point falls outside the source buffer.

● For specifying how to determine the value of a destination pixel when its associated point falls outside the source buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_OVERSCAN_DISABLE	<p>Disables overscanning temporarily. You should disable overscanning to accelerate the convolution for applications in which the overscan data is not important in the resulting buffer.</p> <p>(summarize)</p>
<input type="checkbox"/> M_OVERSCAN_FAST	<p>Specifies that MIL will automatically select the overscan that optimizes speed, according to the specified operation and the target system. The overscan could be hardware-specific thereby having a different behavior than the other supported overscan modes.</p> <p>Note that when using M_OVERSCAN_FAST, the destination pixels in the overscan area are undefined. The pixels can therefore contain different values from one function call to the next, even if the function's parameter values are the same.</p> <p>(summarize)</p>

To specify a predefined IIR filter, set this parameter to one of the following:

● For specifying a predefined IIR filter									
<input type="checkbox"/> Value	Description								
<input type="checkbox"/> M_DERICHE_FILTER(MIL_ID <i>FilterOperation</i> , MIL_INT <i>FilterSmoothness</i>)	<p>Specifies a Canny-Deriche Infinite Support Exponential filter. This is an exponential weighting function, of the general form:</p> $k (a^{ n } + 1) e^{-a^{ n }}$ <p>For the Canny-Deriche filter, the neighborhoods' influence decreases much slower as the distance from the central pixel increases, compared to the Shen-Castan filter (see M_SHEN_FILTER() macro).</p> <p>Possible values for the OperationType and SmoothFactor parameters of M_DERICHE_FILTER() macro are described below.</p> <p>(summarize)</p> <table> <tr> <td colspan="2"><i>Parameters</i></td></tr> <tr> <td colspan="2"><i>FilterOperation</i></td></tr> <tr> <td colspan="2">This parameter specifies the type of operation to perform. You can set this parameter to one of the following:</td></tr> <tr> <td>M_DEFAULT</td><td>Same as M_SMOOTH.</td></tr> </table>	<i>Parameters</i>		<i>FilterOperation</i>		This parameter specifies the type of operation to perform. You can set this parameter to one of the following:		M_DEFAULT	Same as M_SMOOTH .
<i>Parameters</i>									
<i>FilterOperation</i>									
This parameter specifies the type of operation to perform. You can set this parameter to one of the following:									
M_DEFAULT	Same as M_SMOOTH .								

	M_SMOOTH	Performs a smoothing operation on the image using the Canny-Deriche filter.
	M_HORIZ_EDGE	Computes the absolute value of the horizontal derivative of the image using the Canny-Deriche filter.
	M_VERT_EDGE	Computes the absolute value of the vertical derivative of the image using the Canny-Deriche filter.
	M_EDGE_DETECT	Computes the gradient of the image using the Canny-Deriche filter.
	M_EDGE_DETECT_SQR	Computes the square of the gradient of the image using the Canny-Deriche filter.
	M_LAPLACIAN_EDGE	Computes the Laplacian values of the image using the Canny-Deriche filter.
	M_SHARPEN	Performs a sharpening operation on the image using the Canny-Deriche filter.
	M_FIRST_DERIVATIVE_X	Computes the first derivative of the image, with respect to X, using the Canny-Deriche filter.
	M_FIRST_DERIVATIVE_Y	Computes the first derivative of the image, with respect to Y, using the Canny-Deriche filter.
	M_SECOND_DERIVATIVE_X	Computes the second derivative of the image, with respect to X, using the Canny-Deriche filter.
	M_SECOND_DERIVATIVE_Y	Computes the second derivative of the image, with respect to Y, using the Canny-Deriche filter.
	M_SECOND_DERIVATIVE_XY	Computes the second derivative of the image, with respect to X and Y, using the Canny-Deriche filter.
	<i>FilterSmoothness</i>	
	This parameter specifies the degree of smoothness (strength of the denoising) applied by the convolution operation. You can set this parameter to one of the following:	
	M_DEFAULT	The default value is 50.0.
	0.0 to 100.0	Specifies the smoothness value. A value of 100.0 results in a strong noise reduction effect, while a value of 0.0 has almost no noise reduction effect.
<input type="checkbox"/> M_SHEN_FILTER(MIL_ID <i>FilterOperation</i> , MIL_INT <i>FilterSmoothness</i>) 	Specifies a Shen-Castan Infinite Support Exponential filter. This is an exponential weighting function, of the general form:	
	 <p>For the Shen-Castan filter, the neighborhoods' influence decreases much faster as the distance from the central pixel increases, compared to the Canny-Deriche filter (see M_DERICHE_FILTER() macro).</p> <p>Possible values for the OperationType and SmoothFactor parameters of M_SHEN_FILTER() macro are described below.</p> <p>(summarize)</p>	
	<i>Parameters</i>	
	<i>FilterOperation</i>	
	This parameter specifies the type of operation to perform. You can set this parameter to one of the following:	
	M_DEFAULT	Same as M_SMOOTH .
	M_SMOOTH	Performs a smoothing operation on the image using the Shen-Castan filter.
	M_HORIZ_EDGE	Computes the absolute value of the horizontal derivative of the image using the Shen-Castan filter.
	M_VERT_EDGE	Computes the absolute value of the vertical derivative of the image using the Shen-Castan filter.
	M_EDGE_DETECT	Computes the gradient of the image using the Shen-Castan filter.
	M_EDGE_DETECT_SQR	Computes the square of the gradient of the image using the Shen-Castan filter.
	M_LAPLACIAN_EDGE	Computes the Laplacian values of the image using the Shen-Castan filter.
	M_SHARPEN	Performs a sharpening operation on the image using the Shen-Castan filter.
	M_FIRST_DERIVATIVE_X	Computes the first derivative of the image, with respect to X, using the Shen-Castan filter.
	M_FIRST_DERIVATIVE_Y	Computes the first derivative of the image, with respect to Y, using the Shen-Castan filter.
	M_SECOND_DERIVATIVE_X	Computes the second derivative of the image, with respect to X, using the Shen-Castan filter.
	M_SECOND_DERIVATIVE_Y	Computes the second derivative of the image, with respect to Y, using the Shen-Castan filter.
	M_SECOND_DERIVATIVE_XY	Computes the second derivative of the image, with respect to X and Y, using the Shen-Castan filter.
	<i>FilterSmoothness</i>	

	This parameter specifies the degree of smoothness (strength of the denoising) applied by the convolution operation. You can set this parameter to one of the following:	
	M_DEFAULT	The default value is 50.0.
	0.0 to 100.0	Specifies the smoothness value. A value of 100.0 results in a strong noise reduction effect, while a value of 0.0 has almost no noise reduction effect.

Remark

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimCountDifference

Synopsis

Count the number of pixels that differ in each image.

Syntax

```
void MimCountDifference(
    MIL_ID Src1ImageBufId,
    MIL_ID Src2ImageBufId,
    MIL_ID ImResultId
)
```

Description

This function finds the number of differences between the two specified source buffers and stores the resulting number in the specified result buffer.

You can read the number of differences from the result buffer, using [MimGetResultId\(\)](#) or [MimGetResult\(\)](#), specifying [M_VALUE](#) as the result type.

Parameters

Src1ImageBufId

Specifies the identifier of the first image data source. This parameter can be given an image buffer identifier. The image buffer must be one-band.

Src2ImageBufId

Specifies the identifier of the second image data source. This parameter can only be given an image buffer identifier. The image buffer must be one-band.

ImResultId

Specifies the identifier of the buffer in which to store the differences. This parameter must be given the identifier of an image processing result buffer that was allocated with [MimAllocResult\(\)](#) and has an [M_COUNT_LIST](#) type. The buffer needs only one entry.

Remark

- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimDeinterlace

Synopsis

Produce a sequence of deinterlaced images from a sequence of images acquired from an interlaced camera.

Syntax

```
void MimDeinterlace(
    MIL_ID ContextId,
    const MIL_ID *SrcImageArrayPtr,
    const MIL_ID *DstImageArrayPtr,
    MIL_INT SrcImageCount,
    MIL_INT DstImageCount,
    MIL_INT ControlFlag
)
```

Description

This function produces a sequence of deinterlaced images from a sequence of images grabbed from an interlaced camera. When an image is acquired using an interlaced camera, the even and odd fields are not taken at the exact same moment in time. If the object in the image was in motion, there is an offset between the position of the object in one field and its position in the other. Simply combining these two fields to produce a deinterlaced image results in noticeable deformities in moving objects, called interlacing artifacts. **MimDeinterlace()** uses averaging techniques to reduce or remove these interlacing artifacts and produce a higher quality deinterlaced image. To perform the deinterlacing operation, you can select one of several deinterlacing algorithms using **MimControl()** with **M_DEINTERLACE_TYPE**. They can be applied to the entire image or only to the pixels that are considered part of a moving object. To apply the algorithm to the latter, select the adaptive version of the algorithm (for example, **M_ADAPTIVE_DISCARD**).

Often, the order in which the fields are grabbed will depend on the camera being used. When using an algorithm that produces two output frames (for example, **M_BOB** or **M_ADAPTIVE_BOB**), it is important to set the field that is grabbed first using **MimControl()** with **M_FIRST_FIELD**. This will ensure that the sequence of output frames occur in chronological order and that the video stream is fluid.

The adaptive algorithms apply a motion detection algorithm to dynamically determine which pixels are part of an object in motion and which pixels are part of the background, prior to performing the deinterlacing algorithm. Using **MimControl()**, you can set the number of neighboring frames used for motion detection (**M_MOTION_DETECT_NUM_FRAMES**), the location of the frame to be processed within this group of frames (**M_MOTION_DETECT_REFERENCE_FRAME**), and the threshold value used to differentiate between pixels of objects in motion and pixels of background objects (**M_MOTION_DETECT_THRESHOLD**). To visualize which pixels are affected, you can have **MimDeinterlace()** output an image of these pixels, using **MimControl()** with **M_MOTION_DETECT_OUTPUT**; pixels that are part of an object in motion are set to the maximum value (0xFF for an 8-bit image), while the other pixels are set to 0. In this case, the deinterlacing part of the adaptive algorithm is not executed.

Parameters

ContextId

Specifies the identifier of the image processing context to be used for deinterlacing. The image processing context must have been previously allocated on the system using **MimAlloc()**.

SrcImageArrayPtr

Specifies an array containing the identifiers of the buffers of the images to deinterlace. Only 8-bit and 16-bit buffers are supported. All source images must be of the same type, same format, and on the same system. Note that all source and destination images must have the same size.

DstImageArrayPtr

Specifies an array containing the identifiers of the buffers of the images in which to save the deinterlaced images. Only 8-bit and 16-bit buffers are supported. All destination images must be of the same type, same format, and on the same system. Note that all source and destination images must have the same size.

SrcImageCount

Specifies the number of images in the source sequence.

DstImageCount

Specifies the number of deinterlaced images in the destination sequence.

ControlFlag

Reserved for future use. Should be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimDilate

Synopsis

Perform a binary or grayscale dilation-type morphological operation.

Syntax

```
void MimDilate(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT NbIteration,
    MIL_INT ProcMode
)
```

Description

This function performs a binary or grayscale dilation on the given source image for the specified number of iterations.

In binary mode, this function uses a 3 x 3 full rectangular structuring element; in grayscale mode, a 3 x 3 empty one.

The overscan pixels are automatically set to the highest possible buffer value, which will produce the most accurate possible results for the image border pixels.

Parameters

- SrcImageBufId
- Specifies the identifier of the source image buffer.
- DestImageBufId
- Specifies the identifier of the destination image buffer.
- NbIteration
- Specifies the number of times to iterate the operation.
- When this parameter is set to 0 and ProcMode is set to M_BINARY, the source image is binarized and the result is copied into the destination image buffer.
- When this parameter is set to 0 and ProcMode is set to M_GRAYSCALE, the source image is copied into the destination image buffer.
- ProcMode
- Specifies the processing mode to use. This parameter can be set to the following:

For specifying the processing mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BINARY	Performs a binary dilation. In this case, non-zero pixels of the source image are treated as ones (1) during processing. The resulting non-zero pixels will have all bits set to one. (summarize)
<input type="checkbox"/> M_GRAYSCALE	Performs a grayscale dilation. In this case, each pixel of the source image is replaced with the maximum value in its neighborhood. (summarize)

Remarks

- This function is optimized for packed binary buffers.
- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimDistance

Synopsis

Perform a distance transformation.

Syntax

```
void MimDistance(  
    MIL_ID SrcImageBufId,  
    MIL_ID DestImageBufId,  
    MIL_INT DistanceTransform  
)
```

Description

This function determines the shortest distance between each blob pixel and the blob's background, and assigns this distance to the pixel. It produces a type of contour mapping of a blob.

Parameters

SrcImageBufId

Specifies the identifier of the source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the resulting image. This parameter must be given an image buffer identifier.

DistanceTransform

Specifies the way in which the minimum distance from blob pixel to background pixel is calculated. This parameter approximates the true distance from blob pixel to background pixel using a 3 x 3 distance matrix and can be set to one of the following:

For specifying the method used to calculate the minimum distance	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CHAMFER_3_4	<p>Determines the minimum distance using horizontal, vertical or diagonal pixel steps. Horizontal and vertical steps are counted as 3; diagonal steps are counted as 4. Then, the resulting distance is normalized by a factor of 3. This transform provides the best approximation to Euclidean distance.</p> <p>This transform requires that the destination buffer be deep enough to hold a number at least three times the maximum distance from a blob pixel to its edge. For example, an 8-bit buffer (255 max) can be used for a maximum distance of 85 pixels and a 16-bit buffer (65535 max) for a maximum distance of 21845 pixels.</p> <p>3x3 Distance Matrix:</p> <div>$\begin{bmatrix} 4 & 3 & 4 \\ 3 & 0 & 3 \\ 4 & 3 & 4 \end{bmatrix}$</div> <p>(summarize)</p>
<input type="checkbox"/> M_CHESSBOARD	<p>Determines the minimum distance using horizontal, vertical, or diagonal pixel steps. All steps count as 1.</p> <p>3x3 Distance Matrix:</p> <div>$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$</div> <p>(summarize)</p>
<input type="checkbox"/> M_CITY_BLOCK	<p>Determines the minimum distance using only horizontal or vertical pixel steps. Horizontal and vertical steps count as 1.</p>

3x3 Distance Matrix:
$$\begin{bmatrix} \infty & 1 & \infty \\ 1 & 0 & 1 \\ \infty & 1 & \infty \end{bmatrix}$$
[\(summarize\)](#)

Remark

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimDraw

Synopsis

Draw the results of an image processing operation.

Syntax

```
void MimDraw(
    MIL_ID GraphContId,
    MIL_ID Src1ImageOrResultId,
    MIL_ID Src2ImageId,
    MIL_ID DstImageId,
    MIL_INT Operation,
    MIL_DOUBLE Param1,
    MIL_DOUBLE Param2,
    MIL_INT ControlFlag
)
```

Description

The function draws the results of a MIL image processing operation into a destination image buffer.

Parameters

GraphContId

Specifies the graphics context to use. Only grayscale graphics contexts are supported. This parameter must be set to one of the following values:

● For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

Src1ImageOrResultId

Specifies the identifier of the primary source image buffer or the result buffer from which to extract the results to draw. The buffer must have been previously allocated on the required system, using **MbufAlloc...()** or [MimAllocResult\(\)](#).

Src2ImageId

Specifies the identifier of the secondary source image buffer for the drawing operation. The buffer must have been previously allocated on the required system using **MbufAlloc...()**.

This buffer is only required if the image processing operation, used to generate the results, required two buffers to store its results. Otherwise, if two buffers were not used, this parameter should be set to **M_NULL**.

DstImageId

Specifies the identifier of the destination image buffer in which to draw. This destination image buffer can be any supported MIL image buffer. However, when the source buffers are not results buffers, the destination image buffer must be greater than or equal in size and bit depth to the source image buffers.

Operation

Specifies the type of operation to perform.

See the [Parameter associations](#) section for possible values.

Param1

Specifies an attribute of the operation to perform.

See the [Parameter associations](#) section for possible values.

Param2

Specifies an attribute of the operation to perform.

See the [Parameter associations](#) section for possible values.

ControlFlag

Specifies the number of fractional bits in the source values, when they are in a fixed-point format.

For specifying the number of fractional bits

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_FIXED_POINT + 0.
<input type="checkbox"/> M_FIXED_POINT + n	Specifies the number of fractional bits, where n is any integer between 0 and 7, inclusive.

Parameter associations

Possible values for the **Operation**, **Param1**, and **Param2** parameters are described in the table below.

- [For the Operation parameter](#)

For the Operation parameter		
<input type="checkbox"/> Operation	Description	
<input type="checkbox"/> Param1		
<input type="checkbox"/> Param2		
<input type="checkbox"/> M_DRAW_PEAKS +	<p>Draws the position and/or intensity results of MimLocatePeak1d(). Only valid positions are drawn; if a peak intensity was not found for a row/column, nothing will be drawn for that row/column.</p> <p>You should set the Src1ImageOrResultId parameter to the position buffer and, if necessary, the Src2ImageId parameter to the intensity buffer.</p> <p>If you supply an intensity buffer, then the GraphContId parameter must be set to M_DEFAULT so that the intensity buffer defines the drawing color when drawing each position.</p> <p>(summarize)</p>	
<input type="checkbox"/> Param1	<p>Specifies the row in the source image buffers from which to read the results.</p> <p>(summarize)</p>	
<input type="checkbox"/> Param2	<p>Specifies the size of the drawn representation of valid positions. For M_DOTS, this parameter specifies the diameter of the dots that are drawn at valid positions; for M_LINES, this parameter specifies the width of the line segments that connect valid positions.</p> <p>(summarize)</p>	

Combination constants for the values listed in [For the Operation parameter](#)

You must add one of the following values to the above-mentioned value to specify the direction in which the peak detection was performed.

For specifying the direction in which the peak detection was performed

<input type="checkbox"/> Value	Description
--------------------------------	-------------

<input type="checkbox"/> M_1D_COLUMNS	Specifies that the peaks detection was performed in the vertical direction. This is the default value. (summarize)
<input type="checkbox"/> M_1D_ROWS	Specifies that the peaks detection was performed in the horizontal direction.

Combination constants for the values listed in [For the Operation parameter](#)

You must add one of the following values to the above-mentioned value to specify the style in which to draw.

● For specifying the the style in which to draw	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DOTS	Specifies to draw the valid result positions as dots in the destination buffer.
<input type="checkbox"/> M_LINES	Specifies to draw line segments between valid positions in the destination buffer. This is the default value. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimEdgeDetect

Synopsis

Perform a specific edge detection operation and produce a gradient intensity and/or gradient angle image.

Syntax

```
void MimEdgeDetect (
    MIL_ID SrcImageBufId,
    MIL_ID DestIntensityImageBufId,
    MIL_ID DestAngleImageBufId,
    MIL_ID KernelId,
    MIL_INT ControlFlag,
    MIL_INT Threshold
)
```

Description

This function performs an edge detection operation on the specified source image, using the specified filter. It produces a gradient intensity image and/or a gradient angle image in the specified image buffer(s). If one of the destination images is not required, specify **M_NULL** as its image buffer identifier.

For an unsigned destination buffer, the angle is returned from 0° to 360° (counter-clockwise) and mapped to the entire range of the destination buffer. For example, for an 8-bit unsigned buffer, the angles of 0°, 90°, 180°, and 270° map to 0, 64, 128, and 192, respectively. For a signed destination buffer, mapping is done in both the positive and negative range of the buffer and represents angle values from -180° to 180°. For example, for an 8-bit signed buffer, the angles of -180°, -90°, 0°, and 90° map to -128, -64, 0 and 64, respectively.

This function can also be performed on a floating-point buffer. However, the results are not mapped and are returned in the range of 0° to 360°. The degree of precision is equal to the precision of the floating-point buffer for regular computation or to +/- 0.4° for fast computation. The value for undefined results is the maximum positive value of the floating-point buffer.

For a gradient value lower than the threshold value, the angle is not computed (not considered a significant edge) and the resulting angle pixel is undetermined.

You can perform the operation using a combination of fast and full gradient and angle computations.

Type of computation	Description of computation
Full gradient computation	$\text{Gradient} = \sqrt{\text{GradientX}^2 + \text{GradientY}^2}$
Fast gradient computation	$\text{Gradient} = (\text{abs}(\text{GradientX}) + \text{abs}(\text{GradientY}))/2$
Full angle computation	$\text{Angle} = \arctan(\text{GradientY}/\text{GradientX})$

The fast angle approximation is based on the same equation as the full angle computation, however the value is determined from a LUT mapping. Note that the fast angle approximation introduces a maximum error of +/- 0.4°.

Parameters

SrcImageBufId

Specifies the identifier of the source of the operation. This parameter must be given an image buffer identifier.

DestIntensityImageBufId

Specifies the identifier of the destination buffer for the resulting gradient intensity image. This parameter must be given an image buffer identifier or **M_NULL**. Saturation is performed on this buffer.

DestAngleImageBufId

Specifies the identifier of the destination buffer for the resulting gradient angle image. This parameter must be given an image buffer identifier or **M_NULL**.

KernelId

Specifies the filter to use. This parameter must be set to the value below.

● For specifying the filter to use	
☐ Value	Description
☐ M_SOBEL +	<p>Sets the parameter to the predefined 3x3 edge detection filter. This filter is as follows:</p> $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ <p>(summarize)</p>

Combination constant for the values listed in [For specifying the filter to use](#)

You can add the following value to the above-mentioned value to specify that overscan processing is disabled.

● For M_SOBEL	
☐ Value	Description
☐ M_OVERSCAN_DISABLE	Disables overscan processing.

ControlFlag

Specifies how to perform the operations.

The [ControlFlag](#) parameter can be set to one of the following:

● For specifying how to perform the operation	
☐ Value	Description
☐ M_DEFAULT	Same as M_FAST_EDGE_DETECT .
☐ M_FAST_ANGLE + M_REGULAR_GRADIENT	Specifies a full gradient computation and a fast angle approximation.
☐ M_FAST_EDGE_DETECT	<p>Specifies a fast gradient computation and a fast angle approximation. The gradient is computed as an approximation, using the average of the absolute value of the two directional components (X and Y).</p> <p>(summarize)</p>
☐ M_REGULAR_ANGLE + M_FAST_GRADIENT	Specifies a fast gradient computation and a full angle computation.
☐ M_REGULAR_EDGE_DETECT	<p>Specifies a full gradient computation and a full angle computation. The gradient is computed as the square root of the sum of the square of each directional component (X and Y). This method of calculation is slower but more precise.</p> <p>(summarize)</p>

Threshold

Specifies the threshold value for calculating gradient intensity. To perform the full operation, set this parameter to zero or **M_NULL**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimErode

Synopsis

Perform a binary or grayscale erosion-type morphological operation.

Syntax

```
void MimErode(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT NbIteration,
    MIL_INT ProcMode
)
```

Description

This function performs a binary or grayscale erosion on the given source image for the specified number of iterations.

In binary mode, this function uses a 3 x 3 full rectangular structuring element; in grayscale mode, a 3 x 3 empty one.

The overscan pixels are automatically set to the highest possible buffer value, which will produce the most accurate possible results for the image border pixels.

Parameters

- SrcImageBufId
- Specifies the identifier of the source image buffer.
- DestImageBufId
- Specifies the identifier of the destination image buffer.
- NbIteration
- Specifies the number of times to iterate the operation.
- When this parameter is set to 0 and ProcMode is set to M_BINARY, the source image is binarized and the result is copied into the destination image buffer.
- When this parameter is set to 0 and ProcMode is set to M_GRAYSCALE, the source image is copied into the destination image buffer.
- ProcMode
- Specifies the processing mode to use. This parameter can be set to the following:

For specifying the processing mode	
Value	Description
<input type="checkbox"/> M_BINARY	Performs a binary erosion. In this case, non-zero pixels are treated as ones (1) during processing. The resulting non-zero pixels will have all bits set to one. (summarize)
<input type="checkbox"/> M_GRAYSCALE	Performs a grayscale erosion. In this case, each pixel of the source image is replaced with the minimum value in its neighborhood. (summarize)

Remarks

- This function is optimized for packed binary buffers.
- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimFindExtreme

Synopsis

Find an image buffer's extremes (minimum and/or maximum pixel values).

Syntax

```
void MimFindExtreme(
    MIL_ID SrcImageBufId,
    MIL_ID ExtremeImResultId,
    MIL_INT ExtremeType
)
```

Description

This function finds the maximum and/or minimum value of the specified source image and stores results in the specified extreme result buffer.

You can read the minimum and/or maximum from the result buffer, using [MimGetResultId\(\)](#) or [MimGetResult\(\)](#), specifying [M_VALUE](#) as the result type.

Parameters

- SrcImageBufId**
- Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier. The buffer must be one-band.
- ExtremeImResultId**
- Specifies the identifier of the buffer in which to store the extreme values. This parameter must be given the identifier of an image processing result buffer that was allocated with [MimAllocResult\(\)](#) and that has an [M_EXTREME_LIST](#) type. If just the maximum or minimum is calculated, only one entry is needed. If both the minimum and maximum are calculated, the result buffer must have two entries. The minimum value is stored in the first entry and the maximum value is stored in the second.
- ExtremeType**
- Specifies whether to find the minimum value, maximum value, or both. This parameter can be set to one of the values in the table below.

For specifying whether to find the minimum value, maximum value, or both	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MAX_ABS_VALUE	Finds the maximum value, as an absolute value.
<input type="checkbox"/> M_MAX_VALUE	Finds the maximum value.
<input type="checkbox"/> M_MIN_ABS_VALUE	Finds the minimum value, as an absolute value.
<input type="checkbox"/> M_MIN_ABS_VALUE + M_MAX_ABS_VALUE	Finds the minimum and maximum values, as absolute values.
<input type="checkbox"/> M_MIN_VALUE	Finds the minimum value.
<input type="checkbox"/> M_MIN_VALUE + M_MAX_VALUE	Finds the minimum and maximum values.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimFlip

Synopsis

Perform a horizontal or vertical image-flipping operation.

Syntax

```
void MimFlip(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT Operation,
    MIL_INT OpFlag
)
```

Description

This function flips the source image to the destination image according to the specified operation.

Parameters

- SrcImageBufId
- Specifies the identifier of the source image buffer.
- DestImageBufId
- Specifies the identifier of the destination image buffer.
- Operation
- Specifies the operation to perform. This parameter can be set to one of the following:

for specifying the operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FLIP_HORIZONTAL	Flips the image in a horizontal direction (left to right, along a vertical axis).
<input type="checkbox"/> M_FLIP_VERTICAL	Flips the image in a vertical direction (top to bottom, along a horizontal axis).

- OpFlag
- Must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimFree

Synopsis

Free an image processing context or result buffer.

Syntax

```
void MimFree(
    MIL_ID ImResultId
)
```

Description

This function frees the specified image processing context or result buffer, previously allocated with [MimAlloc\(\)](#) or [MimAllocResult\(\)](#), respectively. It should also be used if an image processing context was restored using [MimRestore\(\)](#) or [MimStream\(\)](#) (The control flag in [MimStream\(\)](#) should be set to M_RESTORE in this case).

All image processing contexts and result buffers allocated on a particular system must be freed before the system can be freed.

Parameter

ImResultId

Specifies the identifier of the image processing context or result buffer to free. These must have been successfully allocated (with [MimAlloc\(\)](#) or [MimAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimGetResult1d

Synopsis

Get the specified number of result entries from an image processing result buffer.

Syntax

```
void MimGetResult1d(
    MIL_ID ImResultId,
    MIL_INT OffEntry,
    MIL_INT NbEntries,
    MIL_INT ResultType,
    void *UserArrayPtr
)
```

Description

This function retrieves the specified number of result entries from the specified result buffer and stores them in the specified one-dimensional destination user array.

Note that if your target image has been associated with a calibration object, positional results are by default returned in real-world units. To have results returned in pixel units, use `McalControl()` with the `M_OUTPUT_UNITS` control type set to `M_PIXEL`. If the target image is not calibrated, positional results are returned in pixels, relative to the top-left pixel in the target image.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set `UserArrayPtr` to `M_NULL`. When this parameter is set to `M_NULL`, the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call `MimGetResult1d()` again and you pass an array to the parameter `UserArrayPtr`. You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

Parameters

- ImResultId

Specifies the identifier of the image processing result buffer from which to get results.
- OffEntry

Specifies the offset of the first result to read. The given value must be within the number of allocated result entries.
- NbEntries

Specifies the number of result entries to get, starting from the entry specified by the `OffEntry` parameter.
- ResultType

Specifies the type of result(s) to retrieve. This parameter can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the `UserArrayPtr` parameter the address of an array of type `MIL_DOUBLE` with a size equal to 1 .

• For specifying the type of result to retrieve	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MAX +	Retrieves the maximum pixel value from an <code>M_STAT_LIST</code> result buffer.
<input type="checkbox"/> M_MAX_ABS +	Retrieves the maximum absolute pixel value from an <code>M_STAT_LIST</code> result buffer.
<input type="checkbox"/> M_MEAN +	Retrieves the mean pixel value from an <code>M_STAT_LIST</code> result buffer.
<input type="checkbox"/> M_MIN +	Retrieves the minimum pixel value from an <code>M_STAT_LIST</code> result buffer.

<input type="checkbox"/> M_MIN_ABS +	Retrieves the minimum absolute pixel value from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_NB_EVENT +	Retrieves the number of events from an M_EVENT_LIST result buffer. (summarize)
	<i>UserArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: only one element is required. • Data type: array of type float Array size: only one element is required. Note: use if M_FLOAT was added to the ResultType parameter of MimAllocResult() during result buffer allocation.
<input type="checkbox"/> M_NUMBER +	Retrieves the number of pixels from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_POSITION_X +	Retrieves the image X-coordinate of each event from an M_EVENT_LIST result buffer. (summarize)
	<i>UserArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: the same number of elements as number of entries to obtain. • Data type: array of type float Array size: the same number of elements as number of entries to obtain. Note: use if M_FLOAT was added to the ResultType parameter of MimAllocResult() during result buffer allocation.
<input type="checkbox"/> M_POSITION_Y +	Retrieves the image Y-coordinate of each event from an M_EVENT_LIST result buffer. (summarize)
	<i>UserArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: the same number of elements as number of entries to obtain. • Data type: array of type float Array size: the same number of elements as number of entries to obtain. Note: use if M_FLOAT was added to the ResultType parameter of MimAllocResult() during result buffer allocation.
<input type="checkbox"/> M_STANDARD_DEVIATION +	Retrieves the pixel standard deviation value from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_SUM +	Retrieves the sum of the pixel values from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_SUM_ABS +	Retrieves the sum of the absolute pixel values from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_SUM_OF_SQUARES +	Retrieves the sum of the squared pixel values from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_VALUE +	Retrieves values from an M_HIST_LIST , M_PROJ_LIST , M_EXTREME_LIST , M_EVENT_LIST , or M_COUNT_LIST result buffer. If an M_EXTREME_LIST type result buffer contains both the minimum and maximum value of an image, the minimum is written first, followed by the maximum. (summarize)
	<i>UserArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: the same number of elements as number of entries to obtain (MimAllocResult()). • Data type: array of type float Array size: the same number of elements as number of entries to obtain (MimAllocResult()). Note: use when retrieving results from a M_PROJ_LIST , M_EXTREME_LIST , or M_EVENT_LIST result buffer if M_FLOAT was added to the ResultType parameter of MimAllocResult() during result buffer allocation.

Combination constants for any of the possible values of the [ResultType](#) parameter

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

● For the ResultType parameter to cast the required results to the requested data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_DOUBLE	Casts the requested results to a <i>double</i> . (summarize)
	<i>UserArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type double Array size: depends on the ResultType parameter, see above.

		<p>Note: When multiple results.</p> <ul style="list-style-type: none"> • Data type: double <p>Note: When a single result, see above.</p>
<input type="checkbox"/> M_TYPE_FLOAT	<p>Casts the requested results to a <i>float</i>.</p> <p>(summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type float Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: float Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_LONG	<p>Casts the requested results to a <i>long</i>.</p> <p>(summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type long Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: long Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>.</p> <p>(summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: MIL_DOUBLE Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_MIL_INT	<p>Casts the requested results to a <i>MIL_INT</i>.</p> <p>(summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: MIL_INT Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_MIL_INT32	<p>Casts the requested results to a <i>MIL_INT32</i>.</p> <p>(summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: MIL_INT32 Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_MIL_INT64	<p>Casts the requested results to a <i>MIL_INT64</i>.</p> <p>(summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT64 Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: MIL_INT64 Note: When a single result, see above.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type double
- array of type float

- array of type long
- array of type MIL_DOUBLE
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- double
- float
- long
- MIL_DOUBLE
- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address of the one-dimensional array in which to write the results read from the MIL result buffer.

You can set this parameter to **M_NULL** to put the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimGetResult

Synopsis

Get the specified type of result(s) from an image processing result buffer.

Syntax

```
void MimGetResult(
    MIL_ID ImResultId,
    MIL_INT ResultType,
    void *UserArrayPtr
)
```

Description

This function retrieves all the results of the specified type from the specified result buffer and stores them in the specified one-dimensional destination user array. Results are only available after calling [MimHistogram\(\)](#), [MimFindExtreme\(\)](#), [MimLocateEvent\(\)](#), [MimCountDifference\(\)](#), [MimProject\(\)](#), or [MimStat\(\)](#).

Note that if your target image has been associated with a calibration object, positional results are by default returned in real-world units. To have results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image is not calibrated, positional results are returned in pixels, relative to the top-left pixel in the target image.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [UserArrayPtr](#) to [M_NULL](#). When this parameter is set to [M_NULL](#), the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [MimGetResult\(\)](#) again and you pass an array to the parameter [UserArrayPtr](#). You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

Parameters

ImResultId

Specifies the identifier of the image processing result buffer from which to retrieve results.

ResultType

Specifies the type of result(s) to retrieve. This parameter can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the [UserArrayPtr](#) parameter the address of an array of type [MIL_DOUBLE](#) with a size equal to 1 .

● For a result buffer	
☐ Value	Description
☐ M_MAX +	Retrieves the maximum pixel value from an M_STAT_LIST result buffer.
☐ M_MAX_ABS +	Retrieves the maximum absolute pixel value from an M_STAT_LIST result buffer.
☐ M_MEAN +	Retrieves the mean pixel value from an M_STAT_LIST result buffer.
☐ M_MIN +	Retrieves the minimum pixel value from an M_STAT_LIST result buffer.
☐ M_MIN_ABS +	Retrieves the minimum absolute pixel value from an M_STAT_LIST result buffer.
☐ M_NB_EVENT +	Retrieves the number of events from an M_EVENT_LIST result buffer. (summarize)
	<div> <div>UserArrayPtr info</div> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: only one element is required. • Data type: array of type float Array size: only one element is required. </div>

	Note: use if M_FLOAT was added to the ResultType parameter of MimAllocResult() during result buffer allocation.
<input type="checkbox"/> M_NUMBER +	Retrieves the number of pixels from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_POSITION_X +	Retrieves the image X-coordinate of each event from an M_EVENT_LIST result buffer. (summarize) <i>UserArrayPtr info</i> <ul style="list-style-type: none">• Data type: array of type MIL_INT Array size: the same number of elements as number of entries to obtain (MimAllocResult()).• Data type: array of type float Array size: the same number of elements as number of entries to obtain (MimAllocResult()). Note: use if M_FLOAT was added to the ResultType parameter of MimAllocResult() during result buffer allocation.
<input type="checkbox"/> M_POSITION_Y +	Retrieves the image Y-coordinate of each event from an M_EVENT_LIST result buffer. (summarize) <i>UserArrayPtr info</i> <ul style="list-style-type: none">• Data type: array of type MIL_INT Array size: the same number of elements as number of entries to obtain (MimAllocResult()).• Data type: array of type float Array size: the same number of elements as number of entries to obtain (MimAllocResult()). Note: use if M_FLOAT was added to the ResultType parameter of MimAllocResult() during result buffer allocation.
<input type="checkbox"/> M_STANDARD_DEVIATION +	Retrieves the pixel standard deviation value from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_SUM +	Retrieves the sum of the pixel values from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_SUM_ABS +	Retrieves the sum of the absolute pixel values from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_SUM_OF_SQUARES +	Retrieves the sum of the squared pixel values from an M_STAT_LIST result buffer.
<input type="checkbox"/> M_VALUE +	Retrieves values from an M_HIST_LIST , M_PROJ_LIST , M_EXTREME_LIST , M_EVENT_LIST , or M_COUNT_LIST result buffer. If an M_EXTREME_LIST type result buffer contains both the minimum and maximum value of an image, the minimum is written first, followed by the maximum. (summarize) <i>UserArrayPtr info</i> <ul style="list-style-type: none">• Data type: array of type MIL_INT Array size: the same number of elements as number of entries to obtain (MimAllocResult()).• Data type: array of type float Array size: the same number of elements as number of entries to obtain (MimAllocResult()). Note: use when retrieving results from a M_PROJ_LIST , M_EXTREME_LIST , or M_EVENT_LIST result buffer if M_FLOAT was added to the ResultType parameter of MimAllocResult() during result buffer allocation.

Combination constants for any of the possible values of the **ResultType** parameter

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

● For specifying the data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_DOUBLE	Casts the requested results to a <i>double</i> . (summarize) <i>UserArrayPtr info</i> <ul style="list-style-type: none">• Data type: array of type double Array size: depends on the ResultType parameter, see above. Note: When multiple results.• Data type: double Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_FLOAT	Casts the requested results to a <i>float</i> . (summarize) <i>UserArrayPtr info</i> <ul style="list-style-type: none">• Data type: array of type float Array size: depends on the ResultType parameter, see above.

		<p>Note: When multiple results.</p> <ul style="list-style-type: none"> • Data type: float <p>Note: When a single result, see above.</p>
<input type="checkbox"/> M_TYPE_LONG	<p>Casts the requested results to a <i>long</i>. (summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type long Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: long Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: MIL_DOUBLE Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_MIL_INT	<p>Casts the requested results to a <i>MIL_INT</i>. (summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: MIL_INT Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_MIL_INT32	<p>Casts the requested results to a <i>MIL_INT32</i>. (summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: MIL_INT32 Note: When a single result, see above.
<input type="checkbox"/> M_TYPE_MIL_INT64	<p>Casts the requested results to a <i>MIL_INT64</i>. (summarize)</p>	<p><i>UserArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT64 Array size: depends on the ResultType parameter, see above. Note: When multiple results. • Data type: MIL_INT64 Note: When a single result, see above.

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type double
- array of type float
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- double
- float
- long
- MIL_DOUBLE

- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address of the one-dimensional array in which to write the results read from the MIL result buffer.

You can set this parameter to **M_NULL** to put the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimHistogram

Synopsis

Generate the intensity histogram of an image buffer.

Syntax

```
void MimHistogram(  
    MIL_ID SrcImageBufId,  
    MIL_ID HistImResultId  
)
```

Description

This function calculates the histogram (or pixel intensity distribution) of the specified source buffer and stores results in the specified histogram result buffer.

You can read the histogram values from the result buffer, using [MimGetResultId\(\)](#) or [MimGetResult\(\)](#), specifying [M_VALUE](#) as the result type.

Note that floating-point values will be cast to *MIL_UINT* values before performing the histogram. Therefore, unexpected results can occur if a floating-point value is larger than the *MIL_UINT* range.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the operation. This buffer must be one-band. The buffer data will be treated as unsigned.

HistImResultId

Specifies the identifier of the destination for the histogram results. This parameter must be given the identifier of an image processing result buffer that was allocated with [MimAllocResult\(\)](#) and has an [M_HIST_LIST](#) type. If the result buffer has fewer entries than the full range of source values, the pixel values that are out of range will not be included in the histogram.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimHistogramEqualize

Synopsis

Perform a histogram equalization of an image.

Syntax

```
void MimHistogramEqualize(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT Method,
    MIL_DOUBLE Alpha,
    MIL_DOUBLE Min,
    MIL_DOUBLE Max
)
```

Description

This function performs a histogram equalization of the specified source image. Results are written to a destination buffer, which can be either an image buffer or a LUT buffer.

This function first performs a histogram of the source image buffer. The histogram is then used to calculate a transformation LUT that can be used to enhance the source image (with [MimLutMap\(\)](#)). If the destination buffer is a LUT, the transformation LUT is copied into the destination LUT. If the destination buffer is an image, the source buffer is remapped through the transformation LUT to produce the destination image.

Note that floating-point values will be cast to *MIL_UINT32* values before performing the histogram. Therefore, unexpected results can occur if a floating-point value is larger than the *MIL_UINT32* range.

Parameters

- SrcImageBufId
- Specifies the identifier of the data source of the operation. This parameter will be treated as an unsigned image buffer identifier.
- DestImageBufId
- Specifies the identifier of the destination of the results. This parameter must be given an image buffer or LUT buffer identifier.
- Method
- Specifies the type of equalization to perform. This parameter can be set to the following values. Note that the cumulative probability distribution, $P_A(r)$, of the input image is approximated by its cumulative histogram:

$$P_A(r) \approx \sum_{m=0}^L H_F(m)$$

Refer to "Pratt, William K. *Digital Image Processing* . United States: John Wiley & Sons, 1978. 318."

● For specifying the type of equalization to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EXPONENTIAL	<p>Specifies an equalization density function which generates an Exponential distribution.</p> <p>Output probability density model:</p> $P_F(q) = \alpha \left(e^{-(q - q_{min})} \right)$ <p>$q \geq q_{min}$</p> <p>Transfer function:</p>

	$g = g_{\min} - \frac{1}{\alpha} \ln[1 - p_f(f)]$ <p>(summarize)</p>
<input type="checkbox"/> M_HYPER_CUBE_ROOT	<p>Specifies an equalization density function which generates a Hyperbolic Cube Root distribution.</p> <p>Output probability density model:</p> $p_g(g) = \frac{1}{3} \left[\frac{g^{-2/3}}{\sqrt[3]{g_{\max}} - \sqrt[3]{g_{\min}}} \right]$ <p>Transfer function:</p> $g = \left[\left(\sqrt[3]{g_{\max}} - \sqrt[3]{g_{\min}} \right) (p_f(f)) + \sqrt[3]{g_{\min}} \right]^3$ <p>(summarize)</p>
<input type="checkbox"/> M_HYPER_LOG	<p>Specifies an equalization density function which generates a Hyperbolic Logarithmic distribution.</p> <p>Output probability density model:</p> $p_g(g) = \frac{1}{g [\ln(g_{\max}) - \ln(g_{\min})]}$ <p>Transfer function:</p> $g = g_{\min} \left[\frac{g_{\max}}{g_{\min}} \right]^{p_f(f)}$ <p>(summarize)</p>
<input type="checkbox"/> M_RAYLEIGH	<p>Specifies an equalization density function which generates a Rayleigh distribution.</p> <p>Output probability density model:</p> $p_g(g) = \frac{g - g_{\min}}{\alpha^2} \left[\frac{g - g_{\min}}{2\alpha^2} \right]$ $g \geq g_{\min}$ <p>Transfer function:</p> $g = g_{\min} + \frac{1}{2} \left[2\alpha^2 \ln \left(\frac{1}{1 - p_f(f)} \right) \right]$ <p>(summarize)</p>
<input type="checkbox"/> M_UNIFORM	<p>Specifies an equalization density function which generates a Uniform distribution.</p> <p>Output probability density model:</p> $p_g(g) = \frac{1}{g_{\max} - g_{\min}}$ $g_{\min} \leq g \leq g_{\max}$ <p>Transfer function:</p> $g = [g_{\max} - g_{\min}] p_f(f) + g_{\min}$ <p>(summarize)</p>

Alpha

Specifies the adjustment parameter. Alpha is used with the Method parameter values M_EXPONENTIAL and M_RAYLEIGH.

In the case of the M_EXPONENTIAL method, a greater Alpha yields a lower occurrence of the most frequent pixels of the histogram in the resulting image buffer.

In the case of the [M_RAYLEIGH](#) method, the greater the alpha is, the greater the occurrence of the most frequent pixels of the histogram in the resulting image buffer.

Min

Specifies the lowest pixel value to equalize

Max

Specifies the highest pixel value to equalize

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimInquire

Synopsis

Inquire about an image processing context or result buffer setting.

Syntax

```
MIL_INT MimInquire(
    MIL_ID ContextorResultId,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires information about an image processing context or result buffer setting. The image processing context or result buffer must have been allocated with [MimAlloc\(\)](#) or [MimAllocResult\(\)](#), respectively.

Parameters

ContextorResultId

Specifies the identifier of the image processing context or result buffer about which to inquire information.

InquireType

Specifies the type of information about which to inquire.

For an image processing result buffer, the [InquireType](#) parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

For a result buffer	
Value	Description
<input type="checkbox"/> M_MODIFICATION_COUNT +	Returns the number of modifications made to the buffer since it was allocated.
<input type="checkbox"/> M_OWNER_SYSTEM +	Returns the identifier of the system on which the buffer is allocated. (summarize) <i>UserVarPtr info</i> Data type: MIL_ID
<input type="checkbox"/> M_RESULT_SIZE +	Returns the number of entries in the buffer. (summarize) <i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_RESULT_TYPE +	Returns the attribute or nature of the buffer. (summarize) <i>UserVarPtr info</i> Return values: M_COUNT_LIST; M_EVENT_LIST; M_EXTREME_LIST; M_HIST_LIST; M_PROJ_LIST; M_STAT_LIST; (details)

For an image processing context, the [InquireType](#) parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

● For a context	
☐ Value	Description
☐ M_CONTEXT_TYPE +	<p>Returns the type of the image processing context. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEINTERLACE_CONTEXT; (details)</p>
☐ M_DEINTERLACE_TYPE +	<p>Returns the deinterlacing algorithm. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_ADAPTIVE_AVERAGE; M_ADAPTIVE_BOB; M_ADAPTIVE_DISCARD; M_AVERAGE; M_BOB; M_DEFAULT; M_DISCARD; (details)</p>
☐ M_DISCARD_FIELD +	<p>Returns the field that is discarded when M_DEINTERLACE_TYPE is set to M_DISCARD or M_ADAPTIVE_DISCARD. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_EVEN_FIELD; M_ODD_FIELD; (details)</p>
☐ M_FIRST_FIELD +	<p>Returns the field that is processed first when M_DEINTERLACE_TYPE is set to M_BOB or M_ADAPTIVE_BOB. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_EVEN_FIELD; M_ODD_FIELD; (details)</p>
☐ M_MOTION_DETECT_NUM_FRAMES +	<p>Returns the number of frames to use to perform the motion detection. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 1; (details)</p>
☐ M_MOTION_DETECT_OUTPUT +	<p>Returns whether the output images will be the deinterlaced images or images indicating which pixels are considered to be part of an object in motion (the internal motion detection mask). (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>
☐ M_MOTION_DETECT_REFERENCE_FRAME +	<p>Returns the index of the frame to use for deinterlacing within the group of frames that are used for motion detection. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0<=Value<M_MOTION_DETECT_NUM_FRAMES; M_CENTER_FRAME; M_DEFAULT; M_FIRST_FRAME; M_LAST_FRAME; (details)</p>
☐ M_MOTION_DETECT_THRESHOLD +	<p>Returns the threshold value used for motion detection. (summarize)</p> <p><i>UserVarPtr info</i> Return values: Value >= 0; (details)</p>
☐ M_SOURCE_FIRST_IMAGE +	<p>Returns the index of the input image used to generate the first deinterlaced image. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value >= 0; (details)</p>

Combination constants for any of the possible values of the [InquireType](#) parameter

You can add one of the following values to the above-mentioned values to cast the requested information to a required data type.

● For specifying the data type	
☐ Value	Description

<div><div></div><div>M_TYPE_DOUBLE</div></div>	<div>Casts the requested information to a <i>double</i>. This is the default value. (summarize)</div> <div><div></div><div><div>UserVarPtr info</div><div>Data type: double</div></div></div>
<div><div></div><div>M_TYPE_LONG</div></div>	<div>Casts the requested information to a <i>long</i>. (summarize)</div> <div><div></div><div><div>UserVarPtr info</div><div>Data type: long</div></div></div>
<div><div></div><div>M_TYPE_MIL_DOUBLE</div></div>	<div>Casts the requested information to a <i>MIL_DOUBLE</i>. (summarize)</div> <div><div></div><div><div>UserVarPtr info</div><div>Data type: MIL_DOUBLE</div></div></div>
<div><div></div><div>M_TYPE_MIL_ID</div></div>	<div>Casts the requested information to a <i>MIL_ID</i>. Note that M_TYPE_MIL_ID should only be used with M_OWNER_SYSTEM. (summarize)</div> <div><div></div><div><div>UserVarPtr info</div><div>Data type: MIL_ID</div></div></div>
<div><div></div><div>M_TYPE_MIL_INT</div></div>	<div>Casts the requested information to a <i>MIL_INT</i>. (summarize)</div> <div><div></div><div><div>UserVarPtr info</div><div>Data type: MIL_INT</div></div></div>
<div><div></div><div>M_TYPE_MIL_INT32</div></div>	<div>Casts the requested information to a <i>MIL_INT32</i>. (summarize)</div> <div><div></div><div><div>UserVarPtr info</div><div>Data type: MIL_INT32</div></div></div>
<div><div></div><div>M_TYPE_MIL_INT64</div></div>	<div>Casts the requested information to a <i>MIL_INT64</i>. (summarize)</div> <div><div></div><div><div>UserVarPtr info</div><div>Data type: MIL_INT64</div></div></div>

UserVarPtr

<p>Accepts the address of one of the following (see above for specifics on which is expected):</p> <ul style="list-style-type: none"> • double • long • MIL_DOUBLE • MIL_ID • MIL_INT • MIL_INT32 • MIL_INT64

Specifies the address of the variable in which to write the requested information. Since the `MimInquire()` function also returns the requested information, you can set this parameter to `M_NULL`.

Return value

The returned value is the requested information, cast to *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.

DLL	Requires mil.dll; milim.dll.
-----	------------------------------

MimLabel

Synopsis

Label objects in an image buffer.

Syntax

```
void MimLabel(  
    MIL_ID SrcImageBufId,  
    MIL_ID DestImageBufId,  
    MIL_INT ProcMode  
)
```

Description

This function labels all objects (or blobs) in the specified source image, starting from the top-left corner, with unique consecutive values beginning with the value 1. Each pixel in the same blob is given the same numerical value.

If you want to distinguish between touching blobs, you should separate the blobs. For example, you can use an erosion operation before performing the labeling operation.

Parameters

SrcImageBufId

Specifies identifier of the data source of the operation. This parameter must be given an image buffer identifier. If the source buffer is not binary, all non-zero pixels are considered as part of an object or blob.

DestImageBufId

Specifies the identifier of the destination of the results. This parameter must be given an image buffer identifier. The destination buffer should be large enough to hold the maximum number of objects (blobs). For example, an 8-bit buffer can be used for a maximum of 254 blobs and a 16-bit buffer can be used for a maximum of 65534 blobs). If the destination buffer depth is too small, several blobs might be given the same value.

ProcMode

Specifies the type of connectivity (lattice) to use for processing. This parameter can be set to one of the following values:

● For specifying the type of connectivity to use	
☐ Value	Description
☐ M_4_CONNECTED +	Specifies that each pixel has 4 neighbors. If two blobs touch on the vertical and horizontal axis, they are considered one blob. (summarize)
☐ M_8_CONNECTED +	Specifies that each pixel has 8 neighbors. If two blobs touch on the vertical, horizontal, or diagonal, they are considered one blob. (summarize)

Combination constants for any of the possible values of the [ProcMode](#) parameter

You can add one of the following values to the above-mentioned values to specify the processing mode.

● For the ProcMode parameter	
☐ Value	Description
☐ M_BINARY	Specifies the binary processing mode. If the source image is a binarized image (containing 0 or the maximum value of the buffer, achieved for example with MimBinarize()), you can add M_BINARY to the connectivity mode to accelerate processing, particularly if the labeling is done by the Host.

	(summarize)
<input type="checkbox"/> <code>M_GRAYSCALE</code>	Specifies the grayscale processing mode. This is the default value. (summarize)

Remark

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include <code>mil.h</code> .
Library	Use <code>mil.lib</code> ; <code>milim.lib</code> .
DLL	Requires <code>mil.dll</code> ; <code>milim.dll</code> .

MimLocateEvent

Synopsis

Find pixel coordinates or values that satisfies a specified condition.

Syntax

```
void MimLocateEvent(
    MIL_ID SrcImageBufId,
    MIL_ID EventImResultId,
    MIL_INT Condition,
    MIL_DOUBLE CondLow,
    MIL_DOUBLE CondHigh
)
```

Description

This function finds the coordinates and value of the pixels that satisfy the specified condition in the specified source image. Results are stored in the specified event result buffer.

You can retrieve the values, X- or Y-coordinates, and the total number of pixels that satisfy the condition, using [MimGetResult\(\)](#) or [MimGetResultId\(\)](#).

Parameters

SrcImageBufId

Specifies the identifier of the source image for the operation. This parameter must be given the identifier of a 1-band buffer.

EventImResultId

Specifies the identifier of the buffer in which to store the event results. This parameter must be given the identifier of an image processing result buffer, allocated using [MimAllocResult\(\)](#) with an [M_EVENT_LIST](#) type. The buffer must have enough entries to hold the expected number of events.

If the number of pixels found is less than the number of event result buffer entries, the remaining result buffer entries are set to **M_INVALID**. If the number of events is greater than the number of allocated result buffer entries, the extra entries are discarded.

Condition

Specifies the condition under which pixel values are considered an event. This parameter must be set to one of the values below.

For conditions that use two limits, set the [Condition](#) parameter to one of the values below.

● For conditions that use two limits	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN_RANGE	Specifies that pixels with values between CondLow and CondHigh , inclusive, are considered events.
<input type="checkbox"/> M_OUT_RANGE	Specifies that pixels with values less than CondLow , or greater than CondHigh , are considered events.

For conditions that use one limit, set the [Condition](#) parameter to one of the values below.

● For conditions that use one limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EQUAL	Specifies that only pixels with values equal to CondLow will be considered an event.

<input type="checkbox"/> M_GREATER	Specifies that only pixels with values greater than CondLow will be considered an event.
<input type="checkbox"/> M_GREATER_OR_EQUAL	Specifies that only pixels with values greater than or equal to the CondLow will be considered an event.
<input type="checkbox"/> M_LESS	Specifies that only pixels with values less than CondLow will be considered an event.
<input type="checkbox"/> M_LESS_OR_EQUAL	Specifies that only pixels with values less than or equal to CondLow will be considered an event.
<input type="checkbox"/> M_NOT_EQUAL	Specifies that only pixels with values not equal to CondLow will be considered an event.

CondLow

Specifies the lower limit of the selected condition.

● For specifying the lower limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> Value	Specifies the lower limit.

CondHigh

Specifies the upper limit of the selected condition.

● For specifying the upper limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NULL	Specifies that no upper limit is required. This setting should be used for conditions that use one limit. (summarize)
<input type="checkbox"/> Value	Specifies the upper limit.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimLocatePeak1d

Synopsis

Performs peak intensity detection in every row or column of the source image.

Syntax

```
void MimLocatePeak1d(
    MIL_ID SrcImageId,
    MIL_ID DestPositionId,
    MIL_ID DestIntensityId,
    MIL_INT DestY,
    MIL_INT PeakWidth,
    MIL_DOUBLE MinIntensity,
    MIL_INT ControlFlag,
    MIL_DOUBLE ControlValue
)
```

Description

This function detects the peak intensity above a minimum level in every row or column in the source image. This function is well-suited for laser line capture applications that need to record the three-dimensional position of an object.

For each column or row of the source image, the function finds the neighborhood with the greatest average intensity and records the average as the peak intensity value. The sub-pixel weighted center of the neighborhood is recorded as the position. Therefore, the accuracy of the peak intensity value and position is directly related to the width of the neighborhood.

The function uses two destination image buffers to store the results: one for the position and one for the intensity. Although the function writes to two destination image buffers, it only writes to one row of each. For every row (or column) in the source image, the function determines the X-coordinate (or Y-coordinate) and value of the peak intensity and records it in the specified row in the destination image buffers at an offset corresponding to the row (or column) in the source image buffer. For example, for row five of the source image, the function determines the X-coordinate of the peak intensity and records it in pixel five in the specified row of the destination buffers. Consecutive calls to this function can fill consecutive rows of the destination image buffers, producing a complete pattern of the changes in position and value of the peak intensities in a series of source images.

You can draw the calculated peak intensities at the calculated positions into an image buffer using [MimDraw\(\)](#).

Parameters

SrcImageId

Specifies the identifier of the source image buffer.

Note that this function only supports 8- and 16-bit unsigned source image buffers.

DestPositionId

Specifies the identifier of the destination image buffer in which to store the calculated positional X- or Y-coordinates.

When detecting the peak in every column (using [M_ID_COLUMNS](#)), the destination buffer must be at least as wide as the width of the source buffer. When detecting the peak in every row (using [M_ID_ROWS](#)), the destination buffer must be at least as wide as the height of the source buffer.

Note that this function only supports 8- and 16-bit unsigned destination image buffers. However, you can have results returned in a fixed point representation by adding [M_FIXED_POINT + n](#) to the **ControlFlag** parameter, where n is the required fractional precision.

DestIntensityId

Specifies the identifier of the destination image buffer in which to store the peak intensity values. If the intensity values are not required, set this parameter to **M_NULL**.

When detecting the peak in every column (using [M_ID_COLUMNS](#)), the destination buffer must be at least as wide as the width of the source buffer. When detecting the peak in every row (using [M_ID_ROWS](#)), the destination buffer must be at least as wide as the height of the source buffer.

Note that the intensity destination buffer must have the same type and bit depth as the source buffer.

DestY

Specifies the row in the destination image buffers in which to write the results.

PeakWidth

Specifies the width of the neighborhood, in pixels.

When detecting the peak in every column (using [M_1D_COLUMNS](#)), the width of the neighborhood must be less than the height of the source buffer. When detecting the peak in every row (using [M_1D_ROWS](#)), the width of the neighborhood must be less than the width of the source buffer.

MinIntensity

Specifies the minimum average intensity level required to record a position. The rows (or columns) that do not have a peak detected above this threshold will have their positions written as the maximum possible value of the destination position buffer and will be considered to be invalid positions.

ControlFlag

Specifies the direction of the peak intensity detection.

● For specifying the direction of the peak detection	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_1D_COLUMNS + M_FIXED_POINT + n (where n = 0).
<input type="checkbox"/> M_1D_COLUMNS +	Specifies that the peak detection scans vertically for the peak intensity values.
<input type="checkbox"/> M_1D_ROWS +	Specifies that the peak detection scans horizontally for the peak intensity values.

Combination constant for [M_1D_COLUMNS](#); [M_1D_ROWS](#);
You can add the following value to the above-mentioned values to specify the number of fractional bits to use for fixed-point representation. If the value is not added then there will not be any fractional bits.

● For M_1D_COLUMNS and M_1D_ROWS	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FIXED_POINT + n	Specifies the number of fractional bits to use for fixed-point representation, where n is any integer between 0 and 7, inclusive.

ControlValue

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimLutMap

Synopsis

Perform a point-to-point LUT mapping operation.

Syntax

```
void MimLutMap(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_ID LutBufId
)
```

Description

This function maps each pixel in the specified source image to values determined by the specified lookup table (LUT).

If the source image is signed, negative numbers are treated as unsigned values and address the LUT accordingly. For example, -1 is represented as 0xff for 8-bit image data and 0xffff for 16-bit image data and will therefore address LUT entry 0xff or 0xffff, respectively.

Parameters

- SrcImageBufId
- Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.
- DestImageBufId
- Specifies the identifier of the destination of the results. This parameter must be given an image buffer identifier.
- LutBufId
- Specifies the LUT through which to map source input values. This parameter must be given a valid LUT buffer identifier.
- If the LUT has fewer entries than the full range of source values, unexpected results will occur. If the LUT buffer depth is greater than that of the destination buffer, results will be truncated to fit the destination buffer. LUT entries must have been previously initialized (for example, using [MgenLutRamp\(\)](#) or [MbufPut1d\(\)](#)).

Remark

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimMorphic

Synopsis

Perform a morphological transformation using a user-defined or predefined structuring element.

Syntax

```
void MimMorphic(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_ID StructElemBufId,
    MIL_INT Operation,
    MIL_INT NbIterationOrArea,
    MIL_INT ProcMode
)
```

Description

This function performs one of several morphological transformations on the specified source image, using a user-defined or predefined structuring element.

If the source and destination are multi-band buffers, the same structuring element is applied to every band of the source buffer.

Parameters

SrcImageBufId

Specifies the identifier of the source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the resulting image. This parameter must be given an image buffer identifier.

StructElemBufId

Specifies the structuring element to use. For all operations except area opening and area closing operations, you must specify the identifier of the structuring element buffer which defines the required structuring element.

The structuring element buffer must have been allocated, using [MbufAlloc1d\(\)](#) or [MbufAlloc2d\(\)](#) with an [M_STRUCT_ELEMENT](#) attribute, and loaded with structuring-element values, using [MbufPut\(\)](#). These values can be set to [M_DONT_CARE](#) to specify that the corresponding image pixels should not be taken into account during the operation. You can set the overscan type and the center of the structuring element, using [MbufControlNeighborhood\(\)](#).

For area opening and area closing operations, you must specify one of the following two predefined structuring elements:

● For area opening and area closing operations	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_3X3_CROSS	Specifies a cross (+) structuring element whose neighborhood size is 3 x 3 pixels. In this case, there are 5 valid neighborhood pixels; the function ignores the pixels located at the four corners of the neighborhood. (summarize)
<input type="checkbox"/> M_3X3_RECT	Specifies a structuring element whose neighborhood size is 3 x 3 pixels. In this case, there are 9 valid neighborhood pixels. (summarize)

Operation

Specifies the operation to perform. Supported operations are given in the table below.

● For specifying the type of operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_AREA_CLOSE	Specifies an area close operation.
<input type="checkbox"/> M_AREA_OPEN	Specifies an area open operation.
<input type="checkbox"/> M_DILATE	Specifies a dilation operation.
<input type="checkbox"/> M_ERODE	Specifies an erosion operation.
<input type="checkbox"/> M_HIT_OR_MISS	Specifies a hit or miss transformation.
<input type="checkbox"/> M_MATCH	specifies a matching operation.
<input type="checkbox"/> M_THICK	Specifies a thickening operation.
<input type="checkbox"/> M_THIN	Specifies a thinning operation.

NbIterationOrArea

Specifies a value that is dependent on the morphological operation chosen.

This parameter must be set to a positive integer. If this parameter is set to 0, the destination buffer will remain unchanged.

For an area opening or closing operation, this parameter specifies the minimum area, in pixels. For the other morphological transformations, this parameter specifies the number of times to iterate the operation. For an [M_HIT_OR_MISS](#) or [M_MATCH](#) operation, this parameter must be set to 1.

ProcMode

Specifies the processing mode to use. This parameter can be set to the following:

● For specifying the processing mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BINARY	Treats non-zero pixels as ones (1) during processing.
<input type="checkbox"/> M_GRAYSCALE	Uses the source image's gray values for processing and the resulting buffer will also contain gray values. This is the default value. (summarize)

Remarks

- This function is optimized for packed binary buffers.
- In-place processing is supported, but the source and destination image buffers cannot overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimOpen

Synopsis

Perform a binary or grayscale opening-type morphological operation.

Syntax

```
void MimOpen(  
    MIL_ID SrcImageBufId,  
    MIL_ID DestImageBufId,  
    MIL_INT NbIteration,  
    MIL_INT ProcMode  
)
```

Description

This function performs a binary or grayscale opening operation on the given source image for the specified number of iterations. An opening is an erosion followed by a dilation.

In binary mode, this function uses a 3 x 3 full rectangular structuring element; in grayscale mode, a 3 x 3 empty one.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the resulting image. This parameter must be given an image buffer identifier.

NbIteration

Specifies the number of times to iterate the operation.

When this parameter is set to 0 and [ProcMode](#) is set to [M_BINARY](#), the source image is binarized and the result is copied into the destination image buffer.

When this parameter is set to 0 and [ProcMode](#) is set to [M_GRAYSCALE](#), the source image is copied into the destination image buffer.

ProcMode

Specifies the processing mode to use. This parameter can be set to the following:

For specifying the processing mode	
Value	Description
<input type="checkbox"/> M_BINARY	Treats non-zero pixels as ones (1) during processing. The resulting non-zero pixels will have all bits set to one. (summarize)
<input type="checkbox"/> M_GRAYSCALE	Uses the source image's gray values for processing. The resulting buffer also contains gray values. (summarize)

Remarks

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).
- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimPolarTransform

Synopsis

Perform a polar-to-rectangular or rectangular-to-polar transform.

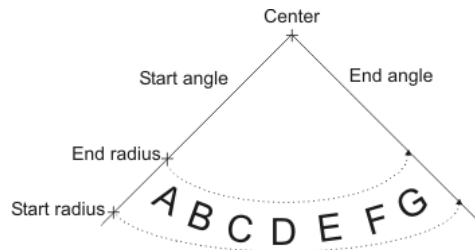
Syntax

```
void MimPolarTransform(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE CenterPosX,
    MIL_DOUBLE CenterPosY,
    MIL_DOUBLE StartRadius,
    MIL_DOUBLE EndRadius,
    MIL_DOUBLE StartAngle,
    MIL_DOUBLE EndAngle,
    MIL_INT OperationMode,
    MIL_INT InterpolationMode,
    MIL_DOUBLE *DestSizeXPtr,
    MIL_DOUBLE *DestSizeYPtr
)
```

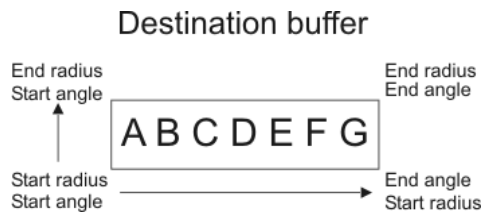
Description

This function performs a rectangular-to-polar or polar-to-rectangular transformation.

For a rectangular-to-polar transformation set the zone to convert by specifying the center coordinates ([CenterPosX](#) and [CenterPosY](#)), the start and end radius ([StartRadius](#) and [EndRadius](#)), and the start and end angle ([StartAngle](#) and [EndAngle](#)) as follows:



The result will be mapped to the destination buffer as follows:



The increment in angle is determined by the length, in pixels, of the outside arc, calculated as follows:

$(\text{endangle} - \text{startangle}) / \text{arclength}$.

A polar-to-rectangular transform performs the reverse of the transform described above. It takes a polar buffer and maps it to a rectangular buffer. Use [CenterPosX](#), [CenterPosY](#), [StartAngle](#), [EndAngle](#), [StartRadius](#), [EndRadius](#) parameters to specify where in the destination buffer the contents of the polar buffer will be mapped.

Parameters

SrcImageBufId

Specifies the identifier of the source buffer.

DestImageBufId

Specifies the identifier of the destination buffer. To determine the required size of the destination buffer, call the function with [DestImageBufId](#) set to **M_NULL** and obtain the size in [DestSizeXPtr](#) and [DestSizeYPtr](#).

CenterPosX

Specifies the X-coordinate of the center in the rectangular image.

CenterPosY

Specifies the Y-coordinate of the center in the rectangular image.

StartRadius

Specifies the start radius of the zone of interest, in pixels.

EndRadius

Specifies the end radius of the zone of interest, in pixels.

StartAngle

Specifies the start of scan in the zone of interest. If the start angle ([StartAngle](#)) is less than the end angle ([EndAngle](#)), the direction of the scan will be counter clockwise. If the start angle is greater than the end angle, the direction of the scan will be clockwise.

The valid range for [StartAngle](#) is between -360.0 to 360.0° and the maximum span between [StartAngle](#) and [EndAngle](#) must not exceed 360.0°.

EndAngle

Specifies the end of scan in the zone of interest.

The valid range for [EndAngle](#) is between -360.0 to 360.0° and the maximum span between [StartAngle](#) and [EndAngle](#) must not exceed 360.0°.

OperationMode

Specifies the transform to perform. This parameter must be set to one of the values below.

● For specifying the transform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_POLAR_TO_RECTANGULAR	Performs polar-to-rectangular transform.
<input type="checkbox"/> M_RECTANGULAR_TO_POLAR	Performs rectangular-to-polar transform. This is the default. (summarize)

InterpolationMode

Specifies the interpolation used in the conversion.

This parameter must be set to one of the values below.

--

● For specifying the interpolation used in the conversion	
Value	Description
<input type="checkbox"/> M_DEFAULT +	Default for interpolation mode and type. Same as M_NEAREST_NEIGHBOR + M_OVERSCAN_ENABLE . (summarize)
<input type="checkbox"/> M_BICUBIC +	Performs bicubic interpolation. Saturation is performed according to the type of the destination buffer. (summarize)
<input type="checkbox"/> M_BILINEAR +	Performs bilinear interpolation. No saturation is performed. (summarize)
<input type="checkbox"/> M_NEAREST_NEIGHBOR +	Performs nearest neighbor interpolation. No saturation is performed. (summarize)

Combination constants for any of the possible values of the [InterpolationMode](#) parameter

You can add one of the following values to the above-mentioned values to specify how to determine the value of a destination pixel when its associated point falls outside the source buffer.

● For specifying how to determine the value of a destination pixel when its associated point falls outside the source buffer																												
Value	Description	corona-II (a)	coronoplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nextis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios glge (v)	vio (w)				
<input type="checkbox"/> M_OVERSCAN_CLEAR	Sets the destination pixel to 0.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
<input type="checkbox"/> M_OVERSCAN_DISABLE	Leaves the destination pixel as is.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
<input type="checkbox"/> M_OVERSCAN_ENABLE	Uses pixels from the source buffer's ancestor buffer. If the source buffer is not a child buffer or if the point falls outside the ancestor buffer, leave the destination pixel as is. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				
	<i>Board specific</i>																											
	Points falling outside the source image are undefined.			d														q	r	s	t	u	v					
<input type="checkbox"/> M_OVERSCAN_FAST	Specifies that MIL will automatically select the overscan that optimizes speed, according to the specified operation and the target system. The overscan could be hardware-specific thereby having a different behavior than the other supported overscan modes. Note that when using M_OVERSCAN_FAST , the destination pixels in the overscan area are undefined. The pixels can therefore contain different values from one function call to the next, even if the function's parameter values are the same. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w				

DestSizeXPtr

Specifies the required width of the destination buffer. If this parameter is not set to **M_NULL**, the width will be returned.

DestSizeYPtr

Specifies the required height of the destination buffer. If this parameter is not set to **M_NULL**, the height will be returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimProject

Synopsis

Project a 2D image into 1D.

Syntax

```
void MimProject(
    MIL_ID SrcImageBufId,
    MIL_ID ProjImResultId,
    MIL_DOUBLE ProjAngle
)
```

Description

This function projects a two-dimensional buffer into a one-dimensional buffer from the specified angle, and writes results to the specified projection result buffer. The results are generated by adding all pixel values along each diagonal in the image at the specified projection angle. The 90° projection of the image is known as the row profile; the 0° projection as the column profile.

Note that, if the sum of any diagonal is larger than the depth of the result buffer, the result will be undefined.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the image. This parameter must be given an image buffer identifier. The buffer must be one-band.

ProjImResultId

Specifies the identifier of the destination of the projection results. This parameter must be given the identifier of an image processing result buffer allocated with [MimAllocResult\(\)](#) and an [M_PROJ_LIST](#) type. The buffer should have as many locations as there are lines to project in the image at the specified angle.

You can read the projection values from the result buffer, using [MimGetResultId\(\)](#) or [MimGetResult\(\)](#), specifying [M_VALUE](#) as the result type.

ProjAngle

Specifies the angle of projection, in degrees. Predefined projection angles are the following:

For specifying the angle of projectuion in degrees	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_0_DEGREE	Specifies a 0° projection (column profile).
<input type="checkbox"/> M_90_DEGREE	Specifies a 90° projection (row profile).

Remark

- Only 0° and 90° projection angles are supported.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimRank

Synopsis

Perform a rank filter on the pixels in an image.

Syntax

```
void MimRank(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_ID StructElemBufId,
    MIL_INT Rank,
    MIL_INT ProcMode
)
```

Description

This function performs a rank filter operation on the specified source buffer. It replaces each pixel with the pixel in its neighborhood whose value is the specified *rankth* value relative to others. It uses the specified structuring element as a mask. This mask determines the neighborhood size and which pixels in the neighborhood to ignore.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the resulting image. This parameter must be given an image buffer identifier.

StructElemBufId

Specifies the identifier of the structuring element buffer to use as a mask. You can either define a custom structuring element as a mask or use a predefined mask buffer.

To use a custom structuring element, you must have previously allocated it with [MbufAlloc1d\(\)](#) or [MbufAlloc2d\(\)](#) (specifying [M_STRUCT_ELEMENT](#) as the attribute) and loaded it with values, using [MbufPut\(\)](#). The structuring element buffer dimensions are used as the neighborhood size. Values in the structuring element set to [M_DONT_CARE](#) represent neighborhood pixels not to be considered in the ranking operation. All other values in the structuring element must be set to 1, representing neighborhood pixels to be considered in the ranking operation.

The following are predefined mask buffers:

● For the structuring element buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_3X3_CROSS +	Applies a cross (+) mask and sets the neighborhood size to 3 x 3 pixels. In this case, there are 5 valid neighborhood pixels; the cross mask makes the function ignore the pixels located at the four corners of the neighborhood. (summarize)
<input type="checkbox"/> M_3X3_RECT +	Applies no mask and sets the neighborhood size to 3 x 3 pixels. In this case, there are 9 valid neighborhood pixels. (summarize)
<input type="checkbox"/> M_5X5_RECT +	Applies no mask and sets the neighborhood size to 5 x 5 pixels. In this case, there are 25 valid neighborhood pixels. (summarize)

Combination constants for any of the possible values of the [StructElemBufId](#) parameter

You can add one of the following values to the above-mentioned values to specify that overscanning is temporarily disabled.

● For temporarily disabling overscanning	
☐ Value	Description
☐ M_OVERSCAN_DISABLE	Disables overscanning temporarily. You should disable overscanning to accelerate the operation for applications in which the overscan data is not important in the resulting buffer. (summarize)
☐ M_OVERSCAN_FAST	Specifies that MIL will automatically select the overscan that optimizes speed, according to the specified operation and the target system. The overscan could be hardware-specific thereby having a different behavior than the other supported overscan modes. Note that when using M_OVERSCAN_FAST , the destination pixels in the overscan area are undefined. The pixels can therefore contain different values from one function call to the next, even if the function's parameter values are the same. (summarize)

Rank

Specifies which of the pixel values to select after valid neighborhood values are sorted in increasing order; valid neighborhood pixel values are those that are not masked-out.

● For specifying which pixel to select	
☐ Value	Description
☐ 1 <= Value <= number of valid neighborhood pixels	Specifies the rank of the pixel value to select. If the number of valid pixels is less than the given rank, the number of valid pixels is used as the rank. (summarize)
☐ M_MEDIAN	Specifies that MimRank() performs a median filter (that is, each pixel is replaced with the median neighborhood value).

ProcMode

Specifies the processing mode to use. This parameter can be set to one of the following:

● For specifying the processing mode to use	
☐ Value	Description
☐ M_BINARY	Treats non-zero pixels as ones (1) during processing. The resulting non-zero pixels will have all bits set to one. (summarize)
☐ M_GRAYSCALE	Uses the source image's gray values for processing. The resulting buffer will also contain gray values. This is the default value. (summarize)

Remarks

- This function is optimized for packed binary buffers.
- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimResize

Synopsis

Resize an image.

Syntax

```
void MimResize(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE ScaleFactorX,
    MIL_DOUBLE ScaleFactorY,
    MIL_INT InterpolationMode
)
```

Description

This function resizes the source image by the specified factors. Results are stored in the destination buffer starting from the top-left corner.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the results. This parameter must be given an image buffer identifier.

ScaleFactorX

Specifies the scaling factor for the width of the source image. This parameter can be set to one of the values below.

For specifying the scaling factor (width)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILL_DESTINATION	Resizes the source image to fill the entire width of the destination buffer. When set to M_FILL_DESTINATION , the source image is resized to fill the entire width of the destination buffer. (summarize)
<input type="checkbox"/> Value	Uses a non-null positive value to multiply the width of the source image. A value greater than 1.0 enlarges the source image, while a factor less than 1.0 reduces it. (summarize)

ScaleFactorY

Specifies the scaling factor for the height of the source image. This parameter can be set to one of the values below.

For specifying the scaling factor (height)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILL_DESTINATION	Resizes the source image to fill the entire height of the destination buffer. When set to M_FILL_DESTINATION , the source image is resized to fill the entire height of the destination buffer. (summarize)
<input type="checkbox"/> Value	Uses a non-null positive value to multiply the height of the source image. A value greater than 1.0 enlarges the source image, while a factor less than 1.0 reduces it.

	it. (summarize)
--	--------------------------------------

InterpolationMode

Specifies the mode of interpolation. This parameter can be set to one of the values below.

● For specifying the mode of interpolation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT +	Same as M_NEAREST_NEIGHBOR + M_OVERSCAN_ENABLE + M_FAST .
<input type="checkbox"/> M_AVERAGE +	Specifies averaging interpolation. No saturation is performed. The scaling factors for the width and height of the source image must be less than or equal to 1. (summarize)
<input type="checkbox"/> M_BICUBIC +	Specifies bicubic interpolation. Saturation is performed according to the type of the destination buffer. (summarize)
<input type="checkbox"/> M_BILINEAR +	Specifies bilinear interpolation. No saturation is performed. (summarize)
<input type="checkbox"/> M_INTERPOLATE +	Specifies interpolated resizing: for zooming = bilinear, for dezooming = averaging. Gives the best speed/result compromise for interpolated resizing. No saturation is performed. (summarize)
<input type="checkbox"/> M_NEAREST_NEIGHBOR +	Specifies the nearest neighbor interpolation. No saturation is performed. (summarize)

Combination constants for any of the possible values of the [InterpolationMode](#) parameter

You can add one of the following values to the above-mentioned values to specify how to determine the value of a destination pixel when its associated point falls outside the source buffer.

● For overscan																										
<div><input type="checkbox"/> Value</div>	Description	corona-II (a)	giga vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xcl (f)	helios ed/xd (g)	ilee 1394 i/dc (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nextis (p)	odyssey ea/xa (q)	odyssey ed/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ed/xd (u)	solios gige (v)	vio (w)			
<div><input type="checkbox"/> M_OVERSCAN_CLEAR</div>	Sets the destination pixel to 0.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<div><input type="checkbox"/> M_OVERSCAN_DISABLE</div>	Leaves the destination pixel as is.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
<div><input type="checkbox"/> M_OVERSCAN_ENABLE</div>	Uses pixels from the source buffer's ancestor buffer. If the source buffer is not a child buffer or if the point falls outside the ancestor buffer, leave the destination pixel as is. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		
	<i>Board specific</i>																									
	Points falling outside the source image are undefined.				d													q	r	s	t	u	v	w		
<div><input type="checkbox"/> M_OVERSCAN_FAST</div>	Specifies that MIL will automatically select the overscan that optimizes speed, according to the specified operation and the target system. The overscan could be hardware-specific thereby having a different behavior than the other supported overscan modes. Note that when using M_OVERSCAN_FAST , the destination pixels in the overscan area are undefined. The pixels can therefore contain different values from one function call to the next, even if the function's parameter values are the same. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

Combination constants for any of the possible values of the [InterpolationMode](#) parameter

You can add one of the following values to the above-mentioned values to specify the speed and precision of the interpolation.

● For the speed and precision	
▣ Value	Description
▣ M_FAST	Uses the fast interpolation method. This method is less precise. The default setting is M_FAST . (summarize)
▣ M_REGULAR	Uses the slow interpolation method. This method is more precise. (summarize)

Remark

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimRestore

Synopsis

Restore an image processing context from disk.

Syntax

```
MIL_ID MimRestore(  
    MIL_TEXT_PTR Filename,  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *ContextIdPtr  
)
```

Description

This function restores an image context that was previously saved to a file, using [MimSave\(\)](#) or [MimStream\(\)](#). This function restores all of the image processing context's settings that were in effect when the image processing context was saved.

Parameters

Filename

Specifies the name and path of the file from which to restore the image processing context. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

● For specifying the file name and path	
☐ Value	Description
<div>MIL_TEXT(MIL_TEXT_PTR FileName)</div>	<div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)</div> <div><div>Parameters</div><div>FileName</div><div>Specifies the drive, directory, and name of the file.</div></div>
<div>☐ M_INTERACTIVE</div>	<div>[This is only applicable to Windows] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.</div>

SystemId

Specifies the system on which to restore the image processing context. This parameter should be set to one of the following values:

● For specifying the system identifier	
☐ Value	Description
<div>☐ M_DEFAULT_HOST</div>	<div>Specifies the default Host system of the current MIL application.</div>
<div>☐ MIL system identifier</div>	<div>Specifies a valid system identifier, previously allocated using MsysAlloc().</div>

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the image processing context identifier. Since this function also returns the image processing context identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the image processing context identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimRotate

Synopsis

Rotate an image.

Syntax

```
void MimRotate(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE Angle,
    MIL_DOUBLE SrcCenX,
    MIL_DOUBLE SrcCenY,
    MIL_DOUBLE DstCenX,
    MIL_DOUBLE DstCenY,
    MIL_INT InterpolationMode
)
```

Description

This function rotates an image by the specified angle of rotation, using the specified interpolation mode. The center of rotation in the source image is determined by the specified X- and Y-source rotation-center coordinates. The rotated image will then be clipped to fit the destination buffer. It will be placed in the destination buffer with its center positioned at the specified X- and Y-destination center coordinates.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the results. This parameter must be given an image buffer identifier.

Angle

Specifies the angle of rotation, in degrees. When a positive angle is specified, the function rotates the image in a counter-clockwise direction.

SrcCenX

Specifies the X-coordinate to use as the center of rotation in the source image. This parameter must be set to one of the values below.

For specifying the X-coordinate (source)	
Value	Description
M_DEFAULT	Causes the image to rotate about its true center.
Value	Specifies the X-coordinate.

SrcCenY

Specifies the Y-coordinate to use as the center of rotation in the source image. This parameter must be set to one of the values below.

For specifying the Y-coordinate (source)	
Value	Description

<input type="checkbox"/> M_DEFAULT	Causes the image to rotate about its true center.
<input type="checkbox"/> Value	Specifies the Y-coordinate.

DstCenX

Specifies the X-coordinate in the destination buffer to which the specified center of the rotated source image will be mapped. This parameter must be set to one of the values below.

● For specifying the X-coordinate (destination)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Causes the true center of the destination buffer to be used.
<input type="checkbox"/> Value	Specifies the X-coordinate.

DstCenY

Specifies the Y-coordinate in the destination buffer to which the specified center of the rotated source image will be mapped. This parameter must be set to one of the values below.

● For specifying the Y-coordinate (destination)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Causes the true center of the destination buffer to be used.
<input type="checkbox"/> Value	Specifies the Y-coordinate.

InterpolationMode

Specifies the interpolation mode. This parameter must be set to one of the values below.

● For specifying the interpolation mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT +	Specifies nearest neighbor interpolation while using pixels from the source buffer's ancestor buffer. This is the same as M_NEAREST_NEIGHBOR + M_OVERSCAN_ENABLE . (summarize)
<input type="checkbox"/> M_BICUBIC +	Specifies bicubic interpolation. Saturation is performed according to the type of the destination buffer. (summarize)
<input type="checkbox"/> M_BILINEAR +	Specifies bilinear interpolation. No saturation is performed. (summarize)
<input type="checkbox"/> M_NEAREST_NEIGHBOR +	Specifies nearest neighbor interpolation. No saturation is performed. (summarize)

Combination constants for any of the possible values of the [InterpolationMode](#) parameter

You can add one of the following values to the above-mentioned values to specify how to determine the value of a destination pixel when its associated point falls outside the source buffer.

● For overscan	
<input type="checkbox"/> Value	Description
	corona-II (a)
	cronoplus (b)
	gige vision (c)
	gpu processing (d)
	helios ea/xa (e)
	helios ea/xd (f)
	helios ec/xd (g)
	ieee 1394 i1dc (h)
	iris (i)
	met-II /cl (j)
	met-II /mc (l)
	met-II /dig (k)
	met-II /std (m)
	morphis (n)
	morphis qxt (o)
	nexis (p)
	odyssey ea/xa (q)
	odyssey ec/xd (r)
	odyssey ed/xd (s)
	solios ea/xa (t)
	solios ec/xd (u)
	solios gige (v)
	v10 (w)

M_OVERSCAN_CLEAR	Sets the destination pixel to 0.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_OVERSCAN_DISABLE	Leaves the destination pixel as is.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_OVERSCAN_ENABLE	Uses pixels from the source buffer's ancestor buffer. If the source buffer is not a child buffer or if the point falls outside the ancestor buffer, leave the destination pixel as is. Points falling outside the source image are undefined. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	Points falling outside the source image are undefined.				d													q	r	s	t	u	v	
M_OVERSCAN_FAST	Specifies that MIL will automatically select the overscan that optimizes speed, according to the specified operation and the target system. The overscan could be hardware-specific thereby having a different behavior than the other supported overscan modes. Note that when using M_OVERSCAN_FAST , the destination pixels in the overscan area are undefined. The pixels can therefore contain different values from one function call to the next, even if the function's parameter values are the same. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Remark

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimSave

Synopsis

Save an image processing context to a file.

Syntax

```
void MimSave(
    MIL_TEXT_PTR FileName,
    MIL_ID ContextId,
    MIL_INT ControlFlag
)
```

Description

This function saves all the information about the previously allocated image processing context to disk. This information can be reloaded, using [MimRestore\(\)](#) or [MimStream\(\)](#).

Parameters

FileName

Specifies the name and path of the file in which to save the image processing context; it is recommended that you use the MIC file extension for easier use with other Matrox Imaging software products. The function internally handles the opening and closing of this file. If this file already exists, it will be overwritten.

This parameter can be set to one of the following values:

For specifying the file name and path					
Value	Description				
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p> <table><tr><th colspan="2">Parameters</th></tr><tr><th>FileName</th><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters		FileName	Specifies the drive, directory, and name of the file.
Parameters					
FileName	Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</p>				

ContextId

Specifies the identifier of the image processing context to save.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimShift

Synopsis

Perform a point-to-point bit shift.

Syntax

```
void MimShift(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT BitsToShift
)
```

Description

This function performs left or right bit-shifting on each pixel in the specified image. The shift operation is signed or unsigned depending on the source image buffer's data type.

Note that floating-point values will be cast to *MIL_UINT32* before performing the shift operation (except when shifting by 0). Therefore, unexpected results can occur if a floating-point value is larger than the *MIL_UINT32* range.

Note that, if you shift by 0, only a copy operation will be performed.

Parameters

- SrcImageBufId
- Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.
- DestImageBufId
- Specifies the identifier of the destination of the results. This parameter must be given an image buffer identifier.
- BitsToShift
- Specifies the number of bits to shift. If the given value is negative, each pixel in the specified image is right bit-shifted by the specified number of bits; otherwise, it is left bit-shifted.

Remark

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimStat

Synopsis

Calculate a variety of statistics on the source image.

Syntax

```
void MimStat(
    MIL_ID SrcImageId,
    MIL_ID StatResultId,
    MIL_INT StatType,
    MIL_INT Condition,
    MIL_DOUBLE CondLow,
    MIL_DOUBLE CondHigh
)
```

Description

This function calculates a variety of statistics for the pixels that satisfy the specified condition in the source image. Results are stored in the specified statistic result buffer.

You can retrieve statistical results using [MimGetResultId\(\)](#) or [MimGetResult\(\)](#).

Parameters

SrcImageId

Specifies the identifier of the source image for the operation. This parameter must be given a 1-band image buffer identifier.

StatResultId

Specifies the identifier of the destination result buffer for the operation.

This parameter must be given the identifier of an image processing result buffer, allocated using [MimAllocResult\(\)](#) function with [M_STAT_LIST](#) as the result type.

StatType

Specifies the type of statistic to compute.

You can set the [StatType](#) parameter to one or more of the following values.

● For specifying the type of statistic to compute	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MAX	Specifies the maximum pixel value that respects the setting of the Condition parameter.
<input type="checkbox"/> M_MAX_ABS	Specifies the maximum absolute pixel value that respects the setting of the Condition parameter.
<input type="checkbox"/> M_MEAN	Specifies the mean value of the pixels that respect the setting of the Condition parameter.
<input type="checkbox"/> M_MIN	Specifies the minimum pixel value that respects the setting of the Condition parameter.
<input type="checkbox"/> M_MIN_ABS	Specifies the minimum absolute pixel value that respects the setting of the Condition parameter.
<input type="checkbox"/> M_NUMBER	Specifies the number of pixels that respect the setting of the Condition parameter.
<input type="checkbox"/> M_STANDARD_DEVIATION	Specifies the standard deviation value of the pixels that respect the setting of the Condition parameter.
<input type="checkbox"/> M_SUM	Specifies the sum of the pixel values that respect the setting of the Condition parameter.

<input type="checkbox"/> M_SUM_ABS	Specifies the sum of the absolute pixel values that respect the setting of the Condition parameter.
<input type="checkbox"/> M_SUM_OF_SQUARES	Specifies the sum of the squared pixel values that respect the setting of the Condition parameter.

To compute all of the different statistics listed above, set [StatType](#) to the following:

● For computing the statistics	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL	Specifies that all statistics are computed.

Condition

Specifies the condition for the statistical computation.

If no condition is required, set the [Condition](#) parameter to the value below.

● For specifying the condition for the statistical computation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NULL	Specifies that no condition is set. Set CondLow and CondHigh parameters to M_NULL . (summarize)

For conditions that use two limits, set the [Condition](#) parameter to one of the values below.

● For conditions that use two limits	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_IN_RANGE	Specifies that pixels with values between CondLow and CondHigh , inclusive, will be used for statistical calculations.
<input type="checkbox"/> M_OUT_RANGE	Specifies that pixels with values less than CondLow , or greater than CondHigh , will be used for statistical calculations.

For conditions that use one limit, set the [Condition](#) parameter to one of the values below.

● For conditions that use one limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EQUAL	Specifies that only pixels with values equal to CondLow will be used for statistical calculations.
<input type="checkbox"/> M_GREATER	Specifies that only pixels with values greater than CondLow will be used for statistical calculations.
<input type="checkbox"/> M_GREATER_OR_EQUAL	Specifies that only pixels with values greater than or equal to CondLow will be used for statistical calculations.
<input type="checkbox"/> M_LESS	Specifies that only pixels with values less than CondLow will be used for statistical calculations.
<input type="checkbox"/> M_LESS_OR_EQUAL	Specifies that only pixels with values less than or equal to CondLow will be used for statistical calculations.
<input type="checkbox"/> M_MASK	Specifies that only pixels corresponding to the non-zero pixels in the mask buffer will be used for statistical calculations. Specify the mask using the CondLow parameter. (summarize)
<input type="checkbox"/> M_NOT_EQUAL	Specifies that only pixels with values not equal to CondLow will be used for statistical calculations.

CondLow

Specifies the lower limit of the selected condition.

This parameter must be set to one of the following:

--

● For the lower limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NULL	Specifies that no lower limit is required. This setting should only be used when the Condition parameter is also set to M_NULL . (summarize)
<input type="checkbox"/> MIL image buffer identifier	Specifies a 1-band image buffer used as the mask. The buffer must be of the same size as the source image. This setting can only be used when the Condition parameter is set to M_MASK . (summarize)
<input type="checkbox"/> Value	Specifies the lower limit.

CondHigh

Specifies the upper limit of the selected condition.

This parameter must be set to one of the following values.

● For the upper limit	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NULL	Specifies that no upper limit is required. This setting should be used for all conditions, except M_IN_RANGE and M_OUT_RANGE . (summarize)
<input type="checkbox"/> Value	Specifies the upper limit.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimStream

Synopsis

Load, restore, or save an image processing context from/to a file or a memory stream.

Syntax

```
void MimStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load or restore an image processing context from a file or memory stream. All of the image processing context's settings that were in effect when the image processing context was saved will be restored.

Moreover, this function can also save an image processing context to a file or memory stream. All information about the previously allocated image processing context is saved.

To inquire the number of bytes necessary to save an image processing context to memory stream, you should call this function (**MimStream()**) first with **M_INQUIRE_SIZE_BYTE**.

The content saved to memory stream is equivalent to the content saved to file. In addition, any file saved with **MimSave()** is equivalent to a file saved using this function.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using **MimStream()**, you can choose to save a backwards-compatible version of the image processing context, which will work using a version of MIL that is up to one major release older than the current version (depending on which version is specified). For example, if you allocate an image processing context using MIL 9.0 and save it to version 8.0, you can restore this context on a computer where MIL 8.0 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore image processing contexts saved using MIL version 9.0 or above. Settings that do not exist in the lower version will be filled with default values when the image processing context is loaded or restored.

Parameters

MemPtrOrFileName

Specifies the file or memory stream.

● For the file or memory stream

<div><div></div>Value</div>	Description
<div><div><div><div></div></div><div>MIL_TEXT(MIL_TEXT_PTR FileName)</div></div></div>	<div><p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving an image processing context to a file, use the MIC file extension. The function internally handles the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p><p>To open or save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p><p>(summarize)</p></div> <div><div></div><div><div>Parameters</div><div>FileName</div><div>Specifies the drive, directory, and name of the file.</div></div></div>

<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE .
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. The parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (MimStream()) first with M_INQUIRE_SIZE_BYTE . (summarize)

SystemId

Specifies the system on which to restore the image processing context. For [M_INQUIRE_SIZE_BYTE](#), [M_LOAD](#), and [M_SAVE](#), [SystemId](#) is ignored and should be set to [M_NULL](#). For an [M_RESTORE](#) operation, this parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform on the image processing context. This parameter must be set to one of the following values:

● For specifying the type of operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save an image processing context to memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated image processing context.
<input type="checkbox"/> M_RESTORE	Restores an image processing context from a file or memory stream and assigns it an identifier.
<input type="checkbox"/> M_SAVE	Saves an image processing context to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the image processing context. This parameter must be set to one of the following values:

● For specifying the type of stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream.
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)

Version

Specifies the MIL version of the image processing context. This parameter must be set to one of the following values.

● For specifying the MIL version	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default version.

	For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL. For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Specifies the function's control flag and should be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the image processing context.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, **ContextIdPtr** specifies the address of the variable from which to read the image processing context identifier.

For an [M_LOAD](#) operation, the **ContextIdPtr** specifies the address of the variable from which to read the identifier of the image processing context where the file or memory stream content will be loaded.

For an [M_RESTORE](#) operation, the **ContextIdPtr** specifies the address in which to return the identifier of the restored image processing context. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the image processing context, in bytes.

If the size is not required, you can set this parameter to **M_NULL**.

Note that the size of an image processing context will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimThick

Synopsis

Perform a binary or grayscale thickening operation on an image.

Syntax

```
void MimThick(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT NbIteration,
    MIL_INT ProcMode
)
```

Description

This function performs a binary or grayscale thickening on the specified source image for the specified number of iterations.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the resulting image. This parameter must be given an image buffer identifier.

NbIteration

Specifies the number of times to iterate the operation. This parameter must be set to one of the values below.

● For specifying the number of times to iterate the operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TO_IDEMPOTENCE	Specifies the operation will iterate until idempotence is reached.
<input type="checkbox"/> Value >= 0	Specifies the number of iterations. If this parameter is set to 0, the destination buffer will remain unchanged. (summarize)

ProcMode

Specifies the processing mode to use. This parameter must be set to one of the values below.

● For specifying the processing mode to use	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BINARY	Treats non-zero pixels as ones (1) during processing. The resulting non-zero pixels will have all bits set to one. (summarize)
<input type="checkbox"/> M_GRAYSCALE	Uses the source image's gray values for processing. The resulting buffer will also contain gray values. (summarize)

Remarks

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).
- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimThin

Synopsis

Perform a binary or grayscale thinning operation on an image.

Syntax

```
void MimThin(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT NbIteration,
    MIL_INT ProcMode
)
```

Description

This function performs a binary or grayscale thinning on the specified source image for the specified number of iterations.

Parameters

SrcImageBufId

Specifies the identifier of the data source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the resulting image. This parameter must be given an image buffer identifier.

NbIteration

Specifies the number of times to iterate the operation. This parameter must be set to one of the values below.

● For specifying the number of times to iterate the operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TO_SKELETON	Specifies that every object will be reduced to its skeleton.
<input type="checkbox"/> Value >= 0	Specifies the number of iterations. If this parameter is set to 0, the destination buffer will remain unchanged. (summarize)

ProcMode

Specifies the processing mode to use. This parameter must be set to one of the values below.

● For specifying the processing mode to use	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BINARY	Treats non-zero pixels as ones (1) during processing. The resulting non-zero pixels will have all bits set to one. (summarize)
<input type="checkbox"/> M_BINARY2	Treats non-zero pixels as ones (1) during processing and produces a result that better preserves the original topology of the objects in the source image. Similar to M_BINARY , the resulting non-zero pixels will also have all bits set to one. However, this mode produces a result that is more consistent with the one produced using the M_GRAYSCALE processing mode. Note that this processing mode is slower than using M_BINARY processing mode. (summarize)
<input type="checkbox"/> M_GRAYSCALE	Uses the source image's gray values for processing. The resulting pixels will be gray scale.

	(summarize)
--	-----------------------------

Remarks

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).
- This function is optimized for packed binary buffers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimTransform

Synopsis

Perform a Fast Fourier transform (FFT), a Discrete Cosine transform (DCT), or a polar-cartesian coordinates transformation.

Syntax

```
void MimTransform(  
    MIL_ID SrcImageRBufId,  
    MIL_ID SrcImageIBufId,  
    MIL_ID DestImageRBufId,  
    MIL_ID DestImageIBufId,  
    MIL_INT TransformType,  
    MIL_INT ControlFlag  
)
```

Description

This function performs a forward or reverse FFT or DCT transformation on an image. It also performs a cartesian-to-polar or polar-to-cartesian point-to-point coordinates transformation.

Parameters

SrcImageRBufId

Specifies the identifier of the source buffer for the real component of the image.

See the [Parameter associations](#) section for possible values.

SrcImageIBufId

Specifies the identifier of the source buffer for the imaginary component of the image.

See the [Parameter associations](#) section for possible values.

DestImageRBufId

Specifies the identifier of the destination buffer for the real component of the image.

See the [Parameter associations](#) section for possible values.

DestImageIBufId

Specifies the identifier of the destination buffer for the imaginary component of the image.

See the [Parameter associations](#) section for possible values.

TransformType

Specifies the type of transform to perform on the image.

See the [Parameter associations](#) section for possible values.

ControlFlag

Specifies if the transform is a forward transform or a reverse transform.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **SrcImageRBufId**, **SrcImageIBufId**, **DestImageRBufId**, **DestImageIBufId**, **TransformType**, and **ControlFlag** parameters are described in the table below.

- [For selecting the transformation type.](#)

For selecting the transformation type.	
<input type="checkbox"/> TransformType	Description
ControlFlag	
SrcImageRBufId	
SrcImageIBufId	
DestImageRBufId	
DestImageIBufId	
<input type="checkbox"/> M_DCT8X8	Performs a discrete cosine transform on each 8x8 pixel block in the image. (summarize)
<input type="checkbox"/> M_DEFAULT +	Same as M_FORWARD . (summarize)
<input type="checkbox"/> M_FORWARD +	Performs a forward transform on the image. (summarize)
<input type="checkbox"/> SrcImageRBufId	Specifies the identifier of the source buffer for the real component of the image in the spatial domain. The width and height of the source buffer must be a multiple of 8. (summarize)
<input type="checkbox"/> ID of a 8-bit signed/unsigned buffer	Performs calculations in fixed-point format and returns to the destinations in 23.9 fixed point format. (summarize)
<input type="checkbox"/> ID of a 16-bit signed/unsigned buffer	Performs calculations in fixed-point format and returns to the destinations in 25.7 fixed point format. (summarize)
<input type="checkbox"/> ID of a 32-bit signed buffer	Conserves the fixed point format between the destination buffer and the source buffer.
<input type="checkbox"/> ID of a floating-point buffer	Specifies that processing will be performed in floating-point arithmetic. The destination buffer should also be a floating-point buffer. (summarize)
<input type="checkbox"/> SrcImageIBufId	Specifies the identifier of the source buffer for the imaginary component of the image in the spatial domain. This buffer must be of the same type and pixel depth, and have the same dimensions as the buffer specified for SrcImageRBufId . (summarize)
<input type="checkbox"/> DestImageRBufId	Specifies the identifier of the destination buffer for the real component of the image in the frequency domain. If SrcImageRBufId is an integer buffer, this must be a 32-bit signed integer buffer. If SrcImageRBufId is a floating-point buffer, this must also be a floating-point buffer. The width and height of the destination buffer should be a multiple of 8. (summarize)
<input type="checkbox"/> DestImageIBufId	Specifies the identifier of the destination buffer for the imaginary component of the image in the frequency domain. This buffer must be of the same type and pixel depth, and have the same dimensions as the buffer specified for DestImageRBufId . (summarize)
<input type="checkbox"/> M_REVERSE +	Performs a reverse transform on the image. (summarize)
<input type="checkbox"/> SrcImageRBufId	Specifies the identifier of the source buffer for the real component of the image in the frequency domain. This must be a 32-bit integer buffer or a floating-point buffer.

	The width and height of the source buffer must be a multiple of 8. (summarize)
<input type="checkbox"/> SrcImageIBufId	Specifies the identifier of the source buffer for the imaginary component of the image in the frequency domain. This buffer must be of the same type and pixel depth, and have the same dimensions as the buffer specified for SrcImageRBufId . (summarize)
<input type="checkbox"/> DestImageRBufId	Specifies the identifier of the destination buffer for the real component of the image in the spatial domain. The width and height of the destination buffer should be a multiple of 8. (summarize)
<input type="checkbox"/> ID of 8-bit signed/unsigned buffer	The format of the source buffer is assumed to be in 23.9 fixed point format. (summarize)
<input type="checkbox"/> ID of 16-bit signed/unsigned buffer	The format of the source buffer is assumed to be in 25.7 fixed point format. (summarize)
<input type="checkbox"/> ID of 32-bit signed buffer	The fixed point format of the destination buffer is the same as that of the source.
<input type="checkbox"/> ID of floating-point buffer	If SrcImageRBufId is a floating-point buffer, this must also be a floating-point buffer and processing will be performed in floating-point arithmetic.
<input type="checkbox"/> DestImageIBufId	Specifies the identifier of the destination buffer for the imaginary component of the image in the spatial domain. This buffer must be of the same type and pixel depth, and have the same dimensions as the buffer specified for DestImageRBufId . (summarize)
<input type="checkbox"/> M_FFT	Performs a Fast Fourier transform. (summarize)
<input type="checkbox"/> M_DEFAULT +	Same as M_FORWARD . (summarize)
<input type="checkbox"/> M_FORWARD +	Performs a forward transform on the image. For floating-point destination buffers, the processing will be performed in floating-point arithmetic. (summarize)
<input type="checkbox"/> SrcImageRBufId	Specifies the identifier of the source buffer for the real component of the image in the spatial domain. The width and height of the source buffer must be a power of 2. (summarize)
<input type="checkbox"/> ID of 8-bit signed/unsigned buffer	Calculations are performed in fixed-point format and returned to the destinations in 23.9 fixed point format. (summarize)
<input type="checkbox"/> ID of 16-bit signed/unsigned buffer	Calculations are performed in fixed-point format and returned to the destinations in 25.7 fixed point format. (summarize)
<input type="checkbox"/> ID of 32-bit signed buffer	The fixed point format of the destination buffer will be the same as that of the source buffer.
<input type="checkbox"/> ID of floating-point buffer	The destination buffer should also be a floating-point buffer and processing will be performed in floating-point arithmetic.
<input type="checkbox"/> SrcImageIBufId	Specifies the identifier of the source buffer for the imaginary component of the image in the spatial domain. This buffer must be of the same type and pixel depth, and have the same dimensions as the buffer specified for SrcImageRBufId . If this parameter is set to M_NULL , a faster version of the forward transform will be performed. (summarize)
<input type="checkbox"/> DestImageRBufId	Specifies the identifier of the destination buffer for the real component of the image in the frequency domain. If SrcImageRBufId is an integer buffer, this can be a 32-bit signed integer buffer or a floating-point buffer. If SrcImageRBufId is a floating-point buffer, this must be a floating-point buffer also. It must be a 32-bit signed integer or a floating-point buffer. The width and height of the destination buffer must be a power of 2. (summarize)
<input type="checkbox"/> DestImageIBufId	Specifies the identifier of the destination buffer for the imaginary component of the image in the frequency domain. This buffer must be of the same type and pixel depth, and have the same dimensions as the buffer specified for DestImageRBufId .

	(summarize)
<input type="checkbox"/> M_REVERSE +	Performs a reverse transform on the image. (summarize)
<input type="checkbox"/> SrcImageRBufId	Specifies the identifier of the source buffer for the real component of the image in the frequency domain. This must be a 32-bit integer buffer or a floating-point buffer. The source buffer must be a 32-bit signed integer or a floating-point buffer. The width and height of the source buffer must be a power of 2. (summarize)
<input type="checkbox"/> SrcImageIBufId	Specifies the identifier of the source buffer for the imaginary component of the image in the frequency domain. This buffer must be of the same type and pixel depth, and have the same dimensions as the buffer specified for SrcImageRBufId . (summarize)
<input type="checkbox"/> DestImageRBufId	Specifies the identifier of the destination buffer for the real component of the image in the spatial domain. The width and height of the destination buffer must be a power of 2. (summarize)
<input type="checkbox"/> ID of 8-bit signed/unsigned buffer	The format of the source buffer is assumed to be in 23.9 fixed point format. (summarize)
<input type="checkbox"/> ID of 16-bit signed/unsigned buffer	The format of the source buffer is assumed to be in 25.7 fixed point format. (summarize)
<input type="checkbox"/> ID of 32-bit signed buffer	The fixed point format of the destination buffer is the same as that of the source.
<input type="checkbox"/> ID of Floating-point buffer	If SrcImageRBufId is a floating-point buffer, this must also be a floating-point buffer and processing will be performed in floating-point arithmetic.
<input type="checkbox"/> DestImageIBufId	Specifies the identifier of the destination buffer for the imaginary component of the image in the spatial domain. This buffer must be of the same type and pixel depth, and have the same dimensions as the buffer specified for DestImageRBufId . If this parameter is set to M_NULL , a faster version of the reverse transform will be performed. (summarize)
<input type="checkbox"/> M_POLAR	Performs a cartesian-to-polar or a polar-to-cartesian transform on the coordinates of the source buffer. (summarize)
<input type="checkbox"/> M_DEFAULT +	Same as M_FORWARD . (summarize)
<input type="checkbox"/> M_FORWARD +	Converts the real and imaginary components of cartesian coordinates to the magnitude and phase components of polar coordinates. (summarize)
<input type="checkbox"/> SrcImageRBufId	Specifies the identifier of the source buffer for the real component of the cartesian coordinates. It must be a 32-bit floating-point buffer. (summarize)
<input type="checkbox"/> SrcImageIBufId	Specifies the identifier of the source buffer for the imaginary component of the cartesian coordinates. It must be provided. (summarize)
<input type="checkbox"/> DestImageRBufId	Specifies the identifier of the destination buffer for the magnitude component of the polar coordinates. It must be a 32-bit floating-point buffers. If this parameter is set to M_NULL , a faster version of the forward transform will be performed. Only one of the destination buffers can be set to M_NULL . (summarize)
<input type="checkbox"/> DestImageIBufId	Specifies the identifier of the destination buffer for the phase component of the polar coordinates. The phase results are comprised in the range of 0-360. If this parameter is set to M_NULL , a faster version of the forward transform will be performed. Note that only one of the destination buffers can be set to M_NULL . (summarize)
<input type="checkbox"/> M_REVERSE +	Converts the magnitude and phase components of polar coordinates to the real and imaginary components of cartesian coordinates. (summarize)
<input type="checkbox"/> SrcImageRBufId	Specifies the identifier of the source buffer for the magnitude component of the polar coordinates. It must be a 32-bit floating-point buffer.

	(summarize)
<input type="checkbox"/> SrcImageIBufId	Specifies the identifier of the source buffer for the phase component of the polar coordinates. It must be provided. (summarize)
<input type="checkbox"/> DestImageRBufId	Specifies the identifier of the destination buffer for the real component of the cartesian coordinates. It must be a 32-bit floating-point buffers. (summarize)
<input type="checkbox"/> DestImageIBufId	Specifies the identifier of the destination buffer for the imaginary component of the cartesian coordinates. The phase results are comprised in the range of 0-360. (summarize)

Combination constants for [M_DEFAULT](#) (of [M_DCT8X8](#)); [M_FORWARD](#) (of [M_DCT8X8](#)); [M_REVERSE](#) (of [M_DCT8X8](#)); [M_DEFAULT](#) (of [M_FFT](#)); [M_FORWARD](#) (of [M_FFT](#)); [M_REVERSE](#) (of [M_FFT](#));

You can add one or more of the following values to the above-mentioned values to set the transform characteristics.

Note that only [M_NORMALIZE](#) is supported with both FFT and DCT. All other values are supported only with the FFT.

For the ControlFlag parameter	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_1D_COLUMNS	Performs a 1-D transform on all columns of the image. If M_1D_COLUMNS has been specified with the forward transform, then it must be specified for the reverse transform as well. (summarize)
<input type="checkbox"/> M_1D_ROWS	Performs a 1-D transform on all rows of the image. If M_1D_ROWS has been specified with the forward transform, then it must be specified for the reverse transform as well. (summarize)
<input type="checkbox"/> M_CENTER	Centers the real part and the imaginary part of the spectrum. The center of the spectrum is put at the (SizeX/2-1, SizeY/2-1). If M_CENTER has been specified with the forward transform, then it must be specified for the reverse transform as well. (summarize)
<input type="checkbox"/> M_MAGNITUDE +	Specifies whether to compute or use the magnitude. When used with M_FORWARD , computes the magnitude of the forward FFT (the square root of $R^2 + I^2$). The magnitude is returned in the DestImageRBufId destination buffer, thereby overwriting the real component of the destination image. When used with M_REVERSE , computes the real component of the transform from the magnitude and phase ($R = M \cos(P)$). The result will be used to perform the reverse FFT. For a reverse FFT, SrcImageRBufId and SrcImageIBufId must contain the magnitude and phase of a forward transform, respectively. M_MAGNITUDE must be used with M_PHASE for a reverse FFT. For a forward FFT, DestImageIBufId can be set to M_NULL if M_MAGNITUDE is used without M_PHASE . M_MAGNITUDE can only be used in transforms processed with floating-point buffers. (summarize)
<input type="checkbox"/> M_NORMALIZE	Normalizes results (divide the final result by 8 for DCT and by (m x n) for FFT where m x n is the size of the image). M_NORMALIZE must be used with fixed-point arithmetic (all integer buffers) to avoid overflows. If M_NORMALIZE has been specified with the forward transform, then it must be specified for the reverse transform as well. (summarize)
<input type="checkbox"/> M_PHASE	Specifies whether to compute or use the phase. When used with M_FORWARD , computes the phase of the forward transform ($P = \text{atan}(I / R)$). The phase is returned in the DestImageIBufId buffer, thereby

	<p>overwriting the imaginary component of the destination image. The phase is returned in the range of 0° to 360°.</p> <p>When used with M_REVERSE, computes the imaginary component of the transform from the magnitude and phase ($I = M \sin(P)$). The result will be used to perform the reverse FFT.</p> <p>For a reverse FFT, SrcImageRBufId and SrcImageIBufId must contain the magnitude and phase of a forward transform, respectively.</p> <p>M_PHASE must be used with M_MAGNITUDE for a reverse FFT.</p> <p>For a forward FFT, DestImageRBufId can be set to M_NULL if M_PHASE is used without M_MAGNITUDE.</p> <p>M_PHASE can only be used in transforms processed with floating-point buffers. (summarize)</p>
<input type="checkbox"/> M_SATURATION	<p>Clips (saturates) the results of a reverse FFT according to the destination buffer's data type.</p> <p>M_SATURATION can only be used with a reverse FFT (M_FFT + M_REVERSE) operation which uses floating-point source buffers and integer destination buffers. (summarize)</p>

Combination constant for [M_MAGNITUDE](#);

You can add the following value to the above-mentioned value to specify that the magnitude of the forward FFT is scaled logarithmically.

● For M_MAGNITUDE	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_LOG_SCALE	<p>Scales logarithmically the magnitude of the forward FFT to be in the range of 0-255. This value is used to scale the spectrum into a displayable range.</p> <p>M_LOG_SCALE can only be used in forward transforms processed with floating-point buffers. (summarize)</p>

Combination constants for [M_DEFAULT](#) (of [M_POLAR](#)); [M_FORWARD](#) (of [M_POLAR](#)); [M_REVERSE](#) (of [M_POLAR](#));

You can add one of the following values to the above-mentioned values to specify the transform characteristics.

● For M_POLAR	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FAST_PHASE	<p>Performs faster phase computation.</p> <p>For a forward polar transformation, it uses a faster computation of the phase results but slightly less accurate. For a reverse polar transformation, it has no effect. (summarize)</p>
<input type="checkbox"/> M_NORMALIZE_PHASE	<p>Rescales the phase results.</p> <p>For a forward polar transformation, it rescales the phase results to be in the range of 0.0-1.0. For a reverse polar transformation, SrcImageRBufId must contain the phase values in the range of 0.0-1.0. (summarize)</p>
<input type="checkbox"/> M_NORMALIZE_PHASE_255	<p>Rescales the phase results.</p> <p>For a forward polar conversion, it rescales the phase results to be in the range of 0-255. For a reverse polar transformation, SrcImageIBufId must contain the phase values in the range of 0-255. (summarize)</p>
<input type="checkbox"/> M_SQUARE_MAGNITUDE	<p>Specifies that the magnitude values are returned or used squared.</p>

	For a forward polar transformation, it gives the square of the magnitude results (i.e. R^2+I^2). For a reverse transformation, SrcImageRBufId must contain the magnitude values squared. (summarize)
--	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimTranslate

Synopsis

Translate an image in X- and/or Y-displacement.

Syntax

```
void MimTranslate(
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_DOUBLE XDisplacement,
    MIL_DOUBLE YDisplacement,
    MIL_INT InterpolationMode
)
```

Description

This function translates the source image position by the specified amount, writing results to the destination buffer. This function can be used to align images to subpixel accuracy before, for example, subtracting them.

Parameters

SrcImageBufId

Specifies the source image buffer identifier.

DestImageBufId

Specifies the destination image buffer identifier.

XDisplacement

Specifies the amount by which to displace the source image in the X-direction. This parameter can be set to any positive or negative value.

YDisplacement

Specifies the amount by which to displace the source image in the Y-direction. This parameter can be set to any positive or negative value.

InterpolationMode

Specifies the interpolation mode. This parameter must be set to one of the values below.

For specifying the interpolation mode	
Value	Description
M_DEFAULT +	Same as M_BILINEAR + M_OVERSCAN_ENABLE .
M_BILINEAR +	Specifies bilinear interpolation.

Combination constants for any of the possible values of the [InterpolationMode](#) parameter

You can add one of the following values to the above-mentioned values to specify how to determine the value of a destination pixel when its associated point falls outside the source buffer.

For overscan	
Value	Description
	vio (
	solia
	solia
	solia
	odys/
	odys/
	odys/
	nexis/
	morp
	morp
	met-
	met-
	met-
	lins (
	leee
	hello
	hello
	gpu i
	glige
	cronc
	coron

MimWarp

Synopsis

Perform a warping.

Syntax

```
void MimWarp(
    MIL_ID SrcImageId,
    MIL_ID DestImageId,
    MIL_ID WarpParam1Id,
    MIL_ID WarpParam2Id,
    MIL_INT OperationMode,
    MIL_INT InterpolationMode
)
```

Description

This function warps an image through either a 3x3 matrix-defined warping or LUTs. You can perform first-order polynomial warpings or perspective polynomial warpings using either a 3x3 coefficients matrix or lookup tables (LUTs). You can perform custom warpings using LUTs.

A warping associates each pixel position of the destination buffer, (x_d, y_d) , with a specific point in the source buffer, (x_s, y_s) , and then determines the pixel value of (x_d, y_d) from its associated point and from a specified interpolation mode.

When performing a first-order polynomial warping, (x_d, y_d) gets associated with (x_s, y_s) through the following equations:

$$x_s = a_0 x_d + a_1 y_d + a_2$$

$$y_s = b_0 x_d + b_1 y_d + b_2$$

When performing a perspective polynomial warping, (x_d, y_d) gets associated with (x_s, y_s) through the following equations:

$$x_s = (a_0 x_d + a_1 y_d + a_2) / (c_0 x_d + c_1 y_d + c_2)$$

$$y_s = (b_0 x_d + b_1 y_d + b_2) / (c_0 x_d + c_1 y_d + c_2)$$

When performing a 3x3 matrix-defined warping, [WarpParam1Id](#) specifies the required coefficients $(a_0 \dots a_2, b_0 \dots b_2)$ or $(a_0 \dots a_2, b_0 \dots b_2, c_0 \dots c_2)$ and [WarpParam2Id](#) must be set to **M_NULL**. The coefficients can be automatically generated using [MgenWarpParameter\(\)](#) or can be user-supplied.

When performing a warping using LUTs, x_s is determined from (x_d, y_d) through one LUT and y_s is determined from (x_d, y_d) through another LUT. In this case, [WarpParam1Id](#) specifies the LUT for x_s and [WarpParam2Id](#) specifies the LUT for y_s . The LUTs can be user-supplied or can be automatically generated from a 3x3 coefficient matrix (for a first-order or polynomial warping) using [MgenWarpParameter\(\)](#).

Parameters

SrcImageId

Specifies the buffer on which to perform the warping. This buffer can be of any type.

DestImageId

Specifies the buffer in which to place the results of the warping. This buffer can be of any type.

WarpParam1Id

Specifies the buffer containing the matrix coefficients or the LUT buffer from which x_s is determined.

When **WarpParam1Id** specifies the $(a_0 \dots a_2, b_0 \dots b_2)$ or $(a_0 \dots a_2, b_0 \dots b_2, c_0 \dots c_2)$ coefficients, the coefficients must be in a single-band, 32-bit floating-point buffer that has an **M_ARRAY** attribute and that has dimensions 3x2 or 3x3. The first row specifies the a_n coefficients, the second row specifies the b_n coefficients, and the third row specifies the c_n coefficients. If the buffer is 3x2, the third row is assumed to be (0, 0, 1).

When **WarpParam1Id** specifies the LUT buffer from which x_s is determined, the buffer must be a signed 16- or 32-bit integer buffer, have the same X and Y size as the destination image buffer, and have an **M_LUT** attribute.

WarpParam2Id

Specifies the LUT buffer from which y_s is determined. This buffer must be a signed 16- or 32-bit integer buffer, have the same X and Y size as the destination image buffer, and have an **M_LUT** attribute.

If you are not using LUTs to perform the warping, set **WarpParam2Id** to **M_NULL**.

OperationMode

Specifies the mode of operation. This parameter must be set to one of the values below.

● For specifying the mode of operation	
☐ Value	Description
☐ M_WARP_LUT +	Performs the warping through LUTs.
☐ M_WARP_POLYNOMIAL	Performs the warping through a 3x3 coefficient matrix. This includes a first-order polynomial warping or a perspective polynomial warping. Note that with a perspective polynomial warping matrix, MimWarp() might return an error if the calculation of $(c_0x_d + c_1y_d + c_2)$ is 0 in the destination buffer, regardless of whether or not the matrix was generated using MgenWarpParameter() . (summarize)

Combination constant for **M_WARP_LUT**;

You can add the following value to the above-mentioned value to specify the number of fractional bits in the coordinates of the source point.

● For M_WARP_LUT	
☐ Value	Description
☐ M_FIXED_POINT + n	Specifies the number of fractional bits for the source point (x_s, y_s) . To do so, add the define M_FIXED_POINT + n to M_WARP_LUT where $n \geq 0$. If nothing is added to M_WARP_LUT , it is assumed that there are no fractional bits in the coordinates of the source point (and therefore interpolation is not required). (summarize)

InterpolationMode

Specifies the interpolation mode. This parameter must be set to one of the values below.

● For specifying the interpolation mode	
☐ Value	Description
☐ M_DEFAULT +	Same as M_NEAREST_NEIGHBOR + M_OVERSCAN_ENABLE .
☐ M_BICUBIC +	Specifies bicubic interpolation. When using bicubic interpolation, saturation is performed according to the type of the destination buffer. (summarize)
☐ M_BILINEAR +	Specifies bilinear interpolation. No saturation is performed. (summarize)
☐ M_NEAREST_NEIGHBOR +	Specifies nearest neighbor interpolation. No saturation is performed. (summarize)

Combination constants for any of the possible values of the **InterpolationMode** parameter

You can add one of the following values to the above-mentioned values to specify how to determine the value of a destination pixel when its associated point falls outside the source buffer.

● For overscan																											
Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ea/xd (f)	helios ed/xd (g)	hls 1394 i1dc (h)	iris (i)	met-II/cl (j)	met-II/dlg (k)	met-II/mc (l)	met-II/std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ed/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ed/xd (u)	solos gige (v)	vio (w)			
<input type="checkbox"/> M_OVERSCAN_CLEAR	Sets the destination pixel to 0, if the associated point falls outside the source buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_OVERSCAN_DISABLE	Leaves the destination pixel as is.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_OVERSCAN_ENABLE	Uses pixels from the source buffer's ancestor buffer. If the source buffer is not a child buffer or if the point falls outside the ancestor buffer, leave the destination pixel as is. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
	<i>Board specific</i> Points falling outside the source image are undefined.				d													q	r	s	t	u	v				
<input type="checkbox"/> M_OVERSCAN_FAST	Specifies that MIL will automatically select the overscan that optimizes speed, according to the specified operation and the target system. The overscan could be hardware-specific thereby having a different behavior than the other supported overscan modes. Note that when using M_OVERSCAN_FAST , the destination pixels in the overscan area are undefined. The pixels can therefore contain different values from one function call to the next, even if the function's parameter values are the same. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimWatershed

Synopsis

Perform a watershed transformation.

Syntax

```
void MimWatershed(
    MIL_ID SrcImageId,
    MIL_ID MarkerImageId,
    MIL_ID DestImageId,
    MIL_INT MinimumVariation,
    MIL_INT ControlFlag
)
```

Description

This function performs a watershed transformation on the specified source buffer. The catchment basins of the source buffer can be determined from extrema (minima or maxima) in the source buffer and/or from a specified marker image. In the case when no marker image is specified, a valid catchment basin is determined from an extrema when the minimum variation in gray levels of an extremum's catchment basin is equal to or greater than the value specified by the [MinimumVariation](#) parameter. In the case when a marker image is specified, each marker of a marker image produces catchment basin(s) by forcing an extremum in the corresponding area(s) of the source buffer.

There are two types of marker images: non-labeled and labeled. In a non-labeled marker image, each group of touching pixels with the value zero in the marker image (known as a marker) starts a catchment basin in the corresponding area of the source image. Specifically, each marker in the marker image forces a extremum in the corresponding area of the source image. Pixels in the marker image are considered touching if they are vertically, horizontally, or diagonally adjacent, that is, if they are "8-connected". Note that each catchment basin in the destination image is given a unique grayscale value, starting at 1.

In a labeled marker image, a marker is a set of pixels that do not have to necessarily touch and that have all the same label value (pixel intensity). Each marker starts a catchment basin in the corresponding area of the source image and each catchment basin of the destination image is assigned the label value of the marker that generated it. Note that, in a labeled marker image, a marker can touch other markers, allowing you to specify adjacent catchment basins. Valid marker label values are 1 to $2^n - 2$ (where n refers to the number of bits per pixel in the buffer). Pixels with the label value of zero are interpreted as the region of the image that is not a marker. Finally, pixels with the label value of $2^n - 1$ are considered to be part of a "don't care" mask and are not processed, accelerating the watershed transformation.

You can specify that the destination buffer contain one of the following:

- Watershed lines. The watershed lines are given the value 0 and all other pixels are given the maximum value in the destination buffer.
- Labeled catchment basins, without watershed lines. Each catchment basin is assigned a grayscale value greater or equal to 1.
- Labeled catchment basins and watershed lines. Each catchment basin is assigned a grayscale value greater or equal to 1. Watershed lines are given the value 0.

For more information, see the [Watershed transformations](#) section in [Chapter 4: Advanced image processing](#).

Parameters

SrcImageId

Specifies the buffer on which to perform the transform. This buffer can be 8-bit or 16-bit, signed or unsigned.

MarkerImageId

Specifies the buffer to use as a marker image. This buffer can be 8-bit or 16-bit, signed or unsigned. If you are not using a marker image, set this parameter to **M_NULL**.

The default marker image type is non-labeled. If you want to use a labeled marker image, you must add **M_LABELED_MARKER** to the [ControlFlag](#) parameter.

DestImageId

Specifies the buffer in which to place the results of the transformation. This buffer can be 8-bit or 16-bit, signed or unsigned. The destination buffer can hold $2^n - 5$ catchment basins, where n refers to the number of bits per pixel in the buffer. Therefore, an 8-bit destination buffer can hold 251 catchment basins and a 16-bit destination buffer can hold 65531 catchment basins.

MinimumVariation

Specifies the minimum variation of gray-level of a catchment basin. This parameter must be set to one of the values below.

● For specifying the minimum variation of gray-level	
Value	Description
<input type="checkbox"/> M_DEFAULT	Produces a catchment basin from each extremum in the source buffer. Setting MinimumVariation to 1 will have the same effect. (summarize)
<input type="checkbox"/> M_OFF	Specifies that catchment basins will not be determined from extrema in the source buffer. M_NULL will do the same thing. (summarize)
<input type="checkbox"/> Value	Sets the minimum variation.

ControlFlag

Specifies how to perform the transformation. It can be set to any combination of the following seven sets of values below. It can also be set to [M_DEFAULT](#), in which case the default value from each set is used (without [M_SKIP_LAST_LEVEL](#), [M_LABELED_MARKER](#) and [M_FILL_SOURCE](#)). If no value from a set is specified, its default value is used.

● For specifying how to perform the transformation	
Value	Description
<input type="checkbox"/> M_DEFAULT +	Same as M_WATERSHED + M_MINIMA_FILL + M_REGULAR + M_4_CONNECTED .
<input type="checkbox"/> M_BASIN +	Labels catchment basins.
<input type="checkbox"/> M_WATERSHED +	Calculates watershed lines.
<input type="checkbox"/> M_WATERSHED + M_BASIN +	Labels catchment basin and calculates watershed lines.

Combination constants for any of the possible values of the [ControlFlag](#) parameter

You can add one of the following values to the above-mentioned values to specify whether to use the source buffer's minima or maxima when determining catchment basins from extrema in the source buffer.

● For specifying whether to use the source buffer's minima or maxima	
Value	Description
<input type="checkbox"/> M_MAXIMA_FILL	Uses the source buffer's maxima.
<input type="checkbox"/> M_MINIMA_FILL	Uses the source buffer's minima. This is the default value. (summarize)

Combination constants for any of the possible values of the [ControlFlag](#) parameter

You can add one of the following values to the above-mentioned values to specify how the watershed lines are drawn.

● For specifying how the watershed lines are drawn	
Value	Description
<input type="checkbox"/> M_REGULAR	Traces the watershed lines exactly. This is the default value. (summarize)
<input type="checkbox"/> M_STRAIGHT_WATERSHED	Forces the watershed lines to be straight. Note that selecting M_STRAIGHT_WATERSHED automatically selects the M_SKIP_LAST_LEVEL setting. (summarize)

Combination constants for any of the possible values of the [ControlFlag](#) parameter

You can add one of the following values to the above-mentioned values to specify whether 4 or 8-connected watershed lines will be used.

● For specifying 4 or 8-connected watershed lines	
Value	Description
M_4_CONNECTED	Uses 4-connected watershed lines. This is the default value. (summarize)
M_8_CONNECTED	Uses 8-connected watershed lines.

Combination constant for any of the possible values of the [ControlFlag](#) parameter

You can add the following value to the above-mentioned values to specify that an extremum's zone of influence cannot extend past the minimum and maximum gray level.

● For preventing an extremum's zone of influence from extending past the minimum and maximum gray level	
Value	Description
M_SKIP_LAST_LEVEL	Prevents an extremum's zone of influence from extending beyond $L_{\max} - 1$ (for a minimum) or $L_{\min} - 1$ (for a maximum), where L_{\max} is the maximum gray level in the image and L_{\min} is the minimum gray level.

Combination constant for any of the possible values of the [ControlFlag](#) parameter

You can add the following value to the above-mentioned values to specify that the marker image is a labeled marker image.

● For denoting that the marker image is labeled	
Value	Description
M_LABELED_MARKER	Specifies that the marker image is a labeled marker image.

Combination constant for any of the possible values of the [ControlFlag](#) parameter

You can add the following value to the above-mentioned values to specify that any unwanted extrema from the source image is removed.

It removes the extrema by filling the catchment basins of unwanted extrema until they become plateaus.

● For removing unwanted extrema	
Value	Description
M_FILL_SOURCE	Fill all local basins of a valid catchment basin in the source image.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

MimZoneOfInfluence

Synopsis

Perform a zone of influence detection.

Syntax

```
void MimZoneOfInfluence (
    MIL_ID SrcImageBufId,
    MIL_ID DestImageBufId,
    MIL_INT OperationFlag
)
```

Description

This function separates an image into zones, according to how much of a blob's surrounding background is within the blob's territorial boundaries, or "zone of influence". The image is considered to be binary, with background pixels equal to 0 (black), and all non-zero pixels treated as blobs. It gives every pixel in a blob's zone of influence the same value. Each zone of influence is numbered consecutively, beginning with 1. A blob's zone of influence consists of all pixels closer to that blob than to any other blob. There are as many zones as blobs.

Parameters

SrcImageBufId

Specifies the identifier of the source of the operation. This parameter must be given an image buffer identifier.

DestImageBufId

Specifies the identifier of the destination of the resulting image. This parameter must be given an image buffer identifier.

The destination buffer should be deep enough to hold the maximum number of zones (blobs). The maximum label value is $2^n - 5$, where n is the depth of the destination buffer in bits. For example, an 8-bit buffer can be used for a maximum of 251 blobs and a 16-bit buffer can be used for a maximum of 65531 blobs. Note that if the destination buffer depth is too small, several zones might be given the same value.

OperationFlag

Controls the type of 3x3 distance matrix used for the operation. The following table shows each value's respective 3x3 distance matrix for calculating the distance to a neighboring pixel, which is then used for the zone of influence computation.

This parameter must be set to one of the values below.

● For controlling the type of 3x3 distance matrix	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_CHESSBOARD .
<input type="checkbox"/> M_CHAMFER_3_4	Uses a distance algorithm that makes a better approximation to actual Euclidean distance, and is therefore more accurate than M_CHESSBOARD . This is the 3x3 distance matrix: $\begin{bmatrix} 4 & 3 & 4 \\ 3 & 0 & 3 \\ 4 & 3 & 4 \end{bmatrix}$ (summarize)
<input type="checkbox"/> M_CHESSBOARD	Specifies a chessboard matrix. This operation is faster than M_CHAMFER_3_4 . This is the 3x3 distance matrix:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$
[\(summarize\)](#)

Remark

- In-place processing is supported, but the source and destination image buffers cannot partially overlap (a situation that can only occur when using child buffers).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milim.lib.
DLL	Requires mil.dll; milim.dll.

Mmeas functions

Synopsis

The functions prefixed with Mmeas make up the Measurement module. The basic Measurement module is a set of functions that can be used to take measurements using spatial reference positions in images. The control allows you to find sets of markers in an image, based on differences in pixel intensities. Upon finding a marker, the control returns the marker's spatial reference position and measures such features as its width and angle. The measurement control can operate on 8-bit or 16-bit unsigned grayscale images. Measurements are made and results are returned with subpixel accuracy.

Functions

- [MmeasAllocContext](#)
- [MmeasAllocMarker](#)
- [MmeasAllocResult](#)
- [MmeasCalculate](#)
- [MmeasControl](#)
- [MmeasDraw](#)
- [MmeasFindMarker](#)
- [MmeasFree](#)
- [MmeasGetResult](#)
- [MmeasGetResultSingle](#)
- [MmeasInquire](#)
- [MmeasRestoreMarker](#)
- [MmeasSaveMarker](#)
- [MmeasSetMarker](#)
- [MmeasStream](#)

MmeasAllocContext

Synopsis

Allocate a measurement context.

Syntax

```
MIL_ID MmeasAllocContext (
    MIL_ID SystemId,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr
)
```

Description

This function allocates a measurement context on the specified system. Measurement context settings are used to control the behavior of measurement operations ([MmeasFindMarker\(\)](#) and [MmeasCalculate\(\)](#)). When the measurement context is no longer required, you should release its memory, using [MmeasFree\(\)](#).

Note, upon allocation of an application, a default measurement context is automatically allocated. Rather than using **MmeasAllocContext()** to allocate a measurement context, you can use this default measurement context, by specifying **M_DEFAULT** wherever a measurement context identifier is required.

Parameters

SystemId

Specifies the system on which the to allocate the measurement context.

This parameter should be set to one of the following values:

For specifying the system identifier	
Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Specifies the allocation control flag and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which the measurement context identifier is to be written. If allocation fails, **M_NULL** is written as the identifier. Since the **MmeasAllocContext()** function also returns the marker identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the measurement context identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Default values

These measurement context control types are set to the following default values upon allocation and can be changed at any time, using [MmeasControl\(\)](#):

Characteristic	Default Value
M_PIXEL_ASPECT_RATIO	1.0
M_PIXEL_ASPECT_RATIO_INPUT	M_CORRECTED

M_PIXEL_ASPECT_RATIO_OUTPUT	M_CORRECTED
-----------------------------	-------------

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasAllocMarker

Synopsis

Allocate a measurement marker.

Syntax

```
MIL_ID MmeasAllocMarker(  
    MIL_ID SystemId,  
    MIL_INT MarkerType,  
    MIL_INT ControlFlag,  
    MIL_ID *MarkerIdPtr  
)
```

Description

This function allocates a measurement marker on the specified system. Once allocated, a marker's characteristics can be specified, using [MmeasSetMarker\(\)](#). For an edge or stripe marker, these characteristics are used as the criteria for finding the marker in a target image. An edge or stripe marker can be located in a target image and its measurements taken, using [MmeasFindMarker\(\)](#). Once found, an edge or stripe can then be used as a reference position in calculations involving two markers ([MmeasCalculate\(\)](#)). A point marker cannot be searched for, but is placed in the required location as a reference position. When the marker is no longer required, release its memory, using [MmeasFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the marker.

This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

MarkerType

Specifies the type of marker to allocate. Any of the types below can be a multiple marker by changing the number of occurrences (**M_NUMBER**) with [MmeasSetMarker\(\)](#). A multiple marker is more than one instance of the same point, edge, or stripe characteristics.

This parameter must be set to one of the values below.

● For specifying the type of marker to allocate	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EDGE	An edge.
<input type="checkbox"/> M_POINT	A single point.
<input type="checkbox"/> M_STRIPE	A pair of edges.

ControlFlag

Specifies the allocation control flag. This parameter must be set to **M_DEFAULT**.

MarkerIdPtr

Specifies the address of the variable in which the marker identifier is to be written. Since the `MmeasAllocMarker()` function also returns the marker identifier, you can set this parameter to `M_NULL`. If allocation fails, `M_NULL` is written as the identifier.

Return value

The returned value is the marker identifier if the allocation is successful. If allocation fails, `M_NULL` is returned.

Default values

Marker characteristics are set to the following default values upon allocation and can be changed at any time, using `MmeasSetMarker()`. Note, a characteristic with the value `M_ANY` will not be considered as a criteria for finding the marker in a target image.

Characteristic	Default Value
Number of points, edges, or stripes	1
Spacing between edges or stripes	<code>M_ANY</code>
Spacing variation	<code>M_ANY</code>
Marker orientation	<code>M_VERTICAL</code> (vertical orientation)
Marker polarity	<code>M_ANY</code> for edge markers. <code>M_ANY</code> for the first (from top-left) edge of a stripe marker. <code>M_OPPOSITE</code> (opposite polarity from the first edge) for the second edge of a stripe marker.
Marker contrast	<code>M_ANY</code>
Marker contrast variation	<code>M_ANY</code>
Marker width	<code>M_ANY</code>
Marker width variation	<code>M_ANY</code>
Marker position	<code>M_ANY</code>
Marker position variation	<code>M_ANY</code>
Size of measurement box	<code>M_DEFAULT</code> (image size)
Origin of measurement box	Top-left corner of the image
	X-coordinate = 0.0
	Y-coordinate = 0.0
Measurement box's center	<code>M_DEFAULT</code> (image size)
Measurement box's angle	0.0
Marker reference	<code>M_DEFAULT</code> (center of the marker)
Edge strength	<code>M_ANY</code>
Edge strength variation	<code>M_ANY</code>
Edge threshold	2.0 (2%)
Stripe inside edges	<code>M_ANY</code>
Stripe inside-edge variation	<code>M_ANY</code>
Stripe inside position	<code>M_ANY</code>
Box angle mode	<code>M_DISABLE</code>
Box angle delta negative and Box angle delta positive	<code>M_DEFAULT</code> (180° search for a symmetrical stripe marker; 360°, complete rotation, for non-symmetrical).
Box angle tolerance	5.0°
Box angle accuracy	<code>M_DISABLE</code>
Box angle interpolation	<code>M_BILINEAR</code>
Box angle reference	<code>M_BOX_CENTER</code>
Weight factors	Weight of 50% to the strength of an edge and the remaining 50% to any other characteristics not set to <code>M_ANY</code> .

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasAllocResult

Synopsis

Allocate a measurement result buffer.

Syntax

```
MIL_ID MmeasAllocResult(  
    MIL_ID SystemId,  
    MIL_INT ResultType,  
    MIL_ID *MeasResultIdPtr  
)
```

Description

This function allocates a buffer, on the specified system, to be used for storing measurement results obtained from a [MmeasCalculate\(\)](#) operation. When the result buffer is no longer required, you should release its memory, using [MmeasFree\(\)](#). Note, a result buffer is not required for a [MmeasFindMarker\(\)](#) operation.

Parameters

SystemId

Specifies the system on which to allocate the result buffer.

This parameter should be set to one of the following values:

For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ResultType

Specifies the type of result buffer to allocate.

This parameter can be set to one of the following values:

For specifying the type of result buffer to allocate	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_CALCULATE .
<input type="checkbox"/> M_CALCULATE	Specifies to allocate a result buffer for a MmeasCalculate() operation.

MeasResultIdPtr

Specifies the address of the variable in which the measurement result identifier is to be written. If allocation fails, **M_NULL** is returned as the identifier. Since the **MmeasAllocResult()** function also returns the measurement result identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasCalculate

Synopsis

Calculate measurements between two markers.

Syntax

```
void MmeasCalculate(
    MIL_ID ContextId,
    MIL_ID Marker1Id,
    MIL_ID Marker2Id,
    MIL_ID MeasResultId,
    MIL_INT MeasurementList
)
```

Description

This function calculates the specified measurements between two markers. Edge and stripe markers must have been previously located in the target image with [MmeasFindMarker\(\)](#). The position of point markers must have been previously set, using [MmeasSetMarker\(\)](#).

The measurement context settings will control the behavior of this function and can be set with [MmeasControl\(\)](#).

Results are stored in the specified measurement result buffer and can be obtained using [MmeasGetResult\(\)](#).

Parameters

ContextId

Specifies the identifier of the measurement context. This parameter must be given a valid measurement context identifier or can be set to **M_DEFAULT**, in which case, the default measurement context of the current MIL application will be used.

Marker1Id

Specifies the identifier of the marker to use as the first reference position for calculating measurements.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The marker ([Marker1Id](#)) must be allocated on the same system as the measurement context ([ContextId](#)). If it is not, an error will occur.

Marker2Id

Specifies the identifier of the marker to use as the second reference position for calculating measurements.

If [Marker1Id](#) and [Marker2Id](#) are multiple markers, then calculations are made using the edges or stripes of the first marker and the corresponding points, edges, or stripes in the second marker, for the entire first marker's number of points, edges, or stripes. The number of calculations performed is limited to the smallest number of points, edges, or stripes held in either marker (that is, if a marker contains only one point, edge, or stripe, then only one calculation is performed, regardless of the number of points, edges, or stripes contained in the first marker).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The marker ([Marker2Id](#)) must be allocated on the same system as the measurement context ([ContextId](#)). If it is not, an error will occur.

MeasResultId

Specifies the identifier of the result buffer in which to place results.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The result buffer ([MeasResultId](#)) must be allocated on the same system as the measurement context ([ContextId](#)). If it is not, an error will occur.

MeasurementList

Specifies which measurement(s) to calculate. The table below lists the measurements that can be calculated. To calculate more than one measurement, add the values together (for example, [M_DISTANCE](#) + [M_ANGLE](#)):

● For specifying which measurement(s) to to calculate	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ANGLE + M_DISTANCE + M_LINE_EQUATION .
<input type="checkbox"/> M_ANGLE	Calculates the angle of the line joining two markers, relative to the positive X-axis. The value can be any between 0° and 360°. (summarize)
<input type="checkbox"/> M_DISTANCE	Calculates the distance between both markers.
<input type="checkbox"/> M_LINE_EQUATION	Calculates the equation of the line joining both markers.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasControl

Synopsis

Control a measurement context setting.

Syntax

```
void MmeasControl(  
    MIL_ID ContextId,  
    MIL_INT ControlType,  
    MIL_DOUBLE ControlValue  
)
```

Description

This function sets the specified measurement context processing control. Measurement context settings control the behavior of measurement operations.

Parameters

- ContextId
- Specifies the identifier of the measurement context.
- ControlType
- Specifies the processing feature to control. Note, when referring to "data relative to the corrected image", this means that the data (for example, a specified position) is given as if it were relative to an image resized by a factor equal to the aspect ratio.
- See the [Parameter associations](#) section for possible values.
- ControlValue
- Specifies the value needed for the control. Note, when referring to "data relative to the corrected image", this means that the data (for example, a specified position) is given as if it were relative to an image resized by a factor equal to the aspect ratio.
- See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the table below.

- [For measurement contexts](#)

For measurement contexts	
ControlType	Description
ControlValue	
M_PIXEL_ASPECT_RATIO	Sets the ratio of the width of the pixel to its height. (summarize)
<div>Value</div>	Specifies the pixel width to pixel height ratio. The default value is 1.0. (summarize)
M_PIXEL_ASPECT_RATIO_INPUT	Sets how the Measurement module interprets specified measurement characteristics relative to the pixel aspect ratio.

	(summarize)
<input type="checkbox"/> M_CORRECTED	Specifies measurement characteristics are relative to the corrected image. This is the default value. (summarize)
<input type="checkbox"/> M_NORMAL	Specifies that input data is relative to the given image (ratio of 1.0). (summarize)
<input type="checkbox"/> M_PIXEL_ASPECT_RATIO_OUTPUT	Sets how the Measurement module returns results relative to the pixel aspect ratio. (summarize)
<input type="checkbox"/> M_CORRECTED	Specifies that results are relative to the corrected image. This is the default value. (summarize)
<input type="checkbox"/> M_NORMAL	Specifies that output data is relative to the given image (ratio of 1.0). (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasDraw

Synopsis

Draw specific expected marker characteristics, or features of marker occurrences or measurement results, in the destination image buffer.

Syntax

```
void MmeasDraw(
    MIL_ID GraphContId,
    MIL_ID MarkerOrResultId,
    MIL_ID DestImageId,
    MIL_INT Operation,
    MIL_INT Index,
    MIL_INT ControlFlag
)
```

Description

This function draws specific expected marker characteristics, or features of marker occurrences or measurement results, in the destination buffer. This function can draw multiple features at once.

Parameters

GraphContId

Specifies the graphics context to use. This parameter must be set to one of the following values:

● For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

MarkerOrResultId

Specifies the identifier of the measurement marker buffer or the measurement result buffer from which to retrieve the features to draw. Note that the measurement marker buffer stores both the expected marker characteristics (specified using [MmeasSetMarker\(\)](#)), and the actual marker characteristics found in the target image (using [MmeasFindMarker\(\)](#)). To specify which features to retrieve from this buffer, use the [ControlFlag](#) parameter.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
The marker or result buffer must be allocated on the same system as the graphics context buffer ([GraphContId](#)). If it is not, an error will occur.

DestImageId

Specifies the identifier of the destination image buffer in which to draw. The buffer can be any supported MIL image buffer. By drawing into a display's overlay buffer, you can also annotate an image non-destructively.

Operation

Specifies the type of operation to perform. The possible [Operation](#) parameter values in the tables below can be added together to draw multiple features at once.

The possible [Operation](#) parameter values for a measurement marker buffer are:

● For specifying the type of operation	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_DRAW_POSITION +	<p>Draws a cross at the expected or found position and angle of the marker.</p> <p>Note that when drawing at the expected position, the expected marker position must have been set to a specific numeric value using MmeasSetMarker() with M_POSITION_X and M_POSITION_Y. The expected position cannot be set to M_ANY (default value); an error message is generated.</p> <p>(summarize)</p>
--------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The possible [Operation](#) parameter values for an edge or stripe measurement marker buffer are:

● For an edge or stripe measurement marker buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_ARROW	Draws an arrow in the center of the measurement box in the direction of the edge transition.
<input type="checkbox"/> M_DRAW_BOX	Draws the measurement box, respecting the position and angle.
<input type="checkbox"/> M_DRAW_BOX_CENTER	Draws the measurement box center, respecting the position and angle.
<input type="checkbox"/> M_DRAW_EDGES	<p>Draws a line along the expected or found edges of the marker.</p> <p>Note that when drawing the expected edges, the expected marker position must have been set to a specific numeric value using MmeasSetMarker() with M_POSITION_X and M_POSITION_Y. The expected position cannot be set to M_ANY (default value); an error message is generated.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DRAW_POSITION_VARIATION	<p>Draws an H-type line (-) indicating possible position variation of the marker.</p> <p>Note that when drawing the expected position variation, the expected position variation must have been set to a specific numeric value using MmeasSetMarker() with M_POSITION_VARIATION. The expected position variation cannot be set to M_ANY (default value); an error message is generated.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DRAW_WIDTH	<p>Draws an H-type line (-) along the expected or found width of the stripe marker.</p> <p>Note that when drawing along the expected stripe width, the expected width must have been set to a specific numeric value using MmeasSetMarker() with M_WIDTH. The expected width cannot be set to M_ANY (default value); an error message is generated.</p> <p>This setting is not available for an edge measurement marker buffer.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DRAW_WIDTH_VARIATION	<p>Draws an H-type line (-) indicating possible width variation of the marker.</p> <p>Note that when drawing the expected width variation, the expected width variation must have been set to a specific numeric value using MmeasSetMarker() with M_WIDTH_VARIATION. The expected width variation cannot be set to M_ANY (default value); an error message is generated.</p> <p>This setting is not available for an edge measurement marker buffer.</p> <p>(summarize)</p>

The possible [Operation](#) parameter values for result occurrences in a measurement marker buffer are:

● For result occurrences in a measurement marker buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_EDGES_PROFILE +	<p>Draws the results' edge profile.</p> <p>This setting can be combined with other values to control its effect. See below.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DRAW_SPACING	Draws an H-type line (-) indicating the found spacing.
<input type="checkbox"/> M_DRAW_SUB_POSITIONS	Draws a cross (+) at the found positions (in the center and at the same angle) of the subedges.

Combination constants for the values listed in [For specifying the type of operation](#); and for the following value: [M_DRAW_EDGES_PROFILE](#);

You can add one of the following values to the above-mentioned values to specify whether to draw the edge profile at the measurement box, or whether to draw the first or second exterior edge of stripe.

● For M_DRAW_POSITION and M_DRAW_EDGE_PROFILE	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_IN_BOX	<p>Draws the edge profile at the position and angle of the measurement box.</p> <p>This can be only added to M_DRAW_EDGES_PROFILE.</p>

	(summarize)
<input type="checkbox"/> M_EDGE_FIRST	<p>Draws the specified feature for the first exterior edge of a stripe.</p> <p>Note that when drawing expected features, the expected width and orientation must have been set to a specific numeric value using MmeasSetMarker() with M_WIDTH_VARIATION and M_ORIENTATION. The expected width and orientation cannot be set to M_ANY (default value); an error message is generated.</p> <p>(summarize)</p>
<input type="checkbox"/> M_EDGE_SECOND	<p>Draws the specified feature for the second exterior edge of a stripe.</p> <p>Note that when drawing expected features, the expected width and orientation must have been set to a specific numeric value using MmeasSetMarker() with M_WIDTH_VARIATION and M_ORIENTATION. The expected width and orientation cannot be set to M_ANY (default value); an error message is generated.</p> <p>(summarize)</p>

The possible [Operation](#) parameter value for results in a measurement result buffer is:

● For results in a measurement result buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_LINE	Draws the line joining 2 markers.

Index

Specifies the index of the result to draw for a measurement marker buffer or the measurement result buffer. This parameter must be set to one of the values below.

● For specifying the index	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	<p>Same as M_ALL.</p> <p>When drawing the expected features of a marker, this parameter must be set to M_DEFAULT. Note that when drawing the expected features of a marker which is a multiple marker, only the first instance of the marker is drawn.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ALL	Draws the result for all occurrences.
<input type="checkbox"/> Value	Specifies the index of the result to draw.

ControlFlag

Specifies whether to draw the expected features of a marker or the features of a result. This parameter must be set to one of the values below.

● For specifying whether to draw the expected features of a marker or the features of a result	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_RESULT .
<input type="checkbox"/> M_MARKER	<p>Draws the expected features of a marker (specified using MmeasSetMarker()).</p> <p>Note that the MarkerOrResultId parameter must be set to a measurement marker buffer.</p> <p>(summarize)</p>
<input type="checkbox"/> M_RESULT	Draws the features of a result.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasFindMarker

Synopsis

Find a marker in an image and take the specified measurements.

Syntax

```
void MmeasFindMarker(
    MIL_ID ContextId,
    MIL_ID ImageBufId,
    MIL_ID MarkerId,
    MIL_INT MeasurementList
)
```

Description

This function finds an edge or a stripe marker in the image and takes the measurements specified in the measurement list. The marker's characteristics are used to help locate the marker and can be set using [MmeasSetMarker\(\)](#).

Measurement context settings will control the behavior of this function and can be set, using [MmeasControl\(\)](#).

It is recommended that the measurement box be set to an angle close to that of the required marker (that is, within the marker's particular rotation tolerance) to ensure that the marker is found and that results are accurate: the greater the angle of the marker relative to the search region, the greater the distortion of the marker's actual characteristics and the chance that the marker might not be successfully located with **MmeasFindMarker()**.

Results are stored with the marker (not in a result buffer), and can be obtained using [MmeasGetResult\(\)](#).

Note, all positional results are relative to the top-left pixel in the target image, or the origin of the coordinate system if the target is a calibrated image.

Parameters

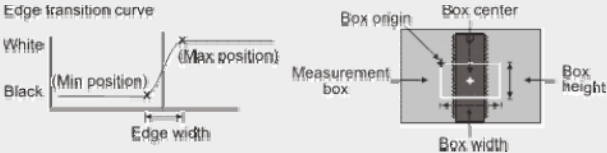
- ContextId
- Specifies the identifier of the measurement context. This parameter must be given a valid measurement context identifier or can be set to **M_DEFAULT**, in which case, the default measurement context of the current MIL application will be used.
- ImageBufId
- Specifies the identifier of the image buffer in which to locate the marker.
- MarkerId
- Specifies the identifier of the marker to be located in the image.
- [Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
- The marker ([MarkerId](#)) must be allocated on the same system as the measurement context ([ContextId](#)). If it is not, an error will occur.
- MeasurementList
- Specifies which measurement(s) to perform. To take more than one measurement, add the values together (for example, [M_POSITION](#) + [M_ANGLE](#)). Use [M_DEFAULT](#) to select all of the measurements.
- The measurement list can contain the following values:

For specifying the measurement(s) to perform	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_DEFAULT	Takes all of the measurements below.
<input type="checkbox"/> M_ANGLE	Determines the angle of the marker relative to the positive X-axis of the target image. The value can be between 0° and 360°. (summarize)
<input type="checkbox"/> M_CONTRAST	Determines the typical difference in average grayscale values between the start and end of the intensity transition (edge width) from which an edge is established. For a stripe marker, the values should be the typical difference in average grayscale values for each of its edges. (summarize)
<input type="checkbox"/> M_EDGE_INSIDE	Determines the number of edges located between the two exterior edges of a stripe marker.
<input type="checkbox"/> M_LENGTH	Determines the length of the marker, in pixels or real-world units. The length is limited to the dimensions of the measurement box. (summarize)
<input type="checkbox"/> M_LINE_EQUATION	Determines the equation of the mean line following an edge. When performing this calculation for a stripe marker, three line equations are calculated: the line equation of the stripe's first edge, the line equation of the stripe's second edge, and the mean of the line equations of both its edges. (summarize)
<input type="checkbox"/> M_NUMBER	Determines the number of edges or stripes found in the target image. This measurement is always selected in the measurement list. (summarize)
<input type="checkbox"/> M_POLARITY	Determines whether the edge (or edges of a stripe marker) are rising (positive polarity) or falling (negative polarity).
<input type="checkbox"/> M_POSITION +	Determines the X- and Y-coordinates of the center of the marker within the measurement box.
<input type="checkbox"/> M_POSITION_VARIATION	Determines the position variation (or uncertainty) of the marker, in pixels or real-world units. If the marker is an edge, the position variation is equal to half of its edge width. If the marker is a stripe, the position variation is the sum of the position variations of both of its edges. (summarize)
<input type="checkbox"/> M_WIDTH	Determines the edge width of an edge marker or the width of a stripe marker in pixels or real-world units. The edge width of an edge marker is a measure of the transition in grayscale values from which it is established. The width of a stripe marker is the average distance between its edges and is calculated at the angle of the measurement box. (summarize)
<input type="checkbox"/> M_WIDTH_VARIATION	Determines the width variation (or uncertainty) of a stripe marker, in pixels or real-world units. It is the sum of the position variations of both of its edges. (summarize)

Combination constants for [M_POSITION](#);

You can add one of the following values to the above-mentioned value to determine edge and marker information.

● For M_POSITION	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BOX_EDGE_VALUES	Determines the edge value for every possible profile value of the measurement box. Therefore, for a vertical marker, the number of returned values is equal to the width of the measurement box; for a horizontal marker, the number is equal to the height. The value is represented as a normalized percentage of the image buffer's maximum possible value. (summarize)
<input type="checkbox"/> M_EDGE_STRENGTH	Determines the minimum/maximum edge value along the edge width of an edge.
<input type="checkbox"/> M_ORIENTATION	Determines the orientation of the marker.
<input type="checkbox"/> M_POSITION_MAX	<p>Determines the X- and Y-coordinates of the maximum position of an edge in a marker.</p> <p>For a stripe marker, this determines the average value of maximum positions unless otherwise specified (using MmeasGetResult() or MmeasGetResultSingle() with M_EDGE_FIRST or M_EDGE_SECOND).</p> <p>The X-coordinate is the maximum position within the edge width from which an edge is established. The Y-coordinate is the Y-coordinate of the measurement box center. The maximum position is always on the side of the measurement box furthest from the origin, regardless of how the measurement box is rotated. The diagram below illustrates an edge transition curve (a profile of gradual transition from dark to light) and a measurement box.</p> 

	(summarize)
<input type="checkbox"/> M_POSITION_MIN	<p>Determines the X- and Y-coordinates of the minimum position of an edge in a marker.</p> <p>For a stripe marker, this determines the average value of minimum positions unless otherwise specified (using MmeasGetResult() or MmeasGetResultSingle() with M_EDGE_FIRST or M_EDGE_SECOND).</p> <p>The X-coordinate is the minimum position within the edge width from which an edge is established. The Y-coordinate is the Y-coordinate of the measurement box center. The minimum position is always on the side of the measurement box adjacent the origin, regardless of how the measurement box is rotated. The diagram in M_POSITION_MAX illustrates an edge transition curve and a measurement box.</p> <p>(summarize)</p>
<input type="checkbox"/> M_SPACING	<p>Determines the inter-edge or inter-stripe spacing of a multiple marker.</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasFree

Synopsis

Free a measurement context, marker, or result buffer.

Syntax

```
void MmeasFree(  
    MIL_ID MeasId  
)
```

Description

This function deletes the specified marker, result buffer, or context identifier and releases any memory associated with it.

Parameter

MeasId

Specifies the measurement identifier to free. The identifier must have been successfully allocated with [MmeasAllocMarker\(\)](#), [MmeasAllocResult\(\)](#), or [MmeasAllocContext\(\)](#) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasGetResult

Synopsis

Get the specified type of result(s) for all edges, stripes, or points, from a measurement marker or result buffer.

Syntax

```
void MmeasGetResult(
    MIL_ID MarkerOrMeasResultId,
    MIL_INT ResultType,
    void *FirstResultArrayPtr,
    void *SecondResultArrayPtr
)
```

Description

This function retrieves the result(s) of the specified type for all edges, stripes, or points from a measurement marker or result buffer. Results should be obtained from a marker if an [MmeasFindMarker\(\)](#) operation was performed or from a measurement result buffer if an [MmeasCalculate\(\)](#) operation was performed. Note that either [MmeasFindMarker\(\)](#) or [MmeasCalculate\(\)](#) must be called prior to using this function, otherwise incorrect results will be returned.

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image.

For certain result types, one value is returned for an edge marker, while two values are returned for a stripe marker. In these cases, for an edge marker, all results are returned to [FirstResultArrayPtr](#), and [SecondResultArrayPtr](#) must be set to [M_NULL](#).

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [FirstResultArrayPtr](#) and [SecondResultArrayPtr](#) parameters to [M_NULL](#). When these parameters are set to [M_NULL](#), the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [MmeasGetResult\(\)](#) again and you pass an array to at least one of the result parameters. You must ensure that the array(s) is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

Note, if a request queued does not return results in the second result array, nothing is added to that array as a placeholder for that particular request.

By setting the [ResultType](#) parameter to [M_INTERACTIVE](#), you can view the results interactively.

Parameters

MarkerOrMeasResultId
Specifies the identifier of the measurement marker buffer (allocated with [MmeasAllocMarker\(\)](#)) or measurement result buffer (allocated with [MmeasAllocResult\(\)](#)) from which to retrieve results.

ResultType
Specifies the type of result(s) to retrieve or opens a dialog box that displays the results stored in the measurement marker or result buffer.

To display the results in an interactive dialog box, select the following.

Unless otherwise specified, the following values require that you pass the [FirstResultArrayPtr](#) parameter and the [SecondResultArrayPtr](#) parameter [M_NULL](#).

● For displaying results in an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	<div>[This is only applicable to Windows]</div> <div>Opens a dialog box containing a spreadsheet that displays the types of results stored in the measurement marker or result buffer. The results are displayed in separate columns. If there are queued requests, only those results are displayed.</div>

([summarize](#))

To retrieve a result from an edge or stripe marker, or from a measurement result buffer, [ResultType](#) can be set to one of the values specified in the table below.

Unless otherwise specified, the following values require that you pass the address of an array of type `MIL_DOUBLE` with a size equal to the number of marker occurrences found to the [FirstResultArrayPtr](#) parameter. In addition, you must pass `M_NULL` to the [SecondResultArrayPtr](#) parameter.

For edge and stripe marker, or measurement result buffers	
Value	Description
M_ANGLE +	<p>Retrieves the angles of the measured lines, in degrees, relative to the positive X-axis.</p> <p>If retrieved from a marker, M_ANGLE returns the angle of the line measured for each marker occurrence. For stripe markers, this line follows the center of the marker occurrence and is calculated as the mean of both outside edge lines.</p> <p>If retrieved from a measurement result buffer, M_ANGLE returns the angle of the lines joining each of the occurrences of the two markers, relative to the positive X-axis.</p> <p>(summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values: 0.0 to 360.0 Angle, in degrees</p>
M_LINE_EQUATION +	<p>Retrieves the measured line equations.</p> <p>If retrieved from a marker, M_LINE_EQUATION returns the line measured for each marker occurrence. For stripe markers, this line follows the center of the marker occurrence and is calculated as the mean of both outside edge lines.</p> <p>If retrieved from a measurement result buffer, M_LINE_EQUATION returns the equations of the lines joining each of the occurrences of the two markers.</p> <p>(summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type <code>MIL_DOUBLE</code> Array size: The number of marker occurrences. Return values: Value Slope of the line. When the equation of the line has an infinite slope, the value returned as the slope is M_INFINITE_SLOPE (1.0E+300).</p> <p><i>SecondResultArrayPtr info</i> Data type: array of type <code>MIL_DOUBLE</code> Array size: The number of marker occurrences. Return values: Value Y-intercept of the line.</p>
M_LINE_EQUATION_INTERCEPT +	<p>Retrieves the Y-intercepts of the measured lines.</p> <p>If retrieved from a marker, M_LINE_EQUATION_INTERCEPT returns the Y-intercept of the line measured for each marker occurrence. For stripe markers, this line follows the center of the marker occurrence and is calculated as the mean of both outside edge lines.</p> <p>If retrieved from a measurement result buffer, M_LINE_EQUATION_INTERCEPT returns the Y-intercept of the lines joining each of the occurrences of the two markers.</p> <p>(summarize)</p>
M_LINE_EQUATION_SLOPE +	<p>Retrieves the slopes of the measured lines.</p> <p>If retrieved from an edge marker, M_LINE_EQUATION_SLOPE returns the slope of the line measured for each marker occurrence. For stripe markers, this line follows the center of the marker occurrence, and is calculated as the mean of both outside edge lines.</p> <p>If retrieved from a measurement result buffer, M_LINE_EQUATION_SLOPE returns the slopes of the lines joining each of the occurrences of the two markers.</p> <p>(summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values: Value Slope of the line. When the equation of the line has an infinite slope, the value returned as the slope is M_INFINITE_SLOPE (1.0E+300).</p>

If retrieving a result from a point, edge, or stripe marker, [ResultType](#) can be set to one of the values specified in the table below.

Unless otherwise specified, the following values require that you pass the address of an array of type `MIL_DOUBLE` with a size equal to the number of marker occurrences found to the [FirstResultArrayPtr](#) parameter. In addition, you must pass `M_NULL` to the [SecondResultArrayPtr](#) parameter.

For point, edge, and stripe marker buffers			
Value	Description		
M_NUMBER +	Retrieves the number of edges or stripes found in the measurement box. After a call to the MmeasFindMarker() function, this number is equal to the number of occurrences of a marker found in the measurement box. After a call to the MmeasCalculate() function, this number is equal to the smallest number of occurrences held in either marker involved in the calculation. (summarize)		
M_POSITION +	Retrieves the coordinates of each of the marker occurrence's centers in the image. (summarize) <table><tr><td><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: ValueX-position.</td></tr><tr><td><i>SecondResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: ValueY-position.</td></tr></table>	<i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: ValueX-position.	<i>SecondResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: ValueY-position.
<i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: ValueX-position.			
<i>SecondResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: ValueY-position.			
M_SCORE +	Retrieves the confidence score, as a percentage, for each marker occurrence.		
M_SPACING +	Retrieves the inter-edge or inter-stripe spacing between consecutive edges or stripes.		
M_TOTAL_SCORE +	Retrieves the average confidence score of all edges or stripes found.		
M_VALID_FLAG +	Retrieves the flag that denotes whether or not a marker was found. (summarize) <table><tr><td><i>FirstResultArrayPtr info</i> Return values: M_FALSEMarker not found.</td></tr><tr><td>M_TRUEMarker found.</td></tr></table>	<i>FirstResultArrayPtr info</i> Return values: M_FALSEMarker not found.	M_TRUEMarker found.
<i>FirstResultArrayPtr info</i> Return values: M_FALSEMarker not found.			
M_TRUEMarker found.			

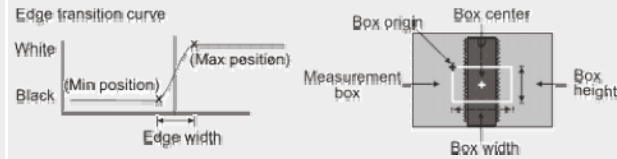
If retrieving a result from an edge or stripe marker, **ResultType** can be set to one of the values specified in the table below.

Unless otherwise specified, the following values require that you pass the address of an array of type `MIL_DOUBLE` with a size equal to the number of marker occurrences found to the [FirstResultArrayPtr](#) parameter. In addition, you must pass `M_NULL` to the [SecondResultArrayPtr](#) parameter.

For edge and stripe marker buffers	
Value	Description
M_BOX_CORNER_BOTTOM_LEFT +	<p>Retrieves the coordinates of the bottom-left corner of the measurement box.</p> <p>If the measurement box was rotated using <code>MmeasSetMarker()</code> with <code>M_BOX_ANGLE</code> or from an angular search, the position returned will take that rotation into account.</p> <p>(summarize)</p> <div><div><i>FirstResultArrayPtr info</i></div><div>Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: ValueX-position.</div></div> <div><i>SecondResultArrayPtr info</i></div>

	<p>Return values:</p> <p>Value Edge value results returned as an array.</p>
<input type="checkbox"/> M_BOX_EDGE_VALUES_NUMBER +	<p>Retrieves the number of edge values to be returned by M_BOX_EDGE_VALUES. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values: Value Number of edge values returned by M_BOX_EDGE_VALUES.</p>
<input type="checkbox"/> M_CONTRAST +	<p>Retrieves the typical difference in average grayscale values between the start and end of the intensity transition (edge width) from which an edge is established. Retrieves one value for each occurrence of an edge marker, and two for each occurrence of a stripe marker. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: Value Average grayscale difference for the first edge.</p> <p><i>SecondResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: Value Average grayscale difference for the second edge if a stripe marker.</p>
<input type="checkbox"/> M_EDGE_INSIDE +	Retrieves the number of edges located between the two exterior edges of each occurrence of a stripe marker.
<input type="checkbox"/> M_EDGE_STRENGTH +	<p>Retrieves the maximum edge value variation (as a normalized percentage of the buffer's maximum possible value) of each edge. Retrieves one value for each occurrence of an edge marker, and two for each occurrence of a stripe marker. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: the number of marker occurrences</p> <p><i>SecondResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: the number of marker occurrences</p>
<input type="checkbox"/> M_LENGTH +	Retrieves the length of each occurrence of the marker, in pixels or real-world units.
<input type="checkbox"/> M_ORIENTATION +	<p>Retrieves the orientation of each occurrence of the marker. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i> Return values: M_HORIZONTAL; M_VERTICAL; (details)</p>
<input type="checkbox"/> M_POLARITY +	<p>Retrieves the polarity of each occurrence of the marker. Retrieves one value for each occurrence of an edge marker, and two for each occurrence of a stripe marker. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: M_NEGATIVE; M_POSITIVE; (details)</p> <p><i>SecondResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of marker occurrences. Return values: M_NEGATIVE; M_POSITIVE; (details)</p>
<input type="checkbox"/> M_POSITION_MAX +	<p>Retrieves the X- and Y-coordinates of the maximum position of an edge in each occurrence of a marker. For a stripe marker, this returns the average value of maximum positions unless otherwise specified (with M_EDGE_FIRST or M_EDGE_SECOND). The X-coordinate is the maximum position within the edge width from which an edge is established. The Y-coordinate is the Y-coordinate of the measurement box center. The maximum position is always on the side of the measurement box furthest from the origin, regardless of how the measurement box is rotated. The</p>

diagram below illustrates an edge transition curve (a profile of gradual transition from dark to light) and a measurement box.



(summarize)

FirstResultArrayPtr info

Data type: array of type MIL_DOUBLE
Array size: The number of marker occurrences.

Return values:

Value X-position.

SecondResultArrayPtr info

Data type: array of type MIL_DOUBLE
Array size: The number of marker occurrences.

Return values:

Value Y-position.

☐ M_POSITION_MIN +

Retrieves the X- and Y-coordinates of the minimum position of an edge in each occurrence of a marker.

For a stripe marker, this returns the average value of minimum positions unless otherwise specified (with [M_EDGE_FIRST](#) or [M_EDGE_SECOND](#)).

The X-coordinate is the minimum position within the edge width from which an edge is established. The Y-coordinate is the Y-coordinate of the measurement box center. The minimum position is always on the side of the measurement box adjacent to the origin, regardless of how the measurement box is rotated. The diagram in [M_POSITION_MAX](#) illustrates an edge transition curve and a measurement box.

(summarize)

FirstResultArrayPtr info

Data type: array of type MIL_DOUBLE
Array size: The number of marker occurrences.

Return values:

Value X-position.

SecondResultArrayPtr info

Data type: array of type MIL_DOUBLE
Array size: The number of marker occurrences.

Return values:

Value Y-position.

☐ M_POSITION_VARIATION +

Retrieves the position variation of each occurrence of a marker, in pixels or real-world units (postive or negative values).

☐ M_SUB_EDGES_INDEX +

Returns the index of the occurrence of the marker on which each subedge is located. This result is only useful when retrieving results for a multiple marker. You can use the marker index to separate by marker the values obtained with [M_SUB_EDGES_POSITION](#) and [M_SUB_EDGES_WEIGHT](#).

For all subedges belonging to the same marker occurrence, the same index is returned. Even if a subedge is not found in a subregion, an index is returned. Use [M_SUB_EDGES_WEIGHT](#) to determine if a subedge was actually found.

(summarize)

FirstResultArrayPtr info

Data type: array of type MIL_DOUBLE
Array size: For edge markers, the array size must be (number of occurrences) x (number of subregions). For stripe markers, the array size must be 2 x (number of occurrences) x (number of subregions).

Return values:

Value For edge markers, FirstResultArrayPtr returns the index of the edge marker occurrence on which each subedge is located. For stripe markers, FirstResultArrayPtr returns the index of the stripe marker occurrence on which each subedge is located.

SecondResultArrayPtr info

Data type: array of type MIL_DOUBLE
Array size: For stripe markers, the array size must be 2 x (number of occurrences) x (number of subregions).

Return values:

	<table><tr><td>Value</td><td>For a stripe marker, SecondResultArrayPtr returns the edge on which each subedge is located (M_EDGE_FIRST or M_EDGE_SECOND).</td></tr></table>	Value	For a stripe marker, SecondResultArrayPtr returns the edge on which each subedge is located (M_EDGE_FIRST or M_EDGE_SECOND).		
Value	For a stripe marker, SecondResultArrayPtr returns the edge on which each subedge is located (M_EDGE_FIRST or M_EDGE_SECOND).				
<div><div></div><div>M_SUB_EDGES_POSITION</div><div>+</div></div>	<div>Returns the X- and Y-position of the center of the subedges of the marker.</div> <div>If a multiple marker was used, the positions of the subedges in the first occurrence of the marker will be followed by the positions of the second occurrence of the marker, and so on.</div> <div>If a subedge is not found in a subregion, the X- and Y- position will both be returned as an arbitrary number. Use M_SUB_EDGES_WEIGHT to determine whether or not a subedge was found.</div> <div>(summarize)</div> <div><div><div>FirstResultArrayPtr info</div><div>Data type: array of type MIL_DOUBLE</div><div>Array size: For edge markers, the array size must be (number of occurrences) x (number of subregions). For stripe markers, the array size must be 2 x (number of occurrences) x (number of subregions).</div><div>Return values:</div><table><tr><td>Value</td><td>X-position of center.</td></tr></table></div><div><div>SecondResultArrayPtr info</div><div>Data type: array of type MIL_DOUBLE</div><div>Array size: For edge markers, the array size must be (number of occurrences) x (number of subregions). For stripe markers, the array size must be 2 x (number of occurrences) x (number of subregions).</div><div>Return values:</div><table><tr><td>Value</td><td>Y-position of center.</td></tr></table></div></div>	Value	X-position of center.	Value	Y-position of center.
Value	X-position of center.				
Value	Y-position of center.				
<div><div></div><div>M_SUB_EDGES_WEIGHT</div><div>+</div></div>	<div>Returns the weight of the subedges of the marker.</div> <div>If a multiple marker was used, the weights of the subedges in the first occurrence of the marker will be followed by the weights of the second occurrence of the marker, and so on.</div> <div>(summarize)</div> <div><div><div>FirstResultArrayPtr info</div><div>Data type: array of type MIL_DOUBLE</div><div>Array size: For edge markers, the array size must be (number of occurrences) x (number of subregions). For stripe markers, the array size must be 2 x (number of occurrences) x (number of subregions).</div><div>Return values:</div><table><tr><td>0</td><td>Indicates the subedge was not found and will not be used to calculate the fit, angle, and line equation values.</td></tr><tr><td>1</td><td>Indicates the subedge was found and will be used to calculate the fit, angle, and line equation values.</td></tr></table></div></div>	0	Indicates the subedge was not found and will not be used to calculate the fit, angle, and line equation values.	1	Indicates the subedge was found and will be used to calculate the fit, angle, and line equation values.
0	Indicates the subedge was not found and will not be used to calculate the fit, angle, and line equation values.				
1	Indicates the subedge was found and will be used to calculate the fit, angle, and line equation values.				
<div><div></div><div>M_WIDTH</div><div>+</div></div>	<div>Retrieves the edge width of each occurrence of the edge marker or the width of each occurrence of the stripe marker, in pixels or real-world units.</div>				

If retrieving a result from a stripe marker, [ResultType](#) can be set to one of the values specified in the table below.

Unless otherwise specified, the following values require that you pass the address of an array of type MIL_DOUBLE with a size equal to the number of marker occurrences found to the [FirstResultArrayPtr](#) parameter. In addition, you must pass M_NULL to the [SecondResultArrayPtr](#) parameter.

● For stripe marker buffers	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_WIDTH_VARIATION +	Retrieves the width variation of each occurrence of the stripe marker, in pixels or real-world units.

Combination constants for [the values listed in](#) all tables **except** **For displaying results in an interactive dialog box**, **For a measurement result buffer**

You can add one of the following values to the above-mentioned values to get specific edge results.

● For all the result types	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EDGE_FIRST	Retrieves results for the first exterior edge of a stripe, for the requested characteristic.

<input type="checkbox"/> M_EDGE_SECOND	Retrieves results for the second exterior edge of a stripe, for the requested characteristic.
<input type="checkbox"/> M_MAX	Retrieves the maximum value for the requested characteristic in all the edges or stripes found. Only one result is returned. (summarize)
	<i>FirstResultArrayPtr and SecondResultArrayPtr info</i> Data type: MIL_DOUBLE
<input type="checkbox"/> M_MEAN	Retrieves the mean value for the requested characteristic in all the edges or stripes found. Only one result is returned. (summarize)
	<i>FirstResultArrayPtr and SecondResultArrayPtr info</i> Data type: MIL_DOUBLE
<input type="checkbox"/> M_MIN	Retrieves the minimum value for the requested characteristic in all the edges or stripes found. Only one result is returned. (summarize)
	<i>FirstResultArrayPtr and SecondResultArrayPtr info</i> Data type: MIL_DOUBLE
<input type="checkbox"/> M_STANDARD_DEVIATION	Retrieves the standard deviation of the values for the requested characteristic in all the edges or stripes found. Only one result is returned. (summarize)
	<i>FirstResultArrayPtr and SecondResultArrayPtr info</i> Data type: MIL_DOUBLE

If results are being obtained from a measurement result buffer ([MmeasCalculate\(\)](#)), the [ResultType](#) parameter can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the address of an array of type MIL_DOUBLE with a size equal to the number of occurrences of the marker found the fewest times to the [FirstResultArrayPtr](#) parameter. In addition, you must pass M_NULL to the [SecondResultArrayPtr](#) parameter.

● For a measurement result buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DISTANCE +	Retrieves the distance between two markers, in pixels or real-world units.
<input type="checkbox"/> M_DISTANCE_X +	Retrieves the distance on the X-axis between two markers, in pixels or real-world units.
<input type="checkbox"/> M_DISTANCE_Y +	Retrieves the distance on the Y-axis between two markers, in pixels or real-world units.

Combination constants for [the values listed in](#) all tables **except** **For displaying results in an interactive dialog box**

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

● For specifying the data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_DOUBLE	Casts the requested results to a <i>double</i> . (summarize)
	<i>FirstResultArrayPtr and SecondResultArrayPtr info</i> <ul style="list-style-type: none"> Data type: array of type double Array size: Large enough to hold results of all occurrences of the marker. Note: When multiple results. Data type: double Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.
<input type="checkbox"/> M_TYPE_LONG	Casts the requested results to a <i>long</i> . (summarize)
	<i>FirstResultArrayPtr and SecondResultArrayPtr info</i> <ul style="list-style-type: none"> Data type: array of type long

	<p>Array size: Large enough to hold results of all occurrences of the marker. Note: When multiple results.</p> <ul style="list-style-type: none"> Data type: long <p>Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.</p>
<input type="checkbox"/> M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> Data type: array of type MIL_DOUBLE Array size: Large enough to hold results of all occurrences of the marker. Note: When multiple results. Data type: MIL_DOUBLE Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.
<input type="checkbox"/> M_TYPE_MIL_INT	<p>Casts the requested results to a <i>MIL_INT</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> Data type: array of type MIL_INT Array size: Large enough to hold results of all occurrences of the marker. Note: When multiple results. Data type: MIL_INT Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.
<input type="checkbox"/> M_TYPE_MIL_INT32	<p>Casts the requested results to a <i>MIL_INT32</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> Data type: array of type MIL_INT32 Array size: Large enough to hold results of all occurrences of the marker. Note: When multiple results. Data type: MIL_INT32 Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.
<input type="checkbox"/> M_TYPE_MIL_INT64	<p>Casts the requested results to a <i>MIL_INT64</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> Data type: array of type MIL_INT64 Array size: Large enough to hold results of all occurrences of the marker. Note: When multiple results. Data type: MIL_INT64 Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.

FirstResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type double
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- double
- long
- M_NULL
- MIL_DOUBLE
- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address of the first array in which to write the requested information.

With a multiple marker, results for each edge or stripe are held in an array. The results for each occurrence of the marker will be stored in a separate element of the specified arrays.

All positional results are relative to the top-left pixel in the target image or to the origin of the coordinate system of a calibrated image.

You must set this parameter to **M_NULL** if you are setting the [ResultType](#) parameter to [M_INTERACTIVE](#).

SecondResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type double
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- double
- long
- M_NULL
- MIL_DOUBLE
- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address of the second array in which to write the requested information.

For most result types, only [FirstResultArrayPtr](#) will be used, and [SecondResultArrayPtr](#) should be set to **M_NULL**.

With a multiple marker, results for each edge or stripe are held in an array. The results for each occurrence of the marker will be stored in a separate element of the specified arrays.

You must set this parameter to **M_NULL** if you are setting the [ResultType](#) parameter to [M_INTERACTIVE](#).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasGetResultSingle

Synopsis

Get the specified type of result for a specific edge, stripe, or point from a measurement marker or result buffer.

Syntax

```
void MmeasGetResultSingle(
    MIL_ID MarkerOrMeasResultId,
    MIL_INT ResultType,
    void *FirstResultArrayPtr,
    void *SecondResultArrayPtr,
    MIL_INT ResultIndex
)
```

Description

This function retrieves the result of the specified type for an edge, stripe, or point from a measurement marker or result buffer. A result should be obtained from a marker if an [MmeasFindMarker\(\)](#) operation was performed or from a measurement result buffer if an [MmeasCalculate\(\)](#) operation was performed. Note that either [MmeasFindMarker\(\)](#) or [MmeasCalculate\(\)](#) must be called prior to using this function, otherwise an incorrect result will be returned.

Note that if your target image was associated with a calibration object, the result is, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image.

For certain result types, one value is returned for an edge marker, while two values are returned for a stripe marker. In these cases, for an edge marker, the result is returned to **FirstResultArrayPtr**, and **SecondResultArrayPtr** must be set to **M_NULL**.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set **FirstResultArrayPtr** and **SecondResultArrayPtr** parameters to **M_NULL**. When these parameters are set to **M_NULL**, the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call **MmeasGetResultSingle()** again and you pass an array to at least one of the result parameters. You must ensure that the array(s) is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

Note, if a request queued does not return a result in the second result array, nothing is added to that array as a placeholder for that particular request.

By setting the **ResultType** parameter to [M_INTERACTIVE](#), you can view the results, for a specified edge or stripe, interactively.

Parameters

MarkerOrMeasResultId

Specifies the identifier of the measurement marker (allocated with [MmeasAllocMarker\(\)](#)) or measurement result buffer (allocated with [MmeasAllocResult\(\)](#)) from which to retrieve results.

ResultType

Specifies the type of result to retrieve or opens a dialog box that displays the results stored in the measurement marker or result buffer.

To display the results in an interactive dialog box, select the following.

Unless otherwise specified, the following values require that you pass the [FirstResultArrayPtr](#) parameter and the [SecondResultArrayPtr](#) parameter **M_NULL**.

● For displaying results in an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] Opens a dialog box containing a spreadsheet that displays the types of results stored in the measurement marker or result buffer. The results are displayed in

separate columns. If there are queued requests, only those results are displayed.
([summarize](#))

To retrieve a result from an edge, or stripe measurement marker occurrence or result buffer, **ResultType** can be set to one of the values specified in the table below.

Unless otherwise specified, the following values require that you pass the address of a MIL_DOUBLE to the [FirstResultArrayPtr](#) parameter. In addition, you must pass M_NULL to the [SecondResultArrayPtr](#) parameter.

● For edge and stripe markers, or measurement result buffers	
☐ Value	Description
☐ M_ANGLE +	<p>Retrieves the angle of the measured line, in degrees, relative to the positive X-axis.</p> <p>If retrieved from a marker, M_ANGLE returns the angle of the line measured for the marker occurrence. For stripe markers, this line follows the center of the marker occurrence and is calculated as the mean of both outside edge lines.</p> <p>If retrieved from a measurement result buffer, M_ANGLE returns the angle of the line joining the occurrence of the two markers, relative to the positive X-axis.</p> <p>(summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values: 0.0 to 360.0 Angle, in degrees</p>
☐ M_LINE_EQUATION +	<p>Retrieves the measured line equation.</p> <p>If retrieved from a marker, M_LINE_EQUATION returns the line measured for the marker occurrence. For stripe markers, this line follows the center of the marker occurrence and is calculated as the mean of both outside edge lines.</p> <p>If retrieved from a measurement result buffer, M_LINE_EQUATION returns the equation of the line joining the occurrence of the two markers.</p> <p>(summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value Slope of the line. When the equation of the line has an infinite slope, the value returned as the slope is M_INFINITE_SLOPE (1.0E+300).</p> <p><i>SecondResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value Y-intercept of the equation.</p>
☐ M_LINE_EQUATION_INTERCEPT +	<p>Retrieves the Y-intercept of the measured line.</p> <p>If retrieved from a marker, M_LINE_EQUATION_INTERCEPT returns the Y-intercept of the line measured for the marker occurrence. For stripe markers, this line follows the center of the marker occurrence and is calculated as the mean of both outside edge lines.</p> <p>If retrieved from a measurement result buffer, M_LINE_EQUATION_INTERCEPT returns the Y-intercept of the line joining the occurrence of the two markers.</p> <p>(summarize)</p>
☐ M_LINE_EQUATION_SLOPE +	<p>Retrieves the slope of the measured line.</p> <p>If retrieved from an edge marker, M_LINE_EQUATION_SLOPE returns the slope of the line measured for the marker occurrence. For stripe markers, this line follows the center of the marker occurrence, and is calculated as the mean of both outside edge lines.</p> <p>If retrieved from a measurement result buffer, M_LINE_EQUATION_SLOPE returns the slope of the line joining the occurrence of the two markers.</p> <p>(summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values: Value Slope of the line. When the equation of the line has an infinite slope, the value returned as the slope is M_INFINITE_SLOPE (1.0E+300).</p>

If retrieving a result from a point, edge, or stripe marker, **ResultType** can be set to one of the values specified in the table below.


Unless otherwise specified, the following values require that you pass the address of a MIL_DOUBLE to the [FirstResultArrayPtr](#) parameter. In addition, you must pass M_NULL to the [SecondResultArrayPtr](#) parameter.



● For point, edge, or stripe measurement markers	
--------------------------------------------------	--

Value	Description						
M_NUMBER +	Retrieves the number of edges or stripes found in the measurement box. After a call to the MmeasFindMarker() function, this number is equal to the number of occurrences of a marker found in the measurement box. After a call to the MmeasCalculate() function, this number is equal to the smallest number of occurrences held in either marker involved in the calculation. (summarize)						
M_POSITION +	Retrieves the coordinates of the marker occurrence's center in the image. (summarize) <table><tr><td><i>FirstResultArrayPtr info</i> Data type: MIL_DOUBLE Return values:</td><td>Value</td><td>X-position.</td></tr><tr><td><i>SecondResultArrayPtr info</i> Data type: MIL_DOUBLE Return values:</td><td>Value</td><td>Y-position.</td></tr></table>	<i>FirstResultArrayPtr info</i> Data type: MIL_DOUBLE Return values:	Value	X-position.	<i>SecondResultArrayPtr info</i> Data type: MIL_DOUBLE Return values:	Value	Y-position.
<i>FirstResultArrayPtr info</i> Data type: MIL_DOUBLE Return values:	Value	X-position.					
<i>SecondResultArrayPtr info</i> Data type: MIL_DOUBLE Return values:	Value	Y-position.					
M_SCORE +	Retrieves the confidence score, as a percentage, for the marker occurrence.						
M_SPACING +	Retrieves the inter-edge or inter-stripe spacing between the specified marker occurrence and the next one found.						
M_TOTAL_SCORE +	Retrieves the average confidence score of all edges or stripes found.						
M_VALID_FLAG +	Retrieves the flag that denotes whether or not a marker was found. (summarize) <table><tr><td><i>FirstResultArrayPtr info</i> Return values:</td><td>M_FALSE</td><td>Marker not found.</td></tr><tr><td></td><td>M_TRUE</td><td>Marker found.</td></tr></table>	<i>FirstResultArrayPtr info</i> Return values:	M_FALSE	Marker not found.		M_TRUE	Marker found.
<i>FirstResultArrayPtr info</i> Return values:	M_FALSE	Marker not found.					
	M_TRUE	Marker found.					

If retrieving an edge or stripe result from a marker, **ResultType** can be set to one of the values specified in the table below.

Unless otherwise specified, the following values require that you pass the address of a MIL_DOUBLE to the [FirstResultArrayPtr](#) parameter. In addition, you must pass M_NULL to the [SecondResultArrayPtr](#) parameter.


For edge or stripe measurement markers

Value	Description
<div>  M_BOX_CORNER_BOTTOM_LEFT + </div>	<div> Retrieves the coordinates of the bottom-left corner of the measurement box. If the measurement box was rotated using <code>MmeasSetMarker()</code> with <code>M_BOX_ANGLE</code> or from an angular search, the position returned will take that rotation into account. (summarize) </div> <div> <div> <i>FirstResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value X-position. </div> <div> <i>SecondResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value Y-position. </div> </div>
<div>  M_BOX_CORNER_BOTTOM_RIGHT + </div>	<div> Retrieves the coordinates of the bottom-right corner of the measurement box. If the measurement box was rotated using <code>MmeasSetMarker()</code> with <code>M_BOX_ANGLE</code> or from an angular search, the position returned will take that rotation into account. (summarize) </div>

	<p><i>FirstResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value X-position.</p> <p><i>SecondResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value Y-position.</p>
M_BOX_CORNER_TOP_LEFT +	<p>Retrieves the coordinates of the top-left corner of the measurement box. If the measurement box was rotated using MmeasSetMarker() with M_BOX_ANGLE or from an angular search, the position returned will take that rotation into account. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value X-position.</p> <p><i>SecondResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value Y-position.</p>
M_BOX_CORNER_TOP_RIGHT +	<p>Retrieves the coordinates of the top-right corner of the measurement box. If the measurement box was rotated using MmeasSetMarker() with M_BOX_ANGLE or from an angular search, the position returned will take that rotation into account. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value X-position.</p> <p><i>SecondResultArrayPtr info</i> Data type: MIL_DOUBLE Return values: Value Y-position.</p>
M_BOX_EDGE_VALUES +	<p>Retrieves the edge values (as a normalized percentage of the buffer's maximum possible value for every profile value of the measurement box). (summarize)</p> <p><i>FirstResultArrayPtr info</i> Data type: array of type MIL_DOUBLE Array size: The number of edge values returned can be determined by calling MmeasGetResultSingle() with M_BOX_EDGE_VALUES_NUMBER. Return values: Value Edge value results returned as an array.</p>
M_BOX_EDGE_VALUES_NUMBER +	<p>Retrieves the number of edge values to be returned by M_BOX_EDGE_VALUES. (summarize)</p> <p><i>FirstResultArrayPtr info</i> Return values: Value Number of edge values returned by M_BOX_EDGE_VALUES.</p>
M_CONTRAST +	<p>Retrieves the typical difference in average grayscale values between the start and end of the intensity transition (edge width) from which an edge is established. Retrieves one value for an edge marker, and two for a stripe marker. (summarize)</p> <p><i>FirstResultArrayPtr info</i></p>

	<div><div>Data type: MIL_DOUBLE</div><div>Return values:</div><div><div>Value</div><div>Average grayscale difference for the first edge.</div></div></div> <div><div>SecondResultArrayPtr info</div><div>Data type: MIL_DOUBLE</div><div>Return values:</div><div><div>Value</div><div>Average grayscale difference for the second edge if a stripe marker.</div></div></div>
<div><div></div><div>M_EDGE_INSIDE +</div></div>	Retrieves the number of edges located between the two exterior edges of the stripe marker occurrence.
<div><div></div><div>M_EDGE_STRENGTH +</div></div>	Retrieves the maximum edge value variation (as a normalized percentage of the buffer's maximum possible value) of each edge of a marker occurrence. Retrieves one value for an edge marker occurrence, and two for a stripe marker occurrence. (summarize) <div><div>FirstResultArrayPtr info</div><div>Data type: MIL_DOUBLE</div><div>SecondResultArrayPtr info</div><div>Data type: MIL_DOUBLE</div></div>
<div><div></div><div>M_LENGTH +</div></div>	Retrieves the length of the marker occurrence, in pixels or real-world units.
<div><div></div><div>M_ORIENTATION +</div></div>	Retrieves the orientation of the marker. (summarize) <div><div>FirstResultArrayPtr and SecondResultArrayPtr info</div><div>Return values: M_HORIZONTAL; M_VERTICAL; (details)</div></div>
<div><div></div><div>M_POLARITY +</div></div>	Retrieves the polarity of the marker occurrence. Retrieves one value for an edge marker occurrence, and two for a stripe marker occurrence. (summarize) <div><div>FirstResultArrayPtr info</div><div>Data type: MIL_DOUBLE</div><div>Return values: M_NEGATIVE; M_POSITIVE; (details)</div><div>SecondResultArrayPtr info</div><div>Data type: MIL_DOUBLE</div><div>Return values: M_NEGATIVE; M_POSITIVE; (details)</div></div>
<div><div></div><div>M_POSITION_MAX +</div></div>	Retrieves the X- and Y-coordinates of the maximum position of an edge in a marker. For a stripe marker, the average value of maximum positions is returned unless otherwise specified (with M_EDGE_FIRST or M_EDGE_SECOND). The X-coordinate is the maximum position within the edge width from which an edge is established. The Y-coordinate is the Y-coordinate of the measurement box center. The maximum position is always on the side of the measurement box furthest from the origin, regardless of how the measurement box is rotated. The diagram below illustrates an edge transition curve (a profile of gradual transition from dark to light) and a measurement box. <div><div>Edge transition curve</div><div></div></div> (summarize) <div><div>FirstResultArrayPtr info</div><div>Data type: MIL_DOUBLE</div><div>Return values:</div><div><div>Value</div><div>X-position.</div></div><div>SecondResultArrayPtr info</div><div>Data type: MIL_DOUBLE</div><div>Return values:</div><div><div>Value</div><div>Y-position.</div></div></div>

<input type="checkbox"/> M_POSITION_MIN +	<p>Returns the X- and Y-coordinates of the minimum position of an edge in a marker.</p> <p>For a stripe marker, the average value of minimum positions is returned unless otherwise specified (with M_EDGE_FIRST or M_EDGE_SECOND).</p> <p>The X-coordinate is the minimum position within the edge width from which an edge is established. The Y-coordinate is the Y-coordinate of the measurement box center. The minimum position is always on the side of the measurement box adjacent to the origin, regardless of how the measurement box is rotated. The diagram in M_POSITION_MAX illustrates an edge transition curve and a measurement box.</p> <p>(summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Data type: MIL_DOUBLE</p> <p>Return values:</p> <table> <tr> <td>Value</td><td>X-position.</td></tr> </table> </div> <div> <p><i>SecondResultArrayPtr info</i></p> <p>Data type: MIL_DOUBLE</p> <p>Return values:</p> <table> <tr> <td>Value</td><td>Y-position.</td></tr> </table> </div>	Value	X-position.	Value	Y-position.
Value	X-position.				
Value	Y-position.				
<input type="checkbox"/> M_POSITION_VARIATION +	<p>Retrieves the position variation of the marker occurrence, in pixels or real-world units (positive or negative values).</p>				
<input type="checkbox"/> M_SUB_EDGES_INDEX +	<p>Returns the index of the occurrence of the marker on which each subedge is located.</p> <p>The same index is returned for each subedge. Even if a subedge is not found in a subregion, an index is returned. Use M_SUB_EDGES_WEIGHT to determine if a subedge was actually found.</p> <p>(summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: For edge markers, the array size must be equal to the number of subregions. For stripe markers, the array size must be 2 x (number of subregions).</p> <p>Return values:</p> <table> <tr> <td>Value</td><td>For edge markers, FirstResultArrayPtr returns the index of the edge marker occurrence on which each subedge is located. For stripe markers, FirstResultArrayPtr returns the index of the stripe marker occurrence on which each subedge is located.</td></tr> </table> </div> <div> <p><i>SecondResultArrayPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: For stripe markers, the array size must be 2 x (number of subregions).</p> <p>Return values:</p> <table> <tr> <td>Value</td><td>For a stripe marker, SecondResultArrayPtr returns the edge on which each subedge is located (M_EDGE_FIRST or M_EDGE_SECOND).</td></tr> </table> </div>	Value	For edge markers, FirstResultArrayPtr returns the index of the edge marker occurrence on which each subedge is located. For stripe markers, FirstResultArrayPtr returns the index of the stripe marker occurrence on which each subedge is located.	Value	For a stripe marker, SecondResultArrayPtr returns the edge on which each subedge is located (M_EDGE_FIRST or M_EDGE_SECOND).
Value	For edge markers, FirstResultArrayPtr returns the index of the edge marker occurrence on which each subedge is located. For stripe markers, FirstResultArrayPtr returns the index of the stripe marker occurrence on which each subedge is located.				
Value	For a stripe marker, SecondResultArrayPtr returns the edge on which each subedge is located (M_EDGE_FIRST or M_EDGE_SECOND).				
<input type="checkbox"/> M_SUB_EDGES_POSITION +	<p>Returns the X- and Y-position of the center of the subedges of the marker occurrence.</p> <p>If a subedge is not found in a subregion, the X- and Y- position will both be returned as an arbitrary number. Use M_SUB_EDGES_WEIGHT to determine whether or not a subedge was found.</p> <p>(summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: For edge markers, the array size must be equal to the number of subregions. For stripe markers, the array size must be 2 x (number of subregions).</p> <p>Return values:</p> <table> <tr> <td>Value</td><td>X-position of center.</td></tr> </table> </div> <div> <p><i>SecondResultArrayPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: For edge markers, the array size must be equal to the number of subregions. For stripe markers, the array size must be 2 x (number of subregions).</p> <p>Return values:</p> <table> <tr> <td>Value</td><td>Y-position of center.</td></tr> </table> </div>	Value	X-position of center.	Value	Y-position of center.
Value	X-position of center.				
Value	Y-position of center.				
<input type="checkbox"/> M_SUB_EDGES_WEIGHT +	<p>Returns the weight of the subedges of the marker occurrence.</p> <p>(summarize)</p> <div> <p><i>FirstResultArrayPtr info</i></p> <p>Data type: array of type MIL_DOUBLE</p> <p>Array size: For edge markers, the array size must be equal to the number of subregions. For stripe markers, the array size must be 2 x (number of subregions).</p> <p>Return values:</p> </div>				

	0	Indicates the subedge was not found and will not be used to calculate the fit, angle, and line equation values.
	1	Indicates the subedge was found and will be used to calculate the fit, angle, and line equation values.
<input type="checkbox"/> M_WIDTH +	Retrieves the edge width of the edge marker occurrence or the width of the stripe marker occurrence, in pixels or real-world units.	

If retrieving a stripe result from a marker, **ResultType** can be set to one of the values specified in the table below.

Unless otherwise specified, the following values require that you pass the address of a `MIL_DOUBLE` to the [FirstResultArrayPtr](#) parameter. In addition, you must pass `M_NULL` to the [SecondResultArrayPtr](#) parameter.

● For stripe measurement markers		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_WIDTH_VARIATION +	Retrieves the width variation of the stripe marker occurrence, in pixels or real-world units.	

Combination constants for the values listed in all tables **except For displaying results in an interactive dialog box**,

You can add one of the following values to the above-mentioned values to get specific marker results.

● For results obtained from a marker		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_EDGE_FIRST	Retrieves results for the first exterior edge of a stripe, for the requested characteristic.	
<input type="checkbox"/> M_EDGE_SECOND	Retrieves results for the second exterior edge of a stripe, for the requested characteristic.	

If retrieving a result from a measurement result buffer (`MmeasCalculate()`), the **ResultType** can be set to one of the values specified in the table below.

Unless otherwise specified, the following values require that you pass the address of a `MIL_DOUBLE` to the [FirstResultArrayPtr](#) parameter. In addition, you must pass `M_NULL` to the [SecondResultArrayPtr](#) parameter.

● For measurement result buffers		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_DISTANCE +	Retrieves the distance between two markers, in pixels or real-world units.	
<input type="checkbox"/> M_DISTANCE_X +	Retrieves the distance on the X-axis between two markers, in pixels or real-world units.	
<input type="checkbox"/> M_DISTANCE_Y +	Retrieves the distance on the Y-axis between two markers, in pixels or real-world units.	

Combination constants for the values listed in all tables **except For displaying results in an interactive dialog box**,

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

● For specifying the required data type		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_TYPE_DOUBLE	Casts the requested results to a <i>double</i> . (summarize)	
	<i>FirstResultArrayPtr and SecondResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type double Array size: Depends on the result type. Note: When multiple results. • Data type: double Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr. 	

<div> <div></div> <div>M_TYPE_LONG</div> </div>	<p>Casts the requested results to a <i>long</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type long Array size: Depends on the result type. Note: When multiple results. • Data type: long Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.
<div> <div></div> <div>M_TYPE_MIL_DOUBLE</div> </div>	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_DOUBLE Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.
<div> <div></div> <div>M_TYPE_MIL_INT</div> </div>	<p>Casts the requested results to a <i>MIL_INT</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_INT Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.
<div> <div></div> <div>M_TYPE_MIL_INT32</div> </div>	<p>Casts the requested results to a <i>MIL_INT32</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_INT32 Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.
<div> <div></div> <div>M_TYPE_MIL_INT64</div> </div>	<p>Casts the requested results to a <i>MIL_INT64</i>. (summarize)</p> <p><i>FirstResultArrayPtr and SecondResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT64 Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_INT64 Note: When a single result is returned to FirstResultArrayPtr and/or SecondResultArrayPtr.

FirstResultArrayPtr

<p>Accepts the address of one of the following (see above for specifics on which is expected):</p> <ul style="list-style-type: none"> • array of type double • array of type long • array of type MIL_DOUBLE • array of type MIL_INT • array of type MIL_INT32 • array of type MIL_INT64 • double • long • M_NULL • MIL_DOUBLE • MIL_INT • MIL_INT32 • MIL_INT64

Specifies the address of the first variable or array in which to write the requested information.

You must set this parameter to **M_NULL** if you are setting the **ResultType** parameter to [M_INTERACTIVE](#).

SecondResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type double
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- double
- long
- M_NULL
- MIL_DOUBLE
- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address of the second variable or array in which to write the requested information.

For most result types, only **FirstResultArrayPtr** will be used and **SecondResultArrayPtr** should be set to **M_NULL**.

You must set this parameter to **M_NULL** if you are setting the **ResultType** parameter to [M_INTERACTIVE](#).

ResultIndex

Specifies the index of the occurrence for which to retrieve information from the marker or the index of the result in the measurement result buffer.

For specifying the index	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 0 to N-1	Specifies the index value where N is the number of edges or stripes found or the number of results calculated.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasInquire

Synopsis

Inquire about a measurement context, marker, or result buffer.

Syntax

```
MIL_INT MmeasInquire(  
    MIL_ID MeasId,  
    MIL_INT InquireType,  
    void *FirstUserVarPtr,  
    void *SecondUserVarPtr  
)
```

Description

This function inquires information about an expected marker characteristic, context setting, or measurement result buffer type. Set marker characteristics using [MmeasSetMarker\(\)](#), and set context settings using [MmeasControl\(\)](#).

Note that for a measurement result buffer, this function only retrieves information about result buffer settings (for example, set with [MmeasSetMarker\(\)](#)). To retrieve information from the measurement result buffer, use [MmeasGetResult\(\)](#) or [MmeasGetResultSingle\(\)](#).

Parameters

MeasId

Specifies the identifier of the marker, measurement context, or measurement result buffer about which to inquire information.

InquireType

Specifies the type of setting about which to inquire.

When performing an inquiry on a marker (to determine the default or user-defined value of a marker characteristic), [InquireType](#) can be set to one of the values listed in the table below.

For certain inquire types, two values can be returned for a stripe marker: or, one for each of the edges. The value for the first edge is returned to [FirstUserVarPtr](#) and that of the second edge to [SecondUserVarPtr](#). For an edge marker, all results are returned to [FirstUserVarPtr](#). Inquire types which return a position always return the X-coordinate to [FirstUserVarPtr](#) and the Y-coordinate to [SecondUserVarPtr](#).

For point, edge, and stripe markers, the [InquireType](#) can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [FirstUserVarPtr](#) parameter and the [SecondUserVarPtr](#) parameter the address of a MIL_DOUBLE.

For point, edge, and stripe markers	
Value	Description
M_MARKER_REFERENCE +	Returns the X- and Y-offsets of the marker reference. (summarize) <div><div>FirstUserVarPtr info</div><div>Return values: Please see MmeasSetMarker() with M_MARKER_REFERENCE. (details)</div></div> <div><div>SecondUserVarPtr info</div><div>Return values: Please see MmeasSetMarker() with M_MARKER_REFERENCE. (details)</div></div>
M_MARKER_TYPE +	Returns the type of marker. (summarize)

	<i>FirstUserVarPtr and SecondUserVarPtr info</i> Return values: M_EDGE; M_POINT; M_STRIPE; (details)
<input type="checkbox"/> M_NUMBER +	Returns the number of points, edges, or stripes. (summarize)
	<i>FirstUserVarPtr and SecondUserVarPtr info</i> Return values: Please see the ResultType parameter of MmeasGetResult() . (details)
<input type="checkbox"/> M_POSITION +	Returns the X- and Y-coordinates of the marker center, except when inquiring about a stripe marker whose M_POSITION_INSIDE_STRIPE control type setting defines the position to be outside the stripe. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY; Value; (details)
	<i>SecondUserVarPtr info</i> Return values: M_ANY; Value; (details)
<input type="checkbox"/> M_POSITION_X +	Returns the X-coordinate of the marker center, except when using a stripe marker and whose M_POSITION_INSIDE_STRIPE control type setting defines the position to be outside the stripe. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY; Value; (details)
	<i>SecondUserVarPtr info</i> Return values: M_ANY; Value; (details)
<input type="checkbox"/> M_POSITION_Y +	Returns the Y-coordinate of the marker center, except when using a stripe marker and whose M_POSITION_INSIDE_STRIPE control type setting defines the position to be outside the stripe. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY; Value; (details)
	<i>SecondUserVarPtr info</i> Return values: M_ANY; Value; (details)
<input type="checkbox"/> M_SPACING +	Returns the typical spacing, in pixels, between consecutive points, edges, or stripes. This setting is only available for a multiple marker. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY; M_SAME; Value; (details)
<input type="checkbox"/> M_SPACING_VARIATION +	Returns the typical variation in spacing, in pixels, between edges or stripes (MmeasSetMarker()). (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY; Value; (details)

For edge and stripe markers, the [InquireType](#) can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [FirstUserVarPtr](#) parameter and the [SecondUserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For edge and stripe markers	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_BOX_ANGLE +	Returns the angle of the measurement box. (summarize)
	<i>FirstUserVarPtr info</i> Return values: 0 to 360; M_ANY; (details)
<input type="checkbox"/> M_BOX_ANGLE_ACCURACY +	Returns the required precision for the resulting angle of the marker.

	(summarize)
	<i>FirstUserVarPtr info</i> Return values: 0.1 to 180.0; M_DISABLE; (details)
<input type="checkbox"/> M_BOX_ANGLE_DELTA_NEG +	Returns the lower limit of the range of angles to be searched for the marker. (summarize)
	<i>FirstUserVarPtr info</i> Return values: 0.1 to 360.0; M_DEFAULT; (details)
<input type="checkbox"/> M_BOX_ANGLE_DELTA_POS +	Returns the upper limit of the range of angles to be searched for the marker. (summarize)
	<i>FirstUserVarPtr info</i> Return values: 0.1 to 360.0; M_DEFAULT; (details)
<input type="checkbox"/> M_BOX_ANGLE_INTERPOLATION_MODE +	Returns the type of interpolation used when performing a search at an angle. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_BICUBIC; M_BILINEAR; M_NEAREST_NEIGHBOR; (details)
<input type="checkbox"/> M_BOX_ANGLE_MODE +	Returns whether multiple-angle search is enabled. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_BOX_ANGLE_REFERENCE +	Returns the center of rotation used when performing a search at an angle. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_BOX_CENTER; M_BOX_ORIGIN; (details)
<input type="checkbox"/> M_BOX_ANGLE_TOLERANCE +	Returns the rotation tolerance of the marker. This is the full range of degrees within which a marker can be rotated from a measurement box that is at a specific angle and still be found. This determines the step angle used for a multiple-angle search. (summarize)
	<i>FirstUserVarPtr info</i> Return values: 0.1 to 360.0; (details)
<input type="checkbox"/> M_BOX_CENTER +	Returns the X- and Y-coordinates of the measurement box's center. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
	<i>SecondUserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_BOX_ORIGIN +	Returns the X- and Y-coordinates of the measurement box's origin. (summarize)
	<i>FirstUserVarPtr info</i> Return values: Please see MmeasSetMarker() with M_BOX_ORIGIN. (details)
	<i>SecondUserVarPtr info</i> Return values: Please see MmeasSetMarker() with M_BOX_ORIGIN. (details)
<input type="checkbox"/> M_BOX_SIZE +	Returns the width and height of the measurement box. The width is returned to FirstUserVarPtr and the height is returned to SecondUserVarPtr . (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
	<i>SecondUserVarPtr info</i>

	Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_CONTRAST +	<p>Returns the contrast of a marker (one value for an edge; two for a stripe). (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: 0 to maximum value of the buffer; M_ANY; (details)</p> <p><i>SecondUserVarPtr info</i> Return values: 0 to maximum value of the buffer; M_ANY; M_NULL; M_SAME; (details)</p>
<input type="checkbox"/> M_CONTRAST_VARIATION +	<p>Returns the contrast variation of a marker. (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: 0 to maximum value of the buffer; M_ANY; (details)</p>
<input type="checkbox"/> M_EDGE_STRENGTH +	<p>Returns the maximum edge value of the edge along the edge width (as a percentage). (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: 0.0 to 100.0; M_ANY; (details)</p> <p><i>SecondUserVarPtr info</i> Return values: 0.0 to 100.0; M_ANY; M_NULL; (details)</p>
<input type="checkbox"/> M_EDGE_STRENGTH_VARIATION +	<p>Returns the tolerance of the maximum edge value of the edge. (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: 0.0 to 100.0; M_ANY; (details)</p>
<input type="checkbox"/> M_EDGE_THRESHOLD +	<p>Returns the edge value beneath which a grayscale variation is not considered an edge. (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: 0.0 to 100.0; (details)</p>
<input type="checkbox"/> M_FILTER_SMOOTHNESS +	<p>Returns the degree of smoothness (strength of denoising) applied to the internal projection buffer of the measurement box during the edge extraction. (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>
<input type="checkbox"/> M_FILTER_TYPE +	<p>Returns the type of the first derivative filter with which to perform the edge extraction. (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: M_DEFAULT; M_EULER; M_PREWITT; M_SHEN; (details)</p>
<input type="checkbox"/> M_NUMBER_MIN +	<p>Returns the minimum number of edges or stripes. (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: M_NUMBER; Value; (details)</p>
<input type="checkbox"/> M_ORIENTATION +	<p>Returns the orientation of the marker. (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: M_HORIZONTAL; M_VERTICAL; (details)</p>
<input type="checkbox"/> M_POLARITY +	<p>Returns the polarity of the marker. (summarize)</p> <p><i>FirstUserVarPtr info</i> Return values: M_NEGATIVE; M_POSITIVE; (details)</p> <p><i>SecondUserVarPtr info</i></p>

	Return values: Please see MmeasSetMarker() with M_POLARITY . (details)
<input type="checkbox"/> M_POSITION_VARIATION +	Returns the tolerance of the position variation. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY ; Value; (details)

For stripe markers, the [InquireType](#) can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [FirstUserVarPtr](#) parameter and the [SecondUserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For stripe markers	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EDGE_INSIDE +	Returns the number of edges between the external edges of a stripe marker. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY ; Value; (details)
<input type="checkbox"/> M_EDGE_INSIDE_VARIATION +	Returns the tolerance of the number of edges between the external edges of a stripe marker. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY ; Value; (details)
<input type="checkbox"/> M_POSITION_INSIDE_STRIPE +	Returns whether the specified position is inside or outside of the stripe. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY ; M_NO ; M_YES ; (details)
<input type="checkbox"/> M_WIDTH +	Returns the width of a stripe marker. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY ; M_SAME ; Value; (details)
<input type="checkbox"/> M_WIDTH_VARIATION +	Returns the width variation of a stripe marker. (summarize)
	<i>FirstUserVarPtr info</i> Return values: M_ANY ; Value; (details)

Combination constant for any of the possible values of the [InquireType](#) parameter

You can add the following value to the above-mentioned values to determine whether an inquire type is supported.

● For inquiring whether an inquire type is supported	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SUPPORTED	Returns whether the specified inquire type is supported. (summarize)
	<i>FirstUserVarPtr and SecondUserVarPtr info</i> Return values: Value!=0 The inquire type is available. 0 The inquire type is not available.

Combination constant for any of the possible values of the [InquireType](#) parameter

You can add the following value to the above-mentioned values to determine the default value of an inquire type.

● For inquiring the default value of an inquire type	
☐ Value	Description
☐ M_DEFAULT	Returns the default value of the specified inquire type. (summarize)
	<i>FirstUserVarPtr and SecondUserVarPtr info</i> Data type: MIL_DOUBLE

Combination constant for [M_POSITION](#); [M_SPACING](#); [M_CONTRAST](#); [M_EDGE_STRENGTH](#); [M_EDGE_INSIDE](#); [M_WIDTH](#);

You can add the following value to the above-mentioned values to get the weight assigned to the specified characteristic.

● For M_CONTRAST, M_EDGE_INSIDE, M_EDGE_STENGTH, M_POSITION, M_SPACING, or M_WIDTH	
☐ Value	Description
☐ M_WEIGHT_FACTOR	Returns the specific weight assigned to the specified characteristic, as a percentage. (summarize)
	<i>FirstUserVarPtr and SecondUserVarPtr info</i> Data type: MIL_DOUBLE

When performing an inquiry on a measurement context buffer, [InquireType](#) can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the [FirstUserVarPtr](#) parameter and the [SecondUserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For a measurement context buffer	
☐ Value	Description
☐ M_PIXEL_ASPECT_RATIO +	Returns the ratio of the width of the pixel to its height. (summarize)
	<i>FirstUserVarPtr and SecondUserVarPtr info</i> Return values: Please see MmeasControl() with M_PIXEL_ASPECT_RATIO . (details)
☐ M_PIXEL_ASPECT_RATIO_INPUT +	Returns how the Measurement module interprets specified measurement characteristics relative to the pixel aspect ratio. (summarize)
	<i>FirstUserVarPtr and SecondUserVarPtr info</i> Return values: M_CORRECTED; M_NORMAL; (details)
☐ M_PIXEL_ASPECT_RATIO_OUTPUT +	Returns how the Measurement module returns results relative to the pixel aspect ratio. (summarize)
	<i>FirstUserVarPtr and SecondUserVarPtr info</i> Return values: M_CORRECTED; M_NORMAL; (details)

When performing a control flag inquiry on point, edge, and stripe markers, or on a measurement context buffer, [InquireType](#) can be set to the following value.

Unless otherwise specified, the following values require that you pass the [FirstUserVarPtr](#) parameter and the [SecondUserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For inquiring the control flag setting given at marker or context allocation time	
Value	Description
M_CONTROL_FLAG +	<p>Returns the ControlFlag parameter setting given at marker or context allocation time. (summarize)</p> <p><i>FirstUserVarPtr and SecondUserVarPtr info</i></p> <p>Return values: Please see the ControlFlag parameter of MmeasAllocMarker(). (details)</p>

When performing an inquiry on a measurement result buffer, [InquireType](#) can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the [FirstUserVarPtr](#) parameter and the [SecondUserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For a measurement result buffer	
Value	Description
M_RESULT_TYPE +	<p>Returns the type of result buffer allocated. (summarize)</p> <p><i>FirstUserVarPtr and SecondUserVarPtr info</i></p> <p>Return values: M_CALCULATE; M_DEFAULT; (details)</p>

Combination constants for any of the possible values of the [InquireType](#) parameter

You can add one of the following values to the above-mentioned values to cast the requested information to the required data type.

● For specifying the data type	
Value	Description
M_TYPE_DOUBLE	<p>Casts the requested information to a <i>double</i>. (summarize)</p> <p><i>FirstUserVarPtr and SecondUserVarPtr info</i></p> <p>Data type: double</p>
M_TYPE_LONG	<p>Casts the requested information to a <i>long</i>. (summarize)</p> <p><i>FirstUserVarPtr and SecondUserVarPtr info</i></p> <p>Data type: long</p>
M_TYPE_MIL_DOUBLE	<p>Casts the requested information to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>FirstUserVarPtr and SecondUserVarPtr info</i></p> <p>Data type: MIL_DOUBLE</p>
M_TYPE_MIL_INT	<p>Casts the requested information to a <i>MIL_INT</i>. (summarize)</p> <p><i>FirstUserVarPtr and SecondUserVarPtr info</i></p> <p>Data type: MIL_INT</p>
M_TYPE_MIL_INT32	<p>Casts the requested information to a <i>MIL_INT32</i>. (summarize)</p> <p><i>FirstUserVarPtr and SecondUserVarPtr info</i></p> <p>Data type: MIL_INT32</p>
M_TYPE_MIL_INT64	<p>Casts the requested information to a <i>MIL_INT64</i>.</p>

	(summarize)
	<i>FirstUserVarPtr and SecondUserVarPtr info</i>
	Data type: MIL_INT64

FirstUserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• double• long• MIL_DOUBLE• MIL_INT• MIL_INT32• MIL_INT64

Specifies the address in which to write the requested information.

SecondUserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• double• long• MIL_DOUBLE• MIL_INT• MIL_INT32• MIL_INT64

Specifies the address in which to write the requested information. If only one value is to be returned, it will be returned to [FirstUserVarPtr](#) and [SecondUserVarPtr](#) should be set to **M_NULL**.

Return value

The returned value is the requested information, typically returned to [FirstUserVarPtr](#), cast to *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasRestoreMarker

Synopsis

Restore a marker from disk.

Syntax

```
MIL_ID MmeasRestoreMarker(  
    MIL_CONST_TEXT_PTR Filename,  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *MarkerIdPtr  
)
```

Description

This function restores a marker that was previously saved, using [MmeasSaveMarker\(\)](#) or [MmeasStream\(\)](#), and returns a handle to it. It also restores all the marker characteristics that were in effect when the marker was saved.

Parameters

Filename

Specifies the name and path of the file from which to restore the marker. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

For specifying the file name and path		
Value	Description	
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)	
	Parameters	
	FileName	Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.	

SystemId

Specifies the system on which to restore the measurement marker.

This parameter should be set to one of the following values:

For specifying the system identifier	
Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Specifies the restore control flag and should be set to **M_DEFAULT**.

MarkerIdPtr

Specifies the address of the variable in which the marker identifier is to be written. Since the **MmeasRestoreMarker()** function also returns the marker identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the marker identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasSaveMarker

Synopsis

Save a marker to disk.

Syntax

```
void MmeasSaveMarker(  
    MIL_CONST_TEXT_PTR FileName,  
    MIL_ID MarkerId,  
    MIL_INT ControlFlag  
)
```

Description

This function saves an allocated marker to disk, including all of its current characteristics. The marker and its characteristics can later be restored, using [MmeasRestoreMarker\(\)](#) or [MmeasStream\(\)](#).

Parameters

FileName

Specifies the name and path of the file in which to save the marker. It is recommended that you use the MRK file extension for easier use with other Matrox Imaging software products. The function internally handles the opening and closing of this file. If this file already exists, it will be overwritten.

This parameter can be set to one of the following values:

For displaying results in an interactive dialog box					
Value	Description				
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p> <table><tr><th colspan="2">Parameters</th></tr><tr><td><i>FileName</i></td><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters		<i>FileName</i>	Specifies the drive, directory, and name of the file.
Parameters					
<i>FileName</i>	Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</p>				

MarkerId

Specifies the identifier of the marker to save.

ControlFlag

Specifies the save control flag and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasSetMarker

Synopsis

Set a marker characteristic.

Syntax

```
void MmeasSetMarker(  
    MIL_ID MarkerId,  
    MIL_INT CharacteristicToSet,  
    MIL_DOUBLE FirstValue,  
    MIL_DOUBLE SecondValue  
)
```

Description

This function sets a marker characteristic. Marker characteristics are used in determining the marker location of an edge or stripe marker in a target image or specifying the position of point markers, generally used as a reference marker. [MmeasFindMarker\(\)](#) locates the edge(s) or stripe(s) that best correspond to the specified marker's characteristics. A marker's characteristics should describe the marker as accurately as possible to ensure that it will be successfully located in an image.

Parameters

MarkerId

Specifies the identifier of the marker.

CharacteristicToSet

Specifies the type of characteristic to set.

See the [Parameter associations](#) section for possible values.

FirstValue

Specifies the first value to assign to the characteristic. The [FirstValue](#) parameter must always be provided.

See the [Parameter associations](#) section for possible values.

SecondValue

Specifies the second value to assign to the characteristic. The [SecondValue](#) parameter should be set to **M_NULL** if no parameter setting is required.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **CharacteristicToSet**, **FirstValue**, and **SecondValue** parameters are described in the following tables:

- [For point, edge, and stripe markers](#)
- [For edge and stripe markers](#)
- [For stripe markers](#)

The following [CharacteristicToSet](#) and corresponding [FirstValue](#) and [SecondValue](#) parameter settings are used to specify the expected measurement marker characteristics. Note that, the [SecondValue](#) parameter should be set to **M_NULL** if no parameter setting is required.

You can set one the following values for point, edge, and stripe markers.

For point, edge, and stripe markers	
CharacteristicToSet	Description
FirstValue	
SecondValue	
M_NUMBER +	<p>Sets the number of edges or stripes to locate in the measurement box, or the number of points used as a reference marker.</p> <p>Unless a minimum number (M_NUMBER_MIN) is specified, no results will be returned if the number of edges or stripes found falls below M_NUMBER.</p> <p>When M_NUMBER is set to a value greater than 1, the marker is considered a multiple marker. (summarize)</p>
FirstValue	(summarize)
M_ALL	<p>Specifies that the search should locate all edges or stripes in the measurement box.</p> <p>This setting cannot be used for a point marker. (summarize)</p>
Value	<p>Specifies the number of points, edges, or stripes to locate.</p> <p>The default value is 1. (summarize)</p>
M_POSITION +	<p>Sets the coordinates of the marker center within the marker (approximate except for a point marker). When using a stripe marker, you can define instead a position that must be outside of the stripe marker, depending on the M_POSITION_INSIDE_STRIPE setting.</p> <p>For an edge or stripe marker, the defined position must be within the measurement box (taking into account the angle or scope of the angular search, if enabled).</p> <p>The position is ignored for a multiple edge or stripe marker. For a multiple point marker, this defines the position of the first point. (summarize)</p>
FirstValue	(summarize)
M_ANY	<p>Specifies any valid value as the X-coordinate. This setting can only be used when setting the marker center within the marker.</p> <p>This setting cannot be used for a point marker.</p> <p>This is the default value. (summarize)</p>
Value	Specifies a value as the X-coordinate, in pixels.
SecondValue	(summarize)
M_ANY	<p>Specifies any valid value as the Y-coordinate. This setting can only be used when setting the marker center within the marker.</p> <p>This setting cannot be used for a point marker.</p> <p>This is the default value. (summarize)</p>
Value	Specifies a value as the Y-coordinate, in pixels.
M_POSITION_X +	<p>Sets the X-coordinate of the marker within the marker (approximate except for a point marker). When using a stripe marker, you can define instead a position that must be outside of the stripe marker depending on the M_POSITION_INSIDE_STRIPE setting.</p> <p>For an edge or stripe marker, the defined position must be within the measurement box (taking into account the angle or scope of the angular search, if enabled).</p> <p>The position is ignored for a multiple edge or stripe marker. For a multiple point marker, this defines the position of the first point. (summarize)</p>
FirstValue	(summarize)

<input type="checkbox"/> M_ANY	<p>Specifies any valid value as the X-coordinate. This setting can only be used when setting the marker center within the marker.</p> <p>This setting cannot be used for a point marker.</p> <p>This is the default value. (summarize)</p>
<input type="checkbox"/> Value	Specifies a value as the X-coordinate, in pixels.
<input type="checkbox"/> M_POSITION_Y +	<p>Sets the Y-coordinate of the marker within the marker (approximate except for a point marker). When using a stripe marker, you can define instead a position that must be outside of the stripe marker depending on the M_POSITION_INSIDE_STRIPE setting.</p> <p>For an edge or stripe marker, the defined position must be within the measurement box (taking into account the angle or scope of the angular search, if enabled).</p> <p>The position is ignored for a multiple edge or stripe marker. For a multiple point marker, this defines the position of the first point. (summarize)</p>
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_ANY	<p>Specifies any valid value as the Y-coordinate. This setting can only be used when setting the marker center within the marker.</p> <p>This setting cannot be used for a point marker.</p> <p>This is the default value. (summarize)</p>
<input type="checkbox"/> Value	Specifies a value as the Y-coordinate, in pixels.
<input type="checkbox"/> M_SPACING +	<p>Sets the typical spacing (distance) between consecutive edges or stripes to locate, or between the points used as a reference marker. This setting is only available for multiple markers. (summarize)</p>
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_ANY	<p>Specifies that spacing is not considered.</p> <p>This setting cannot be used for a point marker.</p> <p>This is the default value. (summarize)</p>
<input type="checkbox"/> M_SAME	<p>Specifies that the average spacing of all located edges or stripes will be used.</p> <p>This setting cannot be set for a point marker. (summarize)</p>
<input type="checkbox"/> Value	Specifies the spacing, in pixels.

You can set one of the following values for edge and stripe markers.

For edge and stripe markers	
<input type="checkbox"/> CharacteristicToSet	Description
FirstValue	
SecondValue	
<input type="checkbox"/> M_BOX_ANGLE +	<p>Sets the angle of the measurement box.</p> <p>The measurement box's center of rotation is set with M_BOX_ANGLE_REFERENCE. For a multiple point marker, the box angle determines the angle from which subsequent points are placed (at the interval set with M_SPACING), proceeding from the coordinates defined by M_POSITION, M_POSITION_X, and M_POSITION_Y. (summarize)</p>
<input type="checkbox"/> FirstValue	(summarize)

<input type="checkbox"/> 0 to 360	Specifies the angle of the measurement box, in degrees. The default value is 0 °. (summarize)
<input type="checkbox"/> M_ANY	Specifies that the content of the measurement box is analyzed to automatically determine the angle of the marker.
<input type="checkbox"/> M_BOX_ANGLE_ACCURACY +	Sets the accuracy of the angular search. This determines the size of the step angle to use once the approximate location of the marker is found. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> 0.1 to 180.0	Specifies the accuracy of the angular search, in degrees.
<input type="checkbox"/> M_DISABLE	Specifies the angle of accuracy to be equal to the angle of tolerance. This is the default value. (summarize)
<input type="checkbox"/> M_BOX_ANGLE_DELTA_NEG +	Sets the negative range of angles within which to search for the marker, in degrees. The search is performed between the range of angles defined by: (M_BOX_ANGLE - M_BOX_ANGLE_DELTA_NEG) to (M_BOX_ANGLE + M_BOX_ANGLE_DELTA_POS), inclusively, starting with an angle closest to that of M_BOX_ANGLE. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default negative range of angles, in degrees. For a symmetrical stripe marker, the default is 180.0. For an edge marker or a non-symmetrical stripe marker, the default is 360.0. (summarize)
<input type="checkbox"/> 0.1 to 360.0	Specifies the negative range of angles, in degrees.
<input type="checkbox"/> M_BOX_ANGLE_DELTA_POS +	Sets the positive range of angles within which to search for the marker, in degrees. The search is performed between the range of angles defined by: (M_BOX_ANGLE - M_BOX_ANGLE_DELTA_NEG) to (M_BOX_ANGLE + M_BOX_ANGLE_DELTA_POS), inclusively, starting with an angle closest to that of M_BOX_ANGLE. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default positive range of angles, in degrees. For a symmetrical stripe marker, the default is 180.0. For an edge marker or a non-symmetrical stripe marker, the default is 360.0. (summarize)
<input type="checkbox"/> 0.1 to 360.0	Specifies the positive range of angles, in degrees.
<input type="checkbox"/> M_BOX_ANGLE_INTERPOLATION_MODE +	Sets the type of interpolation used when M_BOX_ANGLE_MODE is enabled. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_BICUBIC	Specifies bicubic interpolation.
<input type="checkbox"/> M_BILINEAR	Specifies bilinear interpolation. This is the default value. (summarize)
<input type="checkbox"/> M_NEAREST_NEIGHBOR	Specifies nearest-neighbor interpolation.

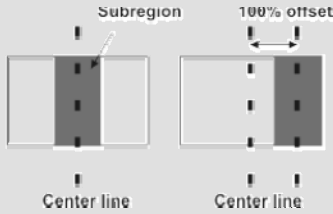
<input type="checkbox"/> M_BOX_ANGLE_MODE +	Enables the use of multiple-angle search, as specified by the settings of other M_BOX_ANGLE... characteristics. The angle of the marker with the highest match score is returned. For an edge marker, this is the score of the edge, and for a stripe marker, this is the mean score of the edges. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the search only be performed at the angle specified by M_BOX_ANGLE . This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies that multiple-angle search is enabled.
<input type="checkbox"/> M_BOX_ANGLE_REFERENCE +	Sets the center of rotation used when M_BOX_ANGLE is not zero. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_BOX_CENTER	Specifies the center of rotation to the values specified in M_BOX_CENTER . This is the default value. (summarize)
<input type="checkbox"/> M_BOX_ORIGIN	Specifies the center of rotation to the values specified in M_BOX_ORIGIN .
<input type="checkbox"/> M_BOX_ANGLE_TOLERANCE +	Sets the rotation tolerance of the marker. This is the full range of degrees within which a marker can be rotated from a measurement box that is at a specific angle and still be found. This determines the step angle used for a multiple-angle search. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 5.0. (summarize)
<input type="checkbox"/> 0.1 to 360.0	Specifies the rotation tolerance, in degrees.
<input type="checkbox"/> M_BOX_CENTER +	Sets the coordinates of the measurement box's center. These coordinates are relative to the top-left corner of the target image. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the target image's corresponding center coordinate is used.
<input type="checkbox"/> Value	Specifies the X-coordinate of the center of the measurement box within the target image.
<input type="checkbox"/> SecondValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the target image's corresponding center coordinate is used.
<input type="checkbox"/> Value	Specifies the Y-coordinate of the center of the measurement box within the target image.
<input type="checkbox"/> M_BOX_ORIGIN +	Sets the coordinates of the measurement box's top-left corner. These coordinates are relative to the top-left corner of the target image. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0 (X-coordinate). (summarize)
<input type="checkbox"/> Value	Specifies the X-coordinate of the measurement box's top-left corner within the target image.
<input type="checkbox"/> SecondValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0 (Y-coordinate). (summarize)
<input type="checkbox"/> Value	Specifies the Y-coordinate of the measurement box's top-left corner within the target image.

<input type="checkbox"/> M_BOX_SIZE +	Sets the width and height of the measurement box. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the width of the entire target image used.
<input type="checkbox"/> Value	Specifies the width of the measurement box within the target image. This value must be positive. (summarize)
<input type="checkbox"/> SecondValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the height of the entire target image used.
<input type="checkbox"/> Value	Specifies the height of the measurement box within the target image. This value must be positive. (summarize)
<input type="checkbox"/> M_CONTRAST +	Sets the typical difference in average grayscale values between the start and end of the intensity transition (edge width) from which an edge is established. For a stripe marker, M_CONTRAST sets the typical difference in average grayscale values for each of its edges. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> 0 to maximum value of the buffer	Specifies the typical difference for the first edge. The maximum value of the buffer is determined by the size of the buffer. For example, for an 8-bit buffer, the value is 255; for a 16-bit buffer, the value is 65535. (summarize)
<input type="checkbox"/> M_ANY	Specifies that the difference in average grayscale values is not considered in the search. This is the default value. (summarize)
<input type="checkbox"/> SecondValue	(summarize)
<input type="checkbox"/> 0 to maximum value of the buffer	Specifies the typical difference for the second edge (for a stripe marker). The maximum value of the buffer is determined by the size of the buffer. For example, for an 8-bit buffer, the value is 255; for a 16-bit buffer, the value is 65535. (summarize)
<input type="checkbox"/> M_ANY	Specifies that the difference in average grayscale values is not considered in the search. This is the default value. (summarize)
<input type="checkbox"/> M_NULL	Specifies that this parameter does not apply. This setting must be used for an edge marker. (summarize)
<input type="checkbox"/> M_SAME	Specifies that the difference in average grayscale values is the same as the first edge.
<input type="checkbox"/> M_CONTRAST_VARIATION +	Sets the contrast variation of an edge. This setting is ignored if M_CONTRAST is set to M_ANY . For a stripe marker, this value should be the maximum of the contrast variation for each of its edges. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> 0 to maximum value of the buffer	Specifies the maximum tolerance of the difference. The maximum value of the buffer is determined by the size of the buffer. For example, for an 8-bit buffer, the value is 255; for a 16-bit buffer, the value is 65535. (summarize)

<input type="checkbox"/> M_ANY	<p>Specifies that the contrast variation is not considered in the search.</p> <p>This is the default value. (summarize)</p>
<input type="checkbox"/> M_EDGE_STRENGTH +	<p>Sets the maximum/minimum edge value along the edge width (depending on the polarity of the edge).</p> <p>In an 8-bit image buffer, the maximum pixel value is 255. Therefore, a 50% edge strength in this buffer represents a maximum filter edge value of 128 and a rising edge. (summarize)</p>
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the maximum/minimum edge value along the edge width of the first edge as a normalized percentage of the maximum pixel value.
<input type="checkbox"/> M_ANY	<p>Specifies that the strongest edge value found in the search is set as the maximum.</p> <p>This is the default value. (summarize)</p>
<input type="checkbox"/> SecondValue	(summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the maximum/minimum edge value along the edge width of the second edge as a normalized percentage of the maximum pixel value (for a stripe marker).
<input type="checkbox"/> M_ANY	<p>Specifies that the strongest edge value found in the search is set as the maximum.</p> <p>This is the default value. (summarize)</p>
<input type="checkbox"/> M_NULL	<p>Specifies that this parameter does not apply.</p> <p>This setting must be used for an edge marker. (summarize)</p>
<input type="checkbox"/> M_EDGE_STRENGTH_VARIATION +	<p>Sets the maximum tolerance of the maximum/minimum edge value of an edge.</p> <p>This setting is ignored if M_EDGE_STRENGTH is set to M_ANY.</p> <p>For a stripe marker, the edge strength variation applies to both edges. (summarize)</p>
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> 0.0 to 100.0	<p>Specifies the maximum tolerance as a normalized percentage of the image buffer's maximum possible value.</p> <p>For example, for an 8-bit image buffer, 10% is an edge strength variation of 25.5. (summarize)</p>
<input type="checkbox"/> M_ANY	<p>Specifies that the edge strength variation is not considered in the search.</p> <p>This is the default value. (summarize)</p>
<input type="checkbox"/> M_EDGE_THRESHOLD +	<p>Sets the edge value beneath which a grayscale variation is not considered an edge. (summarize)</p>
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 2.0. (summarize)</p>
<input type="checkbox"/> 0.0 to 100.0	<p>Specifies the minimum edge value (as a percentage) permissible in the target image buffer.</p> <p>To consider all grayscale variations as edges, irrespective of their values, set FirstValue to 0.0. However, due to the increased number of edges, MmeasFindMarker() will execute relatively slowly. (summarize)</p>
<input type="checkbox"/> M_FILTER_SMOOTHNESS +	Sets the degree of smoothness (strength of denoising) applied to the internal projection buffer of the measurement box during the edge extraction. The

	recursive edge extraction process involves a denoising operation. Note that M_FILTER_SMOOTHNESS is only available if M_FILTER_TYPE is set to M_SHEN . (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 50.0. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the smoothness value. A value of 100.0 results in a strong noise reduction effect, while a value of 0.0 has almost no noise reduction effect. (summarize)
<input type="checkbox"/> M_FILTER_TYPE +	Sets the type of the first derivative filter with which to perform the edge extraction. The edge value calculated for each projection value of the measurement box is dependent on this setting. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default filter type. The default setting is M_EULER . (summarize)
<input type="checkbox"/> M_EULER	Specifies an Euler FIR filter. This filter performs a non-recursive edge extraction operation with a predefined kernel of size 2: [-1,1] This filter is faster but more sensitive to noise, compared to M_PREWITT filter. (summarize)
<input type="checkbox"/> M_PREWITT	Specifies a Prewitt FIR filter. This filter performs a non-recursive edge extraction operation with a predefined kernel of size 3: [-1,0,1] This filter is slower but less sensitive to noise, compared to M_EULER filter. (summarize)
<input type="checkbox"/> M_SHEN	Specifies a Shen-Castan Infinite Support Exponential filter. This is an exponential weighting function, of the general form: $Ke^{-\beta n }$ This filter performs a recursive edge extraction operation using an IIR filter. If this filter actually used a kernel, the kernel size would be theoretically infinite. This IIR filter is slower than FIR filters (M_EULER or M_PREWITT). However, this filter is less sensitive to noise and provides more accurate results. You can also control the strength of denoising applied to the projection buffer of the measurement box during the edge extraction, using the M_FILTER_SMOOTHNESS setting. (summarize)
<input type="checkbox"/> M_MARKER_REFERENCE +	Sets the X- and Y-offsets of the marker's reference position relative to the marker's center. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> Value	Specifies the X-offset value, in pixels.
<input type="checkbox"/> SecondValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> Value	Specifies the Y-offset value, in pixels.

<input type="checkbox"/> M_NUMBER_MIN +	Sets the minimum number of edges or stripes to locate in the measurement box. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_NUMBER	Specifies the number of edges or stripes specified in M_NUMBER . This is the default value. (summarize)
<input type="checkbox"/> Value	Specifies the minimum number of edges or stripes to locate.
<input type="checkbox"/> M_ORIENTATION +	Sets the orientation of an edge or stripe marker. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_ANY	Specifies that the marker orientation is unknown.
<input type="checkbox"/> M_HORIZONTAL	Specifies that the marker has a horizontal orientation.
<input type="checkbox"/> M_VERTICAL	Specifies that the marker has a vertical orientation. This is the default value. (summarize)
<input type="checkbox"/> M_POLARITY +	Specifies whether an edge is a rising edge (increase in grayscale value) or a falling edge (decrease in grayscale value). (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_ANY	Specifies that polarity is not considered. This is the default value. (summarize)
<input type="checkbox"/> M_NEGATIVE	Specifies that the polarity of the first edge is negative (falling edge).
<input type="checkbox"/> M_POSITIVE	Specifies that the polarity of the first edge is positive (rising edge).
<input type="checkbox"/> SecondValue	(summarize)
<input type="checkbox"/> M_ANY	Specifies that the polarity is not considered.
<input type="checkbox"/> M_NEGATIVE	Specifies that the polarity of the second edge is negative (falling edge).
<input type="checkbox"/> M_NULL	Specifies that this parameter does not apply. This setting must be used for an edge marker. (summarize)
<input type="checkbox"/> M_OPPOSITE	Specifies that the polarity of the second edge of a stripe marker is the opposite of that of the first edge. This is the default value. (summarize)
<input type="checkbox"/> M_POSITIVE	Specifies that the polarity of the second edge is positive (rising edge).
<input type="checkbox"/> M_SAME	Specifies that the polarity of the second edge of a stripe marker is the same as that of the first edge.
<input type="checkbox"/> M_POSITION_VARIATION +	Sets the tolerance of position variation. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_ANY	Specifies that the position variation tolerance is not considered in the search. This is the default value. (summarize)
<input type="checkbox"/> Value	Specifies the position variation tolerance within the target image, in pixels.

<input type="checkbox"/> M_SPACING_VARIATION +	Sets the inter-edge or inter-stripe spacing tolerance. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_ANY	Specifies that spacing variation is not considered in the search. This is the default value. (summarize)
<input type="checkbox"/> Value	Specifies the inter-edge or inter-stripe variation, in pixels.
<input type="checkbox"/> M_SUB_REGIONS_NUMBER +	Sets the number of sections in which to divide the measurement box. Since a subregion of each section is processed, this number also corresponds to the number of subregions to process. (summarize)
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1. (summarize)
<input type="checkbox"/> Value >= 1	Specifies the number of sections (subregions).
<input type="checkbox"/> M_SUB_REGIONS_OFFSET +	<p>Sets the offset of the subregions, from the center point of each of their sections.</p> <p>Specify the offset as a percentage of the distance from the center point of the section to the maximum possible offset in each section. 100 corresponds to the maximum possible offset. The direction of the offset (X or Y) is taken from M_ORIENTATION, with positive values shifting the subregion down or to the right, and negative values shifting up or to the left.</p>  <p>Note that subregions are offset as a group. (summarize)</p>
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> -100<=Value<=100	Specifies the offset of the subregions.
<input type="checkbox"/> M_SUB_REGIONS_SIZE +	<p>Sets the size of the subregions, as a percentage of their section.</p> <p>Subregions are always rectangular and extend from one edge of the measurement box to the other in the orientation direction. If made smaller than their section, the subregions become thinner and occupy the specified percentage of their section. In this case, subregions will remain centered unless moved with M_SUB_REGIONS_OFFSET.</p> <p>Note that the size of subregions are set as a group. (summarize)</p>
<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 100. (summarize)
<input type="checkbox"/> Value > 0	Specifies the size of the subregions, as a percentage. If the size is set to 100, subregions will be evenly distributed within the measurement box, with no space between them. (summarize)

You can set one the following values for stripe markers only.

For stripe markers	
CharacteristicToSet	Description
FirstValue	
SecondValue	
M_EDGE_INSIDE +	Sets the typical number of edges occurring inside a stripe marker. (summarize)
FirstValue	(summarize)
M_ANY	Specifies that the number of edges inside a stripe marker is not considered in the search. This is the default value. (summarize)
Value	Specifies the number of edges inside a stripe marker.
M_EDGE_INSIDE_VARIATION +	Sets the maximum variation (tolerance) in the number of inside edges of a stripe. (summarize)
FirstValue	(summarize)
M_ANY	Specifies that the maximum variation in the number of inside edges of a stripe is not considered in the search. This is the default value. (summarize)
Value	Specifies the maximum variation in the number of inside edges of a stripe. Note that this tolerance should be in increments of two if stripes are contained within stripes, since two edges are recognized for each stripe. (summarize)
M_POSITION_INSIDE_STRIPE +	Sets the location of the position, specified with M_POSITION , relative to the stripe. If the position is defined as being within the stripe, the search algorithm proceeds outwards in both directions from that point. If defined as outside, no stripe including the position is considered. (summarize)
FirstValue	(summarize)
M_ANY	Specifies that the position can be either inside or outside the stripe. This is the default value. (summarize)
M_NO	Specifies that the position is outside the stripe.
M_YES	Specifies that the position is inside the stripe.
M_WIDTH +	Sets the typical width of a stripe marker (the distance between the marker's edges). This setting is only available for a stripe marker. (summarize)
FirstValue	(summarize)
M_ANY	Specifies that the typical width of a stripe marker is not considered in the search. This is the default value. (summarize)
M_SAME	Specifies that the typical width of a stripe marker is the average width of all located stripes.
Value	Specifies the typical width of a stripe marker, in pixels.
M_WIDTH_VARIATION +	Sets the maximum variation (tolerance) of a stripe's width, in pixels. This setting must be used with M_WIDTH . (summarize)

<input type="checkbox"/> FirstValue	(summarize)
<input type="checkbox"/> M_ANY	Specifies that the maximum variation of a stripe's width is not considered in the search. This is the default value. (summarize)
<input type="checkbox"/> Value	Specifies the maximum variation of a stripe's width.

Combination constant for [M_POSITION](#); [M_SPACING](#); [M_CONTRAST](#); [M_EDGE_STRENGTH](#); [M_EDGE_INSIDE](#); [M_WIDTH](#);

You can add the following value to the above-mentioned values to set a weight for the specified characteristic.

● For weighting purposes with M_CONTRAST , M_EDGE_INSIDE , M_EDGE_STRENGTH , M_POSITION , M_SPACING , and M_WIDTH	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_WEIGHT_FACTOR	Add a specific weight to the specified characteristic. The value assigned to each characteristic represents the percentage of weight assigned to that characteristics. The sum of weights of the M_WEIGHT_FACTOR + M_... characteristics must be 100. Default assigns 50% to M_EDGE_STRENGTH and 50% to all other characteristics. (summarize)

Combination constants for [the values listed in](#) all tables

You can add one of the following values to the above-mentioned value to set the value for a specific exterior edge of a stripe.

● For an exterior edge or stripe	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_EDGE_FIRST	Specifies the values for the first exterior edge of a stripe for the specified characteristic.
<input type="checkbox"/> M_EDGE_SECOND	Specifies the values for the second exterior edge of a stripe for the specified characteristic.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

MmeasStream

Synopsis

Load, restore, or save a marker from/to a file or a memory stream.

Syntax

```
void MmeasStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *MarkerIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load or restore a previously saved marker from a file or memory stream. All the marker characteristics that were in effect when the marker was saved is restored. This function can also save an allocated marker, including all its characteristics, to a specified file or memory stream.

To inquire the number of bytes necessary to save a marker to memory stream, you should call this function (**MmeasStream()**) first with **M_INQUIRE_SIZE_BYTE**.

The content saved to memory stream is equivalent to the content saved to file. In addition, any file saved with **MmeasSaveMarker()** is equivalent to a file saved using this function.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using **MmeasStream()**, you can choose to save a backwards-compatible version of the marker object, which will work using a version of MIL that is up to one major release older than the current version (depending on which version is specified). For example, if you allocate a marker object using MIL 9.0 and save it to version 8.0, you can restore this object on a computer where MIL 8.0 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore marker objects saved using MIL version 7.0 with Processing Pack 1 or above. Settings that do not exist in the lower version will be filled with default values when the marker object is loaded or restored.

Parameters

MemPtrOrFileName
Specifies the file or memory stream.

For specifying the file or memory stream		
Value	Description	
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving a measurement marker to a file, use the EDG file extension. The function handles (internally) the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p> <p>To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p>	
	Parameters	
	FileName	Specifies the drive, directory, and name of the file.

<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE .
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. This parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (MmeasStream()) first with M_INQUIRE_SIZE_BYTE . (summarize)

SystemId

Specifies the system on which to restore the measurement marker. For [M_INQUIRE_SIZE_BYTE](#), [M_LOAD](#), and [M_SAVE](#), [SystemId](#) is ignored and should be set to [M_NULL](#). For an [M_RESTORE](#) operation, this parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform on the measurement marker. This parameter must be set to one of the following values:

● For the measurement marker	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save a measurement marker to memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated measurement marker.
<input type="checkbox"/> M_RESTORE	Restores a measurement marker from a file or memory stream and assigns it an identifier.
<input type="checkbox"/> M_SAVE	Saves a measurement marker to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the measurement marker. This parameter must be set to one of the following values:

● For specifying the stream that holds the measurement marker	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream.
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)

Version

Specifies the MIL version of the marker object. This parameter must be set to one of the following values.

● For specifying the MIL version	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default version.

	For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL. For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Specifies the function's control flag and should be set to **M_DEFAULT**.

MarkerIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the marker.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, [MarkerIdPtr](#) specifies the address of the variable from which to read the marker identifier.

For an [M_LOAD](#) operation, the [MarkerIdPtr](#) specifies the address of the variable from which to read the identifier of the marker where the file or memory stream content will be loaded.

For an [M_RESTORE](#) operation, the [MarkerIdPtr](#) specifies the address in which to return the identifier of the restored marker. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the measurement marker, in bytes.

If the size is not required, you can set this parameter to **M_NULL**.

Note that the size of a measurement marker will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmeas.lib.
DLL	Requires mil.dll; milmeas.dll.

Mmet functions

Synopsis

The functions prefixed with Mmet make up the Metrology module. The Metrology module measures features and validates geometric tolerances in a target image. Features and tolerances are first added to a virtual template. Added features can be physically measured from an image or geometrically constructed from other features. Once features and tolerances have been added to the template, the template itself can be positioned at a new location, depending on where features have been found in the target image. Positioning in metrology can even be done automatically using the results of other MIL modules, such as Model Finder. Measurements are made and results are returned with a high degree of subpixel accuracy.

Functions

- [MmetAddFeature](#)
- [MmetAddTolerance](#)
- [MmetAlloc](#)
- [MmetAllocResult](#)
- [MmetCalculate](#)
- [MmetControl](#)
- [MmetDraw](#)
- [MmetFree](#)
- [MmetGetResult](#)
- [MmetInquire](#)
- [MmetRestore](#)
- [MmetSave](#)
- [MmetSetPosition](#)
- [MmetSetRegion](#)
- [MmetStream](#)

MmetAddFeature

Synopsis

Add a feature to the metrology template of the metrology context.

Syntax

```
void MmetAddFeature(
    MIL_ID ContextId,
    MIL_INT FeatureType,
    MIL_INT Geometry,
    MIL_INT FeatureLabel,
    MIL_INT BuildOperation,
    const MIL_INT *FeatureLabelArrayPtr,
    const MIL_INT *SubfeatureIndexArrayPtr,
    MIL_INT SizeOfArray,
    MIL_INT ControlFlag
)
```

Description

This function adds a feature to the metrology template of the specified metrology context. To add several features to the template, you must call this function for each feature to add. To delete features, use [MmetControl\(\)](#) with [M_DELETE](#).

Features can either be physically measured from the target image or constructed from a set of other features. When a feature is composed of several instances of the same feature type, it is referred to as a multiple feature. Every instance of the feature type is a subfeature of the multiple feature. A single feature is a feature that has only one instance of the feature type. All supported features are single features, except for the edgel feature ([M_MEASURED](#) or [M_CONSTRUCTED](#) with [M_EDGEL](#)) and measured multiple point feature ([M_MEASURED](#) with [M_POINT](#)). When adding a constructed feature that requires a point, it is possible to isolate and use a specific point subfeature of a measured multiple point feature. This is not true for the edgel feature since it is always used as a group.

Every feature has a position in the metrology template. For a physically measured feature, specify the position as a region in which the geometry specified for the feature is expected to be located, using [MmetSetRegion\(\)](#). By default, the region of interest is an infinite region (the whole target image). For a constructed feature, its position in the template is defined by the position of the features used to construct it. The position of a feature is relative to a reference coordinate system, referred to as the reference frame. The global frame is the default reference frame whose origin is aligned with the target image's top-left corner. You can change the reference frame of a feature to a local frame. A local frame is a coordinate system that can be located anywhere within the metrology template and is defined as a feature. To change the reference frame of a feature, use [MmetControl\(\)](#) with [M_REFERENCE_FRAME](#).

Once a feature is added to the template, it is possible to change its position using [MmetSetPosition\(\)](#).

To define the acceptable geometric relationship between several features in the template or the acceptable deviation from the definition of a feature, add a geometric tolerance, using [MmetAddTolerance\(\)](#). To calculate the added features and validate the geometric tolerances in the target image, use [MmetCalculate\(\)](#). To change the settings of a feature in the template, use [MmetControl\(\)](#).

By setting the [ControlFlag](#) parameter to [M_INTERACTIVE](#), you can add features interactively.

Parameters

ContextId

Specifies the identifier of the metrology context containing the template. The metrology context must have been previously allocated on the required system using [MmetAlloc\(\)](#).

FeatureType

Specifies the type of feature to add ([M_CONSTRUCTED](#) or [M_MEASURED](#)).

See the [Parameter associations](#) section for possible values.

Geometry

Specifies the geometric shape of the feature to add.

See the [Parameter associations](#) section for possible values.

FeatureLabel

Specifies the label of the feature to add. Once added, you can inquire the feature's label using [MmetInquire\(\)](#) with the feature's index.

You must set this parameter to **M_DEFAULT** if you are setting the [ControlFlag](#) parameter to **M_INTERACTIVE**.

● For specifying label values	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the label value, as automatically determined by MIL.
<input type="checkbox"/> Value	Specifies the label value. Note that each feature must have a unique label. (summarize)

BuildOperation

Specifies the operation to use to build the feature.

You must set this parameter to **M_DEFAULT** if you are setting the [ControlFlag](#) parameter to **M_INTERACTIVE**.

See the [Parameter associations](#) section for possible values.

FeatureLabelArrayPtr

Specifies the array that holds the label of every feature to use to construct the feature to add. Set this parameter to **M_NULL** when adding measured features, parametrically constructed features (**M_PARAMETRIC**), or if you are using **M_INTERACTIVE**.

SubfeatureIndexArrayPtr

Specifies the array that holds the index of every subfeature to use to construct the feature to add.

If all features referenced by [FeatureLabelArrayPtr](#) are single features, set [SubfeatureIndexArrayPtr](#) to **M_NULL**. This is also true when adding measured features, parametrically constructed features (**M_PARAMETRIC**), or if you are using **M_INTERACTIVE**. If some features referenced by [FeatureLabelArrayPtr](#) are multiple features, set [SubfeatureIndexArrayPtr](#) to an array that is the same size as the [FeatureLabelArrayPtr](#) array. Set each element of [SubfeatureIndexArrayPtr](#) array to the index of the subfeature to use for the feature at the corresponding element of the [FeatureLabelArrayPtr](#) array; for single features, set their corresponding element in the [SubfeatureIndexArrayPtr](#) array to 0.

SizeOfArray

Specifies the size of the arrays that hold the information that will be used to construct the feature to add ([FeatureLabelArrayPtr](#) and [SubfeatureIndexArrayPtr](#)). When setting the [FeatureLabelArrayPtr](#) parameter to **M_NULL**, set this parameter to 0, unless you are also setting the [ControlFlag](#) parameter to **M_INTERACTIVE**. In this case, set this parameter to **M_NULL**.

ControlFlag


Specifies whether to use this function with an interactive dialog box. This parameter must be set to one of the following values.






● For specifying whether this function is used with an interactive dialog box	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that an interactive dialog box is not used.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively add features.

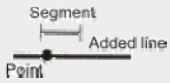

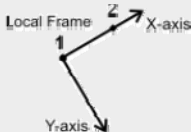
Parameter associations




Possible values for the **FeatureType**, **Geometry**, and **BuildOperation** parameters are described in the table below.



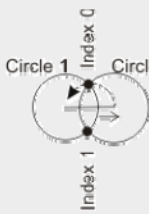
- For selecting the geometry of the feature to add, and the operation used to build the feature



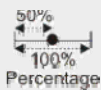
For selecting the geometry of the feature to add, and the operation used to build the feature	
FeatureType	Description
Geometry	
BuildOperation	
<input type="checkbox"/> M_CONSTRUCTED	<p>Specifies to add a constructed feature.</p> <p>Constructed features are geometrically built using other features (added using <code>MmetAddFeature()</code>), or they are defined by their parametric controls (set using <code>MmetControl()</code>). In the latter case, they are referred to as parametrically constructed features (<code>M_PARAMETRIC</code>).</p> <p>Constructed features built using an <code>M_CLONE_FEATURE</code> operation are cloned from the feature specified with the <code>FeatureLabelArrayPtr</code> parameter. For example, an arc can be cloned from an arc feature. Cloned features can be scaled, rotated, and/or moved by using <code>MmetControl()</code> with <code>M_CLONE_....</code></p> <p>For constructed features that require several points or edgels, the actual number of point and edgel features can vary. For example, if 3 points are needed to construct a feature, it is possible to provide one multiple point feature containing three points, three separate point features, or one multiple point feature containing two points and one point subfeature. Only the total number of elements is important, not the number of features. If you do not specify the subfeature indices, then all subfeatures are used.</p> <p>All fit operations for constructed features are affected by the minimum and maximum fit settings. You can adjust these by using <code>MmetControl()</code> with <code>M_FIT_COVERAGE_MIN</code>, <code>M_FIT_DISTANCE_MAX</code>, <code>M_FIT_ITERATIONS_MAX</code>, and <code>M_FIT_VARIATION_MAX</code>. If the constructed feature uses physically measured edgels (any measured feature) as a source, then the fit operations are also affected by the gradient angle settings. You can adjust these by using <code>MmetControl()</code> with <code>M_EDGEL_ANGLE_RANGE</code> and <code>M_EDGEL_RELATIVE_ANGLE</code>.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ARC	<p>Specifies to add a constructed arc feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as <code>M_PARAMETRIC</code> .
<input type="checkbox"/> M_CLONE_FEATURE	Specifies that the arc feature is cloned.
<input type="checkbox"/> M_CONSTRUCTION	<p>Specifies that the arc feature is built from a geometric construction using the features specified with the <code>FeatureLabelArrayPtr</code> and <code>SubfeatureIndexArrayPtr</code> parameters. The specified features must be three points. The first, second, and third points are used as the center, the start, and the end of the arc, respectively. If the radius formed by the first and second points is not equal to the radius formed by the first and third points, the constructed arc will pass as close as possible to both the second and third points. This occurs because arc features are circular arcs, which are portions of a circle. Therefore, every point on the arc must have the same radius. Note that arcs follow the counter-clockwise convention.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_FIT	<p>Specifies that the arc feature is built from the arc that best fits the features specified with the <code>FeatureLabelArrayPtr</code> and <code>SubfeatureIndexArrayPtr</code> parameters. The features that you provide must be three or more edgels or points. It is possible to have a combination of edgels and points. The best fit arc passes as close to as much data as possible while not necessarily touching it. Note that arcs follow the counter-clockwise convention.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_PARAMETRIC	<p>Specifies that the arc feature is built from the following <code>MmetControl()</code> parametric controls: <code>M_POSITION_X</code>, <code>M_POSITION_Y</code>, <code>M_ANGLE_START</code>, <code>M_ANGLE_END</code>, and <code>M_RADIUS</code>.</p>
<input type="checkbox"/> M_CIRCLE	<p>Specifies to add a constructed circle feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as <code>M_PARAMETRIC</code> .
<input type="checkbox"/> M_CLONE_FEATURE	Specifies that the circle feature is cloned.
<input type="checkbox"/> M_CONSTRUCTION	<p>Specifies that the circle feature is built from a geometric construction using the features specified with the <code>FeatureLabelArrayPtr</code> and <code>SubfeatureIndexArrayPtr</code> parameters. The features must be two points. The first point is used as the center of the circle and the second point is located on the contour of the circle. The distance between the two points is the circle's radius.</p>

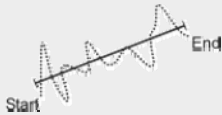
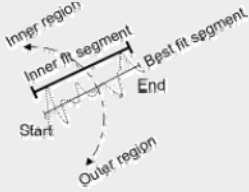
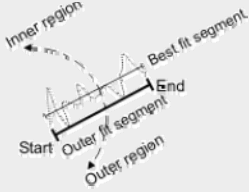
	(summarize)
<input type="checkbox"/> M_FIT	<p>Specifies that the circle feature is built from the circle that best fits the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. The features must be three or more edgels or points. It is possible to have a combination of edgels and points. The best fit circle passes as close to as much data as possible while not necessarily touching it.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_INNER_FIT	<p>Specifies that the circle feature is built from the inner fit of a circle in the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. The features must be three or more edgels or points. It is possible to have a combination of edgels and points. The inner fit for a circle feature is the largest possible circle that does not contain any edgels or points.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_OUTER_FIT	<p>Specifies that the circle feature is built from the best outer fit of a circle around the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. The features must be three or more edgels or points. It is possible to have a combination of edgels and points. The outer fit for a circle feature is the smallest possible circle that contains all the edgels or points.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_PARAMETRIC	Specifies that the circle feature is built from the following MmetControl() parametric controls: M_POSITION_X , M_POSITION_Y , and M_RADIUS .
<input type="checkbox"/> M_EDGEL	<p>Specifies to add a constructed edgel feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_CLONE_FEATURE	Specifies that the edgel feature is cloned.
<input type="checkbox"/> M_COPY_FEATURE_EDGELS	<p>Specifies that the edgel feature is constructed by copying the edgels of the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. You must specify only physically measured features.</p> <p>You can constrain the set of edgels that are copied from the specified features using MmetControl() with M_EDGEL_TYPE.</p> <p>(summarize)</p>
<input type="checkbox"/> M_LINE	<p>Specifies to add a constructed line feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_PARAMETRIC .
<input type="checkbox"/> M_ANGLE	<p>Specifies that the line feature is built at a specified angle from the specified linear feature (segment or line), and passing through a specified point. Specify the linear feature and the point using the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters; the order of the two features does not matter. To specify the angle, use MmetControl() with M_ANGLE; you can specify a value between 0° and 360°. Note that the angle is measured counter-clockwise, relative to the linear feature (from start to end).</p>  <p>(summarize)</p>
<input type="checkbox"/> M_BISECTOR	<p>Specifies that the line feature is built where it divides, in two equal parts, an angle formed by three points or two lines. Specify the points or lines feature with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. If three points are specified, the first point is used as the vertex of the angle.</p> 

	<p>If two lines are specified, the bisector line will divide the angles formed by the first line and the second line in the counter-clockwise direction.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_CLONE_FEATURE	Specifies that the line feature is cloned.
<input type="checkbox"/> M_CONSTRUCTION	Specifies that the line feature is built from a geometric construction using the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. The features must be two points. The line feature will pass through these two points.
	(summarize)
<input type="checkbox"/> M_FIT	Specifies that the line feature is built from the resulting best fit of the segment feature specified with the FeatureLabelArrayPtr parameter.
<input type="checkbox"/> M_PARALLEL	Specifies that the line feature is built parallel to the specified linear feature (segment or line), and passing through a specified point. Specify the linear feature and the point with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters; the order of the two features does not matter.
	 <p>(summarize)</p>
<input type="checkbox"/> M_PARAMETRIC	Specifies that the line feature is built from the following MmetControl() parametric controls: M_LINE_A , M_LINE_B , and M_LINE_C .
<input type="checkbox"/> M_PERPENDICULAR	Specifies that the line feature is built perpendicular to the specified linear feature (segment or line), and passing through a specified point. Specify the linear feature and the point with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters; the order of the two features does not matter.
	 <p>(summarize)</p>
<input type="checkbox"/> M_SECTOR_RELATIVE	<p>Specifies that the line feature is built where it divides, in two parts, an angle formed by three points specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. The first point is used as the vertex of the angle.</p> <p>Specify the size of the angle between the line and the first pair of points (first and second points) using MmetControl() with M_ANGLE. Express the angle as a percentage of the total angle of all three points.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_LOCAL_FRAME	<p>Specifies to add a constructed local frame feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_PARAMETRIC .
<input type="checkbox"/> M_CLONE_FEATURE	Specifies that the local frame feature is cloned.
<input type="checkbox"/> M_CONSTRUCTION	<p>Specifies that the local frame feature is built from a geometric construction using the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. You can specify two points or a single point. If two points are specified, the first point is used as the origin of the local frame and the second point is used to align the X-axis.</p>  <p>If a single point is specified, use MmetControl() with M_ANGLE to build the local frame.</p>

	 <p>(summarize)</p>
<input type="checkbox"/> M_PARAMETRIC	<p>Specifies that the local frame feature is built from the following <code>MmetControl()</code> parametric controls: <code>M_POSITION_X</code>, <code>M_POSITION_Y</code>, and <code>M_ANGLE</code>. Note that the angle is measured counter-clockwise, and is relative to the global frame.</p> <p>(summarize)</p>
<input type="checkbox"/> M_POINT	<p>Specifies to add a constructed point feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Same as <code>M_PARAMETRIC</code>.</p>
<input type="checkbox"/> M_ANGLE_ABSOLUTE	<p>Specifies that the point feature is built at the specified absolute angle, on the contour of the feature specified with the <code>FeatureLabelArrayPtr</code> parameter. You can specify either an arc or a circle. The angle, in degrees, is relative to the start of the contour of the specified feature and is set using <code>MmetControl()</code> with <code>M_ANGLE</code>. The angle is measured counter-clockwise. The start of the contour of a circle is where the circle intersects the positive X-axis at 0°.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_ANGLE_RELATIVE	<p>Specifies that the point feature is built at the specified relative angle, on the contour of the feature specified with the <code>FeatureLabelArrayPtr</code> parameter. You can specify either an arc or a circle. Specify the angle as percentage, relative to the start of the contour using <code>MmetControl()</code> with <code>M_ANGLE</code>. Note that the start of a circle's contour is where the circle intersects with the positive X-axis at 0°.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_CENTER	<p>Specifies that the point feature is built at the center of the feature(s) specified with the <code>FeatureLabelArrayPtr</code> and <code>SubfeatureIndexArrayPtr</code> parameters. You can specify one, or a combination, of the following features: arc, circle, edgel, local frame, point, or segment. If multiple features are provided, the point feature is built at the center of gravity of the centers of all the features.</p>  <p><code>M_CENTER</code> does not necessarily build a point on the feature's edge. Use <code>M_MIDDLE</code> if you want to build a point at the edge's midpoint.</p> <p>(summarize)</p>
<input type="checkbox"/> M_CLONE_FEATURE	<p>Specifies that the point feature is cloned.</p>
<input type="checkbox"/> M_CLOSEST	<p>Specifies that the point feature is built at the point on the first feature closest to the second feature.</p>  <p>Specify both features with the <code>FeatureLabelArrayPtr</code> and <code>SubfeatureIndexArrayPtr</code> parameters.</p> <p>You can specify the following:</p> <ul style="list-style-type: none"> • A segment as the first feature and a point, local frame, circle, line, or edgel as the second feature. • An arc as the first feature and a point, local frame, circle, line, or edgel as the second feature. • A circle as the first feature and a point, local frame, segment, arc, circle, line, or edgel as the second feature.

	<ul style="list-style-type: none"> • A line as the first feature and a point, local frame, segment, arc, circle, or edgel as the second feature. • An edgel feature as the first feature and a point, local frame, segment, arc, circle, line, or edgel as the second feature. <p>The M_CLOSEST operation will fail if the closest point is not unique; for example, a segment and a line that are parallel.</p> <p>(summarize)</p>
☐ M_EXTENDED_INTERSECTION	<p>Specifies that the point feature is built at the point of intersection between the extension of the first and second features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters.</p> <p>The source features which can be extended are: a segment, which is extended into a line, and an arc, which is extended into a circle.</p> <p>You can specify the following:</p> <ul style="list-style-type: none"> • A segment feature and a segment, arc, circle, line, or edgel feature. • An arc feature and a segment, arc, circle, line, or edgel feature. • A circle feature and a segment or arc feature. • A line feature and a segment or arc feature. • An edgel feature and a segment or arc feature. <p>If there is more than one intersection candidate, specify the index of the intersection at which to build the feature, using MmetControl() with M_OCCURRENCE; the intersection is relative to the first feature that has a direction. The default intersection is the one indexed as 0.</p>  <p>(summarize)</p>
☐ M_INTERSECTION	<p>Specifies that the point feature is built at the point of intersection between the first and second features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters.</p> <p>You can specify the following:</p> <ul style="list-style-type: none"> • A segment feature and a segment, arc, circle, line, or edgel feature. • An arc feature and a segment, arc, circle, line, or edgel feature. • A circle feature and a segment, arc, circle, or edgel feature. • A line feature and a segment, arc, line, or edgel feature. • An edgel feature and a segment, arc, circle, line, or edgel feature. <p>If there is more than one intersection candidate, specify the index of the intersection at which to build the feature, using MmetControl() with M_OCCURRENCE; the intersection is relative to the first feature that has a direction. The default intersection is the one indexed as 0.</p>  <p>For two circle intersections, specify the intersections relative to the segment that can be created by joining the circles' centers. The first intersection obtained by scanning the first circle counter-clockwise from the segment is indexed as 0; the other is indexed at 1.</p>  <p>(summarize)</p>
☐ M_MAX_DISTANCE_POINT	<p>Specifies that the point feature is built at the point on the first feature farthest from any point on the second feature. Specify features with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters.</p>

	<p>You can specify the following:</p> <ul style="list-style-type: none"> • A segment as the first feature and a point or local frame as the second feature. • An arc as the first feature and a point, local frame, circle, line, or edgel as the second feature. • A circle as the first feature and a point, local frame, segment, arc, circle, line, or edgel as the second feature. • An edgel feature as the first feature and a point, local frame, segment, arc, circle, line, or edgel as the second feature. <p>The M_MAX_DISTANCE_POINT operation will fail if the farthest point is not unique; for example, a segment and a line that are parallel. Note that the max distance point (on the first feature) is established regardless of how close it is to the other points (on the second feature).</p>  <p>(summarize)</p>
<input type="checkbox"/> M_MIDDLE	<p>Specifies that the point feature is built at the midpoint of the contour (edge) of the feature specified with the FeatureLabelArrayPtr parameter. You can specify a segment, arc, or circle feature. The middle of the contour of a circle is where the circle intersects the negative X-axis at 180°.</p> <p>M_MIDDLE applies to a point, located on the feature's edge, that splits the edge at the halfway mark. Use M_CENTER if you want to build the point feature at the center point of the specified features and not necessarily on the feature's edge.</p> <p>(summarize)</p>
<input type="checkbox"/> M_PARAMETRIC	<p>Specifies that the point feature is built from the following MmetControl() parametric controls: M_POSITION_X and M_POSITION_Y.</p>
<input type="checkbox"/> M_POSITION_ABSOLUTE	<p>Specifies that the point feature is built at the specified position on the contour of the feature specified with the FeatureLabelArrayPtr parameter. You can specify a segment, arc, circle, or edgel feature. The position is relative to the start of the contour of the specified feature and is set using MmetControl() with M_POSITION. In the example below, M_POSITION has a value of 40 and the provided segment feature has a length of 120.</p>  <p>Note that the start of the contour of a circle is where the circle intersects the positive X-axis at 0° and the start of the contour of an edgel feature is the first edgel subfeature of the edgel feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_POSITION_END	<p>Specifies that the point feature is built at the end of the contour of the feature specified with the FeatureLabelArrayPtr parameter.</p> <p>You can specify a segment, arc, circle, or edgel feature. The end of the contour of a circle is where the circle intersects the positive X-axis at 360°, which is the same point as its start of contour. The end of the contour of an edgel feature is the last edgel subfeature of the edgel feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_POSITION_RELATIVE	<p>Specifies that the point feature is built at the specified relative position on the contour of the feature specified with the FeatureLabelArrayPtr parameter. You can specify a segment, arc, circle, or edgel feature. Specify the position as a percentage of the total length of the specified feature's contour, relative to the start of the contour, using MmetControl() with M_POSITION. In the example below, M_POSITION is set to 50%.</p>  <p>Note that the start of the contour of a circle is where the circle intersects the positive X-axis at 0° and the start of the contour of an edgel feature is the first edgel subfeature of the edgel feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_POSITION_START	<p>Specifies that the point feature is built at the start position of the contour of the feature specified with the FeatureLabelArrayPtr parameter.</p> <p>You can specify a segment, arc, circle, or edgel feature. The start of the contour of a circle is where the circle intersects the positive X-axis at 0° and the start of the contour of an edgel feature is the first edgel in the edgel feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_SEGMENT	<p>Specifies to add a constructed segment feature.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Same as M_PARAMETRIC.</p>
<input type="checkbox"/> M_CLONE_FEATURE	<p>Specifies that the segment feature is cloned.</p>

<input type="checkbox"/> M_CONSTRUCTION	<p>Specifies that the segment feature is built from a geometric construction using the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. You can specify two points. The first is the start of the segment and the second is the end.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FIT	<p>Specifies that the segment feature is built from the line that best fits the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. You can specify two or more edgels or points. The best fit segment passes as close to as much data as possible while not necessarily touching it. The start of the segment is the extremity closest to the origin of the global frame.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_INNER_FIT	<p>Specifies that the segment feature is built from the segment that has the best inner fit in the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. You can specify two or more edgels or points.</p> <p>The edgels used to find the inner fit segment are the active edgels on the counter-clockwise side of the best fit segment (from start to end). Active edgels satisfy the edgel constraints and fit constraints that are set using MmetControl(). The best fit segment, whose starting point is the extremity closest to the origin of the global frame, separates the inner and outer fit regions. The inner fit segment passes through the two farthest edgels of the inner fit region such that all the edgels of the inner fit region are located on the same side of the inner fit segment. All the active edgels can be projected on the segment, so the segment's extremities are determined by the two farthest active edgels.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_OUTER_FIT	<p>Specifies that the segment feature is built from the outer fit of the features specified with the FeatureLabelArrayPtr and SubfeatureIndexArrayPtr parameters. You can specify two or more edgels or points.</p> <p>The edgels used to find the outer fit segment are the active edgels on the clockwise side of the best fit segment, relative to its direction. Active edgels satisfy the edgel constraints and fit constraints that are set using MmetControl(). The best fit segment, whose starting point is the extremity closest to the origin of the global frame, separates the inner and outer fit regions. The outer fit segment passes through the two farthest edgels of the outer fit region such that all the edgels of the outer fit region are located on the same side of the outer fit segment. All the active edgels can be projected on the segment, so the segment's extremities are determined by the two farthest active edgels.</p>  <p>(summarize)</p>
<input type="checkbox"/> M_PARAMETRIC	<p>Specifies that the segment feature is built from the following MmetControl() parametric controls: M_POSITION_START_X, M_POSITION_START_Y, M_POSITION_END_X, and M_POSITION_END_Y.</p>
<input type="checkbox"/> M_MEASURED	<p>Specifies to add a physically measured feature.</p> <p>Measured features are built using data (edgels) extracted from the target image within the region of interest.</p> <p>All fit operations for measured features are affected by the minimum and maximum fit settings and the gradient angle settings. You can adjust these by using MmetControl() with M_FIT_COVERAGE_MIN, M_FIT_DISTANCE_MAX, M_FIT_ITERATIONS_MAX, M_FIT_VARIATION_MAX, M_EDGEL_ANGLE_RANGE and M_EDGEL_RELATIVE_ANGLE.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ARC	<p>Specifies to add a physically measured arc feature.</p>

	(summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_FIT .
<input type="checkbox"/> M_FIT	Specifies that the best fit operation will be used to build the measured arc.
<input type="checkbox"/> M_CIRCLE	Specifies to add a physically measured circle feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_FIT .
<input type="checkbox"/> M_FIT	Specifies that the best fit operation will be used to build the measured circle.
<input type="checkbox"/> M_INNER_FIT	Specifies that the inner fit operation will be used to build the measured circle.
<input type="checkbox"/> M_OUTER_FIT	Specifies that the outer fit operation will be used to build the measured circle.
<input type="checkbox"/> M_EDGEL	Specifies to add a physically measured edgel feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the operation with which to build the edgel feature. All the edgels in the region of interest are extracted to add the edgel feature. (summarize)
<input type="checkbox"/> M_POINT	Specifies to add a physically measured point feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_FIT .
<input type="checkbox"/> M_FIT	Specifies that the best fit operation will be used to build the measured point feature.
<input type="checkbox"/> M_SEGMENT	Specifies to add a physically measured segment feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_FIT .
<input type="checkbox"/> M_FIT	Specifies that the best fit operation will be used to build the measured segment.
<input type="checkbox"/> M_INNER_FIT	Specifies that the inner fit operation will be used to build the measured segment.
<input type="checkbox"/> M_OUTER_FIT	Specifies that the outer fit operation will be used to build the measured segment.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetAddTolerance

Synopsis

Add a geometric tolerance to the metrology template of the metrology context.

Syntax

```
void MmetAddTolerance(
    MIL_ID ContextId,
    MIL_INT ToleranceType,
    MIL_INT ToleranceLabel,
    MIL_DOUBLE ValueMin,
    MIL_DOUBLE ValueMax,
    const MIL_INT *FeatureLabelArrayPtr,
    const MIL_INT *SubfeatureIndexArrayPtr,
    MIL_INT SizeOfArray,
    MIL_INT ControlFlag
)
```

Description

This function adds a geometric tolerance to the metrology template. To add several geometric tolerances, you must call this function for each tolerance to add. To delete geometric tolerances, use [MmetControl\(\)](#) with [M_DELETE](#).

A geometric tolerance defines the acceptable deviation from the definition of a feature, or the acceptable relationship between multiple features. For example, you can add a geometric tolerance to the metrology template to specify that one feature must be a certain length, or that two features must be perpendicular. To add a feature, use [MmetAddFeature\(\)](#).

When adding a geometric tolerance that requires a point, you can use a specific point subfeature of a measured multiple point feature by specifying its index with the **SubfeatureIndexArrayPtr** parameter. Note that you cannot do this for other features, including the measured multiple edgel feature, as it is always used as a group.

When adding a geometric tolerance, you must also set its minimum and maximum limit values with the **ValueMin** and **ValueMax** parameters. When defining the tolerance of one feature, these limits represent the acceptable deviation from the definition of the feature. For example, a segment's length tolerance can be between 90 and 100 pixels. For multiple features, these limits represent the valid range of acceptable values between the features. For example, the perpendicular tolerance between two segments can be +/- 0.05° (that is, 89.95° to 90.05°). You can also set warning values to alert you when the tolerance is close to its minimum and maximum limits, using [MmetControl\(\)](#) with [M_VALUE_WARNING_MIN](#) and [M_VALUE_WARNING_MAX](#).

When you call [MmetCalculate\(\)](#), the tolerance is calculated (for example, 95 pixels) and a status is assigned (for example, **M_PASS**). To retrieve the tolerance value and status, use [MmetGetResult\(\)](#) with [M_TOLERANCE_VALUE](#) and [M_STATUS](#).

If a calibration object is associated to the target image, set values in real-world units (for example, tolerance values and warning values, and positional values). Otherwise use pixel units. For more information, see the [Camera calibration - overview](#) section in [Chapter 5: Camera calibration](#).

By setting the **ControlFlag** parameter to [M_INTERACTIVE](#), you can add geometric tolerances interactively.

Parameters

ContextId


Specifies the identifier of the metrology context. The metrology context must have been previously allocated on the required system using [MmetAlloc\(\)](#).

ToleranceType

Specifies the type of geometric tolerance to add. The symbol of every tolerance type is in its description. You can draw this symbol using [MmetDraw\(\)](#) with [M_DRAW_TOLERANCE](#).

For adding geometric tolerances	
<input type="checkbox"/> Value	Description

M_ANGULARITY

Specifies to add an angularity tolerance. Its symbol is  .

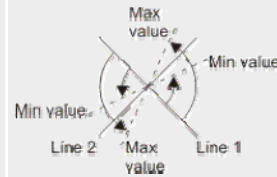
An angularity tolerance validates that the angle between two features falls within the specified angular range (for example, that two lines intersect at an angle between 25° and 35°). An angularity tolerance can be applied between the following features:

- Two segment features.
 - Two line features.
 - A linear feature (segment or line) and an edgel feature.
- Note that an angularity tolerance between a segment and a line produces too many ambiguities and therefore cannot be calculated.

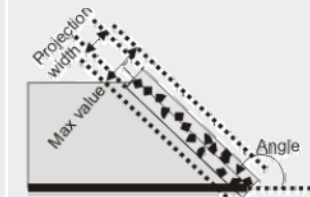
For two segment features, the angle is measured counter-clockwise from the first segment to the second segment. For calculation purposes, the start of each segment is placed at the same point to determine the angle. Therefore, the segments' starting points, as well as the order you list the segments in the **FeatureLabelArrayPtr** parameter, is important. The angle is measured from the first segment to the second segment. The minimum and maximum tolerance parameters set the valid angular range.



For two line features, the angle is measured counter-clockwise from the first segment to the second segment. Therefore, the order you list the segments in the **FeatureLabelArrayPtr** parameter is important. The minimum and maximum tolerance parameters set the valid angular range. Note that there are two ways of measuring this angle; both result in the same value.




For one linear feature (segment or line) and one edgel feature, **M_ANGULARITY** validates the width of the projection of the edgel feature, along the nominal angle specified using **MmetControl()** with **M_ANGLE**. The angle is in the counter-clockwise direction relative to the linear feature. The minimum and maximum tolerance parameters set the valid projection width. Typically, the minimum width value is set to 0. The width of the projection is in pixel or world units. The second feature specified in the **FeatureLabelArrayPtr** parameter must be edgel.

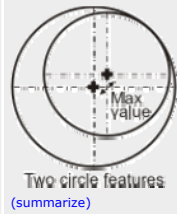


One linear feature and one edgel feature
(summarize)


M_CONCENTRICITY

Specifies to add a concentricity tolerance. Its symbol is  .

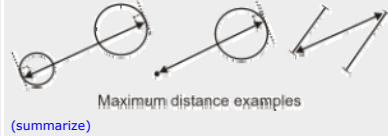
M_CONCENTRICITY validates the concentricity between a combination of two of the following features: arc or circle. This tolerance validates the distance between the centers of the features. Ideally, concentric features have the same center. The order of the features specified in the **FeatureLabelArrayPtr** parameter is inconsequential.




☐ M_DISTANCE_MAX

Specifies to add a maximum distance tolerance. Its symbol is .

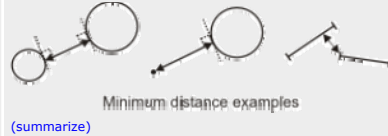
M_DISTANCE_MAX validates that a specified maximum distance is separating any two features (except lines). The order of the features specified in the **FeatureLabelArrayPtr** parameter is inconsequential.



☐ M_DISTANCE_MIN


Specifies to add a minimum distance tolerance. Its symbol is .


M_DISTANCE_MIN validates that a specified minimum distance is separating any two features. The order of the features specified in the **FeatureLabelArrayPtr** parameter is inconsequential.



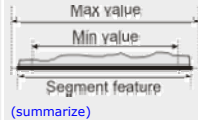
☐ M_LENGTH

Specifies to add a length tolerance.


The symbol for an arc or circle length tolerance is .

The symbol for a segment length tolerance is .

M_LENGTH validates the length of one of the following features: segment, circle, or arc. The length of a circle is its circumference.



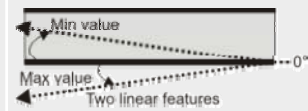
☐ M_PARALLELISM

Specifies to add a parallelism tolerance. Its symbol is .

A parallelism tolerance validates the degree to which two features are parallel. A parallelism tolerance can be applied between the following features:

- Two linear features (segments and lines).
- A linear feature (segment or line) and an edge feature.

For any combination of 2 linear features (segments and lines), **M_PARALLELISM** validates that the angle of the two features, relative to the global reference frame, is the same. The minimum and maximum tolerance parameters set the valid angular range, from the nominal angle (0°). The order of the features specified in the **FeatureLabelArrayPtr** parameter, as well as the start/end of the segments, is inconsequential.

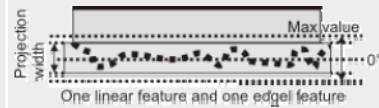


Note that, for parallelism tolerances between two linear features, angles are remapped to 0°; that is, the angle that is measured, and the angular range that you must provide, is the angle that is closest to 0°.




The angle measured Not the angle measured

For a linear feature (segment or line) and an edgel feature, [M_PARALLELISM](#) validates the width of the projection of the edgel feature on a theoretical line that is perpendicular to the segment or line feature. The minimum and maximum tolerance values set the valid projection width. The smaller the width, the more the edgel feature is parallel to the given segment or line feature. Typically, the minimum width value is set to 0. The width of the projection is in pixel or world units. The second feature specified in the **FeatureLabelArrayPtr** parameter must be edgel.



(summarize)

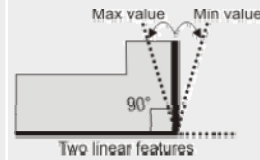
[M_PERPENDICULARITY](#)

Specifies to add a perpendicularity tolerance. Its symbol is .

A perpendicularity tolerance validates the degree to which two features are perpendicular. A perpendicularity tolerance can be applied between the following features:

- Two linear features (segments and lines).
- A linear feature (segment or line) and an edgel feature.

For any combination of 2 linear features (segments and lines), [M_PERPENDICULARITY](#) validates that the angle between the two features is 90°. The minimum and maximum tolerance parameters set the valid angular range, from the nominal angle (90°). The order of the features specified in the **FeatureLabelArrayPtr** parameter, as well as the start/end of the segments, is inconsequential.

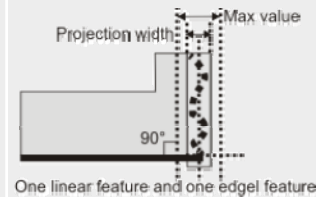


Note that, for perpendicularity tolerances between two linear features, angles are remapped to 90°; that is, the angle that is measured, and the angular range that you must provide, is the angle that is closest to 90°.




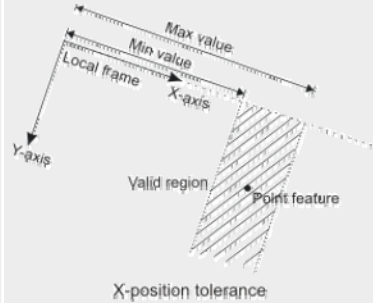

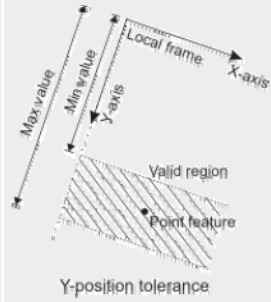


The angle measured Not the angle measured


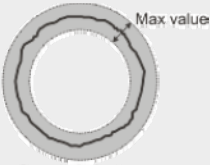


For a linear feature (segment or line) and an edgel feature, [M_PERPENDICULARITY](#) validates the width of the projection of the edgel feature on a theoretical line that is parallel to the segment or line feature. The minimum and maximum tolerance values set the valid projection width. The smaller the width, the more the edgel feature is perpendicular to the given segment or line feature. Typically, the minimum width value is set to 0. The width of the projection is in pixel or world units. The second feature specified in the **FeatureLabelArrayPtr** parameter must be edgel.



One linear feature and one edgel feature

(summarize)

<div data-bbox="191 147 333 175" data-label="Text"> <p><input type="checkbox"/> M_POSITION_X</p> </div>	<div data-bbox="602 167 1484 207" data-label="Text"> <p>Specifies to add a positioning tolerance, along the X-direction of the specified reference frame. Its symbol is .</p> </div> <div data-bbox="602 222 1812 302" data-label="Text"> <p>A positioning tolerance can be used to validate the geometric relationship between a reference frame and one of the following features: local frame, point, or circle. For a circle feature, only the position of its center is validated, not its contour.</p> <p>When using a positioning tolerance, the first element provided to the FeatureLabelArrayPtr parameter must be the label value of the reference frame, while the second element must be the label value of the feature to validate.</p> </div> <div data-bbox="602 310 972 610" data-label="Image">  </div> <div data-bbox="602 613 693 634" data-label="Text"> <p>(summarize)</p> </div>
<div data-bbox="191 638 333 665" data-label="Text"> <p><input type="checkbox"/> M_POSITION_Y</p> </div>	<div data-bbox="602 657 1484 698" data-label="Text"> <p>Specifies to add a positioning tolerance, along the Y-direction of the specified reference frame. Its symbol is .</p> </div> <div data-bbox="602 712 1812 794" data-label="Text"> <p>A positioning tolerance can be used to validate the geometric relationship between a reference frame and one of the following features: local frame, point, or circle. For a circle feature, only the position of its center is validated, not its contour.</p> <p>When using a positioning tolerance, the first element provided to the FeatureLabelArrayPtr parameter must be the label value of the reference frame, while the second element must be the label value of the feature to validate.</p> </div> <div data-bbox="602 802 871 1102" data-label="Image">  </div> <div data-bbox="602 1104 693 1125" data-label="Text"> <p>(summarize)</p> </div>
<div data-bbox="191 1128 298 1156" data-label="Text"> <p><input type="checkbox"/> M_RADIUS</p> </div>	<div data-bbox="602 1148 1050 1188" data-label="Text"> <p>Specifies to add a radius tolerance. Its symbol is .</p> </div> <div data-bbox="602 1203 1694 1226" data-label="Text"> <p>M_RADIUS validates the radius of the specified arc or circle feature. The radius is the linear length between the circle or arc's center and perimeter.</p> </div> <div data-bbox="602 1230 772 1378" data-label="Image">  </div> <div data-bbox="602 1380 693 1401" data-label="Text"> <p>(summarize)</p> </div>
<div data-bbox="191 1406 333 1433" data-label="Text"> <p><input type="checkbox"/> M_ROUNDNESS</p> </div>	

	<p>Specifies to add a roundness tolerance. Its symbol is .</p> <p>M_ROUNDNESS validates the roundness of one of the following features: arc or circle.</p> <p>For a circle, this is done by calculating the distance between the inner and outer circles formed by the given circle feature. Note that the inner and outer circles are concentric. The same concept applies to an arc feature, since an arc is a portion of a circle.</p>  <p>One circle feature</p> <p>The same concept applies to an arc feature, since an arc is a portion of a circle.</p> <p>(summarize)</p>
<input type="checkbox"/> M_STRAIGHTNESS	<p>Specifies to add a straightness tolerance. Its symbol is .</p> <p>M_STRAIGHTNESS validates the straightness of one of the following features: segment or edgel.</p> <p>This is done by calculating the distance between the two parallel lines that are formed by the inner and outer boundaries of the feature. The angle of the two parallel lines is chosen such that the distance between them is the smallest.</p>  <p>One edgel feature</p> <p>(summarize)</p>

ToleranceLabel

Specifies the label of the tolerance to add.

You must set this parameter to **M_DEFAULT** if you are setting the **ControlFlag** parameter to [M_INTERACTIVE](#).

● For labeling the tolerance	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that MIL automatically selects the tolerance's label value. Once a tolerance is added, you can inquire its label using MmetInquire() with the tolerance's index. (summarize)
<input type="checkbox"/> Value > 0	Specifies the label value. Each tolerance must have a unique label otherwise you will get an error. (summarize)

ValueMin

Specifies the minimum value of the geometric tolerance.

You must set this parameter to **M_DEFAULT** if you are setting the **ControlFlag** parameter to [M_INTERACTIVE](#).

● For specifying the minimum geometric tolerance	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the minimum value.

ValueMax

Specifies the maximum value of the geometric tolerance.

You must set this parameter to **M_DEFAULT** if you are setting the **ControlFlag** parameter to [M_INTERACTIVE](#).

● For specifying the maximum geometric tolerance	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the maximum value.

FeatureLabelArrayPtr

Specifies the array that holds the label values of the features whose relationship to validate. The features must have already been added to the metrology template of the metrology context. To add a feature, use [MmetAddFeature\(\)](#).

You must set this parameter to **M_NULL** if you are setting the **ControlFlag** parameter to [M_INTERACTIVE](#).

SubfeatureIndexArrayPtr

Specifies the array that holds the index values of the multiple features' subfeatures whose relationship the added tolerance will validate. For non-multiple features (single features), specify 0 as the index. If the tolerance validates the relationship between only single features (there are no multiple features), set **SubfeatureIndexArrayPtr** to **M_NULL**. You must also set this parameter to **M_NULL** if you are setting the **ControlFlag** parameter to [M_INTERACTIVE](#).

SizeOfArray

Specifies the size of the arrays that hold the information that will be used to add the geometric tolerance ([FeatureLabelArrayPtr](#) and [SubfeatureIndexArrayPtr](#)).

You must set this parameter to **M_NULL** if you are setting the **ControlFlag** parameter to [M_INTERACTIVE](#).

ControlFlag

Specifies whether to use this function with an interactive dialog box. This parameter must be set to one of the following values.

● For specifying whether this function is used with an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that an interactive dialog box is not used.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively add geometric tolerances.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetAlloc

Synopsis

Allocate a metrology context.

Syntax

```
MIL_ID MmetAlloc(  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *ContextIdPtr  
)
```

Description

This function allocates a metrology context on the specified system. A metrology context contains all the information necessary to perform an [MmetCalculate\(\)](#) operation, including global processing settings, geometric tolerances, and features to measure and validate. When the metrology context is no longer required, you should release its memory, using [MmetFree\(\)](#).

You can define features and specify geometric tolerances to a metrology context using [MmetAddFeature\(\)](#) and [MmetAddTolerance\(\)](#), respectively. You can adjust metrology context, feature, and geometric tolerance settings using [MmetControl\(\)](#).

When you allocate a context, a global frame is also automatically created. The global frame is the first coordinate system that will be used to define the features to add to the context. It always exists and therefore cannot be deleted. Its label is **M_GLOBAL_FRAME** and its index is 0.

Parameters

SystemId

Specifies the system on which to allocate the metrology context.

This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the metrology context identifier. Since the **MmetAlloc()** function also returns the metrology context identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the metrology context's identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetAllocResult

Synopsis

Allocate a metrology result buffer.

Syntax

```
MIL_ID MmetAllocResult(  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *ResultIdPtr  
)
```

Description

This function allocates a metrology result buffer, on the specified system, to store results obtained from an [MmetCalculate\(\)](#) operation. When the metrology result buffer is no longer required, release its memory, using [MmetFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the metrology result buffer.

This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ResultIdPtr

Specifies the address of the variable in which to write the metrology result buffer identifier. Since the **MmetAllocResult()** function also returns the metrology result buffer identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the metrology result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetCalculate

Synopsis

Calculate features and validate geometric tolerances.

Syntax

```
void MmetCalculate(
    MIL_ID ContextId,
    MIL_ID TargetBufId,
    MIL_ID ResultId,
    MIL_INT ControlFlag
)
```

Description

This function calculates features and validates geometric tolerances in the target image buffer. To do so, **MmetCalculate()** uses the target image's edge map. For more information on edge maps, see the [Extracting the edges](#) section in [Chapter 9: Edge Finder](#).

To control the Metrology module's calculation and validation behavior, you can set context constraints using [MmetControl\(\)](#). You can also use [MmetControl\(\)](#) to alter settings for features and tolerances that you have already added to the context. To add features to the context, use [MmetAddFeature\(\)](#). To add geometric tolerances to the context and define geometric relationships between features, use [MmetAddTolerance\(\)](#).

Results of a metrology calculation are stored in the specified result buffer and can be obtained using [MmetGetResult\(\)](#).

If the target image is not calibrated, results are calculated in the image coordinate system (pixel units). If the target image is calibrated, results are calculated in the world coordinate system (real-world units), but they can be retrieved in either world or pixel units. Note that, if the target image is calibrated, parametrically constructed feature values, regions, and tolerances are considered as real-world values and are retrieved in world units.

Note that in the presence of distortion, some results are meaningless when converted from real-world to pixel units (for example, angle and scale results). For example, if a feature appears warped in the target image, but the calibration object compensates for this during the calculation, the resulting angle is meaningful in the real-world coordinate system, and meaningless in the image coordinate system.

If complex distortions exist, you must first correct the image before using it with the Metrology module.

Parameters

ContextId

Specifies the identifier of the metrology context. The metrology context must have been previously allocated on the required system using [MmetAlloc\(\)](#).

TargetBufId

Specifies the identifier of the target image buffer from which to extract edges to calculate features and validate geometric tolerances. Typically, the target image buffer should be 1-band 8-bit unsigned. Other buffer depths and types are generally accepted, but can slightly decrease performance. When the target image buffer is 32-bit float, metrology uses floating-point precision calculations.

ResultId

Specifies the metrology result buffer in which to write the results of the calculation. The metrology result buffer must have been previously allocated on the required system using [MmetAllocResult\(\)](#).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The metrology result buffer (**ResultId**) must be allocated on the same system as the metrology context (**ContextId**). If it is not, an error will occur.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetControl

Synopsis

Control a metrology setting for a context, feature, tolerance, or result buffer.

Syntax

```
void MmetControl(
    MIL_ID ContextOrResultId,
    MIL_INT LabelOrIndex,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function sets the specified control for a metrology context, feature, tolerance, or result buffer. These settings control the execution of [MmetCalculate\(\)](#) operations, and how results are retrieved by [MmetGetResult\(\)](#) and [MmetDraw\(\)](#).

Most of the control type settings in **MmetControl()** can be inquired using [MmetInquire\(\)](#). If several calls to **MmetControl()** are made and the **ControlValue** parameter of a particular **ControlType** parameter is changed several times, only the last value is kept. For context, feature, or tolerance settings to take effect, you must call [MmetCalculate\(\)](#).

By setting the **ControlType** parameter to [M_INTERACTIVE](#), you can set the control types interactively.

If a calibration object is associated to the target image, set values in real-world units (for example, tolerance values and warning values, and positional values). Otherwise use pixel units. For more information, see the [Camera calibration - overview](#) section in [Chapter 5: Camera calibration](#).

Parameters

ContextOrResultId

Specifies the identifier of the metrology context or result buffer whose settings to modify. The metrology context or the metrology result buffer must have been previously allocated on the required system using [MmetAlloc\(\)](#) or [MmetAllocResult\(\)](#), respectively.

You must set this parameter to the metrology context if you are setting the **ControlType** parameter to [M_INTERACTIVE](#).

LabelOrIndex

Specifies the metrology context, feature, tolerance, or result buffer to be controlled. Set this parameter to one of the following values.

Unless otherwise specified, you can use the following values to inquire about a metrology context or result buffer, depending on whether you provide a context or result buffer to the **ContextOrResultId** parameter.

For specifying a context, feature, tolerance, or result buffer	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default. If ContextOrResultId is set to a metrology context, M_DEFAULT is the same as M_CONTEXT . If ContextOrResultId is set to a metrology result buffer, M_DEFAULT is the same as M_GENERAL . (summarize)
<input type="checkbox"/> M_FEATURE_LABEL(MIL_INT FeatureLabel)	Specifies the label value of an existing individual feature. (summarize)
	Parameters
	FeatureLabel

	<p>This parameter specifies the label of the individual feature to which the control settings will be applied. You can set this parameter to the following:</p> <table> <tr> <td>Value > 0</td><td>Specifies the label of the individual feature.</td></tr> </table>	Value > 0	Specifies the label of the individual feature.
Value > 0	Specifies the label of the individual feature.		
<input type="checkbox"/> M_FEATURE_INDEX (MIL_INT <i>FeatureIndex</i>)	<p>Specifies the index value of an existing individual feature. (summarize)</p> <p><i>Parameters</i></p> <p><i>FeatureIndex</i> This parameter specifies the index of the individual feature to which the control settings will be applied. You can set this parameter to the following:</p> <table> <tr> <td>Value >= 0</td><td>Specifies the index of the individual feature. Note that the index of the global frame is 0.</td></tr> </table>	Value >= 0	Specifies the index of the individual feature. Note that the index of the global frame is 0.
Value >= 0	Specifies the index of the individual feature. Note that the index of the global frame is 0.		
<input type="checkbox"/> M_TOLERANCE_LABEL (MIL_INT <i>ToleranceLabel</i>)	<p>Specifies the label value of an existing individual tolerance. (summarize)</p> <p><i>Parameters</i></p> <p><i>ToleranceLabel</i> This parameter specifies the label of the individual tolerance to which the control settings will be applied. You can set this parameter to the following:</p> <table> <tr> <td>Value > 0</td><td>Specifies the label of the individual tolerance.</td></tr> </table>	Value > 0	Specifies the label of the individual tolerance.
Value > 0	Specifies the label of the individual tolerance.		
<input type="checkbox"/> M_TOLERANCE_INDEX (MIL_INT <i>ToleranceIndex</i>)	<p>Specifies the index value of an existing individual tolerance. (summarize)</p> <p><i>Parameters</i></p> <p><i>ToleranceIndex</i> This parameter specifies the index of the individual tolerance to which the control settings will be applied. You can set this parameter to the following:</p> <table> <tr> <td>Value >= 0</td><td>Specifies the index of the individual tolerance.</td></tr> </table>	Value >= 0	Specifies the index of the individual tolerance.
Value >= 0	Specifies the index of the individual tolerance.		
<input type="checkbox"/> M_ALL_FEATURES	<p>Applies the specified control setting to all features. This value should only be used with M_DELETE. M_ALL_FEATURES is not supported if the ControlType parameter is set to M_INTERACTIVE. (summarize)</p>		
<input type="checkbox"/> M_ALL_TOLERANCES	<p>Applies the specified control setting to all tolerances. This value should only be used with M_DELETE. M_ALL_TOLERANCES is not supported if the ControlType parameter is set to M_INTERACTIVE. (summarize)</p>		
<input type="checkbox"/> M_CONTEXT	<p>Controls a global setting of a metrology context. When using M_CONTEXT, ContextOrResultId must specify a metrology context. (summarize)</p>		
<input type="checkbox"/> M_GENERAL	<p>Controls a setting of a metrology result buffer. The setting is applied to all features and tolerances in the metrology result buffer. When using M_GENERAL, ContextOrResultId must specify a metrology result buffer. M_GENERAL is not supported if the ControlType parameter is set to M_INTERACTIVE. (summarize)</p>		
<input type="checkbox"/> M_GLOBAL_FRAME	<p>Applies the specified control setting to the global frame of the context. In this case, ContextOrResultId must specify a metrology context. (summarize)</p>		
<input type="checkbox"/> M_MEASURED_FEATURES	<p>Applies the specified control setting to all measured features. In this case, ContextOrResultId must specify a metrology context. (summarize)</p>		

ControlType

Specifies the type of control to set.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the required value for the control.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For opening an interactive dialog box](#)
- [For controlling general context settings](#)
- [For controlling processing settings](#)
- [For controlling features](#)
- [For controlling geometric tolerances](#)
- [For controlling features or geometric tolerances](#)
- [For controlling clone transformations](#)
- [For controlling results](#)
- [For deleting features and tolerances](#)

The following **ControlType** and corresponding **ControlValue** parameter setting can be used to interactively control a metrology context, feature, or tolerance.

For opening an interactive dialog box		
ControlType	Description	
ControlValue		
M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens a dialog box that allows you to interactively edit the control types of the specified metrology context, feature, or tolerance.</p> <p>(summarize)</p>	
M_DEFAULT	Implements the default behavior.	

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control general metrology context settings. For controlling general context settings, the **LabelOrIndex** parameter must be set to [M_CONTEXT](#).

For controlling general context settings		
ControlType	Description	
ControlValue		
M_ASSOCIATED_CALIBRATION	<p>Associates the specified calibration object with the template reference. The template reference can be set using MmetControl() with M_TEMPLATE_REFERENCE_ID. Note that you must re-associate (if necessary) each template reference with its calibration object after restoring a context using MmetRestore() or MmetStream(), since the calibration object is not saved with the context.</p> <p>You cannot associate the template reference with a calibrated object whose Y-axis has been inverted using McalGrid() with M_Y_AXIS_UP.</p> <p>(summarize)</p>	
M_DEFAULT	Same as M_NULL .	
M_NULL	Removes the association between the template reference and a calibration object.	
MIL Calibration object identifier	Specifies the calibration object to associate with the template reference.	
M_TEMPLATE_REFERENCE_ID	<p>Associates a template reference (image) to the context. The template reference is a companion buffer useful for training the metrology context and to draw the results on a typical target image.</p> <p>(summarize)</p>	
M_NULL	Releases the template reference buffer.	
MIL image identifier	Specifies the identifier of the buffer to associate.	
M_TIMEOUT	<p>Sets the maximum measurement and validation time for MmetCalculate(), in msec.</p> <p>(summarize)</p>	
M_DEFAULT	Same as M_DISABLE .	
M_DISABLE	Specifies an infinite amount of measurement and validation time.	
Value > 0.0	Specifies the maximum measurement and validation time, in msec.	

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control processing settings for a measured feature. For controlling processing settings, the **LabelOrIndex** parameter can be set to one of the following: [M_MEASURED_FEATURES](#), or an existing feature label or index using [M_FEATURE_LABEL\(\)](#) or [M_FEATURE_INDEX\(\)](#).

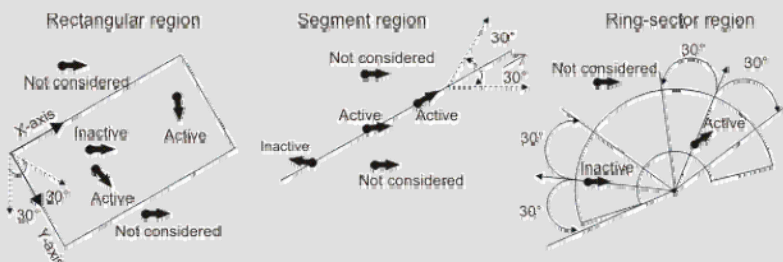
For controlling processing settings	
ControlType	Description
ControlValue	
M_EXTRACTION_SCALE	Sets the scale of the image at which to do the edge extraction. Once the extraction is complete, the results are scaled to the original scale of the image. Note that the default extraction scale setting is typically sufficient. (summarize)
M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
Value > 0.0	Specifies the extraction scale. Note that an extraction scale greater than 1.0 will slow down the calculation time. (summarize)
M_FILTER_MODE	Sets the mode in which to apply the edge extraction filter. For more information on the filter mode, see the Filter mode subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder . (summarize)
M_DEFAULT	Specifies the default filter mode. The default is typically set to the most efficient mode. If special hardware is available to perform the convolution, the default is M_KERNEL . If special hardware is not available to perform the convolution, the default is M_RECURSIVE . (summarize)
M_KERNEL	Specifies the use of a non-recursive implementation of the filter. In this case, a kernel is used to perform the neighborhood operation. To set the size of the filter's convolution kernel, use MmetControl() with M_KERNEL_WIDTH . (summarize)
M_RECURSIVE	Specifies the use of a recursive implementation of an Infinite Impulse Response (IIR) filter, when applicable. If this mode actually used a kernel, the kernel size would be theoretically infinite. (summarize)
M_FILTER_SMOOTHNESS	Sets the degree of smoothness (strength of denoising) applied by the edge extraction filter during the neighborhood operation. For more information on smoothing edges, see the Smoothing subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder . (summarize)
M_DEFAULT	Specifies the default value. The default value is 50.0. (summarize)
0.0 to 100.0	Specifies the smoothness value. A value of 100.0 results in a strong noise reduction effect, while a value of 0.0 has almost no noise reduction effect. (summarize)
M_FILTER_TYPE	Sets the type of filter used to extract edges. The type of filter determines the distribution of the neighborhoods's influence. For more information on filtering, see the Filter type subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder . (summarize)
M_DEFAULT	Same as M_SHEN .
M_DERICHE	Specifies a Canny-Derliche infinite support filter.
M_FREI_CHEN	Specifies a Frei Chen filter.

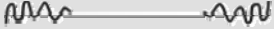
<input type="checkbox"/> M_PREWITT	Specifies a Prewitt filter.
<input type="checkbox"/> M_SHEN	Specifies a Shen-Castan infinite support exponential filter.
<input type="checkbox"/> M_SOBEL	Specifies a Sobel filter.
<input type="checkbox"/> M_FLOAT_MODE	<p>Sets whether to perform all edge extraction operations using floating-point precision calculations.</p> <p>For more information on using floating-point precision to extract edges, see the Calculating and retrieving results section in Chapter 9: Edge Finder. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Specifies that all edge extractions are not forced to be performed using floating-point precision calculations.
<input type="checkbox"/> M_ENABLE	Specifies that all edge extractions are forced to be performed using floating-point precision calculations.
<input type="checkbox"/> M_KERNEL_DEPTH	<p>Sets the depth of the convolution kernel used for kernel filtering mode.</p> <p>For more information on the kernel depth, see the Filter mode subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 8 bit. (summarize)
<input type="checkbox"/> 8	Specifies an 8-bit kernel depth. This is faster, though less accurate, than using a 16-bit kernel depth. (summarize)
<input type="checkbox"/> 16	Specifies a 16-bit kernel depth. This is slower, though more accurate, than using an 8-bit kernel depth. (summarize)
<input type="checkbox"/> M_KERNEL_WIDTH	<p>Sets the maximum X- and Y-size of the convolution kernel used for kernel filtering mode.</p> <p>The size of the kernel can be constrained by the available hardware resources. Larger kernels result in longer processing times.</p> <p>For more information on the kernel width, see the Filter mode subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_AUTO .
<input type="checkbox"/> M_AUTO	Specifies the maximum X- and Y-size of the convolution kernel automatically, for a given smoothness factor. You can set the smoothness factor using MmetControl() with M_FILTER_SMOOTHNESS . (summarize)
<input type="checkbox"/> Value >= 3	<p>Specifies the maximum X- and Y-size of the convolution kernel directly. Only odd integer values are accepted.</p> <p>If your hardware cannot handle the kernel size, the effective smoothness factor will be saturated to the maximum possible value for the specified kernel size. You can set the smoothness factor using MmetControl() with M_FILTER_SMOOTHNESS. (summarize)</p>
<input type="checkbox"/> M_MAGNITUDE_TYPE	<p>Sets how to calculate the magnitude of the edge at each edgel position.</p> <p>For more information on the magnitude, see the Magnitude type subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_SQR_NORM .
<input type="checkbox"/> M_NORM	Specifies that the magnitude will be used.
<input type="checkbox"/> M_SQR_NORM	Specifies that the square of the magnitude will be used.
<input type="checkbox"/> M_SORT	<p>Sets the index sorting order for results obtained for multiple measured features from the starting reference subfeature, which is the subfeature closest to the origin of the region of interest. The indices of the other subfeatures are determined relative to the orientation of the region of interest.</p> <p>M_SORT can actually change how features are added. For example, if 3 point subfeatures are used to add an arc and their order is 1, 2, 3, sorting them descendingly will change the order to 3, 2, 1. This affects the arc's start and end point. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_SORT_UP .
<input type="checkbox"/> M_SORT_DOWN	Specifies to sort in descending order.

<input type="checkbox"/> M_SORT_UP	Specifies to sort in ascending order.
<input type="checkbox"/> M_THRESHOLD_MODE	<p>Sets the threshold of the edge extraction.</p> <p>For more information on thresholding edges, see the Thresholding subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_HIGH .
<input type="checkbox"/> M_DISABLE	<p>Specifies no threshold. All edges are extracted.</p> <p>(summarize)</p>
<input type="checkbox"/> M_HIGH	<p>Specifies a high threshold. M_HIGH should be used for images with some contrast variations, noise, and non-uniform illumination. M_HIGH always results in a lower (or equal) number of edgels than M_MEDIUM.</p> <p>(summarize)</p>
<input type="checkbox"/> M_LOW	<p>Specifies a low threshold. All edges over a minimum noise-based estimated threshold are extracted. M_LOW always results in a lower (or equal) number of edgels than M_DISABLE.</p> <p>(summarize)</p>
<input type="checkbox"/> M_MEDIUM	<p>Specifies a medium threshold. M_MEDIUM should be used for multi-contrast images, or for images with a lot of noise or non-uniform illumination. M_MEDIUM always results in a lower (or equal) number of edgels than M_LOW.</p> <p>(summarize)</p>
<input type="checkbox"/> M_USER_DEFINED	<p>Specifies that the threshold values will be user-defined. Set the threshold values with M_THRESHOLD_VALUE_LOW and M_THRESHOLD_VALUE_HIGH.</p> <p>(summarize)</p>
<input type="checkbox"/> M_VERY_HIGH	<p>Specifies a very high threshold. Only the strongest edges in the image are extracted. M_VERY_HIGH always results in a lower (or equal) number of edgels than M_HIGH.</p> <p>(summarize)</p>
<input type="checkbox"/> M_THRESHOLD_TYPE	<p>Sets the type of hysteresis threshold used when performing the edge extraction.</p> <p>Edge chains are built such that the magnitude values of all connected edgels are stronger than a lower bound threshold value and such that at least one of the edgel's magnitude in each edge chain is stronger than an upper bound threshold value.</p> <p>Note that M_THRESHOLD_TYPE sets how to use the lower and upper threshold bounds, while M_THRESHOLD_MODE defines what the bounds are.</p> <p>For more information on thresholding, see the Thresholding subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_HYSTERESIS .
<input type="checkbox"/> M_FULL_HYSTERESIS	<p>Specifies that the lower bound threshold value is 0.0.</p> <p>(summarize)</p>
<input type="checkbox"/> M_HYSTERESIS	Specifies that both the lower bound threshold value and the upper bound threshold value will be used.
<input type="checkbox"/> M_NO_HYSTERESIS	Specifies that the lower bound threshold value is equal to the upper bound threshold value.
<input type="checkbox"/> M_THRESHOLD_VALUE_HIGH	<p>Sets the upper bound of the hysteresis threshold value. M_THRESHOLD_VALUE_HIGH is ignored unless <code>MmetControl()</code> with M_THRESHOLD_MODE is set to M_USER_DEFINED. Note that lower threshold values result in a more sensitive edgel detection; that is, a higher (or equal) number of edgels are detected.</p> <p>For more information on thresholding, see the Thresholding subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 0.0.</p> <p>(summarize)</p>
<input type="checkbox"/> Value	Specifies the upper bound of the hysteresis threshold.
<input type="checkbox"/> M_THRESHOLD_VALUE_LOW	<p>Sets the lower bound of the hysteresis threshold value. M_THRESHOLD_VALUE_LOW is ignored unless <code>MmetControl()</code> with M_THRESHOLD_MODE is set to M_USER_DEFINED. Note that lower threshold values result in a more sensitive edgel detection; that is, a higher (or equal) number of edgels are detected.</p> <p>For more information on thresholding, see the Thresholding subsection in the Customizing the edge extraction settings section in Chapter 9: Edge Finder. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 0.0.</p> <p>(summarize)</p>

<input type="checkbox"/> Value	Specifies the lower bound of the hysteresis threshold.
--------------------------------	--------------------------------------------------------

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control how the features are calculated. For controlling features, the **LabelOrIndex** parameter can be set to the following: [M_GLOBAL_FRAME](#) or an existing feature label or index using [M_FEATURE_LABEL\(\)](#) or [M_FEATURE_INDEX\(\)](#).

For controlling features	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_ANGLE_END	Sets the end angle of a constructed arc. The angle is relative to the reference frame's X-axis and follows the counter-clockwise convention. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 360.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the end angle, in degrees.
<input type="checkbox"/> M_ANGLE_START	Sets the start angle of a constructed arc. The angle is relative to the reference frame's X-axis and follows the counter-clockwise convention. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the start angle, in degrees.
<input type="checkbox"/> M_EDGEL_ANGLE_RANGE	<p>Sets the gradient angle range. You can use M_EDGEL_ANGLE_RANGE, along with M_EDGEL_RELATIVE_ANGLE, to select the active edgels of a measured feature or the active edgels of a constructed feature based on physically measured edgels.</p> <p>The gradient angle (edgel direction) is the counter-clockwise angle between the target image's horizontal axis and the edge's perpendicular direction (from black to white) at each edgel location. Only edgels with a gradient angle that falls within the angular range can be used to build the feature. The gradient angle range is measured according to the relative angle (M_EDGEL_RELATIVE_ANGLE).</p> <p>In the following examples, the angular range is 60.0° and the relative angle is the same as the region's orientation. Note that for ring-sector, the orientation is radial.</p>  <p>Note that constructed edgel features are not built from a region, but from a specified measured feature. Therefore in this case, the "region's orientation" refers to the orientation of the region from which its specified measured feature was built. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 180.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angular range, in degrees.
<input type="checkbox"/> M_EDGEL_RELATIVE_ANGLE	<p>Sets the relative angle from which to measure the gradient angle range (M_EDGEL_ANGLE_RANGE). You can use M_EDGEL_RELATIVE_ANGLE, along with M_EDGEL_ANGLE_RANGE, to select the active edgels of a measured feature or the active edgels of a constructed feature based on physically measured edgels. Typically, the relative angle is the same as the region's orientation. For an example on how M_EDGEL_RELATIVE_ANGLE is applied, along with the gradient angle range, see M_EDGEL_ANGLE_RANGE.</p> <p>Note that constructed edgel features are not built from a region, but from a specified measured feature. Therefore in this case, the "region's orientation" refers to the orientation of the region from which its specified measured feature was built. (summarize)</p>

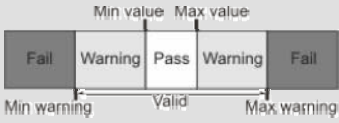
<input type="checkbox"/> M_DEFAULT	Same as M_SAME .
<input type="checkbox"/> M_REVERSE	Specifies that the gradient angle range is measured relative to the reverse angle (orientation) of the region of interest (+ 180°).
<input type="checkbox"/> M_SAME	Specifies that the gradient angle range is measured relative to the same angle (orientation) as the region of interest.
<input type="checkbox"/> M_SAME_OR_REVERSE	Specifies that the gradient angle range is measured relative to either the same or the reverse angle (orientation) of the region of interest.
<input type="checkbox"/> M_EDGEL_TYPE	Sets the type of edgels to use when building constructed edgel features using MmetAddFeature() with M_COPY_FEATURE_EDGELS . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ACTIVE_EDGELS .
<input type="checkbox"/> M_ACTIVE_EDGELS	Specifies the active edgels. Active edgels must satisfy all edgel constraints (M_EDGEL_ANGLE_RANGE and M_EDGEL_RELATIVE_ANGLE) and fit constraints (M_FIT_...). (summarize)
<input type="checkbox"/> M_ALL_EDGELS	Specifies all the edgels.
<input type="checkbox"/> M_FIT_COVERAGE_MIN	Sets the quantity of the fitted feature that must be covered by active edgels. In the following example, 50.0% of the segment feature covers the edgels. M_FIT_COVERAGE_MIN is valid for all fit operations for physically measured or constructed features.  (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the minimum coverage, in percent.
<input type="checkbox"/> M_FIT_DISTANCE_MAX	Sets the greatest possible gap between an active edgel and the projected fit. The higher the value, the farther away an active edgel can be. M_FIT_DISTANCE_MAX is valid for all fit operations for physically measured or constructed features. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_INFINITE .
<input type="checkbox"/> M_INFINITE	Specifies no maximum distance.
<input type="checkbox"/> Value	Specifies the maximum distance, in pixels.
<input type="checkbox"/> M_FIT_ITERATIONS_MAX	Sets the maximum number of fit iterations used to compute the feature. The more iterations, the better the fit, but the slower the calculation. M_FIT_ITERATIONS_MAX is valid for all fit operations for physically measured or constructed features. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_AUTO .
<input type="checkbox"/> M_AUTO	Specifies that the maximum number of fit iterations will be automatically determined.
<input type="checkbox"/> Value >= 1	Specifies the maximum number of fit iterations. Only integer values are accepted. A setting of one will consider all edgels/points in the fit. Settings higher than one will progressively eliminate outlying edgels/points in the fit. (summarize)
<input type="checkbox"/> M_FIT_VARIATION_MAX	Sets the maximum allowable difference in the feature's coefficients' values from one iteration to the next. If at least one coefficient of the feature varies more, between two consecutive fit iterations, than the specified value, the iteration process will continue. Note that: <ul style="list-style-type: none"> • An arc feature has 5 coefficients: the X-coordinate of its center, the Y-coordinate of its center, its radius, its start angle, and its end angle. • A circle feature has 3 coefficients: the X-coordinate of its center, the Y-coordinate of its center and its radius. • A segment feature has 4 coefficients: the X-coordinate of its starting point, the Y-coordinate of its starting point, the X-coordinate of its ending point, and the Y-coordinate of its ending point. <p>These coefficients are the same as those required when building the corresponding features with a parametric operation.</p> <p>M_FIT_VARIATION_MAX is valid for all fit operations for physically measured or constructed features. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_AUTO .
<input type="checkbox"/> 0.0 to 100.0	Specifies the maximum variation, as a percentage.
<input type="checkbox"/> M_AUTO	Specifies that the maximum variation will be determined automatically.
<input type="checkbox"/> M_LINE_A	Sets the coefficient <i>A</i> of the line equation for a parametrically constructed line.

	The line equation is: $A x + B y + C = 0$. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the coefficient's value.
<input type="checkbox"/> M_LINE_B	Sets the B coefficient of the line equation for a parametrically constructed line. The line equation is: $A x + B y + C = 0$. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value	Specifies the coefficient's value.
<input type="checkbox"/> M_LINE_C	Sets the C coefficient of the line equation for a parametrically constructed line. The line equation is: $A x + B y + C = 0$. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the coefficient's value.
<input type="checkbox"/> M_NUMBER_MAX	Sets the maximum number of subfeatures to calculate for a multiple feature. Note that M_NUMBER_MAX is valid for point features only. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1. (summarize)
<input type="checkbox"/> Value	Specifies the maximum number of subfeatures. This value must be greater than 1 for measured multiple point features. (summarize)
<input type="checkbox"/> M_NUMBER_MIN	Sets the minimum number of subfeatures to calculate for a multiple feature. Note that M_NUMBER_MIN is valid for point features only. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> Value	Specifies the minimum number of subfeatures. This value must be greater than 1 for measured multiple point features. (summarize)
<input type="checkbox"/> M_OCCURRENCE	Sets which point occurrence to build when there is more than one candidate. For more information, see the M_INTERSECTION operation of MmetAddFeature() for a constructed point feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> Value	Specifies the index value.
<input type="checkbox"/> M_POSITION	Sets the point on the feature's contour at which to construct the new feature. For more information, see the M_POSITION_ABSOLUTE and M_POSITION_RELATIVE operations of MmetAddFeature() . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> Value	Specifies the position value. Set the position in linear value for M_POSITION_ABSOLUTE and in percent for M_POSITION_RELATIVE . (summarize)
<input type="checkbox"/> M_POSITION_END_X	Sets the X-coordinate of the ending point of a segment constructed using MmetAddFeature() with M_PARAMETRIC . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)

<input type="checkbox"/> Value	Specifies the X-coordinate, in pixel or world units.
<input type="checkbox"/> M_POSITION_END_Y	Sets the Y-coordinate of the ending point of a segment constructed using MmetAddFeature() with M_PARAMETRIC . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the Y-coordinate, in pixel or world units.
<input type="checkbox"/> M_POSITION_START_X	Sets the X-coordinate of the starting point of a segment constructed using MmetAddFeature() with M_PARAMETRIC . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is -1.0. (summarize)
<input type="checkbox"/> Value	Specifies the X-coordinate, in pixel or world units.
<input type="checkbox"/> M_POSITION_START_Y	Sets the Y-coordinate of the starting point of a segment constructed using MmetAddFeature() with M_PARAMETRIC . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the Y-coordinate, in pixel or world units.
<input type="checkbox"/> M_POSITION_X	Sets the X-coordinate of the center of a constructed circle or arc, or sets the X-coordinate of a constructed local frame or point. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the X-coordinate, in pixel or world units.
<input type="checkbox"/> M_POSITION_Y	Sets the Y-coordinate of the center of a constructed circle or arc, or sets the Y-coordinate of a constructed local frame or point. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the Y-coordinate, in pixel or world units.
<input type="checkbox"/> M_RADIUS	Sets the radius of a constructed circle or arc. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the radius.
<input type="checkbox"/> M_REFERENCE_FRAME	Sets the reference frame for any feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies that the reference frame is the global frame. If the target image is not calibrated, the global frame's origin (0,0) is aligned with the target image's top-left corner. If the target image is calibrated, the global frame's origin is aligned with the origin of the calibration. (summarize)
<input type="checkbox"/> Value	Specifies the label of the reference frame.

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control the geometric tolerances calculated for a metrology context. For controlling geometric tolerances, the **LabelOrIndex** parameter can be set to an existing tolerance label or index using [M_TOLERANCE_LABEL\(\)](#) or [M_TOLERANCE_INDEX\(\)](#).

For controlling geometric tolerances	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_VALUE_MAX	Sets the maximum value for a geometric tolerance to have a valid status. To get the status of the geometric tolerance (M_PASS , M_WARNING , or M_FAIL), use MmetGetResult() with M_STATUS .

	<p>The geometric tolerance's status is determined according to the M_VALUE_MAX, M_VALUE_MIN, M_VALUE_WARNING_MAX, and M_VALUE_WARNING_MIN settings. The following is an example of (non-zero) minimum and maximum tolerance values and warning values:</p>  <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the maximum value.
<input type="checkbox"/> M_VALUE_MIN	<p>Sets the minimum value for a geometric tolerance to have a valid status. To get the status of the geometric tolerance (M_PASS, M_WARNING, or M_FAIL), use MmetGetResult() with M_STATUS.</p> <p>The geometric tolerance's status is determined according to the M_VALUE_MAX, M_VALUE_MIN, M_VALUE_WARNING_MAX, and M_VALUE_WARNING_MIN settings. For an example of (non-zero) minimum and maximum tolerance values and warning values, see M_VALUE_MAX. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the minimum value.
<input type="checkbox"/> M_VALUE_WARNING_MAX	<p>Sets the maximum warning value for a geometric tolerance to have a valid status. To get the status of the geometric tolerance (M_PASS, M_WARNING, or M_FAIL), use MmetGetResult() with M_STATUS.</p> <p>The geometric tolerance's status is determined according to the M_VALUE_MAX, M_VALUE_MIN, M_VALUE_WARNING_MAX, and M_VALUE_WARNING_MIN settings. For an example of (non-zero) minimum and maximum tolerance values and warning values, see M_VALUE_MAX. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the maximum warning value.
<input type="checkbox"/> M_VALUE_WARNING_MIN	<p>Sets the minimum warning value for a geometric tolerance to have a valid status. To get the status of the geometric tolerance (M_PASS, M_WARNING, or M_FAIL), use MmetGetResult() with M_STATUS.</p> <p>The geometric tolerance's status is determined according to the M_VALUE_MAX, M_VALUE_MIN, M_VALUE_WARNING_MAX, and M_VALUE_WARNING_MIN settings. For an example of (non-zero) minimum and maximum tolerance values and warning values, see M_VALUE_MAX. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the minimum warning value.

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control features and geometric tolerances calculated for a metrology context. For controlling features, the **LabelOrIndex** parameter can be set to the following: [M_GLOBAL_FRAME](#) or an existing feature label or index using [M_FEATURE_LABEL\(\)](#) or [M_FEATURE_INDEX\(\)](#). For controlling geometric tolerances, the **LabelOrIndex** parameter can be set to an existing tolerance label or index using [M_TOLERANCE_LABEL\(\)](#) or [M_TOLERANCE_INDEX\(\)](#).

● For controlling features or geometric tolerances	
ControlType	Description
ControlValue	
<input type="checkbox"/> M_ANGLE	<p>Sets the angle used to construct a feature or define a tolerance, when applicable. For more information, see MmetAddFeature() with M_ANGLE, M_ANGLE_ABSOLUTE, M_ANGLE_RELATIVE, M_SECTOR_RELATIVE, and M_PARAMETRIC; also see MmetAddTolerance() with M_ANGULARITY. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> Value	Specifies the angle. Depending on the build operation, you must set the angle in either degrees, or as a percentage.

[\(summarize\)](#)

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control the transformation applied to a clone-type feature. ([MmetAddFeature\(\)](#) with **M_CLONE_FEATURE**). For example, using [M_CLONE_SCALE](#), you can clone a circle feature at twice its size. For controlling clone transformations, you must set the **LabelOrIndex** parameter to an existing clone-type feature label or index using [M_FEATURE_LABEL\(\)](#) or [M_FEATURE_INDEX\(\)](#).

● For controlling clone transformations	
ControlType	Description
ControlValue	
M_CLONE_ANGLE	Sets the rotation angle, in the counterclockwise direction, that will be used when cloning the feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
M_CLONE_OFFSET_X	Sets the translation value, in the X-direction, that will be used when cloning the feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the X-offset for the cloned feature, in pixel or world units.
M_CLONE_OFFSET_Y	Sets the translation value, in the Y-direction, that will be used when cloning the feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the Y-offset for the cloned feature, in pixel or world units.
M_CLONE_SCALE	Sets the scale factor, that will be used when cloning the feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0.0	Specifies the scale factor.

The following **ControlType** and corresponding **ControlValue** parameter settings are used to control the results calculated and stored in a metrology result buffer. For controlling results, the **LabelOrIndex** parameter must be set to [M_GENERAL](#).

● For controlling results	
ControlType	Description
ControlValue	
M_DRAW_RELATIVE_ORIGIN_X	Sets the X-coordinate of the top-left corner of the region in the metrology result buffer to use when drawing in the destination image buffer (MmetDraw()). This control type is not saved with the context. If you restore the context, this control type must be set again. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the relative X-offset, in pixel or world units.
M_DRAW_RELATIVE_ORIGIN_Y	Sets the Y-coordinate of the top-left corner of the region in the metrology result buffer to use when drawing in the destination image buffer (MmetDraw()). This control type is not saved with the context. If you restore the context, this control type must be set again. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels.

	(summarize)
<input type="checkbox"/> Value	Specifies the relative Y-offset, in pixel or world units.
<input checked="" type="checkbox"/> M_DRAW_SCALE_X	Sets the scale in the X-direction when drawing results in the destination image buffer (MmetDraw()). This control type is not saved with the context. If you restore the context, this control type must be set again. (summarize)
<input checked="" type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the scale in the X-direction.
<input checked="" type="checkbox"/> M_DRAW_SCALE_Y	Sets the scale in the Y-direction when drawing results in the destination image buffer (MmetDraw()). This control type is not saved with the context. If you restore the context, this control type must be set again. (summarize)
<input checked="" type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the scale in the Y-direction.
<input checked="" type="checkbox"/> M_OUTPUT_FRAME	Sets the output reference frame that will be used to return the feature results. All results are returned relative to the output reference frame that you set. The output reference frame is reset to M_GLOBAL_FRAME after each call to MmetCalculate() . To return results in either pixel or world units, use McalControl() with M_OUTPUT_UNITS . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_GLOBAL_FRAME .
<input checked="" type="checkbox"/> M_GLOBAL_FRAME	Specifies that the global frame is used to return the feature results. If the target image is not calibrated, results are returned relative to the top-left corner of the target image. If the target image is calibrated, results are returned relative to the origin of the calibration. (summarize)
<input type="checkbox"/> M_IMAGE_FRAME	Specifies that the target image reference frame is used to return the feature results. The origin of the target image reference frame is the top-left corner of the target image, even if the target image is calibrated. (summarize)
<input checked="" type="checkbox"/> M_REFERENCE_FRAME	Specifies that the reference frame of each feature is used to return the feature results. The reference frame of a feature is the local frame in which the feature is defined. (summarize)
<input type="checkbox"/> Value	Specifies the label of an existing local frame feature to use to return the feature results.

The following **ControlType** and corresponding **ControlValue** parameter settings are used to delete features and tolerances from the context. The **LabelOrIndex** parameter can be set to one of the following: [M_ALL_FEATURES](#), [M_ALL_TOLERANCES](#), an existing feature label or index using [M_FEATURE_LABEL\(\)](#) or [M_FEATURE_LABEL\(\)](#), or an existing tolerance label or index using [M_TOLERANCE_LABEL\(\)](#) or [M_TOLERANCE_INDEX\(\)](#).

For deleting features and tolerances	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input checked="" type="checkbox"/> M_DELETE	Deletes a specific feature, a specific tolerance, all features, or all tolerances. To delete all features, the LabelOrIndex parameter needs to be set to M_ALL_FEATURES . When deleting a feature, all its derived features and associated tolerances are also deleted. To delete all tolerances, the LabelOrIndex parameter needs to be set to M_ALL_TOLERANCES . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_NULL .
<input checked="" type="checkbox"/> M_NULL	Deletes a specific feature, a specific tolerance, all features, or all tolerances.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetDraw

Synopsis

Draw specific metrology feature results in the destination image buffer.

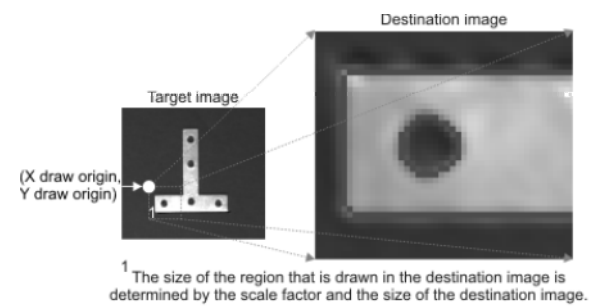
Syntax

```
void MmetDraw(  
    MIL_ID GraphContId,  
    MIL_ID ResultOrContextId,  
    MIL_ID DestImageId,  
    MIL_INT Operation,  
    MIL_INT LabelOrIndex,  
    MIL_INT ControlFlag  
)
```

Description

This function draws specific metrology context, feature or tolerance results in the destination buffer.

You can also draw features of a zoomed region of the target image by specifying the appropriate values for [M_DRAW_RELATIVE_ORIGIN_X](#), [M_DRAW_RELATIVE_ORIGIN_Y](#), [M_DRAW_SCALE_X](#), and [M_DRAW_SCALE_Y](#) control types of [MmetControl\(\)](#). The relative origin values specify, in pixels, the coordinates of the top-left corner of the region in the target image, while the scale values specify the X- and Y-scaling factors used to fill the destination image buffer.



When zooming, **MmetDraw()** will draw into the destination image buffer even if the buffer is not large enough to contain all of the zoomed image. If necessary, the image will be clipped.

If a template reference has been set using [MmetControl\(\)](#) with [M_TEMPLATE_REFERENCE_ID](#), **MmetDraw()** can draw the results of features and tolerances extracted from that template reference. You will need to inquire [M_TEMPLATE_REFERENCE_SIZE_X](#) and [M_TEMPLATE_REFERENCE_SIZE_Y](#) to allocate an appropriate destination buffer.

Parameters

GraphContId

Specifies the graphics context to use when drawing. This parameter must be set to one of the following values:

● For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

ResultOrContextId

Specifies the metrology result buffer or context from which to extract the features to draw. The metrology result buffer must have been previously allocated using [MmetAllocResult\(\)](#). The metrology context must have been previously allocated using [MmetAlloc\(\)](#).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The metrology result buffer must be allocated on the same system as the graphics context (**GraphContId**). If it is not, an error will occur.

DestImageId

Specifies the identifier of the destination image buffer in which to draw. The buffer can be any supported MIL image buffer. By drawing into its display's overlay buffer, you can also annotate the image non-destructively.

When using **MmetDraw()** to draw features with a source model that is associated to a calibration object and if a calibration object is associated to the destination buffer, the latter must also be calibrated to the same world coordinate system. If no calibration object is associated to the destination buffer, the latter will use the same one as the template reference's. To add a template reference to a metrology context, use [MmetControl\(\)](#) with [M_TEMPLATE_REFERENCE_ID](#). **MmetDraw()** draws all the requested features according to whether the destination image buffer is physically corrected or not.

Operation

Specifies the type of drawing operation to perform. This parameter can be set to the following values:

● For specifying the type of drawing operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_ACTIVE_EDGELS	Draws only the edgels that are used to measure features. The LabelOrIndex parameter can be set to M_FEATURE_LABEL() , M_FEATURE_INDEX() , M_ALL_FEATURES , or a specific label/index value (Value). (summarize)
<input type="checkbox"/> M_DRAW_ALL_EDGELS	Draws all the edgels that were calculated in the target image. This includes the edgels that will be used to measure a feature. The LabelOrIndex parameter can be set to M_FEATURE_LABEL() , M_FEATURE_INDEX() , M_ALL_FEATURES , or a specific label/index value (Value). (summarize)
<input type="checkbox"/> M_DRAW_FEATURE +	Draws the feature. The LabelOrIndex parameter can be set to M_GLOBAL_FRAME , M_FEATURE_LABEL() , M_FEATURE_INDEX() , M_ALL_FEATURES , or a specific label/index value (Value). (summarize)
<input type="checkbox"/> M_DRAW_REFERENCE_FEATURES	Draws the features used to define a tolerance. The LabelOrIndex parameter can be set to M_TOLERANCE_LABEL() , M_TOLERANCE_INDEX() , M_ALL_TOLERANCES , M_ALL_PASS_TOLERANCES , M_ALL_FAIL_TOLERANCES , M_ALL_WARNING_TOLERANCES , or a specific label/index value (Value). (summarize)
<input type="checkbox"/> M_DRAW_REGION	Draws the feature's search region. The LabelOrIndex parameter can be set to M_FEATURE_LABEL() , M_FEATURE_INDEX() , M_ALL_FEATURES , or a specific label/index value (Value). (summarize)
<input type="checkbox"/> M_DRAW_TEMPLATE_REFERENCE	Draws the template reference. The LabelOrIndex parameter must be set to M_GENERAL . (summarize)
<input type="checkbox"/> M_DRAW_TOLERANCE +	Draws the geometric tolerance's symbol. The LabelOrIndex parameter can be set to M_TOLERANCE_LABEL() , M_TOLERANCE_INDEX() , M_ALL_TOLERANCES , M_ALL_PASS_TOLERANCES , M_ALL_FAIL_TOLERANCES , M_ALL_WARNING_TOLERANCES , or a specific label/index value (Value). Tolerance symbols are illustrated in MmetAddTolerance() . (summarize)

Combination constant for [M_DRAW_FEATURE](#); [M_DRAW_TOLERANCE](#);

You can add the following value to the above-mentioned values to specify that the label of the feature or tolerance will also be drawn.

● For M_DRAW_FEATURE and M_DRAW_TOLERANCE	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_LABEL	Specifies that the label will also be drawn.

LabelOrIndex

Specifies the label or index of the result in the metrology result buffer for which to perform the drawing operation. Set this parameter to one of the following values:

● For specifying the index of the element in the metrology result buffer			
☐ Value	Description		
☐ <code>M_FEATURE_LABEL(MIL_INT FeatureLabel)</code>	<p>Specifies the label value of an existing individual feature to draw, in either the metrology context or result buffer, depending on whether you provide a context or result buffer to the ResultOrContextId parameter.</p> <p>(summarize)</p> <p><i>Parameters</i></p> <p><i>FeatureLabel</i> This parameter specifies the label of the individual feature to draw. You can set this parameter to the following:</p> <table> <tr> <td>Value > 0</td><td>Specifies the label of the individual feature.</td></tr> </table>	Value > 0	Specifies the label of the individual feature.
Value > 0	Specifies the label of the individual feature.		
☐ <code>M_FEATURE_INDEX(MIL_INT FeatureIndex)</code>	<p>Specifies the index value of an existing individual feature to draw, in either the metrology context or result buffer, depending on whether you provide a context or result buffer to the ResultOrContextId parameter.</p> <p>(summarize)</p> <p><i>Parameters</i></p> <p><i>FeatureIndex</i> This parameter specifies the index of the individual feature to draw. You can set this parameter to the following:</p> <table> <tr> <td>Value >= 0</td><td>Specifies the index of the individual feature. Note that the index of the global frame is 0.</td></tr> </table>	Value >= 0	Specifies the index of the individual feature. Note that the index of the global frame is 0.
Value >= 0	Specifies the index of the individual feature. Note that the index of the global frame is 0.		
☐ <code>M_TOLERANCE_LABEL(MIL_INT ToleranceLabel)</code>	<p>Specifies the label value of an existing individual tolerance to draw, in either the metrology context or result buffer, depending on whether you provide a context or result buffer to the ResultOrContextId parameter.</p> <p>(summarize)</p> <p><i>Parameters</i></p> <p><i>ToleranceLabel</i> This parameter specifies the label of the individual tolerance to draw. You can set this parameter to the following:</p> <table> <tr> <td>Value > 0</td><td>Specifies the label of the individual tolerance.</td></tr> </table>	Value > 0	Specifies the label of the individual tolerance.
Value > 0	Specifies the label of the individual tolerance.		
☐ <code>M_TOLERANCE_INDEX(MIL_INT ToleranceIndex)</code>	<p>Specifies the index value of an existing individual tolerance to draw, in either the metrology context or result buffer, depending on whether you provide a context or result buffer to the ResultOrContextId parameter.</p> <p>(summarize)</p> <p><i>Parameters</i></p> <p><i>ToleranceIndex</i> This parameter specifies the index of the individual tolerance to draw. You can set this parameter to the following:</p> <table> <tr> <td>Value >= 0</td><td>Specifies the index of the individual tolerance.</td></tr> </table>	Value >= 0	Specifies the index of the individual tolerance.
Value >= 0	Specifies the index of the individual tolerance.		
☐ <code>M_ALL_FAIL_TOLERANCES</code>	Specifies that the drawing operation will be performed on all tolerance results that have the status M_FAIL .		
☐ <code>M_ALL_FEATURES</code>	Specifies to draw all feature results.		
☐ <code>M_ALL_PASS_TOLERANCES</code>	Specifies that the drawing operation will be performed on all tolerance results that have the status M_PASS .		
☐ <code>M_ALL_TOLERANCES</code>	Specifies to draw all tolerance results.		
☐ <code>M_ALL_WARNING_TOLERANCES</code>	Specifies that the drawing operation will be performed on all tolerance results that have the status M_WARNING .		
☐ <code>M_GENERAL</code>	Specifies to draw a setting of the context.		
☐ <code>M_GLOBAL_FRAME</code>	Specifies to draw the global frame of the context.		
☐ Value	Specifies the result label or index for which to perform the drawing operation.		

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetFree

Synopsis

Free a metrology context or a metrology result buffer.

Syntax

```
void MmetFree(  
    MIL_ID ResultIdOrContextId  
)
```

Description

This function deletes the specified metrology context or metrology result buffer identifier, and releases any memory associated with it.

All metrology contexts and all metrology result buffers allocated on a particular system must be freed before the system can be freed.

Parameter

ResultIdOrContextId

Specifies the identifier of the metrology context or metrology result buffer to free. These must have been successfully allocated (with [MmetAlloc\(\)](#) or [MmetAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetGetResult

Synopsis

Get the specified type of results from a MIL metrology result buffer.

Syntax

```
void MmetGetResult(
    MIL_ID ResultId,
    MIL_INT LabelOrIndex,
    MIL_INT ResultType,
    void *ResultArrayPtr
)
```

Description

This function retrieves all results of the specified type from a metrology result buffer, after an [MmetCalculate\(\)](#) call. The result entries are ordered by label value.

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with [M_OUTPUT_UNITS](#) set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image.

All results are returned relative to the output reference frame that you set using [MmetControl\(\)](#) with [M_OUTPUT_FRAME](#).

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set **ResultArrayPtr** to **M_NULL**. When this parameter is set to **M_NULL**, the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call **MmetGetResult()** again and you pass an array to the **ResultArrayPtr** parameter. You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

By setting the **LabelOrIndex** parameter to [M_INTERACTIVE](#), you can view the results interactively.

Parameters

ResultId

Specifies the identifier of the metrology result buffer from which to get results.

LabelOrIndex

Specifies the feature or geometric tolerance for which to get results, or opens an interactive dialog box that displays the results stored in the result buffer. Labels must be greater than 0 while indices can be equal to 0.

For specifying a context, feature, tolerance, or result buffer		
Value	Description	
<input type="checkbox"/> M_DEFAULT	Same as M_GENERAL .	
<input type="checkbox"/> M_FEATURE_LABEL(MIL_INT FeatureLabel)	Specifies the label of an existing individual feature for which to get results. (summarize)	
		Parameters
	FeatureLabel	Specifies the label.
<input type="checkbox"/> M_FEATURE_INDEX(MIL_INT FeatureIndex)	Specifies the index value of an existing individual feature for which to get results. (summarize)	

		<i>Parameters</i>
		<i>FeatureIndex</i> Specifies the index.
<input type="checkbox"/> M_TOLERANCE_LABEL(MIL_INT <i>ToleranceLabel</i>)	Specifies the label value of an existing individual tolerance for which to get results. (summarize)	
		<i>Parameters</i>
		<i>ToleranceLabel</i> Specifies the label.
<input type="checkbox"/> M_TOLERANCE_INDEX(MIL_INT <i>ToleranceIndex</i>)	Specifies the index value of an existing individual tolerance for which to get results. (summarize)	
		<i>Parameters</i>
		<i>ToleranceIndex</i> Specifies the index.
<input type="checkbox"/> M_ALL	Specifies to return results about all features and tolerances. This should be used only with M_STATUS . (summarize)	
<input type="checkbox"/> M_ALL_FEATURES	Specifies to return results about all features. This should be used only with M_STATUS . (summarize)	
<input type="checkbox"/> M_ALL_TOLERANCES	Specifies to return results about all tolerances. This should be used only with M_STATUS . (summarize)	
<input type="checkbox"/> M_GENERAL	Specifies that the results relating to the entire metrology context will be returned.	
<input type="checkbox"/> M_GLOBAL_FRAME	Specifies that information about the global frame of the context will be returned.	

To display the results currently stored in the result buffer in an interactive dialog box, select the following.

● For displaying results in an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box containing a spreadsheet that displays the types of results stored in the result buffer. The results are displayed in separate columns. If there are queued requests, only those results are displayed. (summarize)

ResultType

Specifies the type of result to retrieve.

To retrieve general results, the **LabelOrIndex** parameter must be set to [M_GENERAL](#).

You must set this parameter to **M_NULL** if you are setting the **LabelOrIndex** parameter to [M_INTERACTIVE](#).

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a MIL_DOUBLE.

● For retrieving general results	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NUMBER_OF_CONSTRUCTED_FEATURES +	Returns the number of constructed features.
<input type="checkbox"/> M_NUMBER_OF_FEATURES +	Returns the total number of features. This total includes both measured features and constructed features. (summarize)
<input type="checkbox"/> M_NUMBER_OF_FEATURES_FAIL +	Returns the number of features that are not successfully calculated. The status of those features is M_FAIL .

	(summarize)
<input type="checkbox"/> M_NUMBER_OF_FEATURES_PASS +	Returns the number of features that are successfully calculated. The status of those features is M_PASS . (summarize)
<input type="checkbox"/> M_NUMBER_OF_MEASURED_FEATURES +	Returns the number of measured features.
<input type="checkbox"/> M_NUMBER_OF_TOLERANCES +	Returns the number of tolerances.
<input type="checkbox"/> M_NUMBER_OF_TOLERANCES_FAIL +	Returns the number of tolerances that have the status M_FAIL , which is the resulting status when tolerance values are smaller than M_VALUE_MIN or greater than M_VALUE_MAX .
<input type="checkbox"/> M_NUMBER_OF_TOLERANCES_PASS +	Returns the number of tolerances that have the status M_PASS , which is the resulting status when it is not M_FAIL or M_WARNING .
<input type="checkbox"/> M_NUMBER_OF_TOLERANCES_WARNING +	Returns the number of tolerances that have the status M_WARNING , which is the resulting status when tolerance values are greater or equal to M_VALUE_MIN and lower than M_VALUE_WARNING_MIN . It also occurs when tolerance values are lower or equal to M_VALUE_MAX and greater than M_VALUE_WARNING_MAX . (summarize)

To retrieve results for features, the **LabelOrIndex** parameter can be set to **M_GLOBAL_FRAME** or an existing feature label or index using **M_FEATURE_LABEL()** or **M_FEATURE_INDEX()**.

Unless otherwise specified, the following values require that you pass the **ResultArrayPtr** parameter the address of a **MIL_DOUBLE**.

● For retrieving results for features	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ANGLE +	Returns the aperture of an arc or the angle of a segment, a line, or a local frame. This angle is relative to the output frame's X-axis. (summarize)
<input type="checkbox"/> M_ANGLE_END +	Returns the end angle of an arc. This angle is relative to the output frame's X-axis. (summarize)
<input type="checkbox"/> M_ANGLE_START +	Returns the start angle of an arc. This angle is relative to the output frame's X-axis. (summarize)
<input type="checkbox"/> M_LENGTH +	Returns the perimeter of a circle or the length of a segment or arc feature.
<input type="checkbox"/> M_LINE_A +	Returns the A constant of the equation for a constructed line. The line equation is: $Ax + By + C = 0$. (summarize)
<input type="checkbox"/> M_LINE_B +	Returns the B constant of the equation for a constructed line. The line equation is: $Ax + By + C = 0$. (summarize)
<input type="checkbox"/> M_LINE_C +	Returns the C constant of the equation for a constructed line. The line equation is: $Ax + By + C = 0$. (summarize)
<input type="checkbox"/> M_NUMBER +	Returns the number of subfeatures in a multiple feature or the number of edgels in an edgel feature.
<input type="checkbox"/> M_POSITION_END_X +	Returns the X-coordinate of the ending point of a segment or arc.
<input type="checkbox"/> M_POSITION_END_Y +	Returns the Y-coordinate of the ending point of a segment or arc.
<input type="checkbox"/> M_POSITION_START_X +	Returns the X-coordinate of the starting point of a segment or arc.
<input type="checkbox"/> M_POSITION_START_Y +	Returns the Y-coordinate of the starting point of a segment or arc.
<input type="checkbox"/> M_POSITION_X +	Returns the X-coordinate of a point or edgel, the X-coordinate of the center of a circle or arc, or the X-coordinate of the origin of a local frame. For measured points and edgels, you must retrieve M_NUMBER to allocate the appropriate size of the array. (summarize)
<input type="checkbox"/> M_POSITION_Y +	Returns the Y-coordinate of a point or edgel, the Y-coordinate of the center of a circle or arc, or the Y-coordinate of the origin of a local frame. For measured points and edgels, you must retrieve M_NUMBER to allocate the appropriate size of the array. (summarize)
<input type="checkbox"/> M_RADIUS +	Returns the radius of a circle or arc feature.

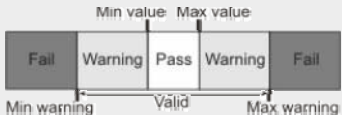
To retrieve tolerance results, the **LabelOrIndex** parameter can be set to an existing tolerance label or index using [M_TOLERANCE_LABEL\(\)](#) or using [M_TOLERANCE_INDEX\(\)](#).

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a `MIL_DOUBLE`.

● For retrieving results for tolerance	
☐ Value	Description
☐ M_TOLERANCE_VALUE +	Returns the calculated geometric tolerance value.

To retrieve the status of the calculation of features and tolerances, the **LabelOrIndex** parameter can be set to one of the following: [M_ALL_FEATURES](#), [M_ALL_TOLERANCES](#), [M_ALL](#), [M_GENERAL](#), an existing feature label or index using [M_FEATURE_LABEL\(\)](#) or [M_FEATURE_INDEX\(\)](#), or an existing tolerance label or index using [M_TOLERANCE_LABEL\(\)](#) or [M_TOLERANCE_INDEX\(\)](#).

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a `MIL_DOUBLE`.

● For retrieving status for features and tolerances							
☐ Value	Description						
☐ M_STATUS +	<p>Returns the status of the calculation of features and tolerances.</p> <p>To retrieve the status of the calculation of a specific feature, set the LabelOrIndex parameter to an existing feature label or index using M_FEATURE_LABEL() or M_FEATURE_INDEX(). To retrieve the status of the calculation of a specific tolerance, set the LabelOrIndex parameter to an existing tolerance label or index using M_TOLERANCE_LABEL() or M_TOLERANCE_INDEX().</p> <p>To retrieve the status of the calculation of all features, set the LabelOrIndex parameter to M_ALL_FEATURES. To retrieve the status of the calculation of all tolerances, set the LabelOrIndex parameter to M_ALL_TOLERANCES. To retrieve the status of the calculation of all features and tolerances, set the LabelOrIndex parameter to M_ALL.</p> <p>To retrieve the global calculation status of all features and tolerances, set the LabelOrIndex parameter to M_GENERAL. In this case, the status will fail unless all features and tolerances were successfully calculated.</p> <p>The geometric tolerance's status is determined by whether it (M_TOLERANCE_VALUE) adheres to the thresholds set using MmetControl() with M_VALUE_MAX, M_VALUE_MIN, M_VALUE_WARNING_MAX, and M_VALUE_WARNING_MIN. The following is an example of (non-zero) minimum and maximum tolerance value and warning value thresholds:</p>  <p>(summarize)</p> <p><i>ResultArrayPtr info</i></p> <p>Return values:</p> <table border="0"> <tr> <td>M_FAIL</td><td>The feature was not calculated successfully, and/or the tolerance has not passed all restrictions.</td></tr> <tr> <td>M_PASS</td><td>The feature was calculated successfully, and/or the geometric tolerance has passed all restrictions.</td></tr> <tr> <td>M_WARNING</td><td>The geometric tolerance has passed all restrictions, with a warning. Warnings can be used to advise you when a tolerance is at its limit, and is in danger of failing.</td></tr> </table>	M_FAIL	The feature was not calculated successfully, and/or the tolerance has not passed all restrictions.	M_PASS	The feature was calculated successfully, and/or the geometric tolerance has passed all restrictions.	M_WARNING	The geometric tolerance has passed all restrictions, with a warning. Warnings can be used to advise you when a tolerance is at its limit, and is in danger of failing.
M_FAIL	The feature was not calculated successfully, and/or the tolerance has not passed all restrictions.						
M_PASS	The feature was calculated successfully, and/or the geometric tolerance has passed all restrictions.						
M_WARNING	The geometric tolerance has passed all restrictions, with a warning. Warnings can be used to advise you when a tolerance is at its limit, and is in danger of failing.						

Combination constant for [the values listed in](#) all tables

You can add the following value to the above-mentioned values to determine whether a result is available.

● For a general result or occurrence	
☐ Value	Description
☐ M_AVAILABLE	Returns whether a result is available to be retrieved for a general result, an occurrence, or for all occurrences. (summarize)
	<i>ResultType info</i>

Return values:

M_NULL	The result is not available to be retrieved.
Value != 0	The result is available to be retrieved.

Combination constants for [the values listed in](#) all tables

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

● For specifying the data type	
☐ Value	Description
☐ M_TYPE_DOUBLE	<p>Casts the requested results to a <i>double</i>. This is the default value. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type double Array size: Depends on the result type. Note: When multiple results. • Data type: double Note: When a single result.
☐ M_TYPE_LONG	<p>Casts the requested results to a <i>long</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type long Array size: Depends on the result type. Note: When multiple results. • Data type: long Note: When a single result.
☐ M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_DOUBLE Note: When a single result.
☐ M_TYPE_MIL_ID	<p>Casts the requested results to a <i>MIL_ID</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_ID Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_ID Note: When a single result.
☐ M_TYPE_MIL_INT	<p>Casts the requested results to a <i>MIL_INT</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_INT Note: When a single result.
☐ M_TYPE_MIL_INT32	<p>Casts the requested results to a <i>MIL_INT32</i>. (summarize)</p>

	<div>ResultArrayPtr info</div> <ul style="list-style-type: none">• Data type: array of type MIL_INT32 Array size: Depends on the result type. Note: When multiple results.• Data type: MIL_INT32 Note: When a single result.
<div><div></div>M_TYPE_MIL_INT64</div>	<div>Casts the requested results to a <i>MIL_INT64</i>. (summarize)</div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none">• Data type: array of type MIL_INT64 Array size: Depends on the result type. Note: When multiple results.• Data type: MIL_INT64 Note: When a single result.

ResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• array of type double• array of type long• array of type MIL_DOUBLE• array of type MIL_ID• array of type MIL_INT• array of type MIL_INT32• array of type MIL_INT64• double• long• MIL_DOUBLE• MIL_ID• MIL_INT• MIL_INT32• MIL_INT64

Specifies the address of the first element of the array in which to write the requested information.

You must set this parameter to **M_NULL** if you are putting the result request on the local queue of the result buffer, or if you are setting the **LabelOrIndex** parameter to **M_INTERACTIVE**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetInquire

Synopsis

Inquire information about a specified metrology context, feature, tolerance, or result buffer.

Syntax

```
MIL_INT MmetInquire(
    MIL_ID ContextOrResultId,
    MIL_INT LabelOrIndex,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires information about a specified metrology context, feature, tolerance, or result buffer. Note that calculated metrology result buffer values can be obtained with [MmetGetResult\(\)](#).

If the inquired setting is set to **M_DEFAULT** (for example, in [MmetControl\(\)](#)), **MmetInquire()** will return **M_DEFAULT**. To inquire the actual default value, add **M_DEFAULT** to the **InquireType** parameter.

By setting the **InquireType** parameter to **M_INTERACTIVE**, you can view the setting of inquire types interactively.

Parameters

ContextOrResultId

Specifies the identifier of the metrology context or result buffer about which to inquire information. Both the metrology context and the metrology result buffer must have been previously allocated on the system using [MmetAlloc\(\)](#) or [MmetAllocResult\(\)](#), respectively.

You can only set this parameter to a metrology context if you are setting the **InquireType** parameter to **M_INTERACTIVE**.

LabelOrIndex

Specifies that information will be inquired about a metrology context, feature, tolerance, or result buffer. This parameter should be set to one of the following values.

Unless otherwise specified, you can use the following values to inquire about a metrology context or result buffer, depending on whether you provide a context or result buffer to the **ContextOrResultId** parameter.

● For specifying a context, feature, tolerance, or result buffer	
☐ Value	Description
☐ M_DEFAULT	Specifies the default. If ContextOrResultId is set to a metrology context, M_DEFAULT is the same as M_CONTEXT . If ContextOrResultId is set to a metrology result buffer, M_DEFAULT is the same as M_GENERAL . (summarize)
☐ M_FEATURE_LABEL(MIL_INT FeatureLabel)	Specifies the label value of an existing individual feature about which to inquire. (summarize)
	Parameters
	FeatureLabel This parameter specifies the label of the individual feature about which to inquire. You can set this parameter to the following:
	Value > 0 Specifies the label of the individual feature.
☐ M_FEATURE_INDEX(Specifies the index value of an existing individual feature.

<input type="checkbox"/> MIL_INT <i>FeatureIndex</i>)	(summarize)
	<i>Parameters</i>
	<i>FeatureIndex</i> This parameter specifies the index of the individual feature about which to inquire. You can set this parameter to the following:
	Value >= 0 Specifies the index of the individual feature. Note that 0 is the index of the global frame.
<input type="checkbox"/> M_TOLERANCE_LABEL(MIL_INT <i>ToleranceLabel</i>)	Specifies the label value of an existing individual tolerance. (summarize)
	<i>Parameters</i>
	<i>ToleranceLabel</i> This parameter specifies the label of the individual tolerance about which to inquire. You can set this parameter to the following:
	Value > 0 Specifies the label of the individual tolerance.
<input type="checkbox"/> M_TOLERANCE_INDEX(MIL_INT <i>ToleranceIndex</i>)	Specifies the index value of an existing individual tolerance. (summarize)
	<i>Parameters</i>
	<i>ToleranceIndex</i> This parameter specifies the index of the individual tolerance about which to inquire. You can set this parameter to the following:
	Value >= 0 Specifies the index of the individual tolerance.
<input type="checkbox"/> M_CONTEXT	Inquires information about a global setting of a metrology context. When using M_CONTEXT , ContextOrResultId must specify a metrology context. (summarize)
<input type="checkbox"/> M_GENERAL	Inquires information about a setting of a metrology result buffer. This setting applies to all features and tolerances in the metrology result buffer. When using M_GENERAL , ContextOrResultId must specify a metrology result buffer. M_GENERAL is not supported if the InquireType parameter is set to M_INTERACTIVE . (summarize)
<input type="checkbox"/> M_GLOBAL_FRAME	Inquires information about the global frame of the metrology template.

InquireType

Specifies the type of setting about which to inquire.

To display the settings of the metrology context's inquire types in an interactive dialog box, select the following:

*Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter *M_NULL*.*

● For opening an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a read-only dialog box that displays the settings of the context's inquire types.

To inquire about general metrology context settings, set this parameter to one of the following values. To inquire general context settings, the **LabelOrIndex** parameter must be set to [M_CONTEXT](#).

*Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a *MIL_DOUBLE*.*

● For inquiring general context settings	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ASSOCIATED_CALIBRATION +	Returns the calibration object associated with the specified template reference.

	(summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: MIL Calibration object identifier; M_DEFAULT; M_NULL; (details)</div> </div>
<input type="checkbox"/> M_MODIFICATION_COUNT +	<div> <div>Returns the current value of the modification counter. The modification counter is increased by one each time settings for the context are modified. Although you cannot identify the modification counter's contents, you can compare them throughout your application to know if the context has been altered. If the modification counter has changed you can, for example, prompt the user to save before closing the application.</div> <div>(summarize)</div> </div> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div>Value Object modification count</div> </div>
<input type="checkbox"/> M_NUMBER_OF_CONSTRUCTED_FEATURES +	<div> <div>Returns the number of constructed features in the template of the metrology context.</div> <div>(summarize)</div> </div> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div>Value Number of constructed features.</div> </div>
<input type="checkbox"/> M_NUMBER_OF_FEATURES +	<div> <div>Returns the total number of features in the template of the metrology context. This total includes both measured features and constructed features.</div> <div>(summarize)</div> </div> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div>Value Total number of features.</div> </div>
<input type="checkbox"/> M_NUMBER_OF_MEASURED_FEATURES +	<div> <div>Returns the number of measured features in the template of the metrology context.</div> <div>(summarize)</div> </div> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div>Value Number of measured features.</div> </div>
<input type="checkbox"/> M_NUMBER_OF_TOLERANCES +	<div> <div>Returns the number of geometric tolerances in the template of the metrology context.</div> <div>(summarize)</div> </div> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div>Value The number of geometric tolerances.</div> </div>
<input type="checkbox"/> M_TEMPLATE_REFERENCE_ID +	<div> <div>Returns whether the template reference (image) was added to the context.</div> <div>(summarize)</div> </div> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div>M_NULL The template reference (image) was not added to the context.</div> <div>non-null (!M_NULL) The template reference (image) was added to the context.</div> </div>
<input type="checkbox"/> M_TEMPLATE_REFERENCE_SIZE_BAND +	<div> <div>Returns the number of bands that template reference has.</div> <div>(summarize)</div> </div> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div>Value Returns the number of bands.</div> </div>
<input type="checkbox"/> M_TEMPLATE_REFERENCE_SIZE_X +	<div> <div>Returns the width of the template reference, in pixels.</div> <div>(summarize)</div> </div> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div>Value Returns the template reference's width.</div> </div>

M_TEMPLATE_REFERENCE_SIZE_Y +	Returns the height of the template reference, in pixels. (summarize)
	<div>UserVarPtr info</div> Return values: Value Returns the template reference's height.
M_TEMPLATE_REFERENCE_TYPE +	Returns the data type and the data depth (per band) of the template reference. (summarize)
	<div>UserVarPtr info</div> Return values: Value Returns the template reference's data type and data depth.
M_TIMEOUT +	Returns the maximum measurement-and-validation time for MmetCalculate() , in msec. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_DISABLE; Value > 0.0; (details)

To inquire about the system on which either the metrology context or metrology result buffer has been allocated, set this parameter to the value below.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

For inquiring about the system	
Value	Description
M_OWNER_SYSTEM +	Returns the identifier of the system on which either the metrology context or metrology result buffer has been allocated. (summarize)
	<div>UserVarPtr info</div> Return values: MIL system identifier; M_DEFAULT_HOST; (details)

To inquire about the processing settings for a measured feature, set this parameter to one of the following values. To inquire global processing settings, the **LabelOrIndex** parameter must be set to an existing measured feature label or index using [M_FEATURE_LABEL\(\)](#) or [M_FEATURE_INDEX\(\)](#).

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

For inquiring global processing settings	
Value	Description
M_CONTEXT_TYPE +	Returns the type of edges to extract from the target image.
M_EXTRACTION_SCALE +	Returns the image scale at which to extract the edges. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; Value > 0.0; (details)
M_FILTER_MODE +	Returns the mode in which to perform the edge extraction filter. (summarize)
	<div>UserVarPtr info</div> Return values: M_DEFAULT; M_KERNEL; M_RECURSIVE; (details)
M_FILTER_SMOOTHNESS +	Returns the degree of smoothness (strength of denoising) of the edge extraction filter. (summarize)
	<div>UserVarPtr info</div> Return values: 0.0 to 100.0; M_DEFAULT; (details)

M_FILTER_TYPE +	<p>Returns the type of filter used when performing the edge extraction. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DERICHE; M_FREI_CHEN; M_PREWITT; M_SHEN; M_SOBEL; (details)</p>
M_FLOAT_MODE +	<p>Returns the mode in which to perform the edge extraction filter. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>
M_KERNEL_DEPTH +	<p>Returns the depth of the convolution kernel used for kernel filtering mode. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 8; 16; M_DEFAULT; (details)</p>
M_KERNEL_WIDTH +	<p>Returns the maximum size (same in X and Y) of the convolution kernel set for kernel filtering mode. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_AUTO; M_DEFAULT; Value >= 3; (details)</p>
M_MAGNITUDE_TYPE +	<p>Returns the type of magnitude value used to calculate the magnitude of the edge at each edgel position. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_NORM; M_SQR_NORM; (details)</p>
M_REGION_ANGLE +	<p>Returns the region's angle. This inquire type is applicable to rectangle and infinite regions. See MmetSetRegion() with M_RECTANGLE and M_INFINITE for more information. (summarize)</p>
M_REGION_ANGLE_END +	<p>Returns the region's ending angle. This inquire type is used for M_RING_SECTOR and M_ARC regions. (summarize)</p>
M_REGION_ANGLE_START +	<p>Returns the region's starting angle. This inquire type is used for M_RING_SECTOR and M_ARC regions. (summarize)</p>
M_REGION_END_X +	<p>Returns the X-coordinate of the region's ending point. This inquire type is used for M_SEGMENT regions. (summarize)</p>
M_REGION_END_Y +	<p>Returns the Y-coordinate of the region's ending point. This inquire type is used for M_SEGMENT regions. (summarize)</p>
M_REGION_GEOMETRY +	<p>Returns the region's geometry. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_ARC; M_INFINITE; M_INTERACTIVE; M_RECTANGLE; M_RING; M_RING_SECTOR; M_SEGMENT;</p>
M_REGION_HEIGHT +	<p>Returns the region's height. This inquire type is used for M_RECTANGLE regions. (summarize)</p>
M_REGION_POSITION_X +	<p>Returns the X-coordinate of the region's origin. This inquire type is used for all regions except M_SEGMENT. (summarize)</p>
M_REGION_POSITION_Y +	<p>Returns the Y-coordinate of the region's origin. This inquire type is used for all regions except M_SEGMENT. (summarize)</p>
M_REGION_RADIUS +	<p>Returns the region's radius. This inquire type is used for M_ARC regions. (summarize)</p>
M_REGION_RADIUS_END +	<p>Returns the ending point of the region's radius. This inquire type is used for M_RING and M_RING_SECTOR regions. (summarize)</p>
M_REGION_RADIUS_START +	<p>Returns the starting point of the region's radius. This inquire type is used for M_RING and M_RING_SECTOR regions. (summarize)</p>

<input type="checkbox"/> M_REGION_START_X +	Returns the X-coordinate of the region's starting point. This inquire type is used for M_SEGMENT regions. (summarize)
<input type="checkbox"/> M_REGION_START_Y +	Returns the Y-coordinate of the region's starting point. This inquire type is used for M_SEGMENT regions. (summarize)
<input type="checkbox"/> M_REGION_WIDTH +	Returns the region's width. This inquire type is used for M_RECTANGLE regions. (summarize)
<input type="checkbox"/> M_THRESHOLD_MODE +	Returns the threshold mode of the edge extraction. Note that lower threshold values result in a more sensitive edgel detection. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_HIGH; M_LOW; M_MEDIUM; M_USER_DEFINED; M_VERY_HIGH; (details)
<input type="checkbox"/> M_THRESHOLD_TYPE +	Returns the type of the hysteresis threshold used when performing the edge extraction. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_FULL_HYSTERESIS; M_HYSTERESIS; M_NO_HYSTERESIS; (details)
<input type="checkbox"/> M_THRESHOLD_VALUE_HIGH +	Returns the user-defined upper bound of the hysteresis threshold. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_THRESHOLD_VALUE_LOW +	Returns the user-defined lower bound of the hysteresis threshold. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)

To inquire about the feature settings for a metrology context, set this parameter to one of the following values. For inquiring feature settings, the **LabelOrIndex** parameter can be set to one of the following: an existing feature label or index using [M_FEATURE_LABEL\(\)](#) or [M_FEATURE_INDEX\(\)](#) or [M_GLOBAL_FRAME](#).

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For inquiring feature settings	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ANGLE +	Returns the angle value used to construct a feature, when applicable. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_ANGLE_END +	Returns the end angle of a constructed arc. The angle is relative to the reference frame's axis. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 360.0; M_DEFAULT; (details)
<input type="checkbox"/> M_ANGLE_START +	Returns the start angle of a constructed arc. The angle is relative to the reference frame's axis. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 360.0; M_DEFAULT; (details)
<input type="checkbox"/> M_EDGEL_ANGLE_RANGE +	Returns the angular range used to select the active edgels. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 360.0; M_DEFAULT; (details)
<input type="checkbox"/> M_EDGEL_RELATIVE_ANGLE +	Returns the relative angle used to select the active edgels.

	(summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: M_DEFAULT; M_REVERSE; M_SAME; M_SAME_OR_REVERSE; (details)</div> </div>
<input type="checkbox"/> M_EDGEL_TYPE +	<div> <div>Returns the type of edgels that will be used for constructed edgel features.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: M_ACTIVE_EDGELS; M_ALL_EDGELS; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_FEATURE_GEOMETRY +	<div> <div>Returns the geometry of the specified feature. Features are added with MmetAddFeature().</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values:</div> <div> <div>M_ARC</div> <div>Arc feature (measured or constructed).</div> </div> <div> <div>M_CIRCLE</div> <div>Circle feature (measured or constructed).</div> </div> <div> <div>M_EDGEL</div> <div>Edgel feature (measured or constructed).</div> </div> <div> <div>M_LINE</div> <div>Line feature (only constructed).</div> </div> <div> <div>M_LOCAL_FRAME</div> <div>Local frame feature (only constructed).</div> </div> <div> <div>M_POINT</div> <div>Point feature (measured or constructed).</div> </div> <div> <div>M_SEGMENT</div> <div>Segment feature (measured or constructed).</div> </div> </div>
<input type="checkbox"/> M_FEATURE_TYPE +	<div> <div>Returns the type of the specified feature.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: M_CONSTRUCTED; M_MEASURED;</div> </div>
<input type="checkbox"/> M_FIT_COVERAGE_MIN +	<div> <div>Returns the minimum portion, in percent, of the feature that must cover the active edgels for a successful fit.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 100.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_FIT_DISTANCE_MAX +	<div> <div>Returns the maximum distance, in pixels, to determine the active edgels used in the fit calculation to compute a feature.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: M_DEFAULT; M_INFINITE; Value; (details)</div> </div>
<input type="checkbox"/> M_FIT_ITERATIONS_MAX +	<div> <div>Returns the maximum number of fit iterations used to compute the feature.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: M_AUTO; M_DEFAULT; Value >= 1; (details)</div> </div>
<input type="checkbox"/> M_FIT_VARIATION_MAX +	<div> <div>Returns the maximum variation of the feature parameter allowed from one iteration to another.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 100.0; M_AUTO; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_LINE_A +	<div> <div>Returns the A parameter of the equation for a parametrically constructed line.</div> <div>The line equation is: $A x + B y + C = 0$.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: M_DEFAULT; Value; (details)</div> </div>
<input type="checkbox"/> M_LINE_B +	<div> <div>Returns the B parameter of the equation for a parametrically constructed line.</div> <div>The line equation is: $A x + B y + C = 0$.</div> <div>(summarize)</div> </div>

	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_LINE_C +	Returns the C parameter of the equation for a parametrically constructed line. The line equation is: $A x + B y + C = 0$. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_NUMBER_MAX +	Returns the maximum number of subfeatures to calculate in a multiple feature (for point features only). (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_NUMBER_MIN +	Returns the minimum number of subfeatures to calculate in a multiple feature (for point features only). (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_POSITION_END_X +	Returns the X-coordinate of the ending point of a parametrically constructed segment. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_POSITION_END_Y +	Returns the Y-coordinate of the ending point of a parametrically constructed segment. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_POSITION_START_X +	Returns the X-coordinate of the starting point of a parametrically constructed segment. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_POSITION_START_Y +	Returns the Y-coordinate of the starting point of a parametrically constructed segment. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_POSITION_X +	Returns the X-coordinate of the center of a constructed circle or arc, or the X-coordinate position of a constructed local frame or point. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_POSITION_Y +	Returns the Y-coordinate of the center of a constructed circle or arc, or the Y-coordinate position of a constructed local frame or point. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_RADIUS +	Returns the radius of a constructed circle or arc. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_REFERENCE_FRAME +	Returns the reference frame for any feature. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)

To inquire about the clone transformation settings used for a cloned feature in a metrology result buffer, set this parameter to one of the following values. For inquiring clone transformations, the **LabelOrIndex** parameter must be set to an existing feature label or index using [M_FEATURE_LABEL\(\)](#) or [M_FEATURE_INDEX\(\)](#).

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For inquiring clone transformations settings	
☐ Value	Description
☐ M_CLONE_ANGLE +	Returns the rotation angle of the cloned feature, in degrees and in the counterclockwise direction. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 360.0; M_DEFAULT; (details)
☐ M_CLONE_OFFSET_X +	Returns the translation value of the cloned feature, in the X-direction, in pixels. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
☐ M_CLONE_OFFSET_Y +	Returns the translation value of the cloned feature, in the Y-direction, in pixels. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
☐ M_CLONE_SCALE +	Returns the scale factor of the cloned feature. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0.0; (details)

To inquire about the geometric tolerance settings of a metrology context, set this parameter to one of the following values. For inquiring geometric tolerances, the **LabelOrIndex** parameter can be set to an existing tolerance label or index using [M_TOLERANCE_LABEL\(\)](#) or [M_TOLERANCE_INDEX\(\)](#).

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For inquiring geometric tolerance settings	
☐ Value	Description
☐ M_CONSTRUCTION_FEATURE_LABELn +	Returns the label value of the <i>n</i> th feature used to build the constructed feature, where <i>n</i> is a value between 0 and the total number of features used in the build operation - 1. For example, if three points are used to build a constructed arc, you can inquire the label of the first point using M_CONSTRUCTION_FEATURE_LABELn , where <i>n</i> = 0. (summarize)
	<i>UserVarPtr info</i> Return values: Please see the FeatureLabelArrayPtr parameter of MmetAddFeature() . (details) Please see the SubfeatureIndexArrayPtr parameter of MmetAddFeature() . (details)
☐ M_TOLERANCE_TYPE +	Return the type of the geometric tolerance. (summarize)
	<i>UserVarPtr info</i> Return values: M_ANGULARITY; M_CONCENTRICITY; M_DISTANCE_MAX; M_DISTANCE_MIN; M_LENGTH; M_PARALLELISM; M_PERPENDICULARITY; M_POSITION_X; M_POSITION_Y; M_RADIUS; M_ROUNDNESS; M_STRAIGHTNESS; (details)
☐ M_VALUE_MAX +	Returns the maximum possible value for a geometric tolerance. If a geometric tolerance value exceeds this maximum threshold, the calculation operation will fail and its status will be M_FAIL . The status of the calculation of a tolerance is found using MmetGetResult() with M_STATUS . (summarize)
	<i>UserVarPtr info</i>

	<div>Return values: M_DEFAULT; Value; (details)</div>
<div><div></div><div>M_VALUE_MIN +</div></div>	<div>Returns the minimum required value for a geometric tolerance. If a geometric tolerance value does not meet this minimum requirement, the calculation operation will fail and its status will be M_FAIL. The status of the calculation of a tolerance is found using MmetGetResult() with M_STATUS. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values: M_DEFAULT; Value; (details)</div></div>
<div><div></div><div>M_VALUE_WARNING_MAX +</div></div>	<div>Returns the maximum warning value for a geometric tolerance. Geometric tolerance values that are greater or equal to M_VALUE_MIN and lower than M_VALUE_WARNING_MIN will result in a warning. Geometric tolerance values that are lower or equal to M_VALUE_MAX and greater than M_VALUE_WARNING_MAX will also result in a warning. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values: M_DEFAULT; Value; (details)</div></div>
<div><div></div><div>M_VALUE_WARNING_MIN +</div></div>	<div>Returns the minimum warning value for a geometric tolerance. Geometric tolerance values that are greater or equal to M_VALUE_MIN and lower than M_VALUE_WARNING_MIN will result in a warning. Geometric tolerance values that are lower or equal to M_VALUE_MAX and greater than M_VALUE_WARNING_MAX will also result in a warning. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values: M_DEFAULT; Value; (details)</div></div>

To inquire the label of a feature or geometric tolerance, set this parameter to the following value.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

For inquiring the label of a feature or geometric tolerance	
Value	Description
M_LABEL_VALUE +	<p>Returns the label value of a specified feature or tolerance. For inquiring the feature's label, the LabelOrIndex parameter can be set to M_FEATURE_INDEX() or M_GLOBAL_FRAME. For inquiring the tolerance's label, the LabelOrIndex parameter can be set to M_TOLERANCE_INDEX(). (summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <div> <div>Value</div> <div>Label value of the specified feature or tolerance.</div> </div> </div>

To inquire about the results calculated for a metrology result buffer, set this parameter to one of the following values. For inquiring result buffer settings, the **LabelOrIndex** parameter must be set to [M_GENERAL](#).

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

For inquiring result buffer settings	
Value	Description
M_DRAW_RELATIVE_ORIGIN_X +	<p>Returns the X-coordinate of the top left corner of the region in the result buffer that will be used when drawing in the destination image (MmetDraw()). This setting is not saved with the context. If you restore the context, it must be set again.</p> <p>(summarize)</p> <div> <div></div> <div> <i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details) </div> </div>
M_DRAW_RELATIVE_ORIGIN_Y +	<p>Returns the Y-coordinate of the top left corner of the region in the result buffer that will be used when drawing in the destination image (MmetDraw()). This setting is not saved with the context. If you restore the context, it must be set again.</p> <p>(summarize)</p> <div> <div></div> <div> <i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details) </div> </div>
M_DRAW_SCALE_X +	<p>Returns the scale in the X-direction when drawing results in the destination image buffer (MmetDraw()).</p>

	This setting is not saved with the context. If you restore the context, it must be set again. (summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: M_DEFAULT; Value > 0; (details)</div> </div>
<input type="checkbox"/> M_DRAW_SCALE_Y +	Returns the scale in the Y-direction when drawing results in the destination image buffer (MmetDraw()). This setting is not saved with the context. If you restore the context, it must be set again. (summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: M_DEFAULT; Value > 0; (details)</div> </div>

Combination constant for [the values listed in](#) all tables **except For opening an interactive dialog box**

You can add the following value to the above-mentioned values to determine the default value of an inquire type.

● For inquiring the default value of an inquire type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Returns the default value of the specified inquire type. (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> </div>

Combination constant for [the values listed in](#) all tables **except For opening an interactive dialog box**

You can add the following value to the above-mentioned values to determine whether a result is supported.

● For inquiring whether an inquire type is supported	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SUPPORTED	Returns whether the specified inquire type is supported for the metrology context. (summarize)
	<div> <div>UserVarPtr info</div> <div>Data type: MIL_DOUBLE</div> <div>Return values:</div> <div> <div>M_NULL</div> <div>Inquire type is not supported.</div> </div> <div> <div>Value != 0</div> <div>Inquire type is supported.</div> </div> </div>

Combination constants for [the values listed in](#) all tables **except For opening an interactive dialog box**

You can add one of the following values to the above-mentioned values to cast the requested information to a required data type.

● For specifying the data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . This is the default value. (summarize)
	<div> <div>UserVarPtr info</div> </div>

		Data type: double
<input type="checkbox"/> M_TYPE_LONG	Casts the requested information to a <i>long</i> . (summarize)	
		UserVarPtr info Data type: long
<input type="checkbox"/> M_TYPE_MIL_DOUBLE	Casts the requested information to a <i>MIL_DOUBLE</i> . (summarize)	
		UserVarPtr info Data type: MIL_DOUBLE
<input type="checkbox"/> M_TYPE_MIL_ID	Casts the requested information to a <i>MIL_ID</i> . Note that M_TYPE_MIL_ID should only be used with M_OWNER_SYSTEM . (summarize)	
		UserVarPtr info Data type: MIL_ID
<input type="checkbox"/> M_TYPE_MIL_INT	Casts the requested information to a <i>MIL_INT</i> . (summarize)	
		UserVarPtr info Data type: MIL_INT
<input type="checkbox"/> M_TYPE_MIL_INT32	Casts the requested information to a <i>MIL_INT32</i> . (summarize)	
		UserVarPtr info Data type: MIL_INT32
<input type="checkbox"/> M_TYPE_MIL_INT64	Casts the requested information to a <i>MIL_INT64</i> . (summarize)	
		UserVarPtr info Data type: MIL_INT64

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• double• long• M_NULL• MIL_DOUBLE• MIL_ID• MIL_INT• MIL_INT32• MIL_INT64

Specifies the address in which the inquiry result will be written.

Return value

The return value is the required information, cast to a *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetRestore

Synopsis

Restore a MIL metrology context from disk.

Syntax

```
MIL_ID MmetRestore(
    MIL_CONST_TEXT_PTR Filename,
    MIL_ID SystemId,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr
)
```

Description

This function restores a MIL metrology context that was previously saved to a file, using [MmetSave\(\)](#) or [MmetStream\(\)](#). This function restores all the metrology context's settings that were in effect when the metrology context was saved.

If you had associated a calibration object with the template reference of your metrology context, you must re-associate the calibration object, using [MmetControl\(\)](#) with [M_ASSOCIATED_CALIBRATION](#). Moreover, if you had previously set drawing control types, using [MmetControl\(\)](#) with [M_DRAW_...](#), you must reset them since they were not saved with the context.

Parameters

Filename

Specifies the name and path of the file from which to restore the metrology context. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:


● For specifying the file name and path		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile". (summarize)	
		Parameters
	FileName	Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.	

SystemId

Specifies the system on which to restore the metrology context.

This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.

 MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .
---------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------

ControlFlag

Reserved for future expansion. This must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the metrology context identifier. Since **MmetRestore()** also returns the metrology context identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the metrology context identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetSave

Synopsis

Save a metrology context to a file.

Syntax

```
void MmetSave(
    MIL_CONST_TEXT_PTR FileName,
    MIL_ID ContextId,
    MIL_INT ControlFlag
)
```

Description

This function saves all the information about the previously allocated metrology context to disk, including all of the feature and geometric tolerance settings. This information can be reloaded, using [MmetRestore\(\)](#) or [MmetStream\(\)](#). However, preprocessing, the template reference's associated calibration object (if any), and all drawing control type settings are not saved.

Parameters

FileName

Specifies the name and path of the file in which to save the metrology context. It is recommended that you use the MET file extension for easier use with other Matrox Imaging software products. The function internally handles the opening and closing of this file. If this file already exists, it will be overwritten.

This parameter can be set to one of the following values:

For specifying the file name and path	
Value	Description
<div>MIL_TEXT(MIL_TEXT_PTR FileName)</div>	<div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile". (summarize)</div> <div><div>Parameters</div><div>FileName</div><div>Specifies the drive, directory, and name of the file.</div></div>
M_INTERACTIVE	<div>[This is only applicable to Windows] Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</div>

ContextId

Specifies the identifier of the metrology context to save.

ControlFlag

Reserved for future use. This parameter must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetSetPosition

Synopsis

Set the position of a feature.

Syntax

```
void MmetSetPosition(
    MIL_ID ContextId,
    MIL_INT FeatureLabelOrIndex,
    MIL_INT TransformationType,
    MIL_DOUBLE Param1,
    MIL_DOUBLE Param2,
    MIL_DOUBLE Param3,
    MIL_DOUBLE Param4,
    MIL_INT ControlFlag
)
```

Description

This function moves the specified feature's position in the template of the specified metrology context.

This function is valid for measured features ([MmetAddFeature\(\)](#) with [M_MEASURED](#)) and parametrically constructed features ([MmetAddFeature\(\)](#) with [M_CONSTRUCTED](#) and [M_PARAMETRIC](#)), unless otherwise specified. Typically, this function is used to move the position of a parametrically constructed local frame. Note that it moves all the features defined and derived from this local frame. Features defined from a local frame have that local frame as their reference frame. Features derived from a local frame are those that are directly constructed from that local frame or from a feature that has that local frame as its reference frame.

The transformation used to set the feature's new position is relative to that feature's reference frame. If the target image is calibrated, values have to be defined in the world coordinate system.

Note that you can also position features using [MmetControl\(\)](#) with [M_POSITION_X](#) and [M_POSITION_Y](#). However, in this case you must enter exact values; you cannot, for example, position features based on the results of another module, as you can using [MmetSetPosition\(\)](#) with [M_RESULT](#).

By setting the [ControlFlag](#) parameter to [M_INTERACTIVE](#), you can move the feature's position interactively.

Parameters

ContextId

Specifies the identifier of the metrology context. The metrology context must have been previously allocated on the required system using [MmetAlloc\(\)](#).

FeatureLabelOrIndex

Specifies the label or index of the feature you want to position.

● For specifying a feature	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the label or index value, as automatically determined by MIL. M_DEFAULT can only be used if you are setting the ControlFlag parameter to M_INTERACTIVE . (summarize)
<input type="checkbox"/> M_FEATURE_LABEL(MIL_INT FeatureLabel)	Specifies the label value of an existing individual feature. (summarize)
	Parameters
	FeatureLabel

<div><div></div><div>M_FEATURE_INDEX(MIL_INT <i>FeatureIndex</i>)</div></div>	This parameter specifies the label of the feature to position. You can set this parameter to the following:	
	Value > 0	Specifies the label.
	Specifies the index value of an existing individual feature. (summarize)	
	<i>Parameters</i>	
	<i>FeatureIndex</i> This parameter specifies the index of the feature to position. You can set this parameter to the following:	
	Value >= 0	Specifies the index. Note that the index of the global frame is 0.

TransformationType

Specifies the type of transformation that will be used to set the new position of the feature.

You must set this parameter to **M_DEFAULT** if you are setting the [ControlFlag](#) parameter to **M_INTERACTIVE**.

See the [Parameter associations](#) section for possible values.

Param1

Specifies an attribute of the position to set. Its definition is dependent on the type of position chosen.

You must set this parameter to **M_DEFAULT** if you are setting the [ControlFlag](#) parameter to **M_INTERACTIVE**.

See the [Parameter associations](#) section for possible values.

Param2

Specifies an attribute of the position to set. Its definition is dependent on the type of position chosen.

You must set this parameter to **M_DEFAULT** if you are setting the [ControlFlag](#) parameter to **M_INTERACTIVE**.

See the [Parameter associations](#) section for possible values.

Param3

Specifies an attribute of the position to set. Its definition is dependent on the type of position chosen.

You must set this parameter to **M_DEFAULT** if you are setting the [ControlFlag](#) parameter to **M_INTERACTIVE**.

See the [Parameter associations](#) section for possible values.

Param4

Specifies an attribute of the position to set. Its definition is dependent on the type of position chosen.

You must set this parameter to **M_DEFAULT** if you are setting the [ControlFlag](#) parameter to **M_INTERACTIVE**.

See the [Parameter associations](#) section for possible values.

ControlFlag

Specifies whether to use this function with an interactive dialog box. This parameter must be set to one of the following values.

● For specifying whether this function is used with an interactive dialog box	
<div><div></div><div>Value</div></div>	Description

<input type="checkbox"/> M_DEFAULT	Specifies that an interactive dialog box is not used.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively move the feature's position.

Parameter associations

Possible values for the **TransformationType**, **Param1**, **Param2**, **Param3**, and **Param4** parameters are described in the table below.

- [For selecting the operation that will be used to position the feature](#)

Note that any unused parameters should be set to **M_DEFAULT**.

For selecting the operation that will be used to position the feature	
<input type="checkbox"/> TransformationType	Description
Param1	
Param2	
Param3	
Param4	
<input type="checkbox"/> M_FORWARD_COEFFICIENTS	Specifies to position the feature using an operation based on forward transformation coefficients, using the following equations: $x_d = ax_s + by_s + c$ $y_d = -bx_s + ay_s + d$ (summarize)
<input type="checkbox"/> Param1	Sets the forward transformation coefficient <i>A</i> . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value	Specifies the coefficient.
<input type="checkbox"/> Param2	Sets the forward transformation coefficient <i>B</i> . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value	Specifies the coefficient.
<input type="checkbox"/> Param3	Sets the forward transformation coefficient <i>C</i> . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value	Specifies the coefficient.
<input type="checkbox"/> Param4	Sets the forward transformation coefficient <i>D</i> . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value	Specifies the coefficient.
<input type="checkbox"/> M_GEOMETRIC	Specifies to position the feature using an operation based on a translation and rotation. (summarize)
<input type="checkbox"/> Param1	Sets the translation value for the feature, in the X-direction.

	(summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the distance to move the feature, in pixel or world units.
<input type="checkbox"/> Param2	Sets the translation value for the feature, in the Y-direction. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the distance to move the feature, in pixel or world units.
<input type="checkbox"/> Param3	Sets the angle of rotation for the feature. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_POSITION	Specifies to position the feature at a specific position and angle. This should be used only for a parametrically constructed local frame. (summarize)
<input type="checkbox"/> Param1	Sets the X-coordinate of the origin of the local frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the X-coordinate.
<input type="checkbox"/> Param2	Sets the Y-coordinate of the origin of the local frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the Y-coordinate.
<input type="checkbox"/> Param3	Sets the angle of the local frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_RESULT	Specifies to position the feature at the position calculated by another MIL module. Supported modules are Code, Geometric Model Finder, Pattern Matching, and String Reader. If you are using a Model Finder result, you can combine M_RESULT with M_FORWARD_COEFFICIENTS . In this case, the transformation uses the forward coefficients of Model Finder's result occurrence. (summarize)
<input type="checkbox"/> Param1	Sets the result buffer to use to specify the feature's position. (summarize)
<input type="checkbox"/> MIL result buffer identifier	Specifies a result buffer from the MIL Model Finder module.
<input type="checkbox"/> Param2	Sets the index value of the result occurrence to use to set the feature's position. (summarize)
<input type="checkbox"/> Value	Specifies the index value. This value must be a valid index that is contained within the result buffer you provided with Param1 . (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetSetRegion

Synopsis

Set the region from which to select data that will be used to add the measured feature.

Syntax

```
void MmetSetRegion(
    MIL_ID ContextId,
    MIL_INT FeatureLabelOrIndex,
    MIL_INT ReferenceFrameLabelOrIndex,
    MIL_INT Geometry,
    MIL_DOUBLE Param1,
    MIL_DOUBLE Param2,
    MIL_DOUBLE Param3,
    MIL_DOUBLE Param4,
    MIL_DOUBLE Param5,
    MIL_DOUBLE Param6
)
```

Description

This function allows you to set a geometrically-defined region (ROI) within the template of the metrology context. The data within this region of the target image will be used to add the specified measured feature. The region that you choose must be compatible with the type of feature specified. To define a measured feature and add its definition to the metrology template of the specified metrology context, use `MmetAddFeature()` with `M_MEASURED`.

Note that valid features must not only fall within the ROI, but their edgels' gradient angle must also fall along the region's orientation. Various control types can be used to set a valid relationship between the ROI's orientation, and the edgels' gradient angle. For more information, see the [Gradient angle](#) subsection in the [Degrees of freedom](#) section in [Chapter 15: Metrology](#).

By setting the **Geometry** parameter to `M_INTERACTIVE`, you can set the feature's region interactively.

If the target image is calibrated, values have to be defined in the world coordinate system.

Parameters

ContextId

Specifies the metrology context of the template in which to set the region. The metrology context must have been previously allocated on the required system using `MmetAlloc()`.

FeatureLabelOrIndex

Specifies the label or index of the feature that will be added from the region defined. Set this parameter to one of the following values:

For specifying a feature	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the label or index value, as automatically determined by MIL. M_DEFAULT can only be used if you are setting the Geometry parameter to <code>M_INTERACTIVE</code> . (summarize)
<input type="checkbox"/> M_FEATURE_LABEL(MIL_INT FeatureLabel)	Specifies the label value of an existing individual feature. (summarize)
	Parameters
	FeatureLabel This parameter specifies the label of the feature to add from the defined region. You can set this parameter to the following:

	Value > 0	Specifies the label.
<input type="checkbox"/> M_FEATURE_INDEX(MIL_INT <i>FeatureIndex</i>)	Specifies the index value of an existing individual feature. (summarize)	
	<i>Parameters</i>	
	<i>FeatureIndex</i> This parameter specifies the index of the feature to add from the defined region. You can set this parameter to the following:	
	Value >= 0	Specifies the index. Note that the index of the global frame is 0.

ReferenceFrameLabelOrIndex

Specifies the label or index of the reference frame feature to use to set the region. The feature index or label that you specify must refer to a reference frame.

Note that you can change the reference frame using `MmetControl()` with `M_REFERENCE_FRAME`.

● For specifying a reference frame feature		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_DEFAULT	Specifies the reference frame's label or index value, as automatically determined by MIL. M_DEFAULT can only be used if you are setting the Geometry parameter to M_INTERACTIVE . (summarize)	
<input type="checkbox"/> M_FEATURE_LABEL(MIL_INT <i>FeatureLabel</i>)	Specifies the label value of an existing reference frame feature. (summarize)	
	<i>Parameters</i>	
	<i>FeatureLabel</i> This parameter specifies the label of the reference frame feature to use to set the region. You can set this parameter to the following:	
	Value > 0	Specifies the label.
<input type="checkbox"/> M_FEATURE_INDEX(MIL_INT <i>FeatureIndex</i>)	Specifies the index value of an existing reference frame feature. (summarize)	
	<i>Parameters</i>	
	<i>FeatureIndex</i> This parameter specifies the index of the reference frame feature to use to set the region. You can set this parameter to the following:	
	Value >= 0	Specifies the index. Note that the index of the global frame is 0.

Geometry

Specifies the type of geometric region from which the feature will be added.

See the [Parameter associations](#) section for possible values.

Param1

Specifies an attribute of the region to set. Its definition is dependent on the region chosen.

See the [Parameter associations](#) section for possible values.

Param2

Specifies an attribute of the region to set. Its definition is dependent on the region chosen.

See the [Parameter associations](#) section for possible values.

Param3

Specifies an attribute of the region to set. Its definition is dependent on the region chosen.

See the [Parameter associations](#) section for possible values.

Param4

Specifies an attribute of the region to set. Its definition is dependent on the region chosen.

See the [Parameter associations](#) section for possible values.

Param5

Specifies an attribute of the region to set. Its definition is dependent on the region chosen.

See the [Parameter associations](#) section for possible values.

Param6

Specifies an attribute of the region to set. Its definition is dependent on the region chosen.


See the [Parameter associations](#) section for possible values.

Parameter associations



Possible values for the **Geometry**, **Param1**, **Param2**, **Param3**, **Param4**, **Param5**, and **Param6** parameters are described in the table below.


- [For selecting the geometry that will be used to set the region, or for using this function with an interactive dialog box](#)

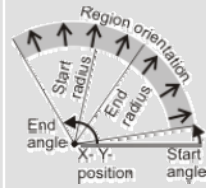
Unless otherwise specified, any unused parameters should be set to **M_NULL**.

● For selecting the geometry that will be used to set the region, or for using this function with an interactive dialog box	
<input type="checkbox"/> Geometry	Description
Param1	
Param2	
Param3	
Param4	
Param5	
Param6	
<input type="checkbox"/> M_ARC	<p>Specifies that the feature will be calculated with an arc region.</p>  <p>M_ARC can be used with point (M_POINT) features. (summarize)</p>
<input type="checkbox"/> Param1	<p>Sets the X-coordinate of the origin of the arc region, relative to the reference frame. (summarize)</p>

<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the origin's X-coordinate, in pixel or world units.
<input type="checkbox"/> Param2	Sets the Y-coordinate of the origin of the arc region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the origin's Y-coordinate, in pixel or world units.
<input type="checkbox"/> Param3	Sets the radius of the arc region. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the radius, in pixels.
<input type="checkbox"/> Param4	Sets the start angle of the arc region. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> Param5	Sets the end angle of the arc region. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 360.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_INFINITE	Specifies that the feature will be calculated with an infinite region. An infinite region is an X- Y-plane with no boundaries. Use Param3 to specify the orientation. Note that M_INFINITE is similar to M_RECTANGLE , however a rectangular region has a width and a height, which are used to bound the region, and also to, in part, set its orientation. M_INFINITE can be used with all features and is the default region. (summarize)
<input type="checkbox"/> Param1	Sets the X-coordinate of the origin of the infinite region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the origin's X-coordinate, in pixel or world units.
<input type="checkbox"/> Param2	Sets the Y-coordinate of the origin of the infinite region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the origin's Y-coordinate, in pixel or world units.
<input type="checkbox"/> Param3	Sets the angle of the infinite region, relative to the reference frame. By default the angle is the same as the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_RADIAL	Specifies a radial angle. In this case, the orientation is the full angular range (0.0° to 360.0°), which radiates out from the specified origin point to create the valid region. Typically, an infinite region with a radial angle is used to define a region for a circle feature whose radius is unknown. For circles, the origin of the region is typically placed at the circle's center. In general, all circles (or arcs) with this origin will be extracted. That is, active edgels must have a gradient angle that points in a direction that adheres to the contour of the circle formed by the origin.

	 <p>Infinite region (radial orientation) (summarize)</p> <p>Circle feature (all edges selected)</p> <p>Circle feature (some edges selected)</p>
<input type="checkbox"/> M_INTERACTIVE	<p>[This is only applicable to Windows]</p> <p>Opens a dialog box that allows you to set the feature's region interactively. (summarize)</p>
<input type="checkbox"/> Param1	<p>Sets an attribute of the region interactively. When using M_INTERACTIVE, Param1 to Param6 must be set to M_DEFAULT. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Implements the default behavior.</p>
<input type="checkbox"/> M_RECTANGLE	<p>Specifies that the feature will be calculated with a rectangular region.</p>  <p>Note that M_RECTANGLE is similar to M_INFINITE, however an infinite region does not have a width and a height (boundaries).</p> <p>M_RECTANGLE can be used with segment (M_SEGMENT) or edge (M_EDGE) features. (summarize)</p>
<input type="checkbox"/> Param1	<p>Sets the X-coordinate of the origin of the rectangular region, relative to the reference frame. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 0.0. (summarize)</p>
<input type="checkbox"/> Value	<p>Specifies the origin's X-coordinate, in pixel or world units.</p>
<input type="checkbox"/> Param2	<p>Sets the Y-coordinate of the origin of the rectangular region, relative to the reference frame. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 0.0. (summarize)</p>
<input type="checkbox"/> Value	<p>Specifies the origin's Y-coordinate, in pixel or world units.</p>
<input type="checkbox"/> Param3	<p>Sets the width of the rectangle. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 0.0 pixels. (summarize)</p>
<input type="checkbox"/> M_INFINITE	<p>Specifies an infinite width.</p>
<input type="checkbox"/> Value	<p>Specifies the rectangle's width, in pixels.</p>
<input type="checkbox"/> Param4	<p>Sets the height of the rectangle. (summarize)</p>

<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> M_INFINITE	Specifies an infinite height.
<input type="checkbox"/> Value	Specifies the rectangle's height, in pixels.
<input type="checkbox"/> Param5	Sets the angle of the rectangular region. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_RING	<p>Specifies that the feature will be calculated with a ring-shaped region.</p>  <p>M_RING can be used with circle (M_CIRCLE) or edgel (M_EDGEL) features. (summarize)</p>
<input type="checkbox"/> Param1	Sets the X-coordinate of the center of the ring-shaped region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the origin's X-coordinate, in pixel or world units.
<input type="checkbox"/> Param2	Sets the Y-coordinate of the center of the ring-shaped region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the origin's Y-coordinate, in pixel or world units.
<input type="checkbox"/> Param3	Sets the starting radius of the ring-shaped region. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the starting radius, in pixels.
<input type="checkbox"/> Param4	Sets the ending radius of the ring-shaped region. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the ending radius, in pixels.
<input type="checkbox"/> M_RING_SECTOR	Specifies that the feature will be calculated with a ring-sector region.



M_RING_SECTOR can be used with radial segments (**M_SEGMENT**), arcs (**M_ARC**), or edgel (**M_EDGEL**) features.
[\(summarize\)](#)

<input type="checkbox"/> Param1	Sets the X-coordinate of the origin of the ring-sector region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the origin's X-coordinate, in pixel or world units.
<input type="checkbox"/> Param2	Sets the Y-coordinate of the origin of the ring-sector region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the origin's Y-coordinate, in pixel or world units.
<input type="checkbox"/> Param3	Sets the starting radius of the ring-sector region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the starting radius, in pixels.
<input type="checkbox"/> Param4	Sets the ending radius of the ring-sector region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the ending radius, in pixels.
<input type="checkbox"/> Param5	Sets the starting angle of the ring-sector region. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> Param6	Sets the ending angle of the ring-sector region. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 360.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_SEGMENT	Specifies that the feature will be calculated with a linear segment region.

	
<p><code>M_SEGMENT</code> can be used with point (<code>M_POINT</code>) features. (summarize)</p>	
<input type="checkbox"/> Param1	Sets the X-coordinate of the starting point of the segment region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the starting point's X-coordinate, in pixel or world units.
<input type="checkbox"/> Param2	Sets the Y-coordinate of the starting point of the segment region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the starting point's Y-coordinate, in pixel or world units.
<input type="checkbox"/> Param3	Sets the X-coordinate of the ending point of the segment region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the ending point's X-coordinate, in pixel or world units.
<input type="checkbox"/> Param4	Sets the Y-coordinate of the ending point of the segment region, relative to the reference frame. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value	Specifies the ending point's Y-coordinate, in pixel or world units.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

MmetStream

Synopsis

Load, restore, or save a MIL metrology context from/to a file or memory stream.

Syntax

```
void MmetStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *ObjectIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load or restore a MIL metrology context from a file or memory stream. All of the metrology context's settings that were in effect when the metrology context was saved will be restored. If you had associated a calibration object with a template reference in your metrology context, you must re-associate the calibration object, using [MmetControl\(\)](#) with [M_ASSOCIATED_CALIBRATION](#). Also, if you had previously set drawing control types, using [MmetControl\(\)](#) with [M_DRAW_...](#), you must reset them since they were not saved with the context.

This function can also save a metrology context to a file or memory stream. All information about the previously allocated metrology context is saved, including all of the feature settings and geometric tolerance settings. However, any associated calibration objects and drawing control type settings are not saved.

To inquire the number of bytes necessary to save a metrology context, you should call this function ([MmetStream\(\)](#)) first with the **Operation** parameter set to [M_INQUIRE_SIZE_BYTE](#).

The content saved to memory stream is equivalent to the content saved to file. In addition, any file saved with [MmetSave\(\)](#) is equivalent to a file saved using this function.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using [MmetStream\(\)](#), you can choose to save a backwards-compatible version of a MIL metrology context, which will work using a version of MIL that is up to one major release older than the current version (depending on which version is specified). For example, if you allocate a metrology context using MIL 9.0 and save it to version 8.0, you can restore this context on a computer where MIL 8.0 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore MIL metrology context saved using MIL version 8.0 or above. Settings that do not exist in the lower version will be filled with default values when the metrology context is loaded or restored.

Parameters

MemPtrOrFileName

Specifies the file or memory stream.

● For specifying the name and path of a file, memory stream or opening an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving a metrology context to a file, use the MET file extension. The function internally handles the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p> <p>To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p>

		<i>Parameters</i>
		<i>FileName</i>
		Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE .	
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. The parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)	
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (MmetStream()) first with the Operation parameter set to M_INQUIRE_SIZE_BYTE . (summarize)	

SystemId

Specifies the system on which to restore the metrology context. For **M_INQUIRE_SIZE_BYTE**, **M_LOAD**, and **M_SAVE**, **SystemId** is ignored and should be set to **M_NULL**.

For an **M_RESTORE** operation, this parameter should be set to one of the following values:

● For M_RESTORE	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform. This parameter must be set to one of the following values:

● For specifying the operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save a metrology context. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated metrology context.
<input type="checkbox"/> M_RESTORE	Restores a metrology context from a file or memory stream and assigns it an identifier.
<input type="checkbox"/> M_SAVE	Saves a metrology context to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the metrology context.

This parameter must be set to one of the following values:

● For the type of stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream.
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)

Version

Specifies the MIL version of the metrology context. This parameter must be set to one of the following values.

● For specifying the MIL version	
Value	Description
M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL. For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Specifies the function's control flag. This should be set to **M_DEFAULT**.

ObjectIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the metrology context.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, **ObjectIdPtr** specifies the address of the variable from which to read the metrology context identifier.

For an [M_LOAD](#) operation, **ObjectIdPtr** specifies the address of the variable from which to read the identifier of the metrology context where the file or memory stream content will be loaded.

For an [M_RESTORE](#) operation, **ObjectIdPtr** specifies the address of the variable in which to return the identifier of the restored metrology context. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the metrology context, in bytes.

If the size is not required, you can set this parameter to **M_NULL**.

Note that the size of the metrology context will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmetrol.lib.
DLL	Requires mil.dll; milmetrol.dll.

Mmod functions

Synopsis

The functions prefixed with Mmod make up the Geometric Model Finder module. The Geometric Model Finder module is a set of functions which find occurrences of models, or sets or image characteristics, in target images or Edge Finder result buffers. The Model Finder module uses a geometric search method to help solve machine vision problems, such as alignment, measurement, and inspection. The Model Finder control also provides complete support for calibration. Searches are performed in the calibrated real-world such that, even without physically correcting your images, complex distortions do not affect performance, and results can be returned in real-world units.

Functions

- [MmodAlloc](#)
- [MmodAllocResult](#)
- [MmodControl](#)
- [MmodDefine](#)
- [MmodDefineFromFile](#)
- [MmodDraw](#)
- [MmodFind](#)
- [MmodFree](#)
- [MmodGetResult](#)
- [MmodInquire](#)
- [MmodMask](#)
- [MmodPreprocess](#)
- [MmodRestore](#)
- [MmodSave](#)
- [MmodStream](#)

MmodAlloc

Synopsis

Allocate a Model Finder context.

Syntax

```
MIL_ID MmodAlloc(  
    MIL_ID SystemId,  
    MIL_INT ModelFinderType,  
    MIL_INT ControlFlag,  
    MIL_ID *ContextIdPtr  
)
```

Description

This function allocates a Model Finder context on the specified system. A Model Finder context contains all the information necessary to perform an [MmodFind\(\)](#) search, including global search settings, and the individual model(s) to locate. When the Model Finder context is no longer required, you should release its memory, using [MmodFree\(\)](#).

Models can be defined and added to a Model Finder context using [MmodDefine\(\)](#).

The Model Finder context and individual model search settings can be adjusted using [MmodControl\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the context. This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ModelFinderType

Sets the type of Model Finder context. This parameter must be set to one of the following values:

● For the type of context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_GEOMETRIC	Specifies that the Model Finder context uses a general geometric search algorithm. This algorithm uses geometric features to locate user-specified model(s). (summarize)
<input type="checkbox"/> M_GEOMETRIC_CONTROLLED	Specifies that the Model Finder context uses a controlled geometric search algorithm. This search method is recommended when there are only small differences in scale between the occurrence and the nominal scale of the model (M_SCALE). When there is a lot of geometric complexity, this search method is often faster and more robust than M_GEOMETRIC . The search method is especially fast when the angle between the occurrence and the nominal angle of the model (M_ANGLE) is small. You can also set M_SEARCH_ANGLE_RANGE to M_ENABLE . When using this type of Model Finder context, M_SEARCH_SCALE_RANGE is disabled by default and cannot be set to M_ENABLE . As well, you cannot get or draw the target edges for the entire target (M_GENERAL); you can only get or draw them in the region of an occurrence. (summarize)

ControlFlag

Specifies the function's control flag. This parameter must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the Model Finder context identifier. Since the **MmodAlloc()** function also returns the Model Finder context identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the Model Finder context identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodAllocResult

Synopsis

Allocate a Model Finder result buffer.

Syntax

```
MIL_ID MmodAllocResult(  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *ModResultIdPtr  
)
```

Description

This function allocates a result buffer, on the specified system, to store results obtained from a [MmodFind\(\)](#) operation. When the result buffer is no longer required, release its memory, using [MmodFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the result buffer.

This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ModResultIdPtr

Specifies the address of the variable in which to write the Model Finder result buffer identifier. Since the **MmodAllocResult()** function also returns the model result buffer identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodControl

Synopsis

Control a Model Finder context, individual model setting, or Model Finder result buffer.

Syntax

```
void MmodControl(
    MIL_ID ContextOrResultId,
    MIL_INT Index,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function sets the specified control for either a Model Finder context itself, for one (or all) of the models contained therein, or for a Model Finder result buffer. These settings control the execution of [MmodFind\(\)](#) operations. Most of the control type settings can be inquired using [MmodInquire\(\)](#).

Changing certain control type settings might require preprocessing of the Model Finder context again, using [MmodPreprocess\(\)](#). To know if the Model Finder context needs to be preprocessed, call [MmodInquire\(\)](#) with the **M_PREPROCESSED** inquire type.

You can automatically define an image-type model that is based on specified control settings using this function with [M_AUTO_DEFINE](#). To have specified settings taken into account during model definition, you must allocate an empty image-type model using [MmodDefine\(\)](#) with [M_AUTO_DEFINE](#), set the appropriate control types using [MmodControl\(\)](#), and then call [MmodControl\(\)](#) with [M_AUTO_DEFINE](#). [M_AUTO_DEFINE](#) defines a unique model by searching the model source image for unique geometric features that match the settings specified using [MmodControl\(\)](#). When the most suitable geometric features are found, the function automatically defines a model from those features. If none are found, no model is allocated and an error is reported. It can take several seconds to find the best model (or more for large or small images). To be useful, the model source image should be a typical target image; otherwise, the selected area for the model might never be present in the target image. For more information on defining an image-type model automatically, see the [Image-type models](#) subsection in the [Defining and adding models to your Model Finder context](#) section in [Chapter 8: Geometric Model Finder](#).

By setting the [ControlType](#) parameter to [M_INTERACTIVE](#), you can set the control types interactively.

Parameters

ContextOrResultId

Specifies the Model Finder context or Model Finder result buffer whose settings you want to modify. The Model Finder context or result buffer must have been previously allocated on the system using [MmodAlloc\(\)](#) or [MmodAllocResult\(\)](#), respectively.

Index

Specifies that the Model Finder context, an individual model, or a Model Finder result buffer is controlled. Set this parameter to one of the following values:

● For specifying a context, model, or result buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default. If a Model Finder context is specified, same as M_ALL . If a Model Finder result buffer is specified, same as M_GENERAL . (summarize)
<input type="checkbox"/> M_ALL	Applies the specified control setting to all models, if a Model Finder context is specified.
<input type="checkbox"/> M_CONTEXT	Controls a general setting of a specified Model Finder context, if one is specified.
<input type="checkbox"/> M_GENERAL	Controls a general setting of a Model Finder result buffer when ContextOrResultId is a result buffer.

<input type="checkbox"/> Value	Specifies the index of the individual model to control, if a Model Finder context is specified. User labels cannot be used as indices; to retrieve the index of a model from its user label, use MmodInquire() with M_INDEX_FROM_LABEL . (summarize)
--------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ControlType

Specifies the setting to change.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the setting's new value.

All values are set in pixels, regardless of whether the model is calibrated.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For opening an interactive dialog box](#)
- [For the context and models](#)
- [For the context](#)
- [For one or all models](#)
- [For one or all models or a result](#)
- [For a result](#)

The following [ControlType](#) and corresponding [ControlValue](#) settings can be specified for both the Model Finder context ([M_CONTEXT](#)) and each individual model in the Model Finder context:

For opening an interactive dialog box	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_INTERACTIVE	<p>[This is only applicable to Windows]</p> <p>Opens a dialog box that allows you to edit the control types interactively.</p> <p>For M_CONTEXT, this setting opens a dialog box that allows you to edit all the control types of the specified Model Finder context interactively.</p> <p>For a model, this setting opens a dialog box that allows you to edit the model's control types interactively. The model is specified by the Index parameter. Once the dialog box is opened, you can change between models interactively using the Model Index field.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.

The following [ControlType](#) and corresponding [ControlValue](#) settings can be specified for both the Model Finder context ([M_CONTEXT](#)) and each individual model in the Model Finder context:

For the context and models	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_NUMBER	<p>Sets the number of occurrences for which to search.</p> <p>For M_CONTEXT, this control type sets the maximum number of all model occurrences for which to search in the target.</p> <p>For a model, this control type sets the maximum number of occurrences of the specified model, for which to search in the target.</p>

	<p>Once the number of occurrences for the context, with scores and target scores greater than or equal to their certainty levels, has been reached, the search will stop, regardless of whether the actual number set for any individual model has been found.</p> <p>To further illustrate the relationship between the M_NUMBER setting for the context and that of the individual models, see the Expected number of occurrences subsection in the Customizing search settings section in Chapter 8: Geometric Model Finder. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value.</p> <p>For M_CONTEXT, the default is M_ALL.</p> <p>For a model, the default value is 1. (summarize)</p>
<input type="checkbox"/> M_ALL	<p>Specifies to find all occurrences.</p> <p>For M_CONTEXT, M_ALL finds all the occurrences specified for each model in the context.</p> <p>For a model, M_ALL finds all the occurrences of the specified model.</p> <p>Note that this setting can increase the search time; always set M_NUMBER to a specific number whenever possible. (summarize)</p>
<input type="checkbox"/> Value	<p>Specifies the number of occurrences.</p> <p>For M_CONTEXT, this value specifies the number of occurrences, for all models within the context, to find in the target.</p> <p>For a model, this value specifies the maximum number of occurrences of the specified model to find in the target. (summarize)</p>

The possible **ControlType** and corresponding **ControlValue** settings for **M_CONTEXT** are:

● For the context																											
<input type="checkbox"/> ControlType	Description																										
ControlValue		corona-II (a)	cronosplus (b)	glige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ec/Xcl (f)	helios ed/Xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ec/Xcl (r)	odyssey ed/Xd (s)	solios ea/Xa (t)	solios ec/Xcl (u)	solios glige (v)	vio (w)			
<input type="checkbox"/> M_ACCURACY	<p>Sets the accuracy with which to find occurrences. The precision achieved is dependent on the quality of the model and the target. In addition, positional accuracy is slightly affected by the setting of the M_SPEED.</p> <p>Note that lower accuracy can speed up the MmodFind() operation. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_DEFAULT	Same as M_MEDIUM .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_HIGH	Sets the accuracy to high.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_MEDIUM	Sets the accuracy to medium.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
<input type="checkbox"/> M_ASPECT_RATIO	<p>Sets the nominal search aspect ratio for the target. When the target is not calibrated, this control type allows you to quickly compensate for aspect ratio distortion in the target, typically a side effect of the sampling rate used by the frame grabber. When the target is calibrated, this control type is ignored.</p> <p>This control type is mainly useful when dealing with synthetic models since it does not compensate for aspect ratio distortion in the model.</p> <p>When performing the search, the nominal search aspect ratio is taken into account horizontally, adjusting the target pixel width to equal the pixel height. As such, results will be returned for this corrected target coordinate system. However, the</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

[illegible]

	(summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_DEFAULT</div>	Same as M_AUTO .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_AUTO</div>	Determines the first resolution automatically.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>0 to 7</div>	Specifies the resolution level. Only integer values are accepted. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_KERNEL_DEPTH</div>	Sets the depth of the convolution kernel used for the kernel filtering mode. M_KERNEL_DEPTH is ignored unless MmodControl() with M_FILTER_MODE is set to M_KERNEL . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_DEFAULT</div>	Specifies the default value. The default value is 8 bit. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>8</div>	Specifies an 8-bit kernel depth. This is faster, though less accurate, than using a 16-bit kernel depth. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>16</div>	Specifies a 16-bit kernel depth. This is slower, though more accurate, than using an 8-bit kernel depth. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_KERNEL_WIDTH</div>	Sets the maximum X and Y size of the convolution kernel used for kernel filtering mode. The size of the kernel can be constrained by the available hardware resources. Larger kernels result in longer processing times. M_KERNEL_WIDTH is ignored unless MmodControl() with M_FILTER_MODE is set to M_KERNEL . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_DEFAULT</div>	Same as M_AUTO .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_AUTO</div>	Specifies the maximum X and Y size of the convolution kernel automatically, for a given smoothness factor. You can set the smoothness factor using MmodControl() with M_SMOOTHNESS . The size of the kernel, for a given smoothness factor, is calculated as the size beyond which the quantified filter values are zero. If the smoothness factor calls for a kernel size that is beyond the limits of your hardware, the smoothness will be sacrificed to achieve the largest possible kernel size. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>Value >= 3</div>	Specifies the maximum X and Y size of the convolution kernel directly. Only odd integer values are accepted. If your hardware cannot handle the kernel size, the effective smoothness factor will be saturated to the maximum possible value for the specified kernel size. You can set the smoothness factor using MmodControl() with M_SMOOTHNESS . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_LAST_LEVEL</div>	Sets the resolution level for the final stage (the highest resolution) of the search. Level 0 is the original target and each higher level is half the size (and resolution) of the previous one. If the specified level is not supported by the search algorithm, the highest possible level will be used. A higher last level speeds up the initial search but might make it less reliable and less accurate because the model might not retain enough distinctive features at a low resolution. M_LAST_LEVEL is for advanced users of the Model Finder module. The default setting usually provides the best results for the search operation. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_DEFAULT</div>	Same as M_AUTO .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

<input type="checkbox"/> M_AUTO	Determines the last resolution automatically.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> 0 to 7	Specifies the resolution level. Only integer values are accepted. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_SAVE_TARGET_EDGES	Sets whether to save the target edges in the result buffer. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISABLE	Disables saving the target edges.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ENABLE	Enables the saving of the target edges. By enabling this value, you can retrieve target chain information with MmodGetResult() and you can draw target edges with MmodDraw() . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_SEARCH_ANGLE_RANGE	<p>Sets whether to perform calculations specific to angular-range search strategies.</p> <p>Typically, to search for models within their specified angular range (M_ANGLE-M_ANGLE_DELTA_NEG to M_ANGLE + M_ANGLE_DELTA_POS) in the target, calculations specific to angular-range search strategies should be enabled for the context. These calculations are not required to search for models at their nominal angle (M_ANGLE). In addition, if you expect that the occurrences sought are close to their model's nominal angle, you can try disabling these calculations to see if Model Finder can still find the required occurrences. Disabling these calculations might speed up the search depending on the model and the target.</p> <p>Note that M_SEARCH_ANGLE_RANGE must be enabled to search for rotation-invariant non-synthetic models (for example, an image-type model of a circle).</p> <p>Also note that candidates can only be returned as occurrences if found within the angular range specified for their model. Therefore, whether you enable or disable the calculation, you can restrict which candidates are returned as occurrences by narrowing the angular range.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value.</p> <p>The default is the same as M_ENABLE if the Model Finder context uses a general geometric search algorithm.</p> <p>The default is the same as M_DISABLE if the Model Finder context uses a controlled geometric search algorithm.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISABLE	Disables calculations specific to angular-range search strategies.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ENABLE	<p>Enables calculations specific to angular-range search strategies.</p> <p>This setting is not available if the Model Finder context uses a controlled geometric search algorithm.</p> <p>(summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_SEARCH_POSITION_RANGE	<p>Sets whether to perform calculations specific to position-range search strategies.</p> <p>Typically, to search for models within their specified position range (M_POSITION_X, M_POSITION_Y, M_POSITION_DELTA_NEG_X, M_POSITION_DELTA_POS_X, M_POSITION_DELTA_NEG_Y, M_POSITION_DELTA_POS_Y) in the target, calculations specific to position-range search strategies should be enabled for the context. These calculations are not required to search for models at their nominal position (M_POSITION_X and M_POSITION_Y). In addition, if you expect that the occurrences sought are close to their model's nominal position, you can try disabling these calculations to see if Model Finder can still find the required occurrences. Disabling these calculations might speed up the search depending on the model and the target.</p> <p>Note that candidates can only be returned as occurrences if found within the position range specified for their model. Therefore, whether you enable or disable the calculations, you can restrict which candidates are returned as occurrences by</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

[illegible]

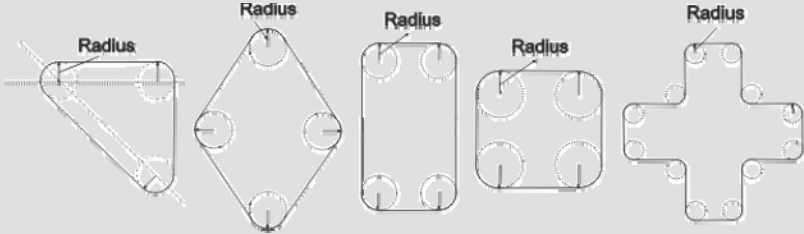
	decrease the robustness and subpixel accuracy of the <code>MmodFind()</code> operation. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Same as <code>M_MEDIUM</code> .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_HIGH	Sets the speed to high.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_LOW	Sets the speed to low.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input checked="" type="checkbox"/> M_MEDIUM	Sets the speed to medium.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_VERY_HIGH	Sets the speed to very high.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input checked="" type="checkbox"/> M_TARGET_CACHING	<p>Sets whether target caching is enabled. When target caching is enabled, the geometric representation of the target, generated by <code>MmodFind()</code>, is kept in the result.</p> <p>Subsequent calls to <code>MmodFind()</code> can reuse the representation, if they use the same result buffer and target. This increases the speed of the searches.</p> <p>Note that the geometric representation generated with a given Model Finder context might not be compatible with another context. In such a case, the cache will not be used.</p> <p>Also note that this constant cannot be set to <code>M_ENABLE</code> if the Model Finder context is set to <code>M_GEOMETRIC_CONTROLLED</code>. (summarize)</p>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Same as <code>M_DISABLE</code> .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input checked="" type="checkbox"/> M_DISABLE	Disables target caching. Each call to <code>MmodFind()</code> will generate the geometric representation of the target. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ENABLE	Enables target caching in the result.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input checked="" type="checkbox"/> M_TIMEOUT	Sets the maximum search time for <code>MmodFind()</code> , in msec. If a search times out, the number of occurrences found at that point will be returned. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 2000.0 msec. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DISABLE	Disables timing out of the search.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> Value	Specifies the maximum search time, in msec.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

The possible **ControlType** and corresponding **ControlValue** settings for models (all or a specified one) are:

For one or all models	
ControlType	Description
ControlValue	
M_ACCEPTANCE	<p>Sets the acceptance level for the score. An occurrence will be returned only if the match score between the target and the model is greater than or equal to this level.</p> <p>The score is a measure, as a percentage, of the presence and fit of the model's active edges in the occurrence. Using M_FIT_ERROR_WEIGHTING_FACTOR, you can specify how much the fit influences the score.</p> <p>(summarize)</p>
M_DEFAULT	<p>Specifies the default value. The default value is 60.0 percent.</p> <p>(summarize)</p>
0.0 to 100.0	<p>Specifies an acceptable score. 100% indicates that for every active edge in the model, a corresponding edge must be found in the occurrence with a perfect fit (generally hard to obtain).</p> <p>(summarize)</p>
M_ACCEPTANCE_TARGET	<p>Sets the acceptance level for the target score. An occurrence will be returned only if the target score between the target and the model is greater than or equal</p>

	to this level.
	The target score is a measure, as a percentage, of edges found in the occurrence that are not found in the original model, weighted by the deviation in position of the common edges. Using M_FIT_ERROR_WEIGHTING_FACTOR , you can specify how much the fit influences the target score. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies an acceptable target score, as a percentage. At 0.0%, any number of extra edges is tolerated (100% allows for no extra edges in the occurrence and the occurrence must have a perfect fit). (summarize)
<input type="checkbox"/> M_ANGLE	Sets the nominal search angle, relative to the model reference axis angle. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 °. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the nominal search angle, in degrees.
<input type="checkbox"/> M_ANGLE_DELTA_NEG	Sets the lower limit of the angular range, relative to the nominal search angle (M_ANGLE). Occurrences with angles outside the angular-range cannot be returned as results. Note that typically, to search for model occurrences within the angular range, calculations specific to angular-range search strategies should be enabled (M_SEARCH_ANGLE_RANGE) for the context. When enabled, the angular range should be used to cover an expected variance in angle. Note that the actual angle of the occurrence does not affect search speed. If you need to search for a model at discrete angles only (for example, at intervals of 90 degrees), it is typically more efficient to define several models with different expected angles, than to search through the full angular range. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 180.0 °. (summarize)
<input type="checkbox"/> 0.0 to 180.0	Specifies the lower limit of the angular range, in degrees.
<input type="checkbox"/> M_ANGLE_DELTA_POS	Sets the upper limit of the angular range, relative to the nominal search angle (M_ANGLE). Occurrences with angles outside the angular-range cannot be returned as results. Note that typically, to search for model occurrences within the angular range, calculations specific to angular-range search strategies should be enabled (M_SEARCH_ANGLE_RANGE) for the context. When enabled, the angular range should be used to cover an expected variance in angle. Note that the actual angle of the occurrence does not affect search speed. If you need to search for a model at discrete angles only (for example, at intervals of 90 degrees), it is typically more efficient to define several models with different expected angles, than to search through the full angular range. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 180.0 °. (summarize)
<input type="checkbox"/> 0.0 to 180.0	Specifies the upper limit of the angular range, in degrees.
<input type="checkbox"/> M_ASSOCIATED_CALIBRATION	Associates the specified calibration object with the specified model. Note that you must reassociate (if necessary) each model with its calibration object after restoring a Model Finder context using MmodRestore() or MmodStream() , since the calibration object is not saved with the Model Finder context. In addition, for a synthetic model, you should use M_ASSOCIATED_CALIBRATION to associate the calibration object of a calibrated target with the model. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_NULL .
<input type="checkbox"/> M_NULL	Removes the association between a model and a calibration object.
<input type="checkbox"/> MIL Calibration object identifier	Specifies the calibration object to associate with the model.
<input type="checkbox"/> M_AUTO_DEFINE	Defines a unique model from a source image, automatically. (summarize)
<input type="checkbox"/> Source image identifier	Specifies the identifier of the image buffer (model source image) from which to extract the model. The image buffer must be a 1-band 8-bit unsigned buffer. (summarize)
<input type="checkbox"/> M_BOX_MARGIN_BOTTOM	Sets a margin at the bottom of the bounding box of the model's active edges. The bounding box of the model's active edges plus the specified margins define the size of the model box.

	<p>The model box is used to compute the target score and to set the different masks of the model. If you change this value, all of this model's previous masks are discarded, since their size is no longer valid.</p> <p>M_BOX_MARGIN_BOTTOM is only available for synthetic models. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 10.0 percent of the height of the bounding box of the model's active edges. (summarize)</p>
<input type="checkbox"/> Value >= 0.0	<p>Specifies the margin, in the user-defined units for the model.</p>
<input type="checkbox"/> M_BOX_MARGIN_LEFT	<p>Sets a margin at the left side of the bounding box of the model's active edges.</p> <p>The bounding box of the model's active edges plus the specified margins define the size of the model box.</p> <p>The model box is used to compute the target score and to set the different masks of the model. If you change this value, all of this model's previous masks are discarded, since their size is no longer valid.</p> <p>M_BOX_MARGIN_LEFT is only available for synthetic models. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 10.0 percent of the width of the bounding box of the model's active edges. (summarize)</p>
<input type="checkbox"/> Value >= 0.0	<p>Specifies the margin, in the user-defined units for the model.</p>
<input type="checkbox"/> M_BOX_MARGIN_RIGHT	<p>Sets a margin at the right side of the bounding box of the model's active edges.</p> <p>The bounding box of the model's active edges plus the specified margins define the size of the model box.</p> <p>The model box is used to compute the target score and to set the different masks of the model. If you change this value, all of this model's previous masks are discarded, since their size is no longer valid.</p> <p>M_BOX_MARGIN_RIGHT is only available for synthetic models. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 10.0 percent of the width of the bounding box of the model's active edges. (summarize)</p>
<input type="checkbox"/> Value >= 0.0	<p>Specifies the margin, in the user-defined units for the model.</p>
<input type="checkbox"/> M_BOX_MARGIN_TOP	<p>Sets a margin at the top of the bounding box of the model's active edges.</p> <p>The bounding box of the model's active edges plus the specified margins define the size of the model box.</p> <p>The model box is used to compute the target score and to set the different masks of the model. If you change this value, all of this model's previous masks are discarded, since their size is no longer valid.</p> <p>M_BOX_MARGIN_TOP is only available for synthetic models. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 10.0 percent of the height of the bounding box of the model's active edges. (summarize)</p>
<input type="checkbox"/> Value >= 0.0	<p>Specifies the margin, in the user-defined units for the model.</p>
<input type="checkbox"/> M_CAD_Y_AXIS	<p>Sets the direction of the Y-axis for a model of type M_DXF_FILE.</p> <p>When the model is added to the Model Finder context from a CAD DXF file, the model will retain its coordinate system (the origin and the axis). Most CAD DXF files are oriented so that the Y-axis is positive going up, whereas the coordinate system MIL uses is oriented with the Y-axis positive going down. So if you take the edge coordinates of an object from a CAD DXF file and put them in an image, the imaged object will look flipped when compared to the original object. This is also the case for a model defined from a CAD DXF file. This means that, unless the object is symmetrical, the match will not be made. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Same as M_FLIP for a model of type M_DXF_FILE. If the model is not of type M_DXF_FILE, this control type must remain set to M_DEFAULT, otherwise an error is generated. (summarize)</p>

<input type="checkbox"/> M_FLIP	Flips the Y-axis for the model so that the Y-axis is positive going down.
<input type="checkbox"/> M_NO_FLIP	Does not flip the Y-axis for the model.
<input type="checkbox"/> M_CERTAINTY	Sets the certainty level for the score, as a percentage. If both the score and target scores are greater than or equal to their respective certainty levels, the occurrence is considered a match, without searching the rest of the target for better matches (provided the specified number of occurrences has been found). (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 90.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the certainty level for the score, as a percentage. If you set the certainty level too high (close to 100.0%), you slow down the search because you force the search algorithm to check the whole position range for the best possible match(es). A good certainty level is slightly lower than the expected score, so that the search can finish as soon as a match is found. However, if you set the certainty level too low, false matches might be found. (summarize)
<input type="checkbox"/> M_CERTAINTY_TARGET	Sets the certainty level for the target score. If both the score and target scores are greater than or equal to their respective certainty levels, the occurrence is considered a match, without searching the rest of the target for better matches (provided the specified number of occurrences has been found). (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the certainty level for the target score, as a percentage. See the possible values of M_CERTAINTY for more details. (summarize)
<input type="checkbox"/> M_CORNER_RADIUS	Sets the radius used to round all the corners of predefined-shaped models that have corners.  This control type is only valid for models of type M_RECTANGLE , M_SQUARE , M_DIAMOND , M_TRIANGLE , and M_CROSS . Attempting to use this value for any other model type will generate an error. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> Value >= 0.0	Specifies the radius, in the user-defined units for the model. If you set the radius to 0.0, there will be no rounding. (summarize)
<input type="checkbox"/> M_FIT_ERROR_WEIGHTING_FACTOR	Sets the fit error weighting factor. This factor determines the contribution of the fit error in the score and target score calculation. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 25.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the fit error weighting factor, as a percentage. The higher the percentage, the greater the contribution of the fit error in determining the score and target score. (summarize)
<input type="checkbox"/> M_MIN_SEPARATION_ANGLE	Sets the minimum angular separation required for two occurrences to be considered two distinct matches (two separate occurrences). Note, only one of the separation criteria needs to be met for occurrences to be considered distinct. DISTINCT OCCURRENCE = (Separation in X) OR (Separation in Y) OR (Separation in Angle) OR (Separation in Scale) (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 10.0 degrees.

	(summarize)
<input type="checkbox"/> 0.0 <Value<= 180.0	Specifies the minimum angular separation required. This value is specified as an absolute angle value. (summarize)
<input type="checkbox"/> M_DISABLE	Disables the minimum angle separation criteria. When disabled (M_DISABLE), the angle is not a factor when determining if occurrences are distinct. (summarize)
<input type="checkbox"/> M_MIN_SEPARATION_SCALE	Sets the minimum separation required in scale for two occurrences to be considered distinct matches (two separate occurrences). This value is specified as a scale factor. Note, only one of the separation criteria needs to be met for occurrences to be considered distinct. DISTINCT OCCURRENCE = (Separation in X) OR (Separation in Y) OR (Separation in Angle) OR (Separation in Scale) (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.1. (summarize)
<input type="checkbox"/> 1.0 <Value<= 4.0	Specifies the criteria for minimum separation in scale.
<input type="checkbox"/> M_DISABLE	Disables the minimum scale separation criteria. When disabled, the scale is not a factor when determining if occurrences are distinct. (summarize)
<input type="checkbox"/> M_MIN_SEPARATION_X	Sets the minimum separation required along the X-axis for two occurrences to be considered distinct matches (two separate occurrences). Note, only one of the separation criteria needs to be met for occurrences to be considered distinct. DISTINCT OCCURRENCE = (Separation in X) OR (Separation in Y) OR (Separation in Angle) OR (Separation in Scale) (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 10.0 percent. (summarize)
<input type="checkbox"/> M_DISABLE	Disables the minimum separation in X criteria. When disabled, separation along the X-axis is not a factor when determining if occurrences are distinct. (summarize)
<input type="checkbox"/> Value	Specifies the minimum separation as a percentage of the model's width at M_SCALE . This value can be greater than 100%. (summarize)
<input type="checkbox"/> M_MIN_SEPARATION_Y	Sets the minimum separation required along the Y-axis for two occurrences to be considered distinct matches (two separate occurrences). Note, only one of the separation criteria needs to be met for occurrences to be considered distinct. DISTINCT OCCURRENCE = (Separation in X) OR (Separation in Y) OR (Separation in Angle) OR (Separation in Scale) (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 10.0 percent. (summarize)
<input type="checkbox"/> M_DISABLE	Disables the minimum separation in Y criteria. When disabled, separation along the Y-axis is not a factor when determining if occurrences are distinct. (summarize)
<input type="checkbox"/> Value	Specifies the minimum separation as a percentage of the model's height at M_SCALE . This value can be greater than 100%. (summarize)
<input type="checkbox"/> M_PIXEL_SCALE	Sets the pixel scale of the model, if it is a synthetic model. This value is the scale to apply to the model to go from model units to pixel units. M_PIXEL_SCALE is used for the match if the target is not calibrated. If the target is calibrated, this value is not used for the match nor for the returned results. However, this value will be used for draw operations. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the pixel scale. If it is a calibrated model, the default is 1/(pixel size in X of the associated calibration). If it is not a calibrated model, the default is 1.0. (summarize)
<input type="checkbox"/> Value > 0.0	Specifies the pixel scale. Note that if the model is not a synthetic model, the only valid value is 1.0. Any other value will generate an error. (summarize)

<input type="checkbox"/> M_POLARITY	<p>Sets the expected polarity of occurrences, compared to that of the model.</p> <div> <div> <div>Model</div> </div> </div> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_SAME .
<input type="checkbox"/> M_ANY	Specifies that the occurrences can be a mixture of polarities.
<input type="checkbox"/> M_REVERSE	Specifies that the polarity of occurrences is the reverse of that of the model.
<input type="checkbox"/> M_SAME	Specifies that the polarity of occurrences is the same as that of the model.
<input type="checkbox"/> M_SAME_OR_REVERSE	Specifies that the polarity of occurrences can be either the same or the reverse of that of the model.
<input type="checkbox"/> M_POSITION_DELTA_NEG_X	<p>Sets the valid position range in the negative X-direction, relative to the nominal position (M_POSITION_X and M_POSITION_Y).</p> <p>The position range limits the region in which the position of a model occurrence can be found; position coordinates which fall outside this region cannot be returned as results (MmodGetResult() with M_POSITION_X and M_POSITION_Y). Note that the position returned for an occurrence is determined by the model's reference axis position. The region defined by the position range can lie partially, or totally, outside the target.</p> <p>Note that typically, to search for model occurrences within the position range, calculations specific to position-range search strategies should be enabled (M_SEARCH_POSITION_RANGE) for the context. When enabled, using a small position range generally decreases the search time, depending on the number of details present in the target. Always set the position range to the minimum required when speed is a consideration.</p> <p>When M_POSITION_X is set to M_ALL, M_POSITION_DELTA_NEG_X will be treated as if set to M_INFINITE regardless of its actual setting.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0.
<input type="checkbox"/> M_INFINITE	Specifies the position range as the entire image plane in the negative X-direction. This means that any X-coordinate in the negative X-direction from the nominal position (even outside the target) can be returned as a result.
<input type="checkbox"/> Value >= 0.0	Specifies the position range's negative X-offset, in pixels, and can be specified with subpixel accuracy.
<input type="checkbox"/> M_POSITION_DELTA_NEG_Y	<p>Sets the valid position range in the negative Y-direction, relative to the nominal position (M_POSITION_X and M_POSITION_Y).</p> <p>The position range limits the region in which the position of a model occurrence can be found; position coordinates which fall outside this region cannot be returned as results (MmodGetResult() with M_POSITION_X and M_POSITION_Y). Note that the position returned for an occurrence is determined by the model's reference axis position. The region defined by the position range can lie partially, or totally, outside the target.</p> <p>Note that typically, to search for model occurrences within the position range, calculations specific to position-range search strategies should be enabled (M_SEARCH_POSITION_RANGE) for the context. When enabled, using a small position range generally decreases the search time, depending on the number of details present in the target. Always set the position range to the minimum required when speed is a consideration.</p> <p>When M_POSITION_Y is set to M_ALL, M_POSITION_DELTA_NEG_Y will be treated as if set to M_INFINITE regardless of its actual setting.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0.
<input type="checkbox"/> M_INFINITE	Specifies the position range as the entire image plane in the negative Y-direction. This means that any Y-coordinate in the negative Y-direction from the nominal position (even outside the target) can be returned as a result.
<input type="checkbox"/> Value >= 0.0	Specifies the position range's negative Y-offset, in pixels, and can be specified with subpixel accuracy.

<input type="checkbox"/> M_POSITION_DELTA_POS_X	<p>Sets the valid position range in the positive X-direction, relative to the nominal position (M_POSITION_X and M_POSITION_Y).</p> <p>The position range limits the region in which the position of a model occurrence can be found; position coordinates which fall outside this region cannot be returned as results (MmodGetResult() with M_POSITION_X and M_POSITION_Y). Note that the position returned for an occurrence is determined by the model's reference axis position. The region defined by the position range can lie partially, or totally, outside the target.</p> <p>Note that typically, to search for model occurrences within the position range, calculations specific to position-range search strategies should be enabled (M_SEARCH_POSITION_RANGE) for the context. When enabled, using a small position range generally decreases the search time, depending on the number of details present in the target. Always set the position range to the minimum required when speed is a consideration.</p> <p>When M_POSITION_X is set to M_ALL, M_POSITION_DELTA_POS_X will be treated as if set to M_INFINITE regardless of its actual setting. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_INFINITE .
<input type="checkbox"/> M_INFINITE	Specifies the position range as the entire image plane in the positive X-direction. This means that any X-coordinate in the positive X-direction from the nominal position (even outside the target) can be returned as a result. (summarize)
<input type="checkbox"/> Value >= 0.0	Specifies the position range's positive X-offset, in pixels, and can be specified with subpixel accuracy.
<input type="checkbox"/> M_POSITION_DELTA_POS_Y	<p>Sets the valid position range in the positive Y-direction, relative to the nominal position (M_POSITION_X and M_POSITION_Y).</p> <p>The position range limits the region in which the position of a model occurrence can be found; position coordinates which fall outside this region cannot be returned as results (MmodGetResult() with M_POSITION_X and M_POSITION_Y). Note that the position returned for an occurrence is determined by the model's reference axis position. The region defined by the position range can lie partially, or totally, outside the target.</p> <p>Note that typically, to search for model occurrences within the position range, calculations specific to position-range search strategies should be enabled (M_SEARCH_POSITION_RANGE) for the context. When enabled, using a small position range generally decreases the search time, depending on the number of details present in the target. Always set the position range to the minimum required when speed is a consideration.</p> <p>When M_POSITION_Y is set to M_ALL, M_POSITION_DELTA_POS_Y will be treated as if set to M_INFINITE regardless of its actual setting. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_INFINITE .
<input type="checkbox"/> M_INFINITE	Specifies the position range as the entire image plane in the positive Y-direction. This means that any Y-coordinate in the positive Y-axis from the nominal position (even outside the target) can be returned as a result. (summarize)
<input type="checkbox"/> Value >= 0.0	Specifies the position range's positive Y-offset, in pixels, and can be specified with subpixel accuracy.
<input type="checkbox"/> M_POSITION_X	<p>Sets the nominal search position for the X-coordinate. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> M_ALL	Specifies all X-coordinates.
<input type="checkbox"/> Value	Specifies the nominal search position, in pixels, and can be specified with subpixel accuracy.
<input type="checkbox"/> M_POSITION_Y	<p>Sets the nominal search position for the Y-coordinate. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> M_ALL	Specifies all Y-coordinates.
<input type="checkbox"/> Value	Specifies the nominal search position, in pixels, and can be specified with subpixel accuracy.
<input type="checkbox"/> M_REFERENCE_ANGLE	<p>Sets the angle of the reference axis for the model. Angle results are returned relative to this axis, rather than the model source image axis. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 degrees. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_REFERENCE_X	Sets the X-offset of the reference axis origin of the model, relative to the model origin. Position results return the X-coordinate of the model's reference axis origin transformed at the model occurrence. Position results are relative to the target origin.

	Note that the reference axis origin need not be in the model. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies that the default X-offset will be used. For a synthetic model, the default value is 0.0. For other types of models, the default value is the center of the model. (summarize)
<input type="checkbox"/> Value	Specifies the X-offset value. The value is in pixels, and can be specified in subpixel accuracy. (summarize)
<input type="checkbox"/> M_REFERENCE_Y	Sets the Y-offset of the reference axis origin of the model, relative to the model origin. Position results return the Y-coordinate of the model's reference axis origin transformed at the model occurrence. Position results are relative to the target origin. Note that the reference axis origin need not be in the model. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies that the default Y-offset will be used. For a synthetic model, the default value is 0.0. For other types of models, the default value is the center of the model. (summarize)
<input type="checkbox"/> Value	Specifies the Y-offset value. The value is in pixels, and can be specified in subpixel accuracy. (summarize)
<input type="checkbox"/> M_SCALE	Sets the nominal search scale. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value	Specifies the value of the nominal search scale. For a synthetic model, the nominal scale can be any positive value. For all other models, the nominal scale must be from 0.5 to 2.0. (summarize)
<input type="checkbox"/> M_SCALE_MAX_FACTOR	Sets the factor used to determine the upper limit (maximum permitted scale) of the scale range. M_SCALE is multiplied by this factor. Occurrences with scales outside the scale-range cannot be returned as results. Note that typically, to search for model occurrences within the scale range, calculations specific to scale-range search strategies should be enabled (M_SEARCH_SCALE_RANGE) for the context. When enabled, the scale range should only cover the expected variance in scale to avoid slowing down the search and avoid finding unwanted occurrences. A search through a range of scales is performed in parallel, meaning that the actual scale of an occurrence has no bearing on which occurrence will be found first. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 2.0. (summarize)
<input type="checkbox"/> 1.0 to 2.0	Specifies the factor that determines the maximum scale of the occurrence.
<input type="checkbox"/> M_SCALE_MIN_FACTOR	Sets the factor used to determine the lower limit (minimum permitted scale) of the scale range. M_SCALE is multiplied by this factor. Occurrences with scales outside the scale-range cannot be returned as results. Note that typically, to search for model occurrences within the scale range, calculations specific to scale-range search strategies should be enabled (M_SEARCH_SCALE_RANGE) for the context. When enabled, the scale range should only cover the expected variance in scale to avoid slowing down the search and avoid finding unwanted occurrences. A search through a range of scales is performed in parallel, meaning that the actual scale of an occurrence has no bearing on which occurrence will be found first. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.5. (summarize)
<input type="checkbox"/> 0.5 to 1.0	Specifies the factor that determines the minimum scale of the occurrence.
<input type="checkbox"/> M_USER_LABEL	Sets a unique user-defined label for the specified model. This label can be used as a means of identifying your model, independently from its index, in the Model Finder context. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_NO_LABEL .

<input type="checkbox"/> M_NO_LABEL	Specifies that no user label is associated with the model.
<input type="checkbox"/> Value	Specifies the user label of the model. The value must be an integer value that is not associated as a label with any other model in the Model Finder context. Since the same label cannot be associated with multiple models, you cannot pass a user label if the Index parameter of this function is set to M_ALL . (summarize)

The possible [ControlType](#) and corresponding [ControlValue](#) settings for a model, all models, or a result are:

● For one or all models or a result	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_X	Sets the X-coordinate of the top left corner of the region in the model source image that will be used when drawing in the destination image (MmodDraw()). This control type is not saved with the context. If you restore the context, this control type must be set again. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. When drawing a model, the default is the current value of MmodInquire() with M_BOX_OFFSET_X . When drawing a result, the default is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the relative X-offset, in pixels.
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_Y	Sets the Y-coordinate of the top left corner of the region in the model source image that will be used when drawing in the destination image (MmodDraw()). This control type is not saved with the context. If you restore the context, this control type must be set again. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. When drawing a model, the default is the current value of MmodInquire() with M_BOX_OFFSET_Y . When drawing a result, the default is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the relative Y-offset, in pixel units.
<input type="checkbox"/> M_DRAW_SCALE_X	Sets the scale in the X-direction when drawing models or results in the destination image buffer. This control type is not saved with the context. If you restore the context, this control type must be set again. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as the current value of M_PIXEL_SCALE , for a model. When drawing a result, the default is 1.0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the scale in the X-direction.
<input type="checkbox"/> M_DRAW_SCALE_Y	Sets the scale in the Y-direction when drawing models or results in the destination image buffer. This control type is not saved with the context. If you restore the context, this control type must be set again. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as the current value of M_PIXEL_SCALE , for a model. When drawing a result, the default is 1.0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the scale in the Y-direction.

The possible [ControlType](#) and corresponding [ControlValue](#) settings for a result are:

For a result	
ControlType	Description
ControlValue	
<input type="checkbox"/> M_MOD_DEFINE_COMPATIBLE	Saves all the necessary information in the result buffer to be able to define a model from a result occurrence (MmodDefine() with M_MOD_RESULT). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Disables saving information.
<input type="checkbox"/> M_ENABLE	Enables saving information.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodDefine

Synopsis

Add a model to, or delete a model from, a Model Finder context.

Syntax

```
void MmodDefine(
    MIL_ID ContextId,
    MIL_INT ModelType,
    MIL_DOUBLE Param1,
    MIL_DOUBLE Param2,
    MIL_DOUBLE Param3,
    MIL_DOUBLE Param4,
    MIL_DOUBLE Param5
)
```

Description

This function allows you to add a model to, or delete a model from, a Model Finder context. You can define a model from an image, from a result buffer, from two other models, or you can define a synthetic model. Image-type, result-type, merge-type, and synthetic models can be mixed in the same Model Finder context.

There are two types of image-type models: models for which you manually define their location in the model source image ([M_IMAGE](#)), and models that MIL automatically defines from the model source image ([M_AUTO_DEFINE](#)). [M_AUTO_DEFINE](#) defines a unique model by searching the model source image for unique geometric features. [MmodDefine\(\)](#) can define an automatic model based on default settings. To have specified settings taken into account during model definition, you must allocate an empty image-type model using [M_AUTO_DEFINE](#), set the appropriate control types using [MmodControl\(\)](#), and then call [MmodControl\(\)](#) with [M_AUTO_DEFINE](#). It can take several seconds to find the best model (or more for large or small images). To be useful, the model source image should be a typical target image; otherwise, the selected area for the model might never be present in the target image. For more information on defining a model automatically, see the [Image-type models](#) subsection in the [Defining and adding models to your Model Finder context](#) section in [Chapter 8: Geometric Model Finder](#).

There are two types of result-type models: models defined from an Edge Finder result buffer (Edge Finder-type models), and models defined from a Model Finder result buffer (Model Finder-type model).

Merge-type models are composed of the edges of two different models. When you define a merge-type model, the two models from which it is merged are not affected.

There are two types of synthetic models: models of a predefined shape (predefined-shape models) and models defined from a CAD DXF file (CAD-file models). Use [MmodDefineFromFile\(\)](#) to define a CAD-file model.

By default, synthetic models are defined to have a 10% margin around the bounding box of their active edges. When finding an occurrence, any extra edges found in this area will reduce the target score. You can change the size of the margins with the [MmodControl\(\)](#) [M_BOX_MARGIN_...](#) control types.

Specify the dimensions of synthetic models in user-defined units. Then, specify the ratio between model units and pixel units using [MmodControl\(\)](#) with [M_PIXEL_SCALE](#). If the target is not calibrated, then [M_PIXEL_SCALE](#) is used for the match. If the target is calibrated, the model units should be the same as the calibrated units. If the target is calibrated, [M_PIXEL_SCALE](#) will be used for draw operations, but not for the match nor the returned results. If you are searching for a synthetic model in a calibrated target, you must also associate the calibration object of the target with the model, using [M_ASSOCIATED_CALIBRATION](#). MIL needs the calibration object for internal purposes at preprocessing time. MIL will not use the calibration object to compensate for incongruencies between the model and the target, nor will MIL map the model's coordinate system to that of the target.

After you have specified your pixel scale and scale settings, you can verify if your synthetic models are valid, using [MmodInquire\(\)](#) with [M_VALID](#). Synthetic models can be invalid if their global size in X or Y (*size of the model box * M_PIXEL_SCALE * M_SCALE*) is greater than 1024.

If the image or the result buffer used to define the model is calibrated, then that calibration object is automatically associated with the model. Models can use different calibration objects within the same Model Finder context, provided all calibrations map to the same world. All image-type or result-type models must be either calibrated or not.

When a call to [MmodFind\(\)](#) is made, all models that are part of the specified context are searched for simultaneously in the target.

Note, when a model is added or deleted, the Model Finder context must be preprocessed again, using [MmodPreprocess\(\)](#).

The search is performed according to the general search settings specified in the Model Finder context ([M_CONTEXT](#)), as well as the individual model search settings. Both the context and individual model search settings can be specified using [MmodControl\(\)](#).

The model index starts at 0, and each subsequent model added to the Model Finder context is given a sequential index number, in the order that the model was added. If a model is deleted, all entries with higher indices are shifted down one.

For more information on defining models, see the [Guidelines for choosing models](#) section in [Chapter 8: Geometric Model Finder](#).

Parameters

ContextId

Specifies the Model Finder context to which to add, or from which to delete, the model. The Model Finder context must have been previously allocated on the required system using `MmodAlloc()`.

ModelType

Specifies the type of model to define when adding models to the context, or specifies to delete a model from the Model Finder context.

See the [Parameter associations](#) section for possible values.

Param1

Specifies an attribute of the model to add. Its definition is dependent on the model type chosen.

See the [Parameter associations](#) section for possible values.

Param2

Specifies an attribute of the model to add. Its definition is dependent on the model type chosen.

See the [Parameter associations](#) section for possible values.

Param3

Specifies an attribute of the model to add. Its definition is dependent on the model type chosen.

See the [Parameter associations](#) section for possible values.

Param4

Specifies an attribute of the model to add. Its definition is dependent on the model type chosen.

See the [Parameter associations](#) section for possible values.

Param5

Specifies an attribute of the model to add. Its definition is dependent on the model type chosen.

See the [Parameter associations](#) section for possible values.

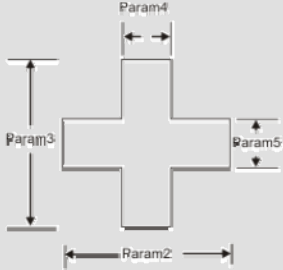
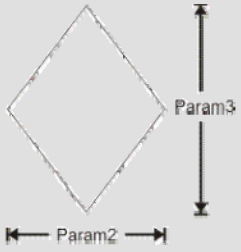
Parameter associations


Possible values for the **ModelType**, **Param1**, **Param2**, **Param3**, **Param4**, and **Param5** parameters are described in the following tables:

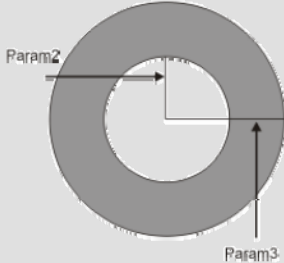

- [For the ModelType parameter](#)
- [For removing a model from the context](#)

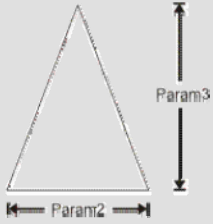
Note that any unused parameters should be set to **M_DEFAULT**.

For the ModelType parameter													
ModelType	Description												
Param1	corona-1	gige visit	gpu proc	hellos ea	hellos ec	hellos ed	ieee 139	iris (i)	met-II /<	met-II /<	met-II /<	met-II /<	vio (w)

<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that white is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param2	Sets the radius of the circle. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_CROSS	Specifies a predefined cross as the model.  You can round the corners of this model using MmodControl() with M_CORNER_RADIUS . (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param1	Sets the foreground color of the cross. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_ANY	Specifies that the model has no specific polarity.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_FOREGROUND_BLACK	Specifies that black is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that white is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param2	Sets the width of the cross. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param3	Sets the height of the cross. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param4	Sets the horizontal thickness of the cross. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param5	Sets the vertical thickness of the cross. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_DIAMOND	Specifies a predefined diamond as the model.  You can round the corners of this model using MmodControl() with M_CORNER_RADIUS . (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param1	Sets the foreground color of the diamond.	a b c d e f g h i j k l m n o p q r s t u v w

M_MERGE_MODEL		Defines a model from two other models. These models have to be in the same Model Finder context.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
		The model is defined with the edges of the first model and the edges of the second model that are not equal to the edges of the first model.																							
		The size of the merge-type model is the size of the first model. To include all of the edges of both models in the new model, set the first parameter to the index of the larger model.																							
		If the models being merged are calibrated, the new merge-type model will be associated with the calibration object of the first model. (summarize)																							
Param1		Sets the index of the first model. User labels cannot be used as indices; to retrieve the index of a model from its user label, use MmodInquire() with M_INDEX_FROM_LABEL . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
Param2		Sets the index of the second model. User labels cannot be used as indices; to retrieve the index of a model from its user label, use MmodInquire() with M_INDEX_FROM_LABEL . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_MOD_RESULT		Defines a model from a Model Finder result buffer.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
		The model is defined with the edges of the target at the position of the specified occurrence.																							
		To define a model from a Model Finder result buffer, enable MmodControl() with M_MOD_DEFINE_COMPATIBLE . In the case of a Model Finder-type model, the model source image is the target from which the results were obtained. (summarize)																							
	Param1	Sets the identifier of the Model Finder result buffer. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
		<i>Board specific</i>																							
		The Model Finder result buffer must be allocated on the same system as the Model Finder context (ContextId). If it is not, an error will occur.																		q	r	s	t	u	v
Param2		Sets the index of the occurrence. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_RECTANGLE		Specifies a predefined rectangle as the model.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
																									
		You can round the corners of this model using MmodControl() with M_CORNER_RADIUS . (summarize)																							
	Param1	Sets the foreground color of the rectangle. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	M_DEFAULT	Same as M_ANY .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	M_ANY	Specifies that the model has no specific polarity.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	M_FOREGROUND_BLACK	Specifies that black is the foreground color of the model.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that white is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param2	Sets the width of the rectangle. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param3	Sets the height of the rectangle. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_RING	Specifies a predefined ring as the model.  (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param1	Sets the foreground color of the ring. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_ANY	Specifies that the model has no specific polarity.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_FOREGROUND_BLACK	Specifies that black is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that white is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param2	Sets the radius of the inner ring. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param3	Sets the radius of the outer ring. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_SQUARE	Specifies a predefined square as the model.  You can round the corners of this model using MmodControl() with M_CORNER_RADIUS . (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param1	Sets the foreground color of the square. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_ANY	Specifies that the model has no specific polarity.	a b c d e f g h i j k l m n o p q r s t u v w

<input type="checkbox"/> M_FOREGROUND_BLACK	Specifies that black is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that white is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param2	Sets the width and height of the square. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_TRIANGLE	Specifies a predefined triangle as the model.  You can round the corners of this model using MmodControl() with M_CORNER_RADIUS . (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param1	Sets the foreground color of the triangle. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_ANY	Specifies that the model has no specific polarity.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_FOREGROUND_BLACK	Specifies that black is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that white is the foreground color of the model.	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param2	Sets the width of the triangle. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w
<input type="checkbox"/> Param3	Sets the height of the triangle. (summarize)	a b c d e f g h i j k l m n o p q r s t u v w

The following **ModelType** parameter setting can be used to remove a model from the context.

Note that any unused parameters should be set to **M_DEFAULT**.

For removing a model from the context		
<input type="checkbox"/> ModelType	Description	
Param1		
Param2		
Param3		
Param4		
Param5		
<input type="checkbox"/> M_DELETE	Deletes the specified model from the context. (summarize)	
<input type="checkbox"/> Param1	Sets the model index. (summarize)	
<input type="checkbox"/> Value	Specifies the model index. User labels cannot be used as indices; to retrieve the index of a model from its user label, use MmodInquire() with M_INDEX_FROM_LABEL . (summarize)	

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodDefineFromFile

Synopsis

Defines a model from a file and adds it to a Model Finder context.

Syntax

```
void MmodDefineFromFile(
    MIL_ID ContextID,
    MIL_INT Filetype,
    MIL_CONST_TEXT_PTR Filename,
    MIL_INT ControlFlag
)
```

Description

This function defines a model from a file and adds it to a Model Finder context. Models defined from a file are CAD-type synthetic models; use [MmodDefine\(\)](#) to add other types of models to the Model Finder context.

You must preprocess the Model Finder context after you add a model.

By default, synthetic models are defined to have a 10% margin around the bounding box of their active edges. When finding an occurrence, any extra edges found in this area will reduce the target score. You can change the size of the margins with the [MmodControl\(\)](#) [M_BOX_MARGIN_...](#) control types.

When the model is added to the Model Finder context from a CAD DXF file, the model will retain the CAD-file's coordinate system (the origin and the axis). Specify the ratio between model units and pixel units using [MmodControl\(\)](#) with [M_PIXEL_SCALE](#).

Note that if the model is used to search in a calibrated target, you must also associate the calibration object of the target with the model, using [M_ASSOCIATED_CALIBRATION](#). MIL needs the calibration object for internal purposes at preprocessing time. MIL will not use the calibration object to compensate for incongencies between the model and the target, nor will MIL map the model's coordinate system to that of the target.

If the target is calibrated, the model units should be the same as the calibrated units. If the target is calibrated, [M_PIXEL_SCALE](#) will be used for draw operations, but not for the match nor the returned results.

Most CAD DXF files are oriented so that the Y-axis is positive going up, whereas the coordinate system MIL uses is oriented with the Y-axis positive going down. So if you take the edge coordinates of an object from a CAD DXF file and put them in an image, the imaged object will look flipped when compared to the original object. This is also the case for a model defined from a CAD DXF file. This means that, unless the object is symmetrical, the match will not be made. You can use [M_CAD_Y_AXIS](#) to set the Y-axis in the appropriate direction.

To delete a model from a Model Finder context, use [MmodDefine\(\)](#) with [M_DELETE](#).

Parameters

ContextID

Specifies the identifier of the Model Finder context.

Filetype

Specifies the type of file from which to define the model.

For specifying the type of file	
Value	Description
<input type="checkbox"/> M_DXF_FILE	Defines the model from the entities in the specified CAD DXF file. This function does not support all entities that are possible in a CAD DXF file. If there are unsupported entities in the file, they are ignored, and the rest of the entities are read. The supported entities are as follows: LINE, POLYLINE, LWPOLYLINE, CIRCLE, ARC, ELLIPSE, INSERT, and BLOCK. A model defined from a CAD DXF file has no polarity.

	(summarize)
--	-----------------------------

Filename

Specifies the name and path of the file from which to define the model. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

● For specifying the file name and path	
☐ Value	Description
☐ MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile". (summarize)
	Parameters
	<i>FileName</i> Specifies the drive, directory, and name of the file.
☐ M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.

ControlFlag

This parameter is reserved for future use. Set this parameter to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodDraw

Synopsis

Draw specific features of models or result occurrences in the destination image buffer.

Syntax

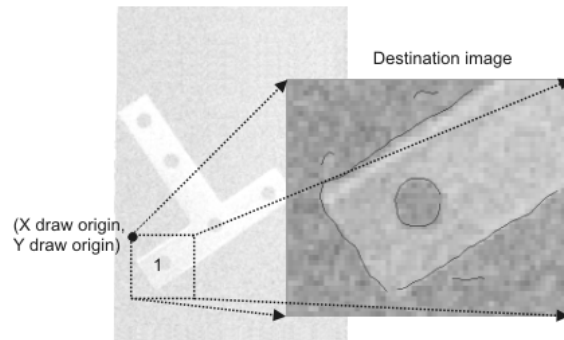
```
void MmodDraw(
    MIL_ID GraphContId,
    MIL_ID ContextOrResultId,
    MIL_ID DestImageId,
    MIL_INT Operation,
    MIL_INT Index,
    MIL_INT ControlFlag
)
```

Description

This function draws specific features of models or result occurrences in the destination buffer.

You can also draw features of a zoomed region of the model image or target by specifying the appropriate values for the [MmodControl\(\)](#) [M_DRAW_RELATIVE_ORIGIN_X](#), [M_DRAW_RELATIVE_ORIGIN_Y](#), [M_DRAW_SCALE_X](#), and [M_DRAW_SCALE_Y](#) control types. The relative origin values specify, in pixels, the coordinates of the top-left corner of the region in the model image or target, while the scale values specify the X- and Y-scaling factors used to fill the destination image buffer.

Drawing a zoomed region of the model image



¹ How much of this is drawn in the destination image is determined by the scale factor and the size of the destination image.

When zooming, **MmodDraw()** will draw into the destination image buffer even if the buffer is not large enough to contain all of the zoomed image. The image will be truncated.

You can use [MmodInquire\(\)](#) with the [M_CHAIN_INDEX](#), [M_CHAIN_X](#), and [M_CHAIN_Y](#) controls to inquire the active edges of the model.

Note that if you try to draw a model's edges and its Model Finder context has not been preprocessed, a preliminary edge extraction operation will be performed for the model.

Parameters

GraphContId

Specifies the graphics context to use when drawing. This parameter must be set to one of the following values:

● For specifying the graphics context	
☐ Value	Description
☐ M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
☐ MIL_graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

ContextOrResultId

Specifies the Model Finder context or result buffer from which to extract the features to draw. The Model Finder context or result buffer must have been previously allocated using [MmodAlloc\(\)](#) or [MmodAllocResult\(\)](#), respectively.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD; Matrox Solios GigE; Matrox Solios eA/XA; Matrox Solios eCL/XCL]

The Model Finder context or result buffer must be allocated on the same system as the graphics context ([GraphContId](#)). If it is not, an error will occur.

DestImageId

Specifies the identifier of the destination image buffer in which to draw. The buffer can be any supported MIL image buffer. By drawing into its display's overlay buffer, you can also annotate an image non-destructively.

When using **MmodDraw()** to draw features extracted from models or targets that are associated with a calibration object, the destination drawing buffer must also be calibrated to the same world coordinate system. **MmodDraw()** draws all the requested features according to whether the destination image buffer is physically corrected or not.

When drawing a model, the destination image must be at least the size of the model. Use [MmodInquire\(\)](#) with the **M_ALLOC_SIZE**... inquire types to get the correct size.

When drawing a synthetic model, the destination image must have a size that matches the size of the model box scaled by [M_PIXEL_SCALE](#).

Operation

Specifies the type of operation to perform. Operations can be added together to draw multiple features at a time. For example, to draw both the result occurrence's position and active edges, you would set the [Operation](#) parameter to [M_DRAW_POSITION](#) + [M_DRAW_EDGES](#). The [Operation](#) parameter can be set to a combination of the following values:

● For specifying the type of operation	
☐ Value	Description
☐ M_DRAW_BOX	Draws the model box, or a bounding box around the occurrence. Note that when drawing an occurrence, the bounding box is drawn maintaining the angle and scale of the occurrence. (summarize)
☐ M_DRAW_DONT_CARES	Draws the model's "don't care" pixels. This is available for models only. (summarize)
☐ M_DRAW_EDGES +	Draws the active edges of the model (not masked out) or the result occurrence. M_MODEL or M_TARGET can be added to M_DRAW_EDGES when you are working with a result occurrence. See the combination values below. (summarize)
☐ M_DRAW_FLAT_REGIONS	Draws the model's "flat regions". This is available for models only. (summarize)
☐ M_DRAW_IMAGE	Draws the model image. This is available for models only. (summarize)
☐ M_DRAW_POSITION	Draws a cross-like symbol at the model's reference axis origin or occurrence position. The cross is drawn maintaining the angle of the model's reference axis or the occurrence. (summarize)
☐ M_DRAW_WEIGHT_REGIONS	Draws the model's weighted region mask. This is available for models only. (summarize)

Combination constants for [M_DRAW_EDGES](#);

You can add one of the following values to the above-mentioned value to specify whether to draw the active edges of the model transformed at the occurrence position or to draw the edges of the target in the region of the occurrence.

● For M_DRAW_EDGES when working with result occurrences	
☐ Value	Description
☐ M_MODEL	Draws the active edges of the model transformed at the occurrence position. This is the default value. (summarize)
☐ M_TARGET	Draws the edges of the target in the region of the occurrence. If the Index parameter is set to M_GENERAL , then all the edges of the target are drawn. To draw the edges of the target, the find operation must have been performed with M_SAVE_TARGET_EDGES set to M_ENABLE . (summarize)

Index

Specifies the index of the model in the specified Model Finder context, or of the result occurrence. User labels cannot be used as indices; to retrieve the index of a model from its user label, use [MmodInquire\(\)](#) with [M_INDEX_FROM_LABEL](#).

This parameter should be set to one of the following values:

● For specifying the index of the model or of the result occurrence	
☐ Value	Description
☐ M_DEFAULT	Refers to index 0 for models and M_ALL for results.
☐ M_ALL	Draws the specified feature of all models or all occurrences.
☐ M_GENERAL	Draws the specified feature of the entire target. This setting can only be used with M_DRAW_EDGES + M_TARGET . (summarize)
☐ Value > = 0	Specifies the model index or the result occurrence.

ControlFlag

Specifies where in the destination image buffer to draw.

● For specifying where to draw	
☐ Value	Description
☐ M_DEFAULT	Draws at the top-left corner of the destination image buffer. When drawing features of a result occurrence, this parameter must be set to M_DEFAULT . (summarize)
☐ M_ORIGINAL	Draws at the offsets used to define the model region in the model source. If M_ORIGINAL is selected, the drawing is done at the original position of the model. M_ORIGINAL is only supported for image-type or Edge Finder-type models. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodFind

Synopsis

Search for the model(s) of the specified Model Finder context in a target image buffer or Edge Finder result buffer.

Syntax

```
void MmodFind(
    MIL_ID ContextId,
    MIL_ID TargetImageOrEdgeResultId,
    MIL_ID ModResultId
)
```

Description

This function searches for the model(s) defined in the specified Model Finder context, in a target image or an Edge Finder result buffer, and writes the results in the provided result buffer. When an Edge Finder result buffer is specified, **MmodFind()** will search for the models only in the included edges in the Edge Finder result buffer. The search is performed using both the global search settings of the Model Finder context and each model's current search settings (see **MmodControl()**). The resulting values can be read with the **MmodGetResult()** function.

If an occurrence has both a score and target score greater than or equal to its model's respective certainty levels (specified using **MmodControl()** with **M_CERTAINTY** and **M_CERTAINTY_TARGET**), it is automatically considered an occurrence (default 80%); the remaining occurrences will be the best of those greater than or equal to the model's acceptance levels (specified using **MmodControl()** with **M_ACCEPTANCE** and **M_ACCEPTANCE_TARGET**).

The maximum number of occurrences found depends on the maximum number specified for the context (specified using **MmodControl()** with **Index** set to **M_CONTEXT** and **ControlType** set to **M_NUMBER**). If it is set to **M_ALL**, **MmodFind()** tries to find the maximum number of occurrences specified for each model (specified using **MmodControl()** with **Index** set to an index value and **ControlType** set to **M_NUMBER**), greater than or equal to the acceptance levels.

Note, the Model Finder context must be preprocessed (**MmodPreprocess()**) before calling this function.

If you are using a Model Finder context whose models are calibrated, the target image or Edge Finder results must also be calibrated. Conversely, if the models are uncalibrated, the target must be uncalibrated as well.

If the target image or Edge Finder results are calibrated, results are calculated in the world coordinate system, otherwise they are calculated in the image coordinate system.

If the target image or Edge Finder results are calibrated, you can retrieve results in real-world or in pixel units. However, in the presence of distortion, some results are meaningless when converted from real-world to pixel units (for example, angle, scale, and transformation coefficient results). For example, if a model appears warped in the target image, but the calibration object of the target image compensates for this during the model search, the resulting angle is meaningful in the real-world coordinate system, and meaningless in the image coordinate system.

If the target image or Edge Finder results are not calibrated and the nominal search aspect ratio is not set to 1 (**MmodControl()** with **M_ASPECT_RATIO**), the nominal search aspect ratio is taken into account horizontally when performing the search, adjusting the target pixel width to equal the pixel height (the model's aspect ratio is not adjusted). As such, results will be returned for this corrected target coordinate system.

If you are searching for model(s) in an Edge Finder result buffer, you can speed up the search by setting **M_EXTRACTION_SCALE** in **MedgeControl()** prior to calling **MedgeCalculate()**. For more information on speeding up your search, see the Interfacing with the Geometric Model Finder module section in Chapter 9: Edge Finder.

Parameters

ContextId

Specifies the Model Finder context to use for the search. The Model Finder context must have been previously allocated on the system using **MmodAlloc()**.

TargetImageOrEdgeResultId

Specifies the target image or Edge Finder result buffer in which to search for the models.

Target images must be 1-band 8-bit unsigned images. The minimum and maximum target image sizes are 16x16 and 32768x32768 pixels, respectively. It is important to note that the minimum and maximum target image sizes are MIL limits. To make sure that you have enough memory, you can call **MmodFindQ()**; if you don't have enough memory, you will get a MIL error.

The Edge Finder result buffer must have been previously allocated on the required system using **MedgeAllocResult()**; **MedgeCalculate()** must have already been called. In addition, the Edge Finder result buffer must be compatible with

the Model Finder context. Enable compatibility using `MedgeControl()` with `M_MODEL_FINDER_COMPATIBLE` prior to calling `MedgeCalculate()`.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

If an Edge Finder result buffer is specified, then this result buffer must be allocated on the same system as the Model Finder context (`ContextId`). If it is not, an error will occur.

`ModResultId`

Specifies the result buffer in which to write the results of the search.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The Model Finder result buffer (`ModResultId`) must be allocated on the same system as the Model Finder context (`ContextId`). If it is not, an error will occur.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodFree

Synopsis

Free a Model Finder context or a result buffer.

Syntax

```
void MmodFree(
    MIL_ID ObjectId
)
```

Description

This function deletes the specified Model Finder context (and all its models) or result buffer identifier, and releases any memory associated with it. Note that to only delete individual models in the context, use [MmodDefine\(\)](#).

All Model Finder contexts and all result buffers allocated on a particular system must be freed before the system can be freed.

Parameter

`ObjectId`

Specifies the identifier of the Model Finder context or result buffer to free. These must have been successfully allocated (with [MmodAlloc\(\)](#) or [MmodAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodGetResult

Synopsis

Get the specified type of result(s) from a Model Finder result buffer.

Syntax

```
void MmodGetResult(
    MIL_ID ResultId,
    MIL_INT ResultIndex,
    MIL_INT ResultType,
    void *ResultArrayPtr
)
```

Description

This function retrieves the result(s) of the specified type from a Model Finder result buffer. Results are only available after calling [MmodFind\(\)](#).

The result entries will be ordered by match score, starting with the highest score (regardless of the result type requested). When searching for multiple models, results are not sorted according to which model occurrence was found; results are still sorted according to score. The [M_INDEX](#) result type returns the index of the model associated with the Model Finder result; whereas the [M_USER_LABEL](#) result type returns its user label.

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image.

If the target is not calibrated and the nominal search aspect ratio is not set to 1 ([MmodControl\(\)](#) with [M_ASPECT_RATIO](#)), results are returned for the corrected target coordinate system (whereby the target pixel width is equal to the pixel height).

All positional results are relative to the center of the top-left pixel in the target or the origin of the world-coordinate if the target is calibrated.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [ResultArrayPtr](#) to [M_NULL](#). When this parameter is set to [M_NULL](#), the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [MmodGetResult\(\)](#) again and you pass an array to the parameter [ResultArrayPtr](#). You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

By setting the [ResultType](#) parameter to [M_INTERACTIVE](#), you can view the results interactively.

Parameters

ResultId

Specifies the identifier of the Model Finder result buffer from which to retrieve results.

ResultIndex

Specifies where to get results. This parameter can be set to one of the following:

● For specifying where to get results	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ALL for all occurrences. Same as M_GENERAL for general results. (summarize)
<input type="checkbox"/> 0 to M_NUMBER-1	Specifies the result index.
<input type="checkbox"/> M_ALL	Retrieves the results of the specified type for all model occurrences found.

<input type="checkbox"/> M_GENERAL	Retrieves a result relating to the entire Model Finder context.
------------------------------------	-----------------------------------------------------------------

ResultType

Specifies the type of result to retrieve or opens an interactive dialog box that displays the results stored in the result buffer, interactively.

To display the results currently stored in the result buffer in an interactive dialog box, select the following.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter `M_NULL`.

● For opening an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	<p>[This is only applicable to Windows]</p> <p>Opens a dialog box containing a spreadsheet that displays the types of results stored in the result buffer. The results are displayed in separate columns. If there are queued requests, only those results are displayed.</p> <p>(summarize)</p>

When retrieving a general result, the [ResultType](#) parameter can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a `MIL_DOUBLE`.

● For general results					
<input type="checkbox"/> Value	Description				
<input type="checkbox"/> M_CONTEXT_ID +	<p>Returns the identifier of the Model Finder context used by MmodFind() to obtain the results in the result buffer. Note that this identifier might not be valid if the Model Finder context has been freed using MmodFree().</p> <p>(summarize)</p>				
<input type="checkbox"/> M_MOD_DEFINE_COMPATIBLE +	<p>Returns whether the result can be used to define a model.</p> <p>(summarize)</p> <div> <p><i>ResultArrayPtr info</i></p> <p>Return values:</p> <table> <tr> <td>M_FALSE</td><td>Returned if the result cannot be used to define a model.</td></tr> <tr> <td>M_TRUE</td><td>Returned if the result can be used to define a model.</td></tr> </table> </div>	M_FALSE	Returned if the result cannot be used to define a model.	M_TRUE	Returned if the result can be used to define a model.
M_FALSE	Returned if the result cannot be used to define a model.				
M_TRUE	Returned if the result can be used to define a model.				
<input type="checkbox"/> M_NUMBER +	Returns the number of occurrences found (all models).				
<input type="checkbox"/> M_TIMEOUT_END +	<p>Returns whether the timeout has been reached. You can set the timeout limit using MmodControl() with M_TIMEOUT. By default, there is no limit.</p> <p>(summarize)</p> <div> <p><i>ResultArrayPtr info</i></p> <p>Return values:</p> <table> <tr> <td>M_FALSE</td><td>Returned if the timeout has not been reached.</td></tr> <tr> <td>M_TRUE</td><td>Returned if the timeout has been reached.</td></tr> </table> </div>	M_FALSE	Returned if the timeout has not been reached.	M_TRUE	Returned if the timeout has been reached.
M_FALSE	Returned if the timeout has not been reached.				
M_TRUE	Returned if the timeout has been reached.				

When retrieving a general result, a result for an individual occurrence, or a result for all occurrences (**M_ALL**), the [ResultType](#) parameter can be set to the following value.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a `MIL_DOUBLE` (when retrieving an individual occurrence or a general result) or the address of an array of type `MIL_DOUBLE` with a size equal to the number of occurrences found; this number can be obtained using [MmodGetResult\(\)](#) with [M_NUMBER](#) (when retrieving all results).

● For an individual occurrence or a general result	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NUMBER_OF_CHAINED_EDGELS +	Returns the total number of chained edgels.

For a general result, [M_NUMBER_OF_CHAINED_EDGELS](#) returns the number of chained edgels in the target.

For a specified occurrence, [M_NUMBER_OF_CHAINED_EDGELS](#) returns the chained edgels in the occurrence. When used with [M_ALL](#), an array of results is returned, with one result per occurrence.

You can use [M_NUMBER_OF_CHAINED_EDGELS](#) to determine the required array size for the **M_CHAIN_...** result types.

[M_NUMBER_OF_CHAINED_EDGELS](#) is not supported for a general result if the Model Finder context uses a controlled geometric search algorithm.

(summarize)

When retrieving a general result or result(s) for a single occurrence, the [ResultType](#) parameter can be set to one of the following values.

When retrieving a general result, the following values are not supported if the Model Finder context is an [M_GEOMETRIC_CONTROLLED](#) type of context.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to the number of chained edgels. This number can be obtained using [MmodGetResult\(\)](#) with [M_NUMBER_OF_CHAINED_EDGELS](#).

● For a general result for a single occurrence	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CHAIN_ANGLE +	Retrieves the angle values of each edgel in all chains in the entire target or in the model occurrence.
<input type="checkbox"/> M_CHAIN_INDEX +	Retrieves the chain index of each edgel in all chains in the entire target, or in the model occurrence. The first chain in the target is identified as index 1, with increasing indices for subsequent chains. (summarize)
<input type="checkbox"/> M_CHAIN_X +	Retrieves the X-coordinate of each edgel in all chains in the entire target or in the model occurrence.
<input type="checkbox"/> M_CHAIN_Y +	Retrieves the Y-coordinate of each edgel in all chains in the entire target or in the model occurrence.

Combination constants for [M_NUMBER_OF_CHAINED_EDGELS](#); [M_CHAIN_INDEX](#); [M_CHAIN_X](#); [M_CHAIN_Y](#);

You can add one of the following values to the above-mentioned values to get the results for a specific occurrence.

● For M_NUMBER_OF_CHAINED_EDGELS, M_CHAIN_INDEX, M_CHAIN_X, or M_CHAIN_Y	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MODEL	Retrieves the edge information of the model transformed at the occurrence position.
<input type="checkbox"/> M_TARGET	Retrieves the edge information of the target in the region of the occurrence.

When retrieving a result for a single occurrence or for all occurrences, the [ResultType](#) parameter can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a `MIL_DOUBLE` (when retrieving an individual occurrence) or the address of an array of type `MIL_DOUBLE` with a size equal to the number of occurrences found; this number can be obtained using [MmodGetResult\(\)](#) with [M_NUMBER](#) (when retrieving all results).

● For a result for a single occurrence or all occurrences	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ANGLE +	Retrieves the angle of the occurrence, relative to the reference axis angle specified for the model (MmodControl() with M_REFERENCE_ANGLE).
<input type="checkbox"/> M_ASPECT_RATIO +	Retrieves the found aspect ratio of all occurrences.
<input type="checkbox"/> M_FIT_ERROR +	Retrieves the fit error as the average quadratic distance, in pixels or calibrated units, between the edges in the occurrence and the corresponding edges in the model. A perfect fit is represented as 0.0. (summarize)
<input type="checkbox"/> M_INDEX +	Retrieves the index of the model that was found.
<input type="checkbox"/> M_MODEL_COVERAGE +	Retrieves the model coverage of the occurrence. The model coverage is the percentage of the total length of the model's active edges found in the occurrence. 100% indicates that for every active edge in the model, a corresponding edge was found in the occurrence.

	Note that using a weighted mask with a model affects the calculation of the model coverage. To view the modified equation, refer to the Determining what is a match section in Chapter 8: Geometric Model Finder . (summarize)
<input type="checkbox"/> M_POLARITY +	Retrieves the polarity of the occurrence. (summarize)
	<i>ResultArrayPtr info</i> Return values: M_ANY; M_DEFAULT; M_REVERSE; M_SAME; M_SAME_OR_REVERSE; (details)
<input type="checkbox"/> M_POSITION_X +	Retrieves the X-coordinate of the occurrence. This is the X-position of the model's reference axis transformed at the occurrence (MmodControl() with M_REFERENCE_X). (summarize)
<input type="checkbox"/> M_POSITION_Y +	Retrieves the Y-coordinate of the occurrence. This is the Y-position of the model's reference axis transformed at the occurrence (MmodControl() with M_REFERENCE_Y). (summarize)
<input type="checkbox"/> M_SCALE +	Retrieves the scale of the occurrence.
<input type="checkbox"/> M_SCORE +	Retrieves the score of the occurrence (as a percentage). The score is calculated as follows: Score = Model coverage x (1 - Fit error weighting factor * Normalized fit error) . Note that the normalized fit error is the fit error converted to a number between 0.0 and 1.0. (summarize)
<input type="checkbox"/> M_SCORE_TARGET +	Retrieves the target score of the occurrence. The target score is calculated as follows: Target Score = Target coverage x (1 - Fit error weighting factor * Normalized fit error) . Note that the normalized fit error is the fit error converted to a number between 0.0 and 1.0. (summarize)
<input type="checkbox"/> M_TARGET_COVERAGE +	Retrieves the target coverage of the model. The target coverage is a percentage of the total length of edges present within the occurrence's bounding box, corresponding to the model's active edges. Lower scores indicate that features or edges found in the target (result occurrence) are not present in the model. Note that using a weighted mask with a model affects the calculation of the target coverage. To view the modified equation, refer to the Determining what is a match section in Chapter 8: Geometric Model Finder . (summarize)
<input type="checkbox"/> M_USER_LABEL +	Retrieves the user label of the model that was found.

When retrieving a result for a single occurrence or for all occurrences, the [ResultType](#) parameter can be set to one of the following values. The following values are not available for all model types. For one of these values to be available for a result occurrence, the result occurrence must be of the appropriate type. Use [M_AVAILABLE](#) to ensure that the value can be used with the occurrence. If the [ResultIndex](#) parameter is set to [M_ALL](#), all the occurrences must support the result type, otherwise an error is generated.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a [MIL_DOUBLE](#) (when retrieving an individual occurrence) or the address of an array of type [MIL_DOUBLE](#) with a size equal to the number of occurrences found; this number can be obtained using [MmodGetResult\(\)](#) with [M_NUMBER](#) (when retrieving all results).

● For a result for a single occurrence or all occurrences (with constraints)	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CORNER_RADIUS +	Retrieves the radius used to round the corners of the models. This control type is only valid for models of type M_RECTANGLE , M_SQUARE , M_DIAMOND , M_TRIANGLE , and M_CROSS . (summarize)
<input type="checkbox"/> M_HEIGHT +	Retrieves the height of the shape. This result type is only available for M_CROSS , M_DIAMOND , M_ELLIPSE , M_RECTANGLE and M_TRIANGLE . (summarize)
<input type="checkbox"/> M_HORIZONTAL_THICKNESS +	Retrieves the horizontal thickness of the cross. This result type is only available for M_CROSS . (summarize)
<input type="checkbox"/> M_INNER_RADIUS +	Retrieves the inner radius of the ring. This result type is only available for M_RING . (summarize)
<input type="checkbox"/> M_OUTER_RADIUS +	Retrieves the outer radius of the ring. This result type is only available for M_RING . (summarize)
<input type="checkbox"/> M_RADIUS +	Retrieves the radius of the circle. This result type is only available for M_CIRCLE .

	(summarize)
<input type="checkbox"/> M_VERTICAL_THICKNESS +	Retrieves the vertical thickness of the cross. This result type is only available for M_CROSS . (summarize)
<input type="checkbox"/> M_WIDTH +	Retrieves the width of the shape. This result type is only available for M_CROSS , M_DIAMOND , M_ELLIPSE , M_RECTANGLE , M_SQUARE and M_TRIANGLE . (summarize)

To retrieve a transformation coefficient for a single occurrence or for all occurrences, the [ResultType](#) parameter can be set to one of the following values. These coefficients allow you to convert coordinates in the model coordinate system to the corresponding coordinates in the target coordinate system for that occurrence (or vice versa). These coefficients handle variations in scale, translation, and angle. If the model is calibrated, these coefficients are given for the real world.

Use the following equations:

$$x_d = ax_s + by_s + c$$
$$y_d = -bx_s + ay_s + d$$

where a, b, c, and d are the transformation coefficients (forward or reverse); x_s and y_s specify the source coordinates (with respect to the origin of the model coordinate system for a forward transformation or target coordinate system for a reverse transformation); and, x_d and y_d are the destination coordinates (with respect to the origin of the target coordinate system for a forward transformation or model coordinate system for a reverse transformation).

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a `MIL_DOUBLE` (when retrieving an individual occurrence) or the address of an array of type `MIL_DOUBLE` with a size equal to the number of occurrences found; this number can be obtained using [MmodGetResult\(\)](#) with [M_NUMBER](#) (when retrieving all results).

● For retrieving a transformation coefficient	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_A_FORWARD +	Retrieves the forward transformation coefficient A for the occurrence.
<input type="checkbox"/> M_A_REVERSE +	Retrieves the reverse transformation coefficient A for the occurrence.
<input type="checkbox"/> M_B_FORWARD +	Retrieves the forward transformation coefficient B for the occurrence.
<input type="checkbox"/> M_B_REVERSE +	Retrieves the reverse transformation coefficient B for the occurrence.
<input type="checkbox"/> M_C_FORWARD +	Retrieves the forward transformation coefficient C for the occurrence.
<input type="checkbox"/> M_C_REVERSE +	Retrieves the reverse transformation coefficient C for the occurrence.
<input type="checkbox"/> M_D_FORWARD +	Retrieves the forward transformation coefficient D for the occurrence.
<input type="checkbox"/> M_D_REVERSE +	Retrieves the reverse transformation coefficient D for the occurrence.

Combination constant for [the values listed in](#) all tables **except For opening an interactive dialog box**

You can add the following value to the above-mentioned values to specify whether a result is available.

● For a general result or occurrence	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_AVAILABLE	Returns whether a result is available to be retrieved for a general result, an occurrence, or for all occurrences. (summarize)
	<div><div>ResultType info</div><div>Return values:</div><div><div>M_NULL</div><div>The result is not available to be retrieved.</div></div><div><div>Value != 0</div><div>The result is available to be retrieved.</div></div></div>

Combination constants for [the values listed in](#) all tables **except For opening an interactive dialog box**

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

● For specifying the data type	
▢ Value	Description
▢ <code>M_TYPE_DOUBLE</code>	<p>Casts the requested results to a <i>double</i>. This is the default value. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>double</i> Array size: Depends on the result type. Note: When multiple results. • Data type: <i>double</i> Note: When a single result.
▢ <code>M_TYPE_LONG</code>	<p>Casts the requested results to a <i>long</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>long</i> Array size: Depends on the result type. Note: When multiple results. • Data type: <i>long</i> Note: When a single result.
▢ <code>M_TYPE_MIL_DOUBLE</code>	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>MIL_DOUBLE</i> Array size: Depends on the result type. Note: When multiple results. • Data type: <i>MIL_DOUBLE</i> Note: When a single result.
▢ <code>M_TYPE_MIL_ID</code>	<p>Casts the requested results to a <i>MIL ID</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>MIL_ID</i> Array size: Depends on the result type. Note: When multiple results. • Data type: <i>MIL_ID</i> Note: When a single result.
▢ <code>M_TYPE_MIL_INT</code>	<p>Casts the requested results to a <i>MIL_INT</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>MIL_INT</i> Array size: Depends on the result type. Note: When multiple results. • Data type: <i>MIL_INT</i> Note: When a single result.
▢ <code>M_TYPE_MIL_INT32</code>	<p>Casts the requested results to a <i>MIL_INT32</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <i>MIL_INT32</i> Array size: Depends on the result type. Note: When multiple results. • Data type: <i>MIL_INT32</i> Note: When a single result.
▢ <code>M_TYPE_MIL_INT64</code>	<p>Casts the requested results to a <i>MIL_INT64</i>. (summarize)</p>

	<i>ResultArrayPtr info</i> <ul style="list-style-type: none">• Data type: array of type MIL_INT64 Array size: Depends on the result type. Note: When multiple results.• Data type: MIL_INT64 Note: When a single result.
--	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• array of type double• array of type long• array of type MIL_DOUBLE• array of type MIL_ID• array of type MIL_INT• array of type MIL_INT32• array of type MIL_INT64• double• long• M_NULL• MIL_DOUBLE• MIL_ID• MIL_INT• MIL_INT32• MIL_INT64

Specifies the address in which to write results.

Set this parameter to **M_NULL** if you are setting the [ResultType](#) parameter to **M_INTERACTIVE** or if you are putting the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodInquire

Synopsis

Inquire information about a specified Model Finder context, a specified model, or a specified result buffer.

Syntax

```
MIL_INT MmodInquire(
    MIL_ID ContextOrResultId,
    MIL_INT Index,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires information about a specified Model Finder context, the models contained therein, or a specified Model Finder result buffer. Note that Model Finder result buffer values can be obtained with [MmodGetResult\(\)](#).

If the inquired setting is set to **M_DEFAULT**, **MmodInquire()** will return **M_DEFAULT**. To inquire its default value, add **M_DEFAULT** to the inquire type.

By setting the **InquireType** parameter to **M_INTERACTIVE**, you can view the setting of inquire types interactively.

Parameters

ContextOrResultId

Specifies the Model Finder context or Model Finder result buffer about which to inquire information. The Model Finder context or result buffer must have been previously allocated on the required system using [MmodAlloc\(\)](#) or [MmodAllocResult\(\)](#), respectively.

Index

Specifies that information will be inquired about the Model Finder context, an individual model, or a Model Finder result buffer. This parameter should be set to one of the following values:

● For specifying a context, model, or result buffer	
☐ Value	Description
☐ M_DEFAULT	Specifies the default. If a Model Finder context is specified, this parameter inquires information about model index 0. If a result buffer is specified, same as M_GENERAL . (summarize)
☐ M_CONTEXT	Inquires information about a Model Finder context, if one is specified.
☐ M_GENERAL	Inquires general information about the Model Finder result buffer, if one is specified.
☐ Value	Inquires information about a model, if a Model Finder context is specified. Set the Index parameter to the index of the required model. To retrieve the index of a model from its user label, set this parameter to the user label and set the inquire type to M_INDEX_FROM_LABEL . (summarize)

InquireType

Specifies the type of setting about which to inquire.

For **M_CONTEXT** or individual model inquiries, the **InquireType** parameter can be set to one of the following:

*Unless otherwise specified, the following values require that you pass the **UserVarPtr** parameter **M_NULL**.*

● For opening an interactive dialog box	
☐ Value	Description
☐ M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens a read-only dialog box that displays the settings of the inquire types.</p> <p>For M_CONTEXT, opens a read-only dialog box that displays the setting of all the inquire types of the specified Model Finder context.</p> <p>For a model, opens a read-only dialog box that displays the setting of the model's inquire types. The model is specified by the Index parameter. Once the dialog box is opened you can change between models using the Model Index field.</p> <p>(summarize)</p>

For **M_CONTEXT** or individual model inquiries, the **InquireType** parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the **UserVarPtr** parameter the address of a **MIL_DOUBLE**.

● For returning the number of models to find	
☐ Value	Description
☐ M_NUMBER +	<p>Returns the number of models for which to search.</p> <p>For M_CONTEXT, this setting returns the maximum number of occurrences of all models to find in the target.</p> <p>For a model, this setting returns the maximum number of occurrences of the specified model to find in the target.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Return values: M_ALL; M_DEFAULT; Value; (details)</p>

The possible settings to inquire for **M_CONTEXT** are:

Unless otherwise specified, the following values require that you pass the **UserVarPtr** parameter the address of a **MIL_DOUBLE**.

● For M_CONTEXT	
☐ Value	Description
☐ M_ACCURACY +	<p>Returns the search accuracy.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Return values: M_DEFAULT; M_HIGH; M_MEDIUM; (details)</p>
☐ M_ASPECT_RATIO +	<p>Returns the nominal search aspect ratio.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Return values: 0.5 to 2.0; M_DEFAULT; (details)</p>
☐ M_CONTEXT_TYPE +	<p>Returns the type of search context allocated.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Return values: M_GEOMETRIC; M_GEOMETRIC_CONTROLLED; (details)</p>
☐ M_DETAIL_LEVEL +	<p>Returns the level of details to extract from the model source and target image.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p> <p>Return values: M_DEFAULT; M_HIGH; M_MEDIUM; M_VERY_HIGH; (details)</p>
☐ M_FILTER_MODE +	<p>Returns the mode in which to perform the edge extraction filter.</p> <p>(summarize)</p>
	<p><i>UserVarPtr info</i></p>

	Return values: M_DEFAULT; M_KERNEL; M_RECURSIVE; (details)
<input type="checkbox"/> M_FIRST_LEVEL +	Returns the resolution level for the initial stage of the search. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 7; M_AUTO; M_DEFAULT; (details)
<input type="checkbox"/> M_KERNEL_DEPTH +	Returns the depth of the convolution kernel used for kernel filtering mode. (summarize)
	<i>UserVarPtr info</i> Return values: 8; 16; M_DEFAULT; (details)
<input type="checkbox"/> M_KERNEL_WIDTH +	Returns the maximum size (same in X and Y) of the convolution kernel set for kernel filtering mode. (summarize)
	<i>UserVarPtr info</i> Return values: M_AUTO; M_DEFAULT; Value >= 3; (details)
<input type="checkbox"/> M_LAST_LEVEL +	Returns the resolution level for the final stage of the search. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 7; M_AUTO; M_DEFAULT; (details)
<input type="checkbox"/> M_MODIFICATION_COUNT +	Returns the current value of the modification counter. The modification counter is increased by one each time settings for the context are modified. Although you cannot identify the modification counter's contents, you can compare them throughout your application to know if the context has been altered. If the modification counter has changed you can, for example, prompt the user to save before closing the application. (summarize)
<input type="checkbox"/> M_NUMBER_MODELS +	Returns the number of models in the context.
<input type="checkbox"/> M_OWNER_SYSTEM +	Returns the identifier of the system on which the context was allocated.
<input type="checkbox"/> M_PREPROCESSED +	Returns whether the Model Finder context is preprocessed. The context must be preprocessed before calling MmodFind() . This inquire type will indicate that the context is not preprocessed after certain settings of the Model Finder context are changed with MmodControl() , and after a model is added or removed with MmodDefine() . (summarize)
	<i>UserVarPtr info</i> Return values: Value!= 0 The context is preprocessed. 0 The context is not preprocessed.
<input type="checkbox"/> M_SAVE_TARGET_EDGES +	Returns whether the target edges in the result will be saved. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SEARCH_ANGLE_RANGE +	Returns whether to perform calculations specific to angular-range search strategies. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SEARCH_POSITION_RANGE +	Returns whether to perform calculations specific to position-range search strategies. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SEARCH_SCALE_RANGE +	Returns whether to perform calculations specific to scale-range search strategies. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)

<input type="checkbox"/> M_SHARED_EDGES +	<p>Returns whether multiple occurrences can share edges. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_SMOOTHNESS +	<p>Returns the degree of smoothness (denoising) applied to the model images and the target images during edge extraction. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>
<input type="checkbox"/> M_SPEED +	<p>Returns the search speed. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_HIGH; M_LOW; M_MEDIUM; M_VERY_HIGH; (details)</p>
<input type="checkbox"/> M_TARGET_CACHING +	<p>Returns whether target caching is enabled. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_TIMEOUT +	<p>Returns the maximum search time for MmodFind(), in msec. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; Value; (details)</p>

The possible settings to inquire for an individual model are:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For models					
<input type="checkbox"/> Value	Description				
<input type="checkbox"/> M_ACCEPTANCE +	<p>Returns the acceptance level for the score. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>				
<input type="checkbox"/> M_ACCEPTANCE_TARGET +	<p>Returns the acceptance level for the target score. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>				
<input type="checkbox"/> M_ALLOC_OFFSET_X +	<p>Returns the X-offset, in pixels, of the model in the model source image. This is only supported for image-type or Edge Finder-type models. For a synthetic model, use M_BOX_OFFSET_X instead. (summarize)</p>				
<input type="checkbox"/> M_ALLOC_OFFSET_Y +	<p>Returns the Y-offset, in pixels, of the model in the model source image. This is only supported for image-type or Edge Finder-type models. For a synthetic model, use M_BOX_OFFSET_Y instead. (summarize)</p>				
<input type="checkbox"/> M_ALLOC_SIGN +	<p>Returns the data range of the model image. (summarize)</p> <p><i>UserVarPtr info</i> Return values:</p> <table> <tr> <td>M_SIGNED</td><td>Data range is signed.</td></tr> <tr> <td>M_UNSIGNED</td><td>Data range is unsigned. Synthetic, Model Finder-type and merge-type models will always return M_UNSIGNED.</td></tr> </table>	M_SIGNED	Data range is signed.	M_UNSIGNED	Data range is unsigned. Synthetic, Model Finder-type and merge-type models will always return M_UNSIGNED.
M_SIGNED	Data range is signed.				
M_UNSIGNED	Data range is unsigned. Synthetic, Model Finder-type and merge-type models will always return M_UNSIGNED.				

M_ALLOC_SIZE_BAND +	<p>Returns the number of bands of the model image. When drawing the model image, use M_ALLOC_SIZE_BAND to establish the required number of bands for the destination buffer.</p> <p>(summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <table> <tr> <td>1</td><td>1-band buffer.</td></tr> <tr> <td>3</td><td>3-band buffer.</td></tr> </table> </div>	1	1-band buffer.	3	3-band buffer.												
1	1-band buffer.																
3	3-band buffer.																
M_ALLOC_SIZE_BIT +	<p>Returns the depth per band, in bits, of the model image.</p> <p>(summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <table> <tr> <td>1</td><td>1-bit depth.</td></tr> <tr> <td>8</td><td>8-bit depth.</td></tr> <tr> <td>16</td><td>16-bit depth.</td></tr> <tr> <td>32</td><td>32-bit depth.</td></tr> </table> </div>	1	1-bit depth.	8	8-bit depth.	16	16-bit depth.	32	32-bit depth.								
1	1-bit depth.																
8	8-bit depth.																
16	16-bit depth.																
32	32-bit depth.																
M_ALLOC_SIZE_X +	<p>Returns the width of the model box, in pixels.</p> <p>For a synthetic model, use M_BOX_SIZE_X to get this value in user-defined units.</p> <p>(summarize)</p>																
M_ALLOC_SIZE_Y +	<p>Returns the height of the model box, in pixels.</p> <p>For a synthetic model, use M_BOX_SIZE_Y to get this value in user-defined units.</p> <p>(summarize)</p>																
M_ALLOC_TYPE +	<p>Returns the data type and depth of the model image.</p> <p>(summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values:</div> <table> <tr> <td>1 + M_UNSIGNED</td><td>1-bit unsigned data.</td></tr> <tr> <td>8 + M_SIGNED</td><td>8-bit signed data.</td></tr> <tr> <td>8 + M_UNSIGNED</td><td>8-bit unsigned data.</td></tr> <tr> <td>16 + M_SIGNED</td><td>16-bit signed data.</td></tr> <tr> <td>16 + M_UNSIGNED</td><td>16-bit unsigned data.</td></tr> <tr> <td>32 + M_FLOAT</td><td>32-bit float data.</td></tr> <tr> <td>32 + M_SIGNED</td><td>32-bit signed data.</td></tr> <tr> <td>32 + M_UNSIGNED</td><td>32-bit unsigned data.</td></tr> </table> </div>	1 + M_UNSIGNED	1-bit unsigned data.	8 + M_SIGNED	8-bit signed data.	8 + M_UNSIGNED	8-bit unsigned data.	16 + M_SIGNED	16-bit signed data.	16 + M_UNSIGNED	16-bit unsigned data.	32 + M_FLOAT	32-bit float data.	32 + M_SIGNED	32-bit signed data.	32 + M_UNSIGNED	32-bit unsigned data.
1 + M_UNSIGNED	1-bit unsigned data.																
8 + M_SIGNED	8-bit signed data.																
8 + M_UNSIGNED	8-bit unsigned data.																
16 + M_SIGNED	16-bit signed data.																
16 + M_UNSIGNED	16-bit unsigned data.																
32 + M_FLOAT	32-bit float data.																
32 + M_SIGNED	32-bit signed data.																
32 + M_UNSIGNED	32-bit unsigned data.																
M_ANGLE +	<p>Returns the nominal search angle, relative to the model reference axis angle.</p> <p>(summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 360.0; M_DEFAULT; (details)</div> </div>																
M_ANGLE_DELTA_NEG +	<p>Returns the lower limit of the angular range, relative to the nominal search angle.</p> <p>(summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 180.0; M_DEFAULT; (details)</div> </div>																
M_ANGLE_DELTA_POS +	<p>Returns the upper limit of the angular range, relative to the nominal search angle.</p> <p>(summarize)</p> <div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 180.0; M_DEFAULT; (details)</div> </div>																
M_ASSOCIATED_CALIBRATION +	<p>Returns the calibration object associated with the specified model.</p>																

	(summarize)
	<i>UserVarPtr info</i> Return values: MIL Calibration object identifier; M_DEFAULT; M_NULL; (details)
<input type="checkbox"/> M_BOX_MARGIN_BOTTOM +	Returns the margin at the bottom of the bounding box of the model's active edges, in the user-defined units for the model. This is only valid for synthetic models. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value >= 0.0; (details)
<input type="checkbox"/> M_BOX_MARGIN_LEFT +	Returns the margin at the left of the bounding box of the model's active edges, in the user-defined units for the model. This is only valid for synthetic models. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value >= 0.0; (details)
<input type="checkbox"/> M_BOX_MARGIN_RIGHT +	Returns the margin at the right of the bounding box of the model's active edges, in the user-defined units for the model. This is only valid for synthetic models. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value >= 0.0; (details)
<input type="checkbox"/> M_BOX_MARGIN_TOP +	Returns the margin at the top of the bounding box of the model's active edges, in the user-defined units for the model. This is only valid for synthetic models. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value >= 0.0; (details)
<input type="checkbox"/> M_BOX_OFFSET_X +	Returns the X-offset of the top-left corner of the model box from the model origin. This is only valid for synthetic models. For predefined shape models, the origin of the model is the center of the shape. For CAD-file models, the origin is the same as the one defined in the CAD file. M_BOX_OFFSET_X is updated when the margins that define the model box are changed. The X-offset is returned in the units of the model. (summarize)
<input type="checkbox"/> M_BOX_OFFSET_Y +	Returns the Y-offset of the top-left corner of the model box from the model origin. This is only valid for synthetic models. For predefined shape models, the origin of the model is the center of the shape. For CAD-file models, the origin is the same as the one defined in the CAD file. M_BOX_OFFSET_Y is updated when the margins that define the model box are changed. The Y-offset is returned in the units of the model. (summarize)
<input type="checkbox"/> M_BOX_SIZE_X +	Returns the size along the X-axis of the model box, in the user-defined units for the model. This is only valid for synthetic models. (summarize)
<input type="checkbox"/> M_BOX_SIZE_Y +	Returns the size along the Y-axis of the model box, in the user-defined units for the model. This is only valid for synthetic models. (summarize)
<input type="checkbox"/> M_CAD_Y_AXIS +	Returns the direction of the Y-axis for a CAD-type model. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_FLIP; M_NO_FLIP; (details)
<input type="checkbox"/> M_CERTAINTY +	Returns the certainty level for the score. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)

M_CERTAINTY_TARGET +	<p>Returns the certainty level for the target score. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>
M_CHAIN_ANGLE +	<p>Returns the angle value of each edgel in the chains composing the model's active edges. (summarize)</p> <p><i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINED_EDGELS to determine the size of the array necessary to hold the chained edgel information.</p>
M_CHAIN_INDEX +	<p>Returns the indices which differentiate each active edge's chain of edgels, for each edgel within the model. The first chain is identified as index 1, with each subsequent chain's index incremented by 1. (summarize)</p> <p><i>UserVarPtr info</i> Data type: array of type MIL_INT Array size: Use M_NUMBER_OF_CHAINED_EDGELS to determine the size of the array necessary to hold the chained edgel information.</p>
M_CHAIN_X +	<p>Returns the X-coordinates, in pixels (with subpixel accuracy), of each edgel in the chains composing the model's active edges. Coordinates are returned for all the active edge chains contained within the model and are relative to the model origin. (summarize)</p> <p><i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINED_EDGELS to determine the size of the array necessary to hold the chained edgel information.</p>
M_CHAIN_Y +	<p>Returns the Y-coordinates, in pixels (with subpixel accuracy), of each edgel in the chains composing the model's active edges. Coordinates are returned for all the active edge chains contained within the model and are relative to the model origin. (summarize)</p> <p><i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: Use M_NUMBER_OF_CHAINED_EDGELS to determine the size of the array necessary to hold the chained edgel information.</p>
M_CORNER_RADIUS +	<p>Returns the radius used to round corners of the model. This is only valid for models of type M_RECTANGLE, M_SQUARE, M_DIAMOND, M_TRIANGLE, and M_CROSS. Attempting to call this value for any other model type will generate an error. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value >= 0.0; (details)</p>
M_FIT_ERROR_WEIGHTING_FACTOR +	<p>Returns the contribution of the fit error in the score and target score calculation. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>
M_FOREGROUND_VALUE +	<p>Returns the foreground value of the model. The foreground is the interior of the shape. This is only valid for predefined shape models. CAD-type models do not have a polarity. For these model types, M_ANY will be returned. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_ANY; M_DEFAULT; M_FOREGROUND_BLACK; M_FOREGROUND_WHITE; (details)</p>
M_HEIGHT +	<p>Returns the height of the shape in model, in the user-defined units for the model. This is only valid for a model of type M_CROSS, M_DIAMOND, M_ELLIPSE, M_RECTANGLE, and M_TRIANGLE. (summarize)</p>
M_HORIZONTAL_THICKNESS +	<p>Returns the horizontal thickness of the cross in model, in the user-defined units for the model. This is only valid for a model of type M_CROSS. (summarize)</p>
M_INNER_RADIUS +	<p>Returns the inner radius of the ring in the model, in the user-defined units for the model. This is only valid for a model of type M_RING.</p>

	(summarize)
<input type="checkbox"/> M_MIN_SEPARATION_ANGLE +	<p>Returns the minimum angular separation required for two occurrences to be considered distinct matches. This value is an absolute angle value.</p> <p>(summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 <Value<= 180.0; M_DEFAULT; M_DISABLE; (details)</p>
<input type="checkbox"/> M_MIN_SEPARATION_SCALE +	<p>Returns the minimum separation required in scale for two occurrences to be considered distinct matches. This value is a scale factor.</p> <p>(summarize)</p> <p><i>UserVarPtr info</i> Return values: 1.0 <Value<= 4.0; M_DEFAULT; M_DISABLE; (details)</p>
<input type="checkbox"/> M_MIN_SEPARATION_X +	<p>Returns the minimum separation required along the X-axis for two occurrences to be considered distinct matches. This value is a percentage of the model's width at M_SCALE.</p> <p>(summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; Value; (details)</p>
<input type="checkbox"/> M_MIN_SEPARATION_Y +	<p>Returns the minimum separation required along the Y-axis for two occurrences to be considered distinct matches. This value is a percentage of the model's height at M_SCALE.</p> <p>(summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; Value; (details)</p>
<input type="checkbox"/> M_MODEL_TYPE +	<p>Returns the type of model.</p> <p>(summarize)</p> <p><i>UserVarPtr info</i> Return values: M_AUTO_DEFINE; M_CIRCLE; M_CROSS; M_DELETE; M_DIAMOND; M_EDGE_RESULT; M_ELLIPSE; M_IMAGE; M_MERGE_MODEL; M_MOD_RESULT; M_RECTANGLE; M_RING; M_SQUARE; M_TRIANGLE;</p>
<input type="checkbox"/> M_NUMBER_OF_CHAINED_EDGELS +	Returns the total number of chained edgels in the active edges of the model.
<input type="checkbox"/> M_ORIGINAL_X +	<p>Returns the X-coordinate of the model's reference axis origin, relative to the top-left corner of the model source image. This is only valid for image-type and Edge Finder-type models.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ORIGINAL_Y +	<p>Returns the Y-coordinate of the model's reference axis origin, relative to the top-left corner of the model source image. This is only valid for image-type and Edge Finder-type models.</p>
<input type="checkbox"/> M_OUTER_RADIUS +	<p>Returns the outer radius of the ring in the model, in the user-defined units for the model.</p> <p>This is only valid for a model of type M_RING.</p> <p>(summarize)</p>
<input type="checkbox"/> M_PIXEL_SCALE +	<p>Returns the pixel scale of the model.</p> <p>This is only valid for synthetic models.</p> <p>(summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0.0; (details)</p>
<input type="checkbox"/> M_POLARITY +	<p>Returns the expected polarity of occurrences relative to the model.</p> <p>(summarize)</p> <p><i>UserVarPtr info</i> Return values: M_ANY; M_DEFAULT; M_REVERSE; M_SAME; M_SAME_OR_REVERSE; (details)</p>
<input type="checkbox"/> M_POSITION_DELTA_NEG_X +	<p>Returns the position range, in pixels, in the negative X-direction relative to the nominal search position.</p> <p>(summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_INFINITE; Value >= 0.0; (details)</p>
<input type="checkbox"/> M_POSITION_DELTA_NEG_Y +	<p>Returns the position range, in pixels, in the negative Y-direction relative to the nominal search position.</p> <p>(summarize)</p>

	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_INFINITE; Value >= 0.0; (details)
<input type="checkbox"/> M_POSITION_DELTA_POS_X +	Returns the position range, in pixels, in the positive X-direction relative to the nominal search position. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_INFINITE; Value >= 0.0; (details)
<input type="checkbox"/> M_POSITION_DELTA_POS_Y +	Returns the position range, in pixels, in the positive Y-direction relative to the nominal search position. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_INFINITE; Value >= 0.0; (details)
<input type="checkbox"/> M_POSITION_X +	Returns the nominal search position for the X-coordinate, in pixels. (summarize)
	<i>UserVarPtr info</i> Return values: M_ALL; M_DEFAULT; Value; (details)
<input type="checkbox"/> M_POSITION_Y +	Returns the nominal search position for the Y-coordinate, in pixels. (summarize)
	<i>UserVarPtr info</i> Return values: M_ALL; M_DEFAULT; Value; (details)
<input type="checkbox"/> M_RADIUS +	Returns the radius of the circle in the model, in the user-defined units for the model. This is only valid for a model of type M_CIRCLE . (summarize)
<input type="checkbox"/> M_REFERENCE_ANGLE +	Returns the reference axis angle of the model. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 360.0; M_DEFAULT; (details)
<input type="checkbox"/> M_REFERENCE_X +	Returns the X-coordinate, in pixels, of the reference axis origin of the model, relative to the model origin. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_REFERENCE_Y +	Returns the Y-coordinate, in pixels, of the reference axis origin of the model, relative to the model origin. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_SCALE +	Returns the nominal search scale. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_SCALE_MAX_FACTOR +	Returns the factor used to determine the upper limit (maximum permitted scale) of the scale range. (summarize)
	<i>UserVarPtr info</i> Return values: 1.0 to 2.0; M_DEFAULT; (details)
<input type="checkbox"/> M_SCALE_MIN_FACTOR +	Returns the factor used to determine the lower limit (minimum permitted scale) of the scale range. (summarize)
	<i>UserVarPtr info</i> Return values: 0.5 to 1.0; M_DEFAULT; (details)
<input type="checkbox"/> M_USER_LABEL +	Returns the user-defined label. (summarize)

	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_NO_LABEL; Value; (details)
<input type="checkbox"/> M_VALID +	Returns whether the model is valid or not. Note that a synthetic model is invalid if <i>size of the model box * M_PIXEL_SCALE * M_SCALE</i> is greater than 1024 pixels in the X-direction or the Y-direction. (summarize)
	<i>UserVarPtr info</i> Return values: <div> M_FALSE The model is not valid. </div> <div> M_TRUE The model is valid. </div>
<input type="checkbox"/> M_VERTICAL_THICKNESS +	Returns the vertical thickness of the cross in the model, in the user-defined units for the model. This is only valid for a model of type M_CROSS . (summarize)
<input type="checkbox"/> M_WIDTH +	Returns the width of the shape in the model, in the user-defined units for the model. This is only valid for a model of type M_CROSS , M_DIAMOND , M_ELLIPSE , M_RECTANGLE , M_SQUARE , and M_TRIANGLE . (summarize)

Combination constant for [the values listed in](#) all tables **except For opening an interactive dialog box, For a result buffer, For user label inquires**

You can add the following value to the above-mentioned values to determine the default value of an inquire type.

● For inquiring the default value of an inquire type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Returns the default value of the specified inquire type. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE

The possible settings to inquire for a model, all models, or a result buffer are:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For one or all models or a result buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_X +	Returns the X-coordinate of the top left corner of the region in the model image or result buffer that will be used when drawing in the destination image. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_Y +	Returns the Y-coordinate of the top left corner of the region in the model image or result buffer that will be used when drawing in the destination image. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_DRAW_SCALE_X +	Returns the scale in the X-direction when drawing models or results in the destination image buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)
<input type="checkbox"/> M_DRAW_SCALE_Y +	Returns the scale in the Y-direction when drawing models or results in the destination image buffer.

	(summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)

Combination constants for [the values listed in](#) all tables **except** For opening an interactive dialog box, For a result buffer, For user label inquires

You can add one of the following values to the above-mentioned values to cast the requested information to the required data type.

● For specifying the data type	
Value	Description
M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . This is the default value. (summarize)
	<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type double Array size: Depends on the inquire type. Note: When multiple inquires. • Data type: double Note: When a single inquire.
M_TYPE_LONG	Casts the requested information to a <i>long</i> . (summarize)
	<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type long Array size: Depends on the inquire type. Note: When multiple inquires. • Data type: long Note: When a single inquire.
M_TYPE_MIL_DOUBLE	Casts the requested information to a <i>MIL_DOUBLE</i> . (summarize)
	<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: Depends on the inquire type. Note: When multiple inquires. • Data type: MIL_DOUBLE Note: When a single inquire.
M_TYPE_MIL_INT	Casts the requested information to a <i>MIL_INT</i> . (summarize)
	<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: Depends on the inquire type. Note: When multiple inquires. • Data type: MIL_INT Note: When a single inquire.
M_TYPE_MIL_INT32	Casts the requested information to a <i>MIL_INT32</i> . (summarize)
	<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: Depends on the inquire type. Note: When multiple inquires. • Data type: MIL_INT32 Note: When a single inquire.
M_TYPE_MIL_INT64	Casts the requested information to a <i>MIL_INT64</i> . (summarize)
	<i>UserVarPtr info</i>

- Data type: array of type MIL_INT64
Array size: Depends on the inquire type.
Note: When multiple inquires.
- Data type: MIL_INT64
Note: When a single inquire.

The possible settings to inquire for a result buffer are:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a MIL_DOUBLE.

● For a result buffer	
☐ Value	Description
☐ M_MOD_DEFINE_COMPATIBLE +	Returns whether the information necessary to define a model from an occurrence will be saved in the result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)

Combination constant for [the values listed in](#) all tables **except** For opening an interactive dialog box, For user label inquires

You can add the following value to the above-mentioned values to determine whether a result is supported.

● For inquiring whether an inquire type is supported	
☐ Value	Description
☐ M_SUPPORTED	Returns whether the specified inquire type is supported. (summarize)
	<i>UserVarPtr info</i> Return values: Value != 0 The inquire type is available. If the index value is different from M_ALL , the inquire type is supported for the model with the appropriate index value. If the index value is M_ALL , the inquire type is supported for all models. If the index value is M_CONTEXT , the inquire type is supported for the context. 0 The inquire type is not available.

For user label inquiries (the label is specified as the index), the *InquireType* parameter can be set to the following:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a MIL_DOUBLE.

● For user label inquires	
☐ Value	Description
☐ M_INDEX_FROM_LABEL	Returns the model index associated with a model user label if the user label is used. Pass the user label as the Index parameter. (summarize)
	<i>UserVarPtr info</i> Return values: M_INVALID Invalid user label. This is returned if the specified model label is not associated with a model. Value Model index associated with the specified user label.

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type double
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- double
- long
- M_NULL
- MIL_DOUBLE
- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address in which to write the requested information. Since the **MmodInquire()** function also returns the requested information, you can set this parameter to **M_NULL**.

Return value

The return value is the required information, cast to a *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodMask

Synopsis

Mask regions of a model.

Syntax

```
void MmodMask (
    MIL_ID ContextId,
    MIL_INT Index,
    MIL_ID MaskBufferId,
    MIL_INT MaskType,
    MIL_INT ControlFlag
)
```

Description

This function allows you to apply masks to a model. Each mask can set pixels in the specified model to one of three states: a "don't care" state, a "flat region" state, or a "weighted region" state. You can apply different types of masks with the same model by making multiple calls to this function. Note that "don't care" and "flat region" masks are not supported for synthetic models.

"Don't care" pixels in a model will not be considered when searching for occurrences of the model in the target.

"Flat region" pixels denote a region where no features are expected, other than consistent pixel values with no edges present; any edge found in a "flat region" reduces the target score.

"Weighted region" pixels denote that the extracted model features will have weights to indicate their importance in the occurrence. The weight of the features will affect the calculation of the score; although the weights themselves don't affect the target score, features corresponding to negative weights will be ignored when calculating the target score.

By setting the **MaskType** parameter to [M_INTERACTIVE](#), you can set the mask interactively.

Note that if you modify the margin of a synthetic model's bounding box after a mask has been applied, the mask will be discarded since it is no longer the same size as the model.

Parameters

ContextId

Specifies the Model Finder context of the model to mask. The Model Finder context must have been previously allocated on the system using [MmodAlloc\(\)](#).

Index

Specifies the individual model for which to create a mask.

For specifying a model	
Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> Value >= 0	Specifies the index of the model for which the mask is being created. User labels cannot be used as indices; to retrieve the index of a model from its user label, use MmodInquire() with M_INDEX_FROM_LABEL . (summarize)

MaskBufferId

Specifies the identifier of the image buffer used to identify the masked pixels in the model. This buffer must be a 1-band 8-bit unsigned buffer for a "don't care" or "flat region" mask. For a "weighted region" mask, the buffer must be a 1-band 8-bit signed buffer. The buffer must be of the same size as the model. This buffer can be calibrated; however, even if the model is calibrated, the mask is applied on a pixel basis. Setting the **MaskBufferId** parameter to

M_NULL when the **MaskType** parameter is set to **M_DONT_CARES**, **M_FLAT_REGIONS**, or **M_WEIGHT_REGIONS** removes the mask.

MaskType

Specifies which type of mask to employ. Set this parameter to one of the following values:

● For specifying the type of mask	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DONT_CARES	Specifies that features present in the masked region (non-zero pixels) are ignored during the search. Features present in this region of the occurrence will not affect the scores. (summarize)
<input type="checkbox"/> M_FLAT_REGIONS	Specifies that no features are expected in the masked region (non-zero pixels). If features are present in this region of the occurrence, the target score will decrease. (summarize)
<input type="checkbox"/> M_INTERACTIVE	<i>[This is only applicable to Windows]</i> Opens a dialog box that allows you to create or modify any of the three types of masks, interactively, on top of the model. To draw the mask, select the mask type, place the cursor over the required area, click your mouse, and drag. You must set the MaskBufferId parameter to M_NULL . (summarize)
<input type="checkbox"/> M_WEIGHT_REGIONS	Specifies that the pixels corresponding to the model's active edges are weighted according to the values of the weighted region mask. The weights affect the calculation of the score; although the weights themselves don't affect the target score, edges corresponding to negative weights will be ignored when calculating the target score. The valid mask weights range from -127 to +127. Positive weights indicate how significant it is for a given pixel to be part of an edge in the occurrence; the more positive the weight, the greater the influence on the score. The absence of a pixel with a very positive weight will reduce the score more than the absence of one with a lower weight. The longer the edges with a positive weight, the more significant the edges to the target coverage and to the target score. Negative weights indicate how significant it is for a pixel not to be part of an edge in the occurrence; the more negative the weight, the greater the influence on the score. The presence of a pixel with a very negative weight will reduce the score more than the presence of one with a less negative weight. Note that weights that have no correspondence to edges in the model have no effect on the score. (summarize)

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodPreprocess

Synopsis

Preprocess a Model Finder context.

Syntax

```
void MmodPreprocess(
    MIL_ID ContextId,
    MIL_INT ControlFlag
)
```

Description

This function preprocesses the specified Model Finder context. It extracts the active edges of the models contained within the Model Finder context and sets internal search settings so that future searches will be optimized for speed and robustness. The procedure is potentially quite lengthy (up to a few seconds) and increases the size of the Model Finder context significantly.

This function must be called before performing the first [MmodFind\(\)](#) search. Call this function after all search settings have been set. When you save the Model Finder context, the model's preprocessing changes are not stored with the model. Upon restoration, the model must be preprocessed again.

Note that if some of the model's search settings are changed after a call to **MmodPreprocess()**, the model must be preprocessed again. To inquire if your model is in a preprocessed state, use [MmodInquire\(\)](#) with [M_PREPROCESSED](#) (this value must be true to perform an [MmodFind\(\)](#) search).

Parameters

- ContextId
- Specifies the Model Finder context of the model to preprocess. The Model Finder context must have been previously allocated on the system using [MmodAlloc\(\)](#).
- ControlFlag
- Specifies whether to perform or undo the preprocessing of the model context. Set this parameter to one of the following values:

● For specifying whether to perform preprocessing	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Performs the preprocessing of the model context.
<input type="checkbox"/> M_RESET	Undoes preprocessing. Undoing preprocessing can be useful if you want to conserve computer memory within an application and preserve Model Finder context settings. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodRestore

Synopsis

Restore a Model Finder context from disk.

Syntax

```
MIL_ID MmodRestore(
    MIL_CONST_TEXT_PTR Filename,
    MIL_ID SystemId,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr
)
```

Description

This function restores a Model Finder context that was previously saved to a file, using [MmodSave\(\)](#) or [MmodStream\(\)](#). This function restores all of the Model Finder context's settings that were in effect when the Model Finder context was saved. A restored Model Finder context is not preprocessed, therefore you must call [MmodPreprocess\(\)](#) before performing a search with [MmodFind\(\)](#).

If you had associated a calibration object with the model(s) in your Model Finder context, you must re-associate the calibration object, using [MmodControl\(\)](#) with [M_ASSOCIATED_CALIBRATION](#). Moreover, if you had previously set drawing control types, using [MmodControl\(\)](#) with [M_DRAW_...](#), then you must reset them since they were not saved with the context.

Parameters

Filename

Specifies the name and path of the file from which to restore the Model Finder context. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:



● For specifying the file name and path	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)</div> <div><div>Parameters</div><div>FileName</div><div>Specifies the drive, directory, and name of the file.</div></div>
<input type="checkbox"/> M_INTERACTIVE	<div>[This is only applicable to Windows] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.</div>

SystemId

Specifies the system on which to restore the Model Finder context.

This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description

 M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
 MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the Model Finder context identifier. Since this function also returns the Model Finder context identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the Model Finder context identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodSave

Synopsis

Save a Model Finder context to a file.

Syntax

```
void MmodSave(  
    MIL_CONST_TEXT_PTR FileName,  
    MIL_ID ContextId,  
    MIL_INT ControlFlag  
)
```

Description

This function saves all the information about the previously allocated Model Finder context to disk, including all of the individual models' current settings. This information can be reloaded, using [MmodRestore\(\)](#) or [MmodStream\(\)](#). However, preprocessing, any associated calibration objects, and all drawing control type settings are not saved.

Parameters

FileName

Specifies the name and path of the file in which to save the Model Finder context; it is recommended that you use the MMF file extension for easier use with other Matrox Imaging software products. The function internally handles the opening and closing of this file. If this file already exists, it will be overwritten.

This parameter can be set to one of the following values:

For specifying the file name and path		
Value	Description	
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile". (summarize)	
	Parameters	
	FileName	Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.	

ContextId

Specifies the identifier of the Model Finder context to save.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

MmodStream

Synopsis

Load, restore, or save a Model Finder context from/to a file or a memory stream.

Syntax

```
void MmodStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load or restore a Model Finder context from a file or memory stream. All of the Model Finder context's settings that were in effect when the Model Finder context was saved will be restored. A loaded or restored Model Finder context is not preprocessed, therefore you must call [MmodPreprocess\(\)](#) before performing a search with [MmodFind\(\)](#). If you had associated a calibration object with the model(s) in your Model Finder context, you must re-associate the calibration object, using [MmodControl\(\)](#) with [M_ASSOCIATED_CALIBRATION](#).

Moreover, this function can also save a Model Finder context to a file or memory stream. All information about the previously allocated Model Finder context is saved, including all of the individual models' current settings. However, preprocessing, any associated calibration objects and drawing control type settings are not saved.

To inquire the number of bytes necessary to save a Model Finder context to memory stream, you should call this function ([MmodStream\(\)](#)) first with [M_INQUIRE_SIZE_BYTE](#).

The content saved to memory stream is equivalent to the content saved to file. In addition, any file saved with [MmodSave\(\)](#) is equivalent to a file saved using this function.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using [MmodStream\(\)](#), you can choose to save a backwards-compatible version of the Model Finder context, which will work using a version of MIL that is up to one major release older than the current version (depending on which version is specified). For example, if you allocate a Model Finder context using MIL 9.0 and save it to version 8.0, you can restore this context on a computer where MIL 8.0 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore Model Finder contexts saved using MIL version 7.0 or above. Settings that do not exist in the lower version will be filled with default values when the Model Finder context is loaded or restored.

Parameters

MemPtrOrFileName

Specifies the file or memory stream.

● For specifying the name and path of a file, memory stream or opening an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving a Model Finder context to a file, use the MMF file extension. The function internally handles the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p> <p>To open or save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p>

		<i>Parameters</i>
		<i>FileName</i> Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE .	
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. The parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)	
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (MmodStream()) first with M_INQUIRE_SIZE_BYTE . (summarize)	

SystemId

Specifies the system on which to restore the Model Finder context. For **M_INQUIRE_SIZE_BYTE**, **M_LOAD**, and **M_SAVE**, **SystemId** is ignored and should be set to **M_NULL**. For an **M_RESTORE** operation, this parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform on the Model Finder context. This parameter must be set to one of the following values:

● For specifying the type of operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save a Model Finder context to memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated Model Finder context.
<input type="checkbox"/> M_RESTORE	Restores a Model Finder context from a file or memory stream and assigns it an identifier.
<input type="checkbox"/> M_SAVE	Saves a Model Finder context to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the Model Finder context. This parameter must be set to one of the following values:

● For specifying the type of stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream.
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)

Version

Specifies the MIL version of the Model Finder context. This parameter must be set to one of the following values.

● For specifying the MIL version	
----------------------------------	--

Value	Description
M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL. For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Specifies the function's control flag and should be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the Model Finder context.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, **ContextIdPtr** specifies the address of the variable from which to read the Model Finder context identifier.

For an [M_LOAD](#) operation, the **ContextIdPtr** specifies the address of the variable from which to read the identifier of the Model Finder context where the file or memory stream content will be loaded.

For an [M_RESTORE](#) operation, the **ContextIdPtr** specifies the address in which to return the identifier of the restored Model Finder context. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the Model Finder context, in bytes.

If the size is not required, you can set this parameter to **M_NULL**.

Note that the size of a Model Finder context will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milmod.lib.
DLL	Requires mil.dll; milmod.dll.

Mocr functions

Synopsis

The functions prefixed with Mocr make up the Optical Character Recognition module. The Optical Character Recognition module is template-based and reads/verifies mechanically generated, character strings in 8-bit grayscale images, and provides results such as quality scores and validity flags. It is especially designed to operate on character strings with known lengths in degraded images with a few degrees of rotation in the target image. Functions are available to create, save, restore, modify, and inquire about fonts. The module also allows you to calibrate fonts, define search constraints, and read/verify character strings in the target image. The font information can be saved in a MIL-compatible (.mfo) font file format that can be used with other Matrox Imaging software products.

Functions

- [MocrAllocFont](#)
- [MocrAllocResult](#)
- [MocrCalibrateFont](#)
- [MocrControl](#)
- [MocrCopyFont](#)
- [MocrDraw](#)
- [MocrFree](#)
- [MocrGetResult](#)
- [MocrHookFunction](#)
- [MocrImportFont](#)
- [MocrInquire](#)
- [MocrModifyFont](#)
- [MocrPreprocess](#)
- [MocrReadString](#)
- [MocrRestoreFont](#)
- [MocrSaveFont](#)
- [MocrSetConstraint](#)
- [MocrStream](#)
- [MocrVerifyString](#)

MocrAllocFont

Synopsis

Allocate an OCR font context.

Syntax

```
MIL_ID MocrAllocFont(
    MIL_ID SystemId,
    MIL_INT FontType,
    MIL_INT CharNumber,
    MIL_INT CharCellSizeX,
    MIL_INT CharCellSizeY,
    MIL_INT CharOffsetX,
    MIL_INT CharOffsetY,
    MIL_INT CharSizeX,
    MIL_INT CharSizeY,
    MIL_INT CharThickness,
    MIL_INT StringLength,
    MIL_INT InitFlag,
    MIL_ID *FontIdPtr
)
```

Description

This function allocates an OCR font context on the specified system. An OCR font context contains the OCR font, target image information, and processing controls. When the OCR font context is no longer required, you should release its memory, using [MocrFree\(\)](#).

If the OCR font context already exists as an OCR font file that matches your constraints and character sizes, such as *SEMI_M12-92.mfo* (SEMI M12-92) or *SEMI_M13-88.mfo* (SEMI M13-88), restore it using [MocrRestoreFont\(\)](#). If you need a modified version of a SEMI font (changing its character size, offset, thickness, or the number of characters in the OCR font context) or need a user-defined font, you should allocate a new OCR font context. Note that using either [M_SEMI_M12_92](#) or [M_SEMI_M13_88](#) automatically sets the OCR font context type to [M_CONSTRAINED](#).

A newly allocated OCR font context must have its grayscale character representations initialized using [MocrImportFont\(\)](#) or [MocrCopyFont\(\)](#). Each part of the OCR font context can be changed using [MocrControl\(\)](#) and [MocrModifyFont\(\)](#). Constraints are set using [MocrSetConstraint\(\)](#).

When allocating an OCR font context, you must specify an OCR font context type. This can be modified later using [MocrControl\(\)](#) with [M_CONTEXT_CONVERT](#). If your target image has a visible threshold difference between the target characters and the background, uniform illumination, and unknown spacing between characters, and unknown string lengths, and if your font is proportional, start by allocating an [M_GENERAL](#) OCR font context. If your target image contains noisy, scratched, or low contrast characters, or if the illumination is not uniform, or if your font includes broken characters start by allocating an [M_CONSTRAINED](#) OCR font context.

You must calibrate the OCR font context when the cell, offset, size, or thickness of the characters in your target image differs from the OCR font. Calibration can be either automatic (using [MocrCalibrateFont\(\)](#)) or manual (using [MocrControl\(\)](#)). Note, to use [MocrCalibrateFont\(\)](#), you must use an [M_CONSTRAINED](#) OCR font context.

If you change any controls or constraints, use [MocrPreprocess\(\)](#) to speed up any following read or verify operation.

If you intend to reuse a font that you modified, save it using [MocrSaveFont\(\)](#), and restore it using [MocrRestoreFont\(\)](#) when you need it.

Parameters

SystemId

Specifies the system on which to allocate the OCR font context. This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

FontType

Specifies the OCR font context type and the type of its font. This parameter should be set to one of the following values:

● For specifying the context type and the type of its font	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_USER_DEFINED .
<input type="checkbox"/> M_SEMI_M12_92	Specifies a font respecting the SEMI M12-92 standard as the type of font. StringLength must be set to 12, the constraints (MocrSetConstraint()) for the target image are preset, and a checksum calculation is activated. (summarize)
<input type="checkbox"/> M_SEMI_M13_88	Specifies a font respecting the SEMI M13-88 standard as the type of font. StringLength must be set to 18, the constraints (MocrSetConstraint()) for the target image are preset, and a checksum calculation is activated. (summarize)
<input type="checkbox"/> M_USER_DEFINED +	Specifies a general, user-defined type of font. No maximum string length is set, the scale is set to 1.0, and the character spacing is the same as CharCellSizeX . No checksum is automatically associated to this OCR font context. You must specify a combination value from the table below . (summarize)

Combination constants for [M_USER_DEFINED](#);

You must add one of the following values to the above-mentioned value to set the OCR font context type.

● For M_USER_DEFINED to specify the context type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CONSTRAINED	Specifies an OCR font context that works well with degraded target images and requires more information about the target string, but provides a more robust search. Note that using either M_SEMI_M12_92 or M_SEMI_M13_88 automatically sets the OCR font context type to M_CONSTRAINED . For more information, see the Guidelines for choosing context types section in Chapter 11: Optical character recognition . This is the default value. (summarize)
<input type="checkbox"/> M_GENERAL	Specifies an OCR font context that works well with clean target images. It requires less information about the target string, but provides a less robust search. For more information, see the Guidelines for choosing context types section in Chapter 11: Optical character recognition . (summarize)

CharNumber

Specifies how many characters can be stored in the OCR font context. The maximum number of characters supported is 500.

CharCellSizeX

Specifies the width of the characters in the OCR font context, in pixels.

● For specifying the width of the characters in the context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 6 <= Value <= 256	Sets the width, in pixels.

CharCellSizeY

Specifies the height of the characters in the OCR font context, in pixels.

● For specifying the height of the characteres in the context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 6 <= Value <= 256	Sets the height, in pixels.

CharOffsetX

Specifies the distance between the edge of each character and their surrounding cell, along the X-axis, in pixels. The minimum recommended value is 1.

Typically, this is set to the same value as the character thickness.

CharOffsetY

Specifies the distance between the edge of each character and their surrounding cell, along the Y-axis, in pixels. The minimum recommended value is 1.

CharSizeX

Specifies the width of the characters without their surrounding cell, in pixels.

● For specifying the width of the charcaters without their surrounding cell	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 6 <= Value <= 256	Sets the width, in pixels. This should be equal to: <code>CharCellSizeX - (CharOffsetX *2)</code> . (summarize)

CharSizeY

Specifies the height of the characters without their surrounding cell, in pixels.

● For specifying the height of the characters without their surrounding cell	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 6 <= Value <= 256	Sets the height, in pixels. This should be equal to: <code>CharCellSizeY - (CharOffsetY *2)</code> . (summarize)

CharThickness

Specifies the maximum thickness (stroke width) of the characters in the OCR font context, in pixels.

StringLength

Specifies the maximum length of the string that can be read or verified using the OCR font context. Note that if [M_SEMI_M12_92](#) is used, the string length must be 12. If [M_SEMI_M13_88](#) is used, the string length must be 18. The maximum string length is 100 characters.

InitFlag

Specifies whether the characters are brighter than the background. This parameter should be set to one of the following values:

● For specifying the character brightness	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FOREGROUND_BLACK	Specifies that the characters to read or verify are darker than the background.
<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that the characters to read or verify are brighter than the background.

FontIdPtr

Specifies the address of the variable to which the OCR font context identifier is to be written. Since the **MocrAllocFont()** function also returns the OCR font context identifier, you can set this parameter to **M_NULL**.

Return value

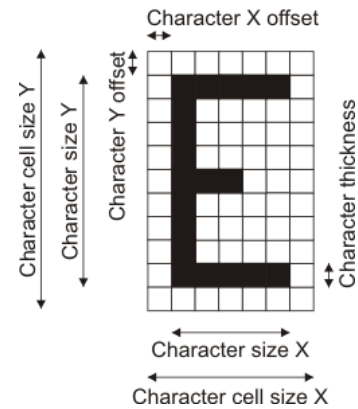
The returned value is the OCR font context identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Example

An example of one character from the font, the letter 'E', follows:

```
MIL_ID FontID = M_NULL;
MocrAllocFont(M_DEFAULT, M_USER_DEFINED+M_GENERAL,
              26, 8, 11, 1, 1, 6, 9, 1, 26, M_FOREGROUND_BLACK, &FontID);
```

Would look as follows:



Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrAllocResult

Synopsis

Allocate an OCR result buffer.

Syntax

```
MIL_ID MocrAllocResult(  
    MIL_ID SystemId,  
    MIL_INT InitFlag,  
    MIL_ID *OcrResultIdPtr  
)
```

Description

This function allocates a result buffer for use with other OCR functions. You should check if the operation was successful by verifying that the OCR result buffer identifier returned is not **M_NULL**.

When the result buffer is no longer required, you should release it using the [MocrFree\(\)](#) function.

Parameters

SystemId

Specifies the system on which to allocate the OCR result buffer.

This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

InitFlag

Specifies the initialization flag and should be set to **M_DEFAULT**.

OcrResultIdPtr

Specifies the address of the variable to which the OCR result buffer identifier is to be written. Since the **MocrAllocResult()** function also returns the OCR result buffer identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrCalibrateFont

Synopsis

Calibrate the OCR font context's character size, and target spacing to match the characters in a sample target image.

Syntax

```
void MocrCalibrateFont(
    MIL_ID ImageBufId,
    MIL_ID FontId,
    MIL_CONST_TEXT_PTR String,
    MIL_DOUBLE TargetCharSizeXMin,
    MIL_DOUBLE TargetCharSizeXMax,
    MIL_DOUBLE TargetCharSizeXStep,
    MIL_DOUBLE TargetCharSizeYMin,
    MIL_DOUBLE TargetCharSizeYMax,
    MIL_DOUBLE TargetCharSizeYStep,
    MIL_INT Operation
)
```

Description

This function automatically calibrates the X and Y size and spacing of the font of a specified OCR font context to match that of a string in the sample target image. Note that these values can also be set manually with [MocrControl\(\)](#) with [M_TARGET_CHAR_SPACING](#), [M_TARGET_CHAR_SIZE_X](#), and [M_TARGET_CHAR_SIZE_Y](#). This function can only calibrate an [M_CONSTRAINED](#) OCR font context. You can change the type of context using [MocrControl\(\)](#) with [M_CONTEXT_CONVERT](#).

The sample image must contain the string specified in the [String](#) parameter, and both the size and spacing must be representative of the images to be read or verified with the calibrated OCR font context. This function might not return the expected results if the sample image contains more or fewer characters than the specified target string length of the target ([MocrControl\(\)](#) with [M_STRING_SIZE](#)).

The better the sample image, the better the calibration results. If a string cannot be located in the sample image, an error will be generated. Use [MappGetError\(\)](#) to learn the error code and related information.

MocrCalibrateFont() should be called after defining a new OCR font context or when the character size in the target images changes.

The resulting target values are the best match between the minimum and the maximum target character sizes specified in the parameters of this function. Calibration will increment the minimum target character size by a step value until it reaches the maximum target character size. You can inquire the target character sizes and spacing with [MocrInquire\(\)](#). For best results, use a range of +/- 1 pixel with, for example, a step of between 0.1 and 0.2 pixels. Note that the greater the number of steps between the minimum and the maximum target character sizes, the longer this operation will take.

Parameters

- ImageBufId
- Specifies the identifier of the sample target image buffer. The characters of the string in this sample image must be of the same type and size as that in the images to be read or verified using the OCR font context.
- FontId
- Specifies the identifier of the OCR font context to be calibrated.
- String
- Specifies the character string present in the sample target image. This string must be null-terminated.

● For specifying the character string	
▢ Value	Description
▢ MIL_TEXT(Specifies the character string that is present in the sample target image.

MIL_TEXT_PTR String)	(summarize)
	Parameters
	String Specifies the character string. This string must be null-terminated.

TargetCharSizeXMin

Specifies the minimum width of the characters in the sample target image, in pixels.

For specifying the minimum width of the characters	
Value	Description
<input type="checkbox"/> 6 <= Value <= 256	Sets the minimum width, in pixels. Note that the specified value must satisfy the following: (TargetCharSizeXMin / CharSizeX) >= 0.25 . (summarize)

TargetCharSizeXMax

Specifies the maximum width of the characters in the sample target image, in pixels.

For specifying the maximum width of the characters	
Value	Description
<input type="checkbox"/> 6 <= Value <= 256	Sets the maximum width, in pixels. Note that the specified value must satisfy the following: (TargetCharSizeXMax / CharSizeX) <= 4.0 . (summarize)

TargetCharSizeXStep

Specifies the increment to use when searching for the character width between [TargetCharSizeXMin](#) and [TargetCharSizeXMax](#). Note that very small step sizes should only be used within small ranges, otherwise application performance can be adversely affected.

TargetCharSizeYMin

Specifies the minimum height of the characters in the sample target image, in pixels.

For specifying the minimum height of the characters	
Value	Description
<input type="checkbox"/> 6 <= Value <= 256	Sets the minimum height, in pixels. Note that the specified value must satisfy the following: (TargetCharSizeYMin / CharSizeY) >= 0.25 . (summarize)

TargetCharSizeYMax

Specifies the maximum height of the characters in the sample target image, in pixels.

For specifying the maximum height of the characters	
Value	Description
<input type="checkbox"/> 6 <= Value <= 256	Sets the maximum height, in pixels. Note that the specified value must satisfy the following: (TargetCharSizeYMax / CharSizeY) <= 4.0 . (summarize)

TargetCharSizeYStep

Specifies the increment to use when searching for the character height between [TargetCharSizeYMin](#) and [TargetCharSizeYMax](#). Note that very small step sizes should only be used within small ranges, otherwise application performance can be adversely affected.

Operation

Specifies the operation to be performed. This should be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrControl

Synopsis

Control an OCR font context or an OCR result buffer setting.

Syntax

```
void MocrControl(
    MIL_ID FontContextorResultId,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function sets various OCR controls for a read/verify operation. This function also sets various controls that affect how results are retrieved by [MocrGetResult\(\)](#). To inquire the current value of a particular control type, use [MocrInquire\(\)](#).

After changing the OCR controls or constraints, use [MocrPreprocess\(\)](#) to speed up any following read or verify operation.

Parameters

FontContextorResultId
Specifies the identifier of the OCR font context or the OCR result buffer to control.

ControlType
Specifies the type of control to set.
See the [Parameter associations](#) section for possible values.

ControlValue
Specifies the required value for the control.
See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For the characteristics of the font](#)
- [For operation controls for read and verify operations](#)
- [For the characteristics of the target characters](#)
- [For the result buffer](#)

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are used to specify the characteristics of the font.

For the characteristics of the font	
ControlType	Description
ControlValue	
M_CHAR_ERASE	Erases a character from the font of the OCR font context. (summarize)

<input type="checkbox"/> Value	Specifies the ASCII character associated with the character. Note that the character must be present in the OCR font context. (summarize)
--------------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are used to specify operational controls for read and verify operations.

For operation controls for read and verify operations	
ControlType	Description
ControlValue	
<input type="checkbox"/> M_BLANK_CHARACTERS	Sets whether the space between characters in the target image is read or not. Note that this is available only when using an M_GENERAL OCR font context. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that blank spaces will not be read from the target image into the result string. Note that spaces could still be erroneously read if your OCR font context is not manually calibrated properly or the string length for your target image is wrong. This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies that blank spaces will be read from the target image into the result string.
<input type="checkbox"/> M_BROKEN_CHAR	Sets the capability to read/verify a broken character. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that, during the read/verify, OCR should not try to compensate for broken characters. Note that this enables a faster algorithm for reading/verifying a string within the target image. Note that broken characters could still be erroneously read if your target image is noisy, scratched, has low contrast characters, or if the illumination is not uniform. This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies that broken characters should be identifies as possible characters.
<input type="checkbox"/> M_CHAR_ACCEPTANCE	Sets the acceptance level used to determine a successful match between the font and the characters found within the target image. During a read/verify operation, the character found with the highest match score, either equal to or greater than the acceptance level, will be returned. If the match score is less than the acceptance level, the result with the highest match score closest to the acceptance level will be returned. If the match score is 0% the invalid character (M_CHAR_INVALID) is returned. A perfect match is 100%, no correlation is 0%. The match score depends on the image quality. You should experiment to decide what is a typical match score for your application. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the acceptance level for a target character. The default value is 1.0. (summarize)
<input type="checkbox"/> M_CHAR_INVALID	Sets the ASCII character for all characters within the selected string or strings during the next read or verify operation whose match scores fall below the character acceptance level. (summarize)
<input type="checkbox"/> 1 to 255	Specifies the character that will replace unrecognized characters.
<input type="checkbox"/> M_NULL	Specifies that no special character will replace unrecognized characters. This is the default value. (summarize)
<input type="checkbox"/> M_CONTEXT_CONVERT	Changes the type of OCR font context. Note that not all controls are available with both types of OCR font contexts. The restrictions are indicated within each control. When you change the type of OCR font context, invalid controls are replaced by their default value.

	(summarize)
<input type="checkbox"/> M_CONSTRAINED	Specifies an OCR font context that works well with degraded target images. It requires more information about the target string, but provides a more robust search. (summarize)
<input type="checkbox"/> M_GENERAL	Specifies an OCR font context that works well with clean target images. It requires less information about the target string, but provides a less robust search. (summarize)
<input type="checkbox"/> M_MORPHOLOGIC_FILTERING	Sets the number of iterations of morphological filtering. Morphological filtering can add robustness to the OCR operation by internally enhancing the contrast of the image, but it requires processing time. The optimal number of iterations depends on the complexity of the target image. (summarize)
<input type="checkbox"/> 0 to 100	Specifies the number of iterations. 0 disables the morphologic filtering. Only integer values are accepted. The default value is 2. (summarize)
<input type="checkbox"/> M_SKIP_STRING_LOCATION	Sets whether to first locate the strings in the target image before trying to identify characters against the OCR font context or to skip this step and save processing time. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the step will not be skipped. This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies that the step will be skipped.
<input type="checkbox"/> M_SPEED	Sets the algorithm's search speed. Note that increasing the search speed can decrease the robustness and subpixel accuracy of a read/verify operation. This is most effective on large-sized characters. (summarize)
<input type="checkbox"/> M_HIGH	Sets the speed to high.
<input type="checkbox"/> M_LOW	Sets the speed to low.
<input type="checkbox"/> M_MEDIUM	Sets the speed to medium. This is the default value. (summarize)
<input type="checkbox"/> M_VERY_HIGH	Sets the speed to very high.
<input type="checkbox"/> M_VERY_LOW	Sets the speed to very low.
<input type="checkbox"/> M_STRING_ACCEPTANCE	Sets the acceptance level used to determine a successful match between the font and a read/verified string. During a read/verify operation, the string found with the highest match score, either equal to or greater than the acceptance level, will be returned. If the match score is less than the acceptance level, the result with the highest match score closest to the acceptance level will be returned. If the match score is 0%, a blank string is returned. Note that, in this case, <code>MocrGetResult()</code> with <code>M_STRING_VALID_FLAG</code> would be false. A perfect match is 100%, no correlation is 0%. The match score depends on the image quality. You should experiment to decide what is a typical match score for your application. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the acceptance level for a string. The default value is 1.0. (summarize)
<input type="checkbox"/> M_STRING_ANGLE_INTERPOLATION_MODE	Sets the interpolation mode to use when reading/verifying a string at an angle. (summarize)
<input type="checkbox"/> M_BICUBIC	Specifies that bicubic interpolation should be used.
<input type="checkbox"/> M_BILINEAR	Specifies that bilinear interpolation should be used.

	This is the default value. (summarize)
<input type="checkbox"/> M_NEAREST_NEIGHBOR	Specifies that nearest neighbor interpolation should be used.
<input type="checkbox"/> M_TOUCHING_CHAR	Sets the capability to read/verify characters that touch each other in the target image. (summarize)
<input type="checkbox"/> M_DISABLE	Disables the identification of touching characters. Enables a faster algorithm for read or verify operations. This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Enables the identification of touching characters.

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are used to set the characteristics of the target characters.

● For the characteristics of the target characters	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_CHAR_POSITION_VARIATION_X	Sets the amount by which the position of the characters in the target string can vary along the X-axis. This tolerance is relative to the position of each character, as determined by M_TARGET_CHAR_SPACING . This tolerance increases the region being searched for characters within the target image. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the position tolerance, in pixels of the character, not of the target image. The default value is 0.0. (summarize)
<input type="checkbox"/> M_CHAR_POSITION_VARIATION_Y	Sets the amount by which the position of the characters in the target string can vary along the Y-axis. This tolerance is relative to the position of each character, as determined by M_TARGET_CHAR_SPACING . This tolerance increases the region being searched for characters within the target image. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the position tolerance, in pixels of the character, not of the target image. The default value is 0.0. (summarize)
<input type="checkbox"/> M_STRING_ANGLE	Sets the expected angle at which the string can be found. (summarize)
<input type="checkbox"/> 0 <= Value <= 360°	Specifies the angle, in degrees. The default value is 0.0. (summarize)
<input type="checkbox"/> M_STRING_ANGLE_DELTA_NEG	Sets the possible angle variation in a clockwise rotation, relative to M_STRING_ANGLE . (summarize)
<input type="checkbox"/> 0 <= Value <= 180°	Specifies the possible clockwise angle variation. The default value is 0.0. (summarize)
<input type="checkbox"/> M_STRING_ANGLE_DELTA_POS	Sets the possible angle variation in a counter-clockwise rotation, relative to M_STRING_ANGLE . (summarize)
<input type="checkbox"/> 0 <= Value <= 180°	Specifies the possible counter-clockwise angle variation. The default value is 0.0. (summarize)
<input type="checkbox"/> M_STRING_NUMBER	Sets the number of strings to be read/verified from the target image. Multiple strings can only be found if each string resides on a different line within the target image and each line of text does not overlap the previous.

	Note that, for best results, all strings read in an image with multiple strings should be of similar length, have a consistent inter-line spacing and should start at a similar location along the X-axis. (summarize)
<input type="checkbox"/> M_ALL	Specifies that the number of strings in the target image should be determined automatically.
<input type="checkbox"/> Value	Specifies the number of lines of text in the target image. Note that specifying this value can improve the speed of a following read/verify operation. The default value is 1. (summarize)
<input type="checkbox"/> M_STRING_SIZE	Sets the length of the string to be read/verified from the target image. (summarize)
<input type="checkbox"/> M_ANY	Specifies that the length of the string is unknown. Note that this is available only when using an M_GENERAL OCR font context. (summarize)
<input type="checkbox"/> Value	Specifies the string length to read/verify, which must be less than or equal to the maximum string length of the OCR font context. Note that when using an M_GENERAL OCR font context and M_BLANK_CHARACTERS are enabled, blank characters must be counted in the string length. The default value is set to the maximum string length of the OCR font context. When OCR font context constraints are set, specifying a string length can improve the speed of a following read/verify operation. Note that if M_SEMI_M12_92 is used, the string length must be 12. If M_SEMI_M13_88 is used, the string length must be 18. (summarize)
<input type="checkbox"/> M_TARGET_CHAR_SIZE_X	Sets the width of the target characters. Note that this can also be automatically set using MocrCalibrateFont() . (summarize)
<input type="checkbox"/> M_SAME	Specifies that the character width is found automatically. Use this value when the font uses a fixed width for all its characters. Note that this is available only when using an M_GENERAL OCR font context. (summarize)
<input type="checkbox"/> Value > 1	Specifies the character width in pixels (with subpixel accuracy). The default value is set to the character size X of the font. (summarize)
<input type="checkbox"/> M_TARGET_CHAR_SIZE_Y	Sets the height of the target characters. Note that this can also be automatically set using MocrCalibrateFont() . (summarize)
<input type="checkbox"/> M_SAME	Specifies that the character height is found automatically. Use this value when the font uses a fixed height for all its characters. Note that this is available only when using an M_GENERAL OCR font context. (summarize)
<input type="checkbox"/> Value > 1	Specifies the character height in pixels (with subpixel accuracy). The default value is set to the character size Y of the font. (summarize)
<input type="checkbox"/> M_TARGET_CHAR_SPACING	Sets the amount of space between characters in the string. Note that this can also be automatically set using MocrCalibrateFont() . (summarize)
<input type="checkbox"/> M_ANY	Specifies that the inter-character spacing is unknown and not the same between the characters. This enables automatic spacing detection. Note that this is available only when using an M_GENERAL OCR font context. (summarize)
<input type="checkbox"/> M_SAME	Specifies that the inter-character spacing is the same throughout the target string. This enables automatic spacing detection. Note that this is available only when using an M_GENERAL OCR font context. (summarize)

<input type="checkbox"/> Value > 1	Specifies the inter-character spacing, with subpixel accuracy. The default value is set to the character cell size of the font. (summarize)
<input type="checkbox"/> M_TEXT_STRING_SEPARATOR	Sets the ASCII character to be used as a string separator within the text read/verified. This separator will be inserted between each string. Note that this must be set before the text is read/verified. (summarize)
<input type="checkbox"/> Value	Specifies the ASCII code of the character. The default value is '\n'. (summarize)
<input type="checkbox"/> M_THICKEN_CHAR	Sets the number of character thickening iterations. Each iteration enlarges the thickness of the target character. This is useful for some types of fonts (for example, dotted characters). You should choose a value large enough for the intra character dots to connect but not too large to connect separate characters together. Note that the characters included in the font should not be dotted. Instead a thickened version should be included. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to 100	Specifies the number of times a character should be thickened. Only integer values are accepted. (summarize)
<input type="checkbox"/> M_THRESHOLD	Sets the threshold value used to internally binarize the target image for segmentation of characters. Note that this control type is available only when using an M_GENERAL OCR font context. (summarize)
<input type="checkbox"/> M_AUTO	Specifies to use automatically computed threshold value.
<input type="checkbox"/> Value	Specifies the threshold value. The default value is 1. (summarize)

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings are used to retrieve results from the result buffer.

● For the result buffer	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_SELECT_STRING	Selects the line of text to return from the result buffer, when multiple lines of text are read/verified. (summarize)
<input type="checkbox"/> M_ALL	Specifies that all strings are selected. This is the default value. (summarize)
<input type="checkbox"/> Value	Specifies a specific string. Strings are identified by numbers from 0 to (M_STRING_NUMBER -1). (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrCopyFont

Synopsis

Copy a font character to or from an image buffer.

Syntax

```
void MocrCopyFont(
    MIL_ID ImageBufId,
    MIL_ID FontId,
    MIL_INT Operation,
    MIL_CONST_TEXT_PTR CharListString
)
```

Description

This function copies a character representation of one, or many, font characters to/from the specified image buffer. This buffer can then be used to initialize, change or obtain a visual representation of the font's characters.

If character representations from the image buffer are being copied to the OCR font context, the OCR font context must be large enough to hold the representations of all the specified characters. You can use [MocrInquire\(\)](#) to determine the maximum number of characters that can be stored in the font and the size of each character.

The characters are stored in contiguous cells in the image buffer.

The character representations are read from either a font or an image buffer, assuming that the source image is a grid of character representations matching the specified font size. The character representations are read from left to right, and the characters in **CharListString** must exist in the font.

If the character representation being added already exists in the font, it will be replaced by the new character representation. If the character representation does not already exist in the font, it is appended to the end of the font. If the maximum string length, as set using [MocrAllocFont\(\)](#), is equal to or greater than the number of characters in the font (use [MocrInquire\(\)](#) with [M_CHAR_NUMBER_IN_FONT](#)), an error will occur and the copy operation will fail.

Note that it is crucial that the character representation respects the foreground value as set in [MocrAllocFont\(\)](#). If the foreground value of the character representation does not match, the font will be unusable.

Parameters

ImageBufId

Specifies the identifier of the image buffer to/from which characters are copied.

FontId

Specifies the identifier of the OCR font context to/from which the characters are copied.

Operation

Specifies the direction of the copy operation. This parameter can be set to one of the following values:

● For specifying the direction of the copy operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_COPY_FROM_FONT +	Copies character(s) from a font context to an image buffer. Note that the maximum number of characters that can be copied from a font context to an image buffer is limited to the number of characters in the font. Note that The buffer height must be at least the character Cell size Y multiplied by the number of lines necessary to hold all the characters. For more details see MocrInquire() with M_CHAR_CELL_SIZE_Y . (summarize)
<input type="checkbox"/> M_COPY_TO_FONT +	Copies character(s) from an image buffer to a font context.

Combination constant for any of the possible values of the [Operation](#) parameter

You can add the following value to the above-mentioned values to specify that all font characters are copied.

● For the Operations parameter	
☐ Value	Description
☐ M_ALL_CHAR	Copies all font characters to/from the image buffer, stacked from left to right. Note that when M_ALL_CHAR is added, the CharListString parameter should be set to M_NULL and is ignored. (summarize)

Combination constant for [M_COPY_FROM_FONT](#);

You can add the following value to the above-mentioned value to specify that fonts are sorted by their ACII character values.

● For M_COPY_FROM_FONT	
☐ Value	Description
☐ M_SORT	Sorts font by its ASCII character values.

CharListString

Specifies a string containing the list of characters to be copied.

● For specifying the string					
☐ Value	Description				
☐ MIL_TEXT(MIL_TEXT_PTR <i>CharListString</i>)	<p>Specifies the string containing the list of characters to be copied. Character representations copied to the font are associated with the characters, and their ASCII code, in the string. If a character exists in both the list and the font, the character representation will be overwritten with the new character representation. New characters will be added to the font provided that there is sufficient space in the OCR font context.</p> <p>Note that the CharListString parameter must be set to M_NULL if M_ALL_CHAR is added to the Operation parameter.</p> <p>(summarize)</p> <table><tr><td colspan="2"><i>Parameters</i></td></tr><tr><td><i>CharListString</i></td><td>Specifies the string. This string must be null-terminated.</td></tr></table>	<i>Parameters</i>		<i>CharListString</i>	Specifies the string. This string must be null-terminated.
<i>Parameters</i>					
<i>CharListString</i>	Specifies the string. This string must be null-terminated.				

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrDraw

Synopsis

Draw specific features of OCR fonts or OCR result occurrences in the destination image buffer.

Syntax

```
void MocrDraw(
    MIL_ID GraphContId,
    MIL_ID OcrFontOrResultId,
    MIL_ID DestImageId,
    MIL_INT Operation,
    MIL_INT Index,
    MIL_CONST_TEXT_PTR CharList,
    MIL_INT ControlFlag
)
```

Description

This function draws specific features of the OCR fonts or OCR results in the destination image buffer.

Parameters

GraphContId

Specifies the graphics context to use when drawing. This parameter must be set to one of the following values:

For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

OcrFontOrResultId

Specifies the OCR Font context or result buffer from which to extract the features to draw. or the OCR font context from which to retrieve the features to draw. The OCR Font context or result buffer must have been previously allocated using [MocrAllocFont\(\)](#) or [MocrAllocResult\(\)](#).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
The OCR Font context or result buffer must be allocated on the same system as the graphics context buffer ([GraphContId](#)). If it is not, an error will occur.

DestImageId

Specifies the identifier of the destination image buffer in which to draw. The buffer can be any supported MIL image buffer. By drawing into a display's overlay buffer, you can also annotate an image non-destructively.

Operation

Specifies the type of operation to perform. The possible [Operation](#) parameter values in the tables below can be added together to draw multiple features at once.

The possible [Operation](#) parameter values for a OCR Font context are:

For specifying the type of operation to perform	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_DRAW_CHAR	Draws a character representation of the font in the destination image buffer. Characters are drawn from left to right, and top to bottom. On each line, as many characters as possible will be drawn where the size of each character is the Cell size X. For more information see MocrInquire() with M_CHAR_CELL_SIZE_X . The buffer height must be at least the character Cell size Y multiplied by the number of lines necessary to hold all the characters. For more details see MocrInquire() with M_CHAR_CELL_SIZE_Y . (summarize)
--------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The possible [Operation](#) parameter values for an OCR result buffer are:

● For an OCR Font context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_STRING_BOX	Draws a box around the string that is read in the target. Note that the bounding box of the string is the bounding box of all the string's characters bounding box, including the draw margin. (summarize)
<input type="checkbox"/> M_DRAW_STRING_CHAR_BOX	Draws all the character boxes of the result(s) of the string(s) read. The character boxes of the strings are drawn at the location read in the target with the correct angle, scale and aspect ratio. The character box drawn is the bounding box of the character plus the draw margin. (summarize)
<input type="checkbox"/> M_DRAW_STRING_CHAR_POSITION	Draws a cross at the center of the bounding box of each string's character. Note that the angle, scale, and aspect ration of the character are taken into account. (summarize)

Index

Reserved for future expansion and must be set to **M_DEFAULT**.

CharList

Specifies a string containing a list of valid characters to draw when drawing an OCR font. To draw all the characters in the font or to draw results, this parameter should be set to **M_NULL**.

● For specifying the string	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR CharList)	Specifies that, when drawing an OCR font, this parameter can be a null-terminated string specifying a list of valid characters to draw. Characters will be drawn in the specified order. (summarize)
	Parameters
	CharList Specifies the string. This string must be null-terminated.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrFree

Synopsis

Free an OCR font or result buffer.

Syntax

```
void MocrFree(  
    MIL_ID FontOrResultId  
)
```

Description

This function deletes the specified OCR font context or result buffer identifier, and releases any memory associated with it.

Parameter

FontOrResultId

Specifies the identifier of the OCR font context, or result buffer to free. The OCR font context, or result buffer, must have been successfully allocated with [MocrAllocFont\(\)](#) or [MocrAllocResult\(\)](#), respectively, prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrGetResult

Synopsis

Get the specified type of result(s) from an OCR result buffer.

Syntax

```
void MocrGetResult(
    MIL_ID OcrResultId,
    MIL_INT ResultType,
    void *ResultPtr
)
```

Description

This function retrieves the result(s) of the specified type from an OCR result buffer. The information can be retrieved as a text, a string or a character. A text contains one or more strings. A string contains one or more characters. Results are only available after calling [MocrReadString\(\)](#) or [MocrVerifyString\(\)](#).

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image.

Note that, by default, the first string found is returned as a result. To get the result for a specific string or for all strings from a result buffer, use [MocrControl\(\)](#) with [M_SELECT_STRING](#).

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [ResultPtr](#) to [M_NULL](#). When this parameter is set to [M_NULL](#), the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [MocrGetResult\(\)](#) again and you pass an array to the parameter [ResultPtr](#). You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

By setting the [ResultType](#) parameter to [M_INTERACTIVE](#), you can view the results interactively.

Parameters

OcrResultId

Specifies the identifier of the OCR result buffer from which to retrieve results.

ResultType

Specifies the type of result(s) to retrieve or opens an interactive dialog box that displays the results stored in the result buffer, interactively. The following values are available to retrieve results from a read or a verify operation, unless otherwise specified.

To display the results currently stored in the result buffer in an interactive dialog box, select the following.

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter [M_NULL](#).

● For specifying the type of result or opening an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	<div>[<i>This is only applicable to Windows</i>]</div> <div>Opens a dialog box containing a spreadsheet that displays the types of results stored in the result buffer. The results are displayed in separate columns. If there are queued requests, only those results are displayed.</div> <div>(summarize)</div>

To get a result for all strings, use one of the following:

Unless otherwise specified, the following values require that you pass the *ResultPtr* parameter the address of a *MIL_DOUBLE*.

● For all strings	
☐ Value	Description
☐ M_TEXT +	Retrieves the entire text. This includes all available strings, their separators, and the terminating character. Note that the string separator can be controlled using MocrControl() with M_TEXT_STRING_SEPARATOR . (summarize)
	<i>ResultPtr info</i> Data type: array of type char Array size: M_TEXT_LENGTH
☐ M_TEXT_LENGTH +	Retrieves the total number of characters in the entire text. This includes all available strings, their separators, and the terminating character. Note that the string separator can be controlled using MocrControl() with M_TEXT_STRING_SEPARATOR . (summarize)
☐ M_TEXT_SCORE +	Retrieves the match score for the entire text as determined during the read/verify operation.
☐ M_THRESHOLD +	Retrieves the value used to binarize the target image. Note that this is available only when using an M_GENERAL OCR font context, or when using an M_CONSTRAINED OCR font context if your target image has multiple lines, or its string angle is not 0.0°. (summarize)

The following values are available to retrieve results from a read or a verify operation, unless otherwise specified. To get a result for a string or all strings, use one of the following:

Unless otherwise specified, the following values require that you pass the *ResultPtr* parameter the address of a *MIL_DOUBLE*.

● For retrieving results from a read or verify operation	
☐ Value	Description
☐ M_NB_STRING +	Retrieves the number of strings read by MocrReadString() . If no strings are found, 0 is returned. (summarize)
☐ M_STRING +	Retrieves the null-terminated string found during the read/verify operation. By default, any unrecognized character in the string will be replaced by the most-likely character, even if the match score for that character is lower than the specified character acceptance level (MocrControl() with M_CHAR_ACCEPTANCE). You can replace all unrecognized characters in the string with an invalid character symbol by calling MocrControl() with M_CHAR_INVALID . Note that no string separator is inserted between the different strings when getting all strings simultaneously. Refer to M_TEXT to retrieve the strings with separators. (summarize)
	<i>ResultPtr info</i> <ul style="list-style-type: none"> • Data type: array of type char Array size: M_STRING_SIZE + 1 Note: for a single string • Data type: array of type char Array size: the sum of all string lengths + 1 (M_STRING_SIZE + M_SUM) + 1 Note: for multiple strings
☐ M_STRING_ALLOC_SIZE +	Retrieves the total size of the string or each string. This includes the null-termination character. (summarize)
☐ M_STRING_ANGLE +	Retrieves the angle of the string or each string, in degrees.
☐ M_STRING_SCORE +	Retrieves the score calculated during the read/verify operation for the entire string, or for each string.
☐ M_STRING_SIZE +	Retrieves the length of the null-terminated string, or each string, found during the read/verify operation. Note that this does not include the null-termination character. (summarize)
☐ M_STRING_VALID_FLAG +	Retrieves the result of the read or verify operation as a flag that denotes the validity of the string. Each flag is set by comparing the match score with the acceptance level (MocrControl() with M_STRING_ACCEPTANCE). (summarize)
	<i>ResultPtr info</i>

Return values:

M_FALSE	The string was not found by the read/verify operation.
M_TRUE	The string was successfully found by the read/verify operation.

Combination constant for [M_STRING_SIZE](#);

You can add the following value to the above-mentioned value to get the total length of all available strings.

● For M_STRING_SIZE	
▢ Value	Description
▢ M_SUM	Returns the total length of all available strings without the NULL terminating character or string separators.

To get a result for individual characters, use one of the following.

Unless otherwise specified, for a single string, the following values require that you pass the [ResultPtr](#) parameter the address of an array of type MIL_DOUBLE with a size equal to [M_STRING_SIZE](#). For multiple strings, the following values require that you pass the [ResultPtr](#) parameter the address of an array of type MIL_DOUBLE with a size equal to [M_STRING_SIZE](#) + [M_SUM](#).

● For individual characters					
▢ Value	Description				
▢ M_CHAR_POSITION_X +	Gets the X-position of each individual character in the string, or strings.				
▢ M_CHAR_POSITION_Y +	Gets the Y-position of each individual character in the string, or strings.				
▢ M_CHAR_SCORE +	Gets the match score of each individual character within the string, or strings.				
▢ M_CHAR_SIZE_X +	Gets the cell width of each character within the string, or strings.				
▢ M_CHAR_SIZE_Y +	Gets the cell height of each character within the string, or strings.				
▢ M_CHAR_SPACING +	Gets the spacing between each pair of characters in the selected string, or strings. Note that the spacing of the last character in each string is M_NULL . (summarize)				
▢ M_CHAR_VALID_FLAG +	Gets the result of the read or verify operation as flags that denote the validity of each character within the string. Each flag is set by comparing the character's match score with the character's acceptance level (MocrControl() with M_CHAR_ACCEPTANCE). (summarize)				
	<i>ResultPtr info</i> Return values: <table> <tr> <td>M_FALSE</td><td>The character was not found in the read/verify string.</td></tr> <tr> <td>M_TRUE</td><td>The character was successfully found in the read/verify string.</td></tr> </table>	M_FALSE	The character was not found in the read/verify string.	M_TRUE	The character was successfully found in the read/verify string.
M_FALSE	The character was not found in the read/verify string.				
M_TRUE	The character was successfully found in the read/verify string.				

Combination constants for [the values listed in](#) all tables **except** For specifying the type of result or opening an interactive dialog box

You can add one of the following values to the above-mentioned values to cast the requested results to a required data type.

● For specifying the data type	
▢ Value	Description
▢ M_TYPE_CHAR	Casts the requested results to a <i>char</i> . (summarize)
	<i>ResultPtr info</i> Data type: char

M_TYPE_DOUBLE	Casts the requested results to a <i>double</i> . This is the default value. (summarize)
	ResultPtr info Data type: double
M_TYPE_FLOAT	Casts the requested results to a <i>float</i> . (summarize)
	ResultPtr info Data type: float
M_TYPE_LONG	Casts the requested results to a <i>long</i> . (summarize)
	ResultPtr info Data type: long
M_TYPE_MIL_DOUBLE	Casts the requested results to a <i>MIL_DOUBLE</i> . (summarize)
	ResultPtr info <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: Depends on the result type. Note: When multiple results. • Data type: MIL_DOUBLE Note: When a single result.
M_TYPE_MIL_INT	Casts the requested results to a <i>MIL_INT</i> . (summarize)
	ResultPtr info Data type: MIL_INT
M_TYPE_MIL_INT32	Casts the requested results to a <i>MIL_INT32</i> . (summarize)
	ResultPtr info Data type: MIL_INT32
M_TYPE_MIL_INT64	Casts the requested results to a <i>MIL_INT64</i> . (summarize)
	ResultPtr info Data type: MIL_INT64
M_TYPE_SHORT	Casts the requested results to a <i>short</i> . (summarize)
	ResultPtr info Data type: short
M_TYPE_TEXT_CHAR	Cast the requested results to a <i>MIL_TEXT_CHAR</i> . (summarize)
	ResultPtr info <ul style="list-style-type: none"> • Data type: MIL_TEXT_CHAR • Data type: array of type MIL_TEXT_CHAR Array size: Size depends on the contents of the array

ResultPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type char
- array of type MIL_DOUBLE
- array of type MIL_TEXT_CHAR

- char
- double
- float
- long
- M_NULL
- MIL_DOUBLE
- MIL_INT
- MIL_INT32
- MIL_INT64
- MIL_TEXT_CHAR
- short

Specifies the address in which to write the results. Each line of text found within the target image is read as a separate string. If there is only one line of text, there is only one string. Only the results from the first string or selected strings ([MocrControl\(\)](#) with [M_SELECT_STRING](#)) are obtained.

Set this parameter to [M_NULL](#) if you are setting the [ResultType](#) parameter to [M_INTERACTIVE](#) or if you are putting the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrHookFunction

Synopsis

Hook a function to an event.

Syntax

```
MIL_OCR_HOOK_FUNCTION_PTR MocrHookFunction(
    MIL_ID FontId,
    MIL_INT HookType,
    MIL_OCR_HOOK_FUNCTION_PTR HookHandlerPtr,
    void MPTYPE *UserDataPtr
)
```

Description

This function allows you to attach or detach a user-defined function to an event when the specified font is used. A type of event to which a user-defined function can be hooked is string validation. This would immediately follow a read or verify operation. When this type of event occurs, the `MocrReadString()` or `MocrVerifyString()` function will call the hooked function one or many times during the operation to validate each string after it is read but before being written to the OCR result buffer. This function allows you to impose global string constraints, and can be used to implement custom checksum functions or to reject strings that would have otherwise met the character constraints imposed.

Note that a function hooked to an event executes on a distinct thread. This permits the functions to run asynchronously from the operation that fired the event and from functions hooked to another event.

Hooked functions should not take longer to execute than the period in which two of their associated events can occur. You cannot determine the instance of the event that fired the function, and even if this were possible, this information would generally not be very useful. Typically, a hooked function performs the minimum number of operations required and, if necessary, performs longer processes by launching other threads.

Parameters

FontId

Specifies the identifier of the font.

HookType

Specifies the event type. This parameter can be set to the following value:

For specifying the event type																						
Value	Description																					
	corona-II (a)	cnosplur (b)	gige vision (c)	gpu processing (d)	hellor ea/Xa (e)	hellor ecl/Xcl (f)	hellor ed/Xd (g)	IEEE 1394 iIIdc (h)	irls (i)	met-II /cl (j)	met-II /dig (k)	met-II /fmc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	odyssey ea/Xa (p)	odyssey ed/Xcl (r)	odyssey ed/Xcl (s)	solios ea/Xa (t)	solios ecl/Xcl (u)	solios gige (v)	vio (w)
M_STRING_VALIDATION +	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p			t	u	v	w

Combination constant for any of the possible values of the [HookType](#) parameter

You must add the following value to the above-mentioned values to specify to unhook the event that was hooked to the function.

For the HookType parameter to unhook the function	
<input type="checkbox"/> Value	Description

<div><div></div><div>M_UNHOOK</div></div>	Unhooks the function that was hooked to the event.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p					t	u	v	w
-------------------------------------------	----------------------------------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	--	--	--	--	---	---	---	---

HookHandlerPtr

Specifies the address of the function that should be called when the event occurs.

The hook handler function, pointed to by [HookHandlerPtr](#), must be declared as follows:

```
MIL_INT MFTYPE HookHandler(  
    MIL_INT HookType,  
    char MFTYPE *StringPtr,  
    void MFTYPE *UserDataPtr  
)
```

Parameters:

- HookType*
Specifies the type of event hooked.
- StringPtr*
Points to the string to validate.
- UserDataPtr*
*Specifies the user data pointer passed to **MocrHookFunction()**.*

The value returned by the hook function must contain the string validity status: either **M_TRUE** or **M_FALSE**. Note that, *MFTYPE* and *MIL_OCR_HOOK_FUNCTION_PTR* are reserved MIL predefined types for functions and data pointers.

UserDataPtr

Specifies the address of the user data that you want to make available to the hook-handler function. This address is passed to the hook-handler function, through its [UserDataPtr](#) parameter, when the specified event occurs. Set this parameter to **M_NULL** if the [UserDataPtr](#) is not required.

Return value

This function returns a *NULL* pointer.

Remark

- Note that the prototype of this function is kept for backwards-compatibility.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrImportFont

Synopsis

Import character representations into an OCR font context.

Syntax

```
void MocrImportFont(  
    MIL_CONST_TEXT_PTR Filename,  
    MIL_INT FileFormat,  
    MIL_INT Operation,  
    MIL_CONST_TEXT_PTR CharListString,  
    MIL_ID FontId  
)
```

Description

This function imports character representations from a file to initialize or overwrite those of an existing OCR font context. This function's main use is to initialize custom fonts.

Font character representations can come from either a grayscale image or a text file.

When taken from an image file, character representations are associated to the ASCII characters in the string, specified by **CharListString**. The image file must be a grid of character representations matching the specifications of the font. The character representations are read from left to right and top to bottom, and the number of characters in the source image must equal the number of characters in the string, specified by **CharListString**.

When taken from an ASCII file, font character representations must be presented as in the table below.

Parameters

Specifies the name and path of the file from which to import the character representation. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

For specifying the file name and path	
Value	Description
MIL_TEXT (MIL_TEXT_PTR <i>FileName</i>)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p>

	(summarize)	
	<i>Parameters</i>	
	<i>FileName</i>	A string that specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.	

FileFormat

Specifies how the data is stored in the file. This parameter can be set to one of the following values:

● For specifying how the data is stored		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_FONT_ASCII	Specifies a user-drawn ASCII file. Note that the CharListString must be set to M_NULL . (summarize)	
<input type="checkbox"/> M_MIL	Specifies a MIL format image file. Note that only one-band images can be used. (summarize)	
<input type="checkbox"/> M_TIFF	Specifies a TIFF format image file. Note that only one-band images can be used. (summarize)	

Operation

Indicates the type of import operation to be performed. This parameter must be set to the following value:

● For specifying the type of import operation		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> M_LOAD_CHARACTER	Specifies that the font character representations should be imported.	

CharListString

Lists the ASCII characters to associate with the font character representations in the file. This parameter can be set to one of the following values

● For specifying the ASCII characters		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR <i>StringOfCharacters</i>)	Specifies the list of ASCII characters to associate with the font character representations in the image file. Note that the number of characters in this list must match the number of font character representations present in the source file. (summarize)	
	<i>Parameters</i>	
	<i>StringOfCharacters</i>	Specifies the string with the list of ASCII characters. This string must be null-terminated.
<input type="checkbox"/> M_NULL	Specifies that this parameter is ignored. Use this setting when the FileFormat parameter is set to M_FONT_ASCII . (summarize)	

FontId

Specifies the identifier of the OCR font context.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrInquire

Synopsis

Inquire about OCR font context or an OCR result buffer setting.

Syntax

```
MIL_INT MocrInquire(
    MIL_ID FontContextorResultId,
    MIL_INT InquireType,
    void *ResultPtr
)
```

Description

This function inquires about an OCR font context or an OCR result buffer setting.

Note that for an OCR result buffer, this function only retrieves information about result buffer settings (set with [MocrControl\(\)](#), for example). To retrieve results from the OCR result buffer, use [MocrGetResult\(\)](#).

Parameters

FontContextorResultId

Specifies the identifier of the OCR font context or the OCR result buffer from which to read information.

InquireType

Specifies the type of setting about which to inquire. It can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the *ResultPtr* parameter the address of a *MIL_DOUBLE*.

For OCR settings	
Value	Description
<input type="checkbox"/> M_BLANK_CHARACTERS +	<div>Returns whether the space between characters in the target image is read. Note that this is available only when using an M_GENERAL OCR font context. (summarize)</div> <div>ResultPtr info Return values: M_DISABLE; M_ENABLE; (details)</div>
<input type="checkbox"/> M_BROKEN_CHAR +	<div>Returns the capacity to read/verify broken characters. (summarize)</div> <div>ResultPtr info Return values: M_DISABLE; M_ENABLE; (details)</div>
<input type="checkbox"/> M_CHAR_ACCEPTANCE +	<div>Returns the acceptance level used to determine a successful match between the font and the characters found within the target image. (summarize)</div> <div>ResultPtr info Return values: 0.0 to 100.0; (details)</div>
<input type="checkbox"/> M_CHAR_CELL_SIZE_X +	<div>Returns the font's character cell width, in pixels. (summarize)</div> <div>ResultPtr info Return values: 6 <= Value <= 256; (details)</div>

M_CHAR_CELL_SIZE_Y +	<p>Returns the font's character cell height, in pixels. (summarize)</p> <p><i>ResultPtr Info</i> Return values: 6 <= Value <= 256; (details)</p>
M_CHAR_IN_FONT +	<p>Returns a list of the ASCII characters associated to the character representations present in the font. (summarize)</p> <p><i>ResultPtr Info</i> Data type: array of type char Array size: M_CHAR_NUMBER_IN_FONT +1. Return values: M_NULL The parameter is ignored. StringOfCharacters The string with the list of ASCII characters.</p>
M_CHAR_INVALID +	<p>Returns the symbol for the unrecognized characters within the string, or strings. (summarize)</p> <p><i>ResultPtr Info</i> Return values: 1 to 255; M_NULL; (details)</p>
M_CHAR_NUMBER +	<p>Returns the number of characters that can be stored in the font of the OCR font context. (summarize)</p> <p><i>ResultPtr Info</i> Return values: Please see the CharNumber parameter of MocrAllocFont(). (details)</p>
M_CHAR_NUMBER_IN_FONT +	<p>Returns the number of characters that are actually stored in the OCR font context.</p>
M_CHAR_OFFSET_X +	<p>Returns the font's character offset along the X-axis, in pixels. (summarize)</p> <p><i>ResultPtr Info</i> Return values: Please see the CharOffsetX parameter of MocrAllocFont(). (details)</p>
M_CHAR_OFFSET_Y +	<p>Returns the font's character offset along the Y-axis, in pixels. (summarize)</p> <p><i>ResultPtr Info</i> Return values: Please see the CharOffsetY parameter of MocrAllocFont(). (details)</p>
M_CHAR_POSITION_VARIATION_X +	<p>Returns the amount by which the position of the characters in the target image can vary along the X-axis. This value is expressed in pixels of the character, not of the target image. (summarize)</p> <p><i>ResultPtr Info</i> Return values: 0.0 to 100.0; (details)</p>
M_CHAR_POSITION_VARIATION_Y +	<p>Returns the amount by which the position of the characters in the target image can vary along the Y-axis. This value is expressed in pixels of the character, not of the target image. (summarize)</p> <p><i>ResultPtr Info</i> Return values: 0.0 to 100.0; (details)</p>
M_CHAR_SIZE_X +	<p>Returns the width of the font's characters, in pixels. (summarize)</p> <p><i>ResultPtr Info</i> Return values: 6 <= Value <= 256; (details)</p>
M_CHAR_SIZE_Y +	<p>Returns the height of the font's characters, in pixels. (summarize)</p>

	<p>Return values:</p> <p>Value > 0 Preprocessing is not necessary.</p> <p>0 Preprocessing is necessary.</p>
<input type="checkbox"/> M_SKIP_STRING_LOCATION +	<p>Returns whether the string location step will be skipped or not. (summarize)</p> <p><i>ResultPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)</p>
<input type="checkbox"/> M_SPEED +	<p>Returns the algorithm's search speed. (summarize)</p> <p><i>ResultPtr info</i> Return values: M_HIGH; M_LOW; M_MEDIUM; M_VERY_HIGH; M_VERY_LOW; (details)</p>
<input type="checkbox"/> M_STRING_ACCEPTANCE +	<p>Returns the acceptance level used to determine a successful match between the font and the read/verified string. (summarize)</p> <p><i>ResultPtr info</i> Return values: 0.0 to 100.0; (details)</p>
<input type="checkbox"/> M_STRING_ANGLE +	<p>Returns the angle at which the string is expected to be found. (summarize)</p> <p><i>ResultPtr info</i> Return values: 0 <= Value <= 360°; (details)</p>
<input type="checkbox"/> M_STRING_ANGLE_DELTA_NEG +	<p>Returns the possible angle variation in a clockwise rotation, relative to M_STRING_ANGLE. (summarize)</p> <p><i>ResultPtr info</i> Return values: 0 <= Value <= 180°; (details)</p>
<input type="checkbox"/> M_STRING_ANGLE_DELTA_POS +	<p>Returns the possible angle variation in a counter-clockwise rotation, relative to M_STRING_ANGLE. (summarize)</p> <p><i>ResultPtr info</i> Return values: 0 <= Value <= 180°; (details)</p>
<input type="checkbox"/> M_STRING_ANGLE_INTERPOLATION_MODE +	<p>Returns the interpolation mode to use when reading/verifying a string at an angle. (summarize)</p> <p><i>ResultPtr info</i> Return values: M_BICUBIC; M_BILINEAR; M_NEAREST_NEIGHBOR; (details)</p>
<input type="checkbox"/> M_STRING_NUMBER +	<p>Returns the number of strings to read from the target image. (summarize)</p> <p><i>ResultPtr info</i> Return values: M_ALL; Value; (details)</p>
<input type="checkbox"/> M_STRING_SIZE +	<p>Returns the expected length of the string within the target image. (summarize)</p> <p><i>ResultPtr info</i> Return values: M_ANY; Value; (details)</p>
<input type="checkbox"/> M_STRING_SIZE_MAX +	<p>Returns the maximum string length that will be read/verified using the OCR font context. (summarize)</p> <p><i>ResultPtr info</i> Return values: Please see the StringLength parameter of MocrAllocFont(). (details)</p>
<input type="checkbox"/> M_TARGET_CHAR_SIZE_X +	<p>Returns the width of the expected target characters, in pixels. (summarize)</p>

	<i>ResultPtr info</i> Return values: M_SAME; Value > 1; (details)
<input type="checkbox"/> M_TARGET_CHAR_SIZE_Y +	Returns the height of the expected target characters, in pixels. (summarize)
	<i>ResultPtr info</i> Return values: M_SAME; Value > 1; (details)
<input type="checkbox"/> M_TARGET_CHAR_SPACING +	Returns the expected amount of space, in pixels, between characters in the string. (summarize)
	<i>ResultPtr info</i> Return values: M_ANY; M_SAME; Value > 1; (details)
<input type="checkbox"/> M_TEXT_STRING_SEPARATOR +	Returns the ASCII code of the character to be used as a string separator within the text read. (summarize)
	<i>ResultPtr info</i> Return values: Please see MocrControl() with M_TEXT_STRING_SEPARATOR . (details)
<input type="checkbox"/> M_THICKEN_CHAR +	Returns the number of times a character should be thickened. (summarize)
	<i>ResultPtr info</i> Return values: 0 to 100; M_DEFAULT; (details)
<input type="checkbox"/> M_THRESHOLD +	Returns the threshold value used to internally binarize the target image. (summarize)
	<i>ResultPtr info</i> Return values: M_AUTO; Value; (details)
<input type="checkbox"/> M_TOUCHING_CHAR +	Returns whether the capability to read characters that touch is enabled or not. (summarize)
	<i>ResultPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)

Combination constant for [M_CHAR_IN_FONT](#);

You can add the following value to the above-mentioned value to specify that the returned string is sorted in ascending ASCII order.

● For M_CHAR_IN_FONT	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SORT	Sorts the returned string in ascending ASCII order.

Combination constants for any of the possible values of the [InquireType](#) parameter

You can add one of the following values to the above-mentioned values to cast the requested information to the required data type.

● For specifying the data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_CHAR	Casts the requested information to a <i>char</i> . (summarize)
	<i>ResultPtr info</i> Data type: char

M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . (summarize)
	ResultPtr info Data type: double
M_TYPE_FLOAT	Casts the requested information to a <i>float</i> . (summarize)
	ResultPtr info Data type: float
M_TYPE_LONG	Casts the requested information to a <i>long</i> . (summarize)
	ResultPtr info Data type: long
M_TYPE_MIL_DOUBLE	Casts the requested information to a <i>MIL_DOUBLE</i> . (summarize)
	ResultPtr info Data type: MIL_DOUBLE
M_TYPE_MIL_ID	Casts the requested information to a <i>MIL_ID</i> . (summarize)
	ResultPtr info Data type: MIL_ID
M_TYPE_MIL_INT	Casts the requested information to a <i>MIL_INT</i> . (summarize)
	ResultPtr info Data type: MIL_INT
M_TYPE_MIL_INT32	Casts the requested information to a <i>MIL_INT32</i> . (summarize)
	ResultPtr info Data type: MIL_INT32
M_TYPE_MIL_INT64	Casts the requested information to a <i>MIL_INT64</i> . (summarize)
	ResultPtr info Data type: MIL_INT64
M_TYPE_SHORT	Casts the requested information to a <i>short</i> . (summarize)
	ResultPtr info Data type: short
M_TYPE_TEXT_CHAR	Casts the requested information to a <i>MIL_TEXT_CHAR</i> . (summarize)
	ResultPtr info <ul style="list-style-type: none"> • Data type: MIL_TEXT_CHAR • Data type: array of type MIL_TEXT_CHAR Array size: Size depends on the contents of the array

The following inquire type specifies the result for [MocrGetResult\(\)](#).

Unless otherwise specified, the following values require that you pass the [ResultPtr](#) parameter the address of a *MIL_DOUBLE*.

• For specifying the result for MocrGetResult

Value	Description
M_SELECT_STRING +	Returns the index of the string, selected from a multiple line text, for which to get the results with MocrGetResult() . (summarize)
	<i>ResultPtr info</i> Return values: M_ALL; Value; (details)

ResultPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• array of type char• array of type MIL_TEXT_CHAR• char• double• float• long• MIL_DOUBLE• MIL_ID• MIL_INT• MIL_INT32• MIL_INT64• MIL_TEXT_CHAR• short

Specifies the address in which to write the requested information. Since this function also returns the requested information, you can set this parameter to M_NULL.

Return value

This function returns the result of the inquire, cast to a *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrModifyFont

Synopsis

Invert or resize the font of an OCR font context to match the target image characters.

Syntax

```
void MocrModifyFont(  
    MIL_ID FontId,  
    MIL_INT Operation,  
    MIL_INT OperationFlag  
)
```

Description

This function physically modifies the polarity and sizing of the font of an OCR font context.

Parameters

FontId

Specifies the identifier of the OCR font context to modify.

Operation

Specifies the type of operation to perform. This parameter can be set to one or more of the following values:

● For specifying the type of operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INVERT	Physically inverts the character representation of the font (for example, if the foreground is white, this will change it to black).
<input type="checkbox"/> M_RESIZE	<p>Scales the font.</p> <p>This operation physically changes the size of the characters of the font to match the target character size (M_TARGET_CHAR_SIZE_X and M_TARGET_CHAR_SIZE_Y). Once performed, all of the font's specifications, which were set at allocation (MocrAllocFont()), are modified to this new size.</p> <p>Note that changing the size of the font permanently in the OCR font context can save time when compared to resizing the font before each read/verify operation. Increasing the size of the font might be slower because the resize time might be less than the additional processing time required during a read or verify operation to find characters of a larger font.</p> <p>(summarize)</p>

OperationFlag

Specifies the interpolation mode for the function. This parameter can be set to one of the following:

● For specifying the interpoaltion mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	<p>Same as M_BILINEAR.</p> <p>Note that if the Operation parameter is set to M_INVERT, this parameter should be set to M_DEFAULT. No interpolation is done with M_INVERT.</p> <p>(summarize)</p>
<input type="checkbox"/> M_BICUBIC	Sets the interpolation mode to bicubic interpolation.
<input type="checkbox"/> M_BILINEAR	Sets the interpolation mode to bilinear interpolation.
<input type="checkbox"/> M_NEAREST_NEIGHBOR	Sets the interpolation mode to nearest neighbor interpolation.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrPreprocess

Synopsis

Preprocess an OCR font context.

Syntax

```
void MocrPreprocess(  
    MIL_ID FontId,  
    MIL_INT ControlFlag  
)
```

Description

This function preprocesses the specified OCR font context. This function must be called before any read or verify operation and after all the control and constraints are set to speed up all subsequent read or verify operations. Note that preprocessing will be done automatically during the first read or verify operation if this function is not explicitly called.

Each time a font-specific control or target constraint is changed the OCR font context should be preprocessed. To learn if the OCR font context requires preprocessing, call [MocrInquire\(\)](#) with [M_PREPROCESSED](#).

Parameters

FontId

Specifies the OCR font context identifier.

ControlFlag

Specifies the type of operation to perform. This parameter must be set to the following value.

For specifying the type of operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Preprocesses the OCR font context.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrReadString

Synopsis

Read an unknown string from a target image.

Syntax

```
void MocrReadString(
    MIL_ID ImageBufId,
    MIL_ID FontId,
    MIL_ID OcrResultId
)
```

Description

This function reads an unknown string from the specified target image using the specified OCR font context. All existing font controls and constraints (which can be set with [MocrControl\(\)](#) and [MocrSetConstraint\(\)](#) respectively) are taken into account, and results are stored in the specified result buffer. Results can be read from the result buffer using the [MocrGetResult\(\)](#) function.

This function assumes that the string to be read has the same length as specified in the font, and that the target string characters have the same type, size, and spacing as the characters in OCR font context used for calibration (either automatically with [MocrCalibrateFont\(\)](#) or manually with [MocrControl\(\)](#)). This is particularly important when using an [M_CONSTRAINED](#) OCR font context.

For best results, the angle of the string(s) to be read should be as close to 0 as possible.

When reading multiple strings, a search performed at an angle or any use of the [M_GENERAL](#) OCR font context, the target image should have a clearly-defined threshold between the characters and their background. Note that the threshold must preserve the shape of the characters. Broken and/or touching characters can degrade the results.

Note that before performing a read operation, the OCR font context must be preprocessed ([MocrPreprocess\(\)](#)).

Parameters

ImageBufId

Specifies the identifier of the image which contains the string to be read.

FontId

Specifies the identifier of the OCR font context to use to read the string from the target image.

OcrResultId

Specifies the OCR result buffer in which to place results of the read operation.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
The OCR result buffer ([OcrResultId](#)) must be allocated on the same system as the OCR font context ([FontId](#)). If it is not, an error will occur.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrRestoreFont

Synopsis

Restore an OCR font context, font constraint data, or control data from file.

Syntax

```
MIL_ID MocrRestoreFont(  
    MIL_CONST_TEXT_PTR Filename,  
    MIL_INT Operation,  
    MIL_ID SystemId,  
    MIL_ID *FontIdPtr  
)
```

Description

This function restores an OCR font context, previously saved with [MocrSaveFont\(\)](#), from a file. It can also restore the two semi fonts available in MIL by specifying their file names, "Semi1292.mfo" or "Semi1388.mfo". Alternatively, this function can load only font constraint or control data from the file into the specified OCR font context.

When a font is no longer required, it should be freed with [MocrFree\(\)](#).

Parameters

Filename

Specifies the name and path of the file from which to restore the MIL OCR font context. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

● For specifying the file name and path		
☐ Value	Description	
☐ MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)	
		Parameters
	<i>FileName</i>	Specifies the drive, directory, and name of the file.
☐ M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.	

Operation

Specifies which type of OCR font context data to restore. It can be set to one of the following values:

● For specifying the type of data to restore	
☐ Value	Description
☐ M_DEFAULT	Same as M_RESTORE .
☐ M_LOAD_CONSTRAINT	Loads only font constraint data into the specified OCR font context.

<input type="checkbox"/> M_LOAD_CONTROL	Loads only font control data into the specified OCR font context.
<input type="checkbox"/> M_RESTORE	Allocates a new OCR font context on the specified system, and initializes it with the font, control, and constraint data, restored from file.

SystemId

Specifies the system on which the OCR font context is allocated.

This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

FontIdPtr

Specifies the address of the variable to/from which the font identifier is to be written/read.

Since after performing a restore operation, **MocrRestoreFont()** also returns the OCR font context identifier, this parameter can be set to **M_NULL**. If the restore operation fails, **M_NULL** is returned as the identifier.

Return value

Returns the identifier of the existing or newly allocated OCR font context. If allocation fails, **M_NULL** is returned as the identifier.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrSaveFont

Synopsis

Save an existing OCR font context to disk.

Syntax

```
void MocrSaveFont(
    MIL_CONST_TEXT_PTR FileName,
    MIL_INT Operation,
    MIL_ID FontId
)
```

Description

This function saves an existing OCR font context to disk using the MIL font file format. The OCR font context's control, constraint, and/or font character data can all be saved; which data is saved depends on the value of the [Operation](#) parameter.

The font character data contains the character representation of each character in the OCR font context. The control data includes controls used to specify the operational controls for a read/verify operation (such as [M_BLANK_CHARACTERS](#), and [M_TOUCHING_CHAR](#)) and those used to set the characteristics of the target characters (such as [M_THICKEN_CHAR](#), and [M_TARGET_CHAR_SPACING](#)). The constraint data specifies which characters can appear at given positions in the search string.

Parameters

FileName

Specifies the name and path of the file in which to save the font data. It is recommended that you use the MFO file extension for easier use with other Matrox Imaging software products. The function internally handles the opening and closing of this file. If this file already exists, it will be overwritten.

This parameter can be set to one of the following values:

● For specifying the file name and path		
<input type="checkbox"/> Value	Description	
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile". (summarize)	
	Parameters	
	FileName	Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[This is only applicable to Windows] Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.	

Operation

Specifies which data to save to disk. The [Operation](#) parameter can take one of the following values:

● For specifying the data to save	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Saves the same data as M_SAVE .

<input type="checkbox"/> M_SAVE	Saves the OCR font context.
<input type="checkbox"/> M_SAVE_CONSTRAINT	Saves only character constraint data.
<input type="checkbox"/> M_SAVE_CONTROL	Saves only control data.

FontId

Specifies the OCR font context to be saved.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrSetConstraint

Synopsis

Set character position constraints.

Syntax

```
void MocrSetConstraint(
    MIL_ID FontId,
    MIL_INT CharPos,
    MIL_INT CharPosType,
    MIL_CONST_TEXT_PTR CharValidString
)
```

Description

This function specifies which character value constraints to apply to each position of the strings in the target image, when using the specified OCR font context. This function specifies which characters can appear at given positions in the string to be read, and associates these constraints with the OCR font context. Using this function increases both speed and reliability as long as the strings in the target image have a known format and obey certain grammatical rules.

Note that constraints are set in regards to the character's position in the string. When dealing with multiple strings, the same constraints are applied to each string.

If you change any constraints, use [MocrPreprocess\(\)](#) to speed up any following read or verify operation.

Parameters

FontId

Specifies the OCR font context identifier with which to associate the constraints.

CharPos

Specifies the character position in the string for which a constraint is being set. Valid values range from zero to the length of the string -1.

CharPosType

Specifies the type of character that can appear at the specified string position.

This parameter should be set to one of the following values:

● for specifying the type of character that can appear	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ANY .
<input type="checkbox"/> M_ANY	Sets all characters present in the font as valid.
<input type="checkbox"/> M_DIGIT	Sets only characters "0...9" (ASCII codes 48 to 57) as valid. (summarize)
<input type="checkbox"/> M_LETTER +	Sets only characters "A...Z" and "a...z" (ASCII codes 65 to 90 and 97 to 122) as valid. (summarize)

Combination constants for [M_LETTER](#);
You can add one of the following values to the above-mentioned value to specify the letter case of the characters accepted.

● For M_LETTER to specify which characters are accepted	
▢ Value	Description
▢ M_LOWERCASE	Sets only characters "a...z" (ASCII codes 97 to 122) as valid. (summarize)
▢ M_UPPERCASE	Sets only characters "A...Z" (ASCII codes 65 to 90) as valid. (summarize)

CharValidString

Specifies that any character matching the given **CharPosType** will be accepted as valid, or explicitly lists valid characters for the specified position.

● For specifying whether all characters matching the CharPosType will be accepted	
▢ Value	Description
▢ MIL_TEXT(MIL_TEXT_PTR <i>StringOfCharacters</i>)	Specifies an explicit list of valid characters for the specified position. The accepted characters must match the CharPosType parameter definition and must exist in the font. (summarize)
	<i>Parameters</i>
	<i>StringOfCharacters</i> Specifies an explicit list of valid characters for the specified position. This string must be null-terminated.
▢ M_NULL	Sets all font characters matching the CharPosType parameter value as valid.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrStream

Synopsis

Load, restore, or save an OCR font context from/to a file or a memory stream.

Syntax

```
void MocrStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load or restore an OCR font context from a file or memory stream. All of the OCR font context settings that were in effect when the OCR font context was saved will be restored. A loaded or restored OCR font context is not preprocessed, therefore you must call [MocrPreprocess\(\)](#) before performing a read operation with [MocrReadString\(\)](#).

Moreover, this function can also save an OCR font context to a file or memory stream. All information about the previously allocated OCR font context is saved, including the OCR font, target image information, and processing controls. However, preprocessing changes are not saved.

To inquire the number of bytes necessary to save an OCR font context to memory stream, you should first call this function ([MocrStream\(\)](#)) with [M_INQUIRE_SIZE_BYTE](#).

The content saved to memory stream is equivalent to the content saved to file. In addition, any file saved with [MocrSaveFont\(\)](#) is equivalent to a file saved using this function.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using [MocrStream\(\)](#), you can choose to save a backwards-compatible version of the OCR font context, which will work using a version of MIL that is up to one major release older than the current version (depending on which version is specified). For example, if you allocate a OCR font context using MIL 9.0 and save it to version 8.0, you can restore this context on a computer where MIL 8.0 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore OCR font contexts saved using MIL version 8.0 or above. Settings that do not exist in the lower version will be filled with default values when the OCR font context is loaded or restored.

Parameters

MemPtrOrFileName

Specifies the file or memory stream.

For specifying the file or memory stream		
Value	Description	
<div>MIL_TEXT(MIL_TEXT_PTR FileName)</div>	<div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving an OCR font context to a file, use the MFO file extension. The function handles (internally) the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</div> <div>To open or save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</div> <div>(summarize)</div>	
		Parameters

	<i>FileName</i> Specifies the drive, directory, and name of the file.
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE .
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. This parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (MocrStream()) first with M_INQUIRE_SIZE_BYTE . (summarize)

SystemId

Specifies the system on which to restore the OCR font context. For [M_INQUIRE_SIZE_BYTE](#), and [M_SAVE](#) operations, **SystemId** is ignored and should be set to [M_NULL](#). For an [M_RESTORE](#) operation, this parameter should be set to one of the following values:

● For specifying the system on which to restore the OCR font context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform on the OCR font context. This parameter must be set to one of the following values:

● For specifying the operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save an OCR font context to memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
<input type="checkbox"/> M_RESTORE	Restores an OCR font context from a file or memory stream and assigns it an identifier.
<input type="checkbox"/> M_SAVE	Saves an OCR font context to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the OCR font context. This parameter must be set to one of the following values:

● For specifying the type of stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream.
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)

Version

Specifies the MIL version of the OCR font context. This parameter must be set to one of the following values.

● For specifying the MIL version	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL. For an M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the OCR font context.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, **ContextIdPtr** specifies the address of the variable from which to read the OCR font context identifier.

For an [M_RESTORE](#) operation, **ContextIdPtr** specifies the address in which to return the identifier of the restored OCR font context. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the OCR font context, in bytes. If the size is not required, you can set this parameter to **M_NULL**.

Note that the size of an OCR font context will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

MocrVerifyString

Synopsis

Verify a known string in an image.

Syntax

```
void MocrVerifyString(  
    MIL_ID ImageBufId,  
    MIL_ID FontId,  
    MIL_CONST_TEXT_PTR String,  
    MIL_ID OcrResultId  
)
```

Description

This function determines whether a known string is present in the target image, and evaluates the quality of the string. These operations can be performed more quickly with **MocrVerifyString()** than with **MocrReadString()**. After verification, the specified result buffer can be read with the **MocrGetResult()** function.

If string location is disabled (using **MocrControl()** with **M_SKIP_STRING_LOCATION**), use a child buffer to limit the search area for best results.

The verification can only be performed on a single string and not multiple strings. When dealing with multiple lines of text in a target image, the returned string is the first one found within the target image. Note that this is not necessarily the first string within the target image. The string is located according to its OCR font context type, constraints, and processing controls. Once located, the string can then be read and compared against the validation string provided by this function.

For a more robust search, use an **M_CONSTRAINED** OCR font context. You can change the type of context using **MocrControl()** with **M_CONTEXT_CONVERT**.

Blank characters cannot be verified and will be ignored.

Parameters

- ImageBufId
- Specifies the identifier of the target image containing the string to be verified.
- FontId
- Specifies the identifier of the OCR font context to use to verify the string in the target image.
- String
- Specifies the character string to be verified in the sample target image.

● For specifying the ASCII characters	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR String)	Specifies the character string to be verified in the sample target image. This string must be null-terminated. Note that blank characters cannot be verified and will be ignored. (summarize)
	Parameters
	String Specifies the character string to be verified in the sample target image.

OcrResultId

Specifies the OCR result buffer in which to place the verification results.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
The OCR result buffer ([OcrResultId](#)) must be allocated on the same system as the OCR font context ([FontId](#)). If it is not, an error will occur.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milocr.lib.
DLL	Requires mil.dll; milocr.dll.

Mpat functions

Synopsis

The functions prefixed with Mpat make up the Pattern Matching module. The Pattern Matching module is a set of functions that uses normalized grayscale correlation (NGC) to search for occurrences of a pattern in an image.

Functions

- MpatAllocAutoModel
- MpatAllocModel
- MpatAllocResult
- MpatAllocRotatedModel
- MpatCopy
- MpatDraw
- MpatFindModel
- MpatFindMultipleModel
- MpatFree
- MpatGetNumber
- MpatGetResult
- MpatInquire
- MpatPreprocModel
- MpatRestore
- MpatSave
- MpatSetAcceptance
- MpatSetAccuracy
- MpatSetAngle
- MpatSetCenter
- MpatSetCertainty
- MpatSetDontCare
- MpatSetNumber
- MpatSetPosition
- MpatSetSearchParameter
- MpatSetSpeed
- MpatStream

MpatAllocAutoModel

Synopsis

Automatically allocate unique pattern matching models of the specified type, from a source image.

Syntax

```
MIL_ID MpatAllocAutoModel(
    MIL_ID SystemId,
    MIL_ID SrcImageBufId,
    MIL_INT SizeX,
    MIL_INT SizeY,
    MIL_INT PosUncertaintyX,
    MIL_INT PosUncertaintyY,
    MIL_INT ModelType,
    MIL_INT Mode,
    MIL_ID *ModelIdArrayPtr
)
```

Description

This function searches for the specified number of most-suitable unique areas, of the specified dimensions, in the model's source image. From each area found, the function automatically allocates a model. If none are found, no model is allocated and an error is reported. It can take several seconds to find the best models (more for large or small images).

To be effective, the model's target image should be a typical target image. Therefore, **MpatAllocAutoModel()** is useful, for example, when you want to perform whole image alignment, for which allocation of a unique model is essential.

You can determine the offset of a model's origin relative to its source image, using the **MpatInquire()** function.

If the eventual model can appear at an angle, in most cases, it is better to allocate an **M_NORMALIZED** + **M_CIRCULAR_OVERSCAN** type of model and then use **MpatSetAngle()** to specify the angular range.

You can change a model's search parameter settings at any time, using the appropriate **MpatSet...Q** function. When the model(s) is no longer required, you should release its memory, using **MpatFree()**.

You can only define models that respect the following condition: $(maxvalue^2 * SizeX * SizeY) < 2^{63}$ where *maxvalue* is the maximum pixel value (typically, 255) in the target image and the model. This restriction is imposed to avoid overflows in the internal 64-bit accumulators.

The total area of the defined model must be greater or equal to 4 pixels (**SizeX * SizeY** >= 4).

Parameters

SystemId

Specifies the system on which to allocate the models.

This parameter should be set to one of the following values:

For specifying the system identifier	
Value	Description
M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
MIL system identifier	Specifies a valid system identifier, which you have allocated using the MsysAlloc() function.

SrcImageBufId

Specifies the identifier of the image buffer from which to extract the models. This image should be representative of the target images. This function currently supports only 8-bit unsigned grayscale images.

SizeX

Defines the width of the required models.

SizeY

Defines the height of the required models.

PosUncertaintyX

Specifies the maximum displacement (shift) expected between the reference position of the models in the source image and their position when found in the target images in the horizontal direction. This information is used to select models that are far enough from the image borders to be present in the target images.

This parameter must be set to one of the values below.

● For specifying the maximum displacement in the X-direction	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Selects models from anywhere in the whole width of the image. The default search region of each model is automatically set using the original position of the model, plus or minus the specified positional uncertainty. (summarize)
<input type="checkbox"/> Value	Sets PosUncertaintyX to the expected maximum pixel displacement in the horizontal direction. This value will select models that are far enough from the image borders to be present in the target images. (summarize)

PosUncertaintyY

Specifies the maximum displacement (shift) expected between the reference position of the models in the source image and their position when found in the target images in the vertical direction. This information is used to select models that are far enough from the image borders to be present in the target images.

This parameter must be set to one of the values below.

● For specifying the maximum displacement in the Y-direction	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Selects models from anywhere in the whole height of the image. The default search region of each model is automatically set using the original position of the model, plus or minus the specified positional uncertainty. (summarize)
<input type="checkbox"/> Value	Sets PosUncertaintyY to the expected maximum pixel displacement in the vertical direction. This value will select models that are far enough from the image borders to be present in the target images. (summarize)

ModelType

Specifies the type of the model. The parameter must be set to the following:

● For specifying the type of model	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NORMALIZED +	Specifies a model used to search for the position, match score, and angle (if angular search is enabled) of a model occurrence in a target image, using MpatFindModel() and MpatFindMultipleModel() .

Combination constant for any of the possible values of the [ModelType](#) parameter

You can add the following value to the above-mentioned values to extract the model, as well as circular overscan data.

● For the <code>ModelType</code> parameter	
☐ Value	Description
☐ <code>M_CIRCULAR_OVERSCAN</code>	Extracts the model, as well as circular overscan data from the source image. The overscan data is determined by rotating the area from which to extract the model, about its center. The overscan data of an M_CIRCULAR_OVERSCAN type model is used if you search for an occurrence of the model at an angle. For this model, you cannot define "don't care" pixels (MpatSetDontCare()). (summarize)

Mode

Specifies the speed of the allocation process. This parameter must be set to one of the values below.

● For specifying the speed of the allocation	
☐ Value	Description
☐ <code>M_DEFAULT +</code>	Allocates the models at high-speed, the same as <code>M_FAST</code> .
☐ <code>M_BEST +</code>	Allocates the models with high precision.
☐ <code>M_FAST +</code>	Allocates the models at high-speed.

Combination constant for any of the possible values of the [Mode](#) parameter

You can add the following value to the above-mentioned values to set the number of models to allocate from the same image.

● For the <code>Mode</code> parameter	
☐ Value	Description
☐ <code>M_MULTIPLE + n</code>	Allocates several models from the same image. Set <i>n</i> to the number of models to allocate, where <i>n</i> cannot be > 15. For example, set Mode to <code>M_BEST + M_MULTIPLE + 4</code> to find and allocate the four most unique models available in your image. Models can overlap within the image by half the size of the model (in X and Y). (summarize)

ModelIdArrayPtr

Specifies the address of the array in which to write the model identifiers. The identifier is required to use a model with other pattern matching functions. Since `MpatAllocAutoModel()` also returns the model identifier, you can set this parameter to `M_NULL` when allocating a single model. If allocation fails, `M_NULL` is written as the identifier.

Return value

The returned value is the model identifier if the allocation is successful. If allocation fails, `M_NULL` is returned.

Default values

Models are associated with some default search parameters. For an [M_NORMALIZED](#) model type, the model's search parameters are set to the following defaults:

Characteristic	Default Value
Search region	The default search region of each model is automatically set using the original position of the model, plus or minus the specified positional uncertainty.
Positional accuracy	M_MEDIUM (typically ± 0.10 pixels)
Search number	1
Search speed	M_MEDIUM
Acceptance level	70%
Certainty level	80%
Search angle	0°
Center of model (reference position)	A <i>MIL_DOUBLE</i> value equal to: $((\text{SizeX} - 1) / 2.0, (\text{SizeY} - 1) / 2.0)$ (relative to the model origin).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatAllocModel

Synopsis

Allocate a pattern matching model from a source image.

Syntax

```
MIL_ID MpatAllocModel(
    MIL_ID SystemId,
    MIL_ID SrcImageBufId,
    MIL_INT OffX,
    MIL_INT OffY,
    MIL_INT SizeX,
    MIL_INT SizeY,
    MIL_INT ModelType,
    MIL_ID *ModelIdPtr
)
```

Description

This function allocates a model, using data from the specified area of the model's source image.

You can change a model's search parameter settings at any time, using the appropriate **MpatSet...()** function. When the model is no longer required, you should release its memory, using **MpatFree()**.

An **M_NORMALIZED** type model (without overscan data) is usually used for an image with an inconsistent surrounding region, such as an image of loose nuts and bolts lying on a metal sheet. To perform an angular search, define the angular range in which the model can appear using **MpatSetAngle()**. When preprocessing an **M_NORMALIZED** model (without overscan data) for which an angular search range is specified, rotated versions of the model are created assigning "don't care" pixels to regions that do not have corresponding data in the original model.

Angular search is fastest when performed with an **M_CIRCULAR_OVERSCAN** model; however, **M_CIRCULAR_OVERSCAN** should only be used when the region around the model is consistent, for example, when searching for a chip in the image of an integrated circuit.

When preprocessing an **M_CIRCULAR_OVERSCAN** model for which an angular search range has been specified, a set of models is extracted from rotated versions of the **M_CIRCULAR_OVERSCAN** model, creating models that would appear upright if the target image were rotated. For this type of model, a larger region than the one defined is extracted from the model's source image so as to allow creation of models at different angles.

You can only define models that respect the following condition: $(maxvalue^2 * SizeX * SizeY) < 2^{63}$ where *maxvalue* is the maximum pixel value (typically, 255) in the target image and the model. This restriction is imposed to avoid overflows in the internal 64-bit accumulators.

The total area of the defined model must be greater or equal to 4 pixels (**SizeX * SizeY** >= 4).

Parameters

SystemId

Specifies the system on which to allocate the model.

This parameter should be set to one of the following values:

● For specifying the system identifier	
☐ Value	Description
☐ M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
☐ MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

SrcImageBufId

Specifies the identifier of the image buffer from which to extract the model (that is, the model's source image). This function currently supports only 8-bit unsigned grayscale images.

OffX

Specifies the coordinates of the model origin in the source image ([SrcImageBufId](#)). The specified model position ([OffX](#)) must be valid in the source image.

OffY

Specifies the coordinates of the model origin in the source image ([SrcImageBufId](#)). The specified model position ([OffY](#)) must be valid in the source image.

SizeX



Specifies the width of the model. The specified model width ([SizeX](#)) must be valid in the source image.

SizeY

Specifies the height of the model. The specified model height ([SizeY](#)) must be valid in the source image.



ModelType

Specifies the type of the model. This parameter must be set to the following:

● For specifying the type of model	
 Value	Description
 M_NORMALIZED +	Specifies a model used to search for the position, match score, and angle (if angular search is enabled) of a model occurrence in a target image, using MpatFindModel() and MpatFindMultipleModel() .

Combination constant for any of the possible values of the [ModelType](#) parameter

You can add the following value to the above-mentioned values to extract the model, as well as circular overscan data.

● For the ModelType parameter	
 Value	Description
 M_CIRCULAR_OVERSCAN	Extracts the model, as well as circular overscan data from the source image. The overscan data is determined by rotating the area from which to extract the model, about its center. The overscan data of an M_CIRCULAR_OVERSCAN type model is used if you search for an occurrence of the model at an angle. The model must not be extracted from a region too close to the edge of the model's source image. In addition, the M_CIRCULAR_OVERSCAN model should be complex enough so that if models are created from it at the required angles, they are representative of the pattern being sought. For this model, you cannot define "don't care" regions (MpatSetDontCare()). (summarize)

ModelIdPtr

Specifies the address of the variable in which to write the model identifier. This identifier is required to use the model with other pattern matching functions. Since [MpatAllocModel\(\)](#) also returns the model identifier, you can set this parameter to [M_NULL](#).

Return value

The returned value is the model identifier if the allocation is successful. If allocation fails, [M_NULL](#) is returned.

Default values

Models are associated with some default search parameter settings. For an [M_NORMALIZED](#) model type, the model's search parameter settings are set to the following defaults:

Characteristic	Default Value
Positional accuracy	M_MEDIUM (typically ± 0.10 pixels)

Positional uncertainty (search region)	M_ALL (full image)
Search number	1
Search speed	M_MEDIUM
Acceptance level	70%
Certainty level	80%
Search angle	Disabled
Center of model (reference position)	A double value equal to: ((SizeX - 1) / 2.0, (SizeY - 1) / 2.0) (relative to the model origin).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatAllocResult

Synopsis

Allocate a pattern matching result buffer.

Syntax

```
MIL_ID MpatAllocResult(  
    MIL_ID SystemId,  
    MIL_INT NbEntries,  
    MIL_ID *PatResultIdPtr  
)
```

Description

This function allocates a result buffer with the specified number of entries. When the result buffer is no longer required, release its memory, using [MpatFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the result buffer.

This parameter should be set to one of the following values:

● For the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

NbEntries

Specifies the number of result entries to allocate. If [NbEntries](#) is set to [M_DEFAULT](#), the number of entries will be allocated dynamically to match the number of actual occurrences found at runtime; used in conjunction with an [M_ALL](#) search ([MpatSetNumber\(\)](#)), this provides an efficient method of allocating the correct size result buffer.

This parameter must be set to one of the values below.

● For specifying the number of result entries	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the number of entries will be allocated dynamically to match the number of actual occurrences found at runtime.
<input type="checkbox"/> Value	Specifies the number of result entries to allocate. Value should be less than or equal to the number of occurrences specified with MpatSetNumber() . (summarize)

PatResultIdPtr

Specifies the address of the variable in which the pattern matching result buffer identifier is to be written. Since the **MpatAllocResult()** function also returns the pattern matching result buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatAllocRotatedModel

Synopsis

Rotate a pattern matching model.

Syntax

```
MIL_ID MpatAllocRotatedModel(  
    MIL_ID SystemId,  
    MIL_ID SrcModelId,  
    MIL_DOUBLE Angle,  
    MIL_INT InterpolationMode,  
    MIL_INT ModelType,  
    MIL_ID *NewModelIdPtr  
)
```

Description

This function allocates a new model and initializes it with the rotated version of the specified source model.

This function has the same effect as creating an [M_NORMALIZED](#) model (without [M_CIRCULAR_OVERSCAN](#)) for which a single angle is specified (that is, delta minimum and maximum are set to 0).

Note that the allocated model size is usually greater than the size of its source, due to the rectangular shape of the model and the nature of the rotation operation. The additional pixel locations produced by the rotation, which have no corresponding pixels in the source model, are set to "don't care" values. The size (X and Y) of the new model can be inquired, using [MpatInquire\(\)](#).

Note that the settings of the source model are copied to the new model. You can change any of the model's search parameters (using any appropriate **MpatSet...Q** function), except its angle ([MpatSetAngle\(\)](#) function).

Once you have a rotated version of a model, if you no longer require the original source model, it can be deleted with [MpatFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the rotated model.

This parameter should be set to one of the values in the following table.

● For specifying the system identifier	
☐ Value	Description
☐ M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
☐ MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

SrcModelId

Specifies the identifier of the source model. It must be an [M_NORMALIZED](#) model without overscan data. A new rotated model is generated from this model.

Angle

Specifies the angle at which to rotate the model in a counter-clockwise direction. This parameter can be set to any value from 0° to 360°.

InterpolationMode

Specifies the mode of interpolation. This parameter must be set to one of the values below.

For specifying interpolation mode	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_NEAREST_NEIGHBOR .
<input type="checkbox"/> M_BICUBIC	Specifies bicubic interpolation.
<input type="checkbox"/> M_BILINEAR	Specifies bilinear interpolation.
<input type="checkbox"/> M_NEAREST_NEIGHBOR	Specifies nearest neighbor interpolation.

ModelType

Specifies the type of the new model. This parameter must be set to the value below.

For specifying the type of model	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NORMALIZED	Searches for the position and match score of a model occurrence in a target image, using MpatFindModel() and MpatFindMultipleModel() .

NewModelIdPtr

Specifies the address of the variable in which to write the rotated model identifier. Since the [MpatAllocRotatedModel\(\)](#) function also returns the model identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the model identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Default values

Models are associated with a set of default search parameters. For an [M_NORMALIZED](#) model type, the model's search parameters are set to the defaults in the table that follows.

Characteristic	Default Value
Positional accuracy	M_MEDIUM (typically ± 0.10 pixels)
Positional uncertainty (search region)	M_ALL (full image)
Search number	1
Search speed	M_MEDIUM
Acceptance level	70%
Certainty level	80%
Search angle	0°
Center of model (reference position)	A double value equal to $((\text{SizeX} - 1) / 2, (\text{SizeY} - 1) / 2)$ (relative to the model origin).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatCopy

Synopsis

Copy a pattern matching model to an image buffer.

Syntax

```
void MpatCopy(
    MIL_ID ModelId,
    MIL_ID DestImageBufId,
    MIL_INT CopyMode
)
```

Description

This function copies the specified model to the specified destination image buffer, starting at the top-left corner of the buffer. Once the model is copied to the destination buffer, it can then be displayed (if the destination buffer is displayable).

By making two calls to this function, one in which `M_DEFAULT` is used as the copy mode and the other in which `M_DONT_CARE` is used, it is possible to achieve the effect of overlaying the "don't care" pixels onto the original model.

Parameters

ModelId

Specifies the identifier of the model that you want copied to the image buffer.

DestImageBufId

Specifies the identifier of the destination image buffer in which to place the model. You must ensure that the buffer is at least as large as the model. Note that the function only supports 8-bit unsigned grayscale images.

CopyMode

Specifies how the model will be copied. This parameter must be set to one of the values below.

For specifying how to copy the model	
Value	Description
M_DEFAULT	Copies the portion of the source image from which the model was extracted to the destination buffer. That is, the model as it was first defined before any "don't care" pixels where associated with it (see <code>MpatSetDontCare()</code>). Overscan data is not copied. (summarize)
M_DONT_CARE +	Copies only the "don't care" pixels of the model to the destination buffer. When copied, these pixels are given the value zero. To give these pixels a value other than zero, add the value to the <code>M_DONT_CARE</code> copy mode. Note, by default when using the <code>M_DONT_CARE</code> copy mode, any pixel value other than the "don't care" pixels in the destination image buffer will not be overwritten. (summarize)

Combination constant for `M_DONT_CARE`;

You can add the following value to the above-mentioned value to set all pixels in the destination buffer that are not "don't care" pixels to zero.

For M_DONT_CARE to affect the destination buffer	
Value	Description
M_CLEAR_BACKGROUND	Clears pixels in the resulting destination buffer that are not "don't care" pixels to zero.

Example

The following example demonstrates how to copy "don't care" pixels as value 255 and set the other pixels in the destination buffer to zero.

```
MpatCopy(ModelId, ImageBufId, M_DONT_CARE +255 +M_CLEAR_BACKGROUND);
```

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatDraw

Synopsis

Draw specific features of the model or result occurrence in the destination image buffer.

Syntax

```
void MpatDraw(
    MIL_ID GraphContId,
    MIL_ID ModelOrResultId,
    MIL_ID DestImageId,
    MIL_INT Operation,
    MIL_INT Index,
    MIL_INT ControlFlag
)
```

Description

This function draws specific model or result occurrence features in the destination buffer. **MpatDraw()** can be used with multiple results.

Parameters

GraphContId

Specifies the graphics context to use. This parameter must be set to one of the following values:

For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

ModelOrResultId

Specifies the model or result buffer identifier from which to extract the features to draw.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
The model or result buffer must be allocated on the same system as the graphics context ([GraphContId](#)). If it is not, an error will occur.

DestImageId

Specifies the identifier of the destination image buffer into which to draw. You can also choose to annotate an image non-destructively, by drawing into its display's overlay buffer.

Operation

Specifies the type of operation to perform. Operations can be added together to draw multiple features at a time. For example, to draw both the result occurrence's bounding box and position, you would specify [M_DRAW_BOX](#) + [M_DRAW_POSITION](#) as the [Operation](#) parameter. The [Operation](#) parameter can be set to one or a combination of the values below.

For specifying the type of operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_BOX	Draws the bounding box of the model or of the result occurrence(s). For a result occurrence, the box is drawn at the coordinates at which its origin was found in the target image (not the reference position), respecting the occurrence's angle. (summarize)

<input type="checkbox"/> M_DRAW_DONT_CARES	Draws the model's don't care pixels (for models only).
<input type="checkbox"/> M_DRAW_IMAGE	Draws the region of the model source image from which the model was extracted. (equivalent to MpatCopy()) (for models only). (summarize)
<input type="checkbox"/> M_DRAW_POSITION	Draws a cross at the model's reference position, or at the result occurrence(s) position and angle found in the target image (with respect to the reference position).

Index

Specifies the index of the result occurrence to draw. **M_DEFAULT** always refers to all results. Note, when drawing features of a model, the index must be set to **M_DEFAULT**.

ControlFlag

Specifies where to draw a model. This parameter must be set to one of the values below.

● For specifying where to draw the model	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies to draw the model at the top-left corner of the destination image.
<input type="checkbox"/> M_ORIGINAL	Specifies to draw the model at the offsets used to define the model region in the source image.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatFindModel

Synopsis

Find a pattern matching model in the target image buffer.

Syntax

```
void MpatFindModel(
    MIL_ID ImageBufId,
    MIL_ID ModelId,
    MIL_ID PatResultId
)
```

Description

This function finds occurrences of the specified model in the given image and returns the position of each occurrence. The search is performed using the model's current search parameters. Note, this function assumes that the model has been preprocessed ([MpatPreprocModel\(\)](#)).

This function assumes that the model was extracted from a source image with the same scaling as the target image. In addition, if using a normalized grayscale non-rotational model, it assumes that the source image and the model's target image are of the same orientation (with approx. 5° tolerance). For angular search, use [MpatSetAngle\(\)](#) to set the angular search range of the model.

The model's search parameters specify the maximum number of occurrences for which to look in the target image ([MpatSetNumber\(\)](#)).

If a correlation has a match score greater than or equal to the certainty level ([MpatSetCertainty\(\)](#)), it is automatically considered an occurrence (default 80%). The remaining occurrences will be the best of those greater than or equal to the acceptance level ([MpatSetAcceptance\(\)](#)).

Results are written to the specified result buffer. Use the **MpatGet...()** functions to read the results. See [MpatAllocResult\(\)](#) for more information on allocating result buffers.

This function returns results in decreasing match-score order. This means that the most-likely occurrence is always returned first.

Patterns that are very close to the edge of the image might be found with lower match scores than usual due to edge effects. For this reason, you should use [MpatSetPosition\(\)](#), rather than a child buffer, to restrict the search to a portion of the image.

Parameters

ImageBufId

Specifies the identifier of the target image. Note that this function only supports 8-bit unsigned grayscale images.

ModelId

Specifies the identifier of the model for which to search in the specified target image buffer.

PatResultId

Specifies the identifier of the pattern matching result buffer in which to store results.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The pattern matching result buffer (**PatResultId**) must be allocated on the same system as the model (**ModelId**). If it is not, an error will occur.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatFindMultipleModel

Synopsis

Find multiple pattern matching models in the target image buffer.

Syntax

```
void MpatFindMultipleModel(
    MIL_ID ImageBufId,
    const MIL_ID *ModelIdArray,
    const MIL_ID *PatResultIdArray,
    MIL_INT NumModels,
    MIL_INT SearchMode
)
```

Description

This function finds occurrences of the specified models in the given image and returns the position of each occurrence for each model or of the best matches from the group of models. If you want to search for several models in a given image region, a single call to this function is more efficient than multiple calls to [MpatFindModel\(\)](#). This function assumes all models have been preprocessed, and requires that all models be of the same size.

This function assumes that each model was extracted from an image with the same scaling as the target image. In addition, if using a normalized grayscale non-rotational model, it assumes that the model's source image and the target image are at the same orientation (with approximately 5° tolerance). For an angular search, use [MpatSetAngle\(\)](#) to set the angular search range of the model.

Results are written to the specified result buffers. This function returns results in decreasing match-score order. This means that the most-likely occurrence is always returned first. Use the **MpatGet...Q** functions to read the results. See [MpatAllocResult\(\)](#) for more information on allocating result buffers.

Patterns that are very close to the edge of the image might be found with lower match scores than usual due to edge effects. For this reason, you should use [MpatSetPosition\(\)](#), rather than a child buffer, to restrict the search to a portion of the image. Note that all models must use the same search region.

Parameters

- ImageBufId**
- Specifies the identifier of the target image buffer. Note that this function only supports 8-bit unsigned grayscale images.
- ModelIdArray**
- Specifies the address of the one-dimensional array that contains the identifiers of the models for which to search in the specified target image buffer. Note, all models must be of the same size and must use the same search region in the target image.
- PatResultIdArray**
- Specifies the address of the one-dimensional array that contains the identifiers of the pattern matching result buffers in which to store results.
- [Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

All pattern matching result buffers and models must be allocated on the same system. If they are not, an error will occur.
- NumModels**
- Specifies the number of models.
- SearchMode**
- Specifies the search mode. This parameter must be set to:

● For specifying the search mode

Value	Description	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_FIND_ALL_MODELS	Searches for all the specified models in the target image and writes the result of each model search in the corresponding buffer. The search is performed using each model's current search parameters. Each model's search parameters also determine the maximum number of occurrences for which to look in the target image (MpatSetNumber()). If a correlation has a match score greater than or equal to the model's certainty level (MpatSetCertainty()), it is automatically considered an occurrence (default 80%), the remaining occurrences will be the best of those greater than or equal to the acceptance level (MpatSetAcceptance()). Note, the results for the model in the ModelIdArray [n] are written in the corresponding result buffer PatResultIdArray [n]. Therefore, you must specify the same number of result buffers as the number of models. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
M_FIND_BEST_MODELS	Searches for the specified models in the target image and writes the results which have the highest match scores in a single result buffer (PatResultIdArray [0]). Note that the number of different matches that are found depends on MpatSetNumber() , MpatSetCertainty() , and MpatSetAcceptance() of the first model in the array but cannot exceed 100 occurrences. The certainty, speed and all other search parameters are also specified by the first model in the array; the search parameter settings of the other models are ignored. The model index of the model that generated the best match scores can be read from the result buffer by setting the MpatGetResult() ResultType parameter to M_MODEL_INDEX . You cannot use M_FIND_BEST_MODELS to perform an angular search (MpatSetAngle()). (summarize)	a				e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatFree

Synopsis

Free a pattern matching model or a result buffer.

Syntax

```
void MpatFree(  
    MIL_ID PatId  
)
```

Description

This function deletes the specified pattern matching model or result buffer identifier and releases any memory associated with it.

Parameter

PatId

Specifies the identifier of the pattern matching model or result buffer to free. These must have been successfully allocated (with [MpatAllocModel\(\)](#), [MpatAllocAutoModel\(\)](#), or [MpatAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatGetNumber

Synopsis

Get the number of model occurrences found in the target image.

Syntax

```
MIL_INT MpatGetNumber(  
    MIL_ID PatResultId,  
    MIL_INT *CountPtr  
)
```

Description

This function retrieves the number of matches found after searching for a model. It returns the number of occurrences of the model that were found with match scores greater than or equal to the model's acceptance level to a maximum of [MpatSetNumber\(\)](#). The returned number will never be bigger than the number of entries allocated for the result buffer.

This function determines the number of results retrieved with [MpatGetResult\(\)](#). Call **MpatGetNumber()** prior to checking results with [MpatGetResult\(\)](#). Note that when no occurrences of the model are found, there is no need to check the results.

Parameters

- PatResultId
Specifies the identifier of the pattern matching result buffer that was used to store results obtained by [MpatFindModel\(\)](#).
- CountPtr
Specifies the address of the variable in which to write the number of occurrences. Since the **MpatGetNumber()** function also returns the requested information, you can set this parameter to **M_NULL**.

Return value

The returned value is the number of occurrences of the model that were found and that were greater than or equal to the acceptance level.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatGetResult

Synopsis

Get the specified type of result(s) from a pattern matching result buffer.

Syntax

```
void MpatGetResult(
    MIL_ID PatResultId,
    MIL_INT ResultType,
    void *UserArrayPtr
)
```

Description

This function retrieves the result(s) of the specified type from a specified pattern matching result buffer. Results are only available after calling [MpatFindModel\(\)](#) or [MpatFindMultipleModel\(\)](#).

Note that if your target image was associated with a calibration object, results are, by default, returned with respect to the real-world coordinate system. To have positional and dimensional results returned in pixel units, use [McalControl\(\)](#) with the [M_OUTPUT_UNITS](#) control type set to [M_PIXEL](#). If the target image was not calibrated, positional and dimensional results are returned in pixels, relative to the top-left pixel in the target image.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [UserArrayPtr](#) to [M_NULL](#). When this parameter is set to [M_NULL](#), the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [MpatGetResult\(\)](#) again and you pass an array to the parameter [UserArrayPtr](#). You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

By setting the [ResultType](#) parameter to [M_INTERACTIVE](#), you can view the results interactively.

Parameters

PatResultId

Specifies the identifier of the pattern matching result buffer from which to retrieve result(s).

ResultType

Specifies the type of result to retrieve for each model occurrence or opens a dialog box that displays all the results stored in the result buffer interactively.

To display the features currently stored in the result buffer in an interactive dialog box, select the following.

Unless otherwise specified, the following values require that you pass the [UserArrayPtr](#) parameter [M_NULL](#).

● For displaying an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	<div>[This is only applicable to Windows]</div> <div>Opens a dialog box containing a spreadsheet that displays the types of results stored in the result buffer. The results are displayed in separate columns. If there are queued requests, only those results are displayed.</div> <div>(summarize)</div>

To specify the type of result for each model occurrence this parameter must be set to one of the following.

Unless otherwise specified, the following values require that you pass the [UserArrayPtr](#) parameter the address of an array of type [MIL_DOUBLE](#) with a size equal to the same number of elements returned by [MpatGetNumber\(\)](#) or the address of a [MIL_DOUBLE](#) (for a single result).

● For specifying the results

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ANGLE	Retrieves the angle of the occurrence (if found) using an MpatFindModel() search at the specified angle.
<input type="checkbox"/> M_MODEL_INDEX	Retrieves the index of the model, in the model identifier array, for which the occurrence was found when using MpatFindMultipleModel() with M_FIND_BEST_MODELS .
<input type="checkbox"/> M_NUMBER_OF_PIXELS	Retrieves the number of pixels in the model used to compute the correlation function. Note that to retrieve this result type, the sums must be saved by using MpatSetSearchParameter() with M_SAVE_SUMS set to M_ENABLE . (summarize)
<input type="checkbox"/> M_POSITION_X	Retrieves the X-coordinate of the occurrence.
<input type="checkbox"/> M_POSITION_Y	Retrieves the Y-coordinate of the occurrence.
<input type="checkbox"/> M_SCORE	Retrieves the match score of the occurrence, as a percentage.
<input type="checkbox"/> M_SUM_I	Retrieves the sum of the values of the pixels in the image over the N pixels corresponding to the model occurrence. Note that to retrieve this result type, the sums must be saved by using MpatSetSearchParameter() with M_SAVE_SUMS set to M_ENABLE . (summarize)
<input type="checkbox"/> M_SUM_II	Retrieves the sum of the squares of the values of the pixels in the image over the N pixels corresponding to the model occurrence. Note that to retrieve this result type, the sums must be saved by using MpatSetSearchParameter() with M_SAVE_SUMS set to M_ENABLE . (summarize)
<input type="checkbox"/> M_SUM_IM	Retrieves the sum of the products of the values of the pixels in the image and the pixels in the model over the N pixels corresponding to the model occurrence. Note that to retrieve this result type, the sums must be saved by using MpatSetSearchParameter() with M_SAVE_SUMS set to M_ENABLE . (summarize)
<input type="checkbox"/> M_SUM_M	Retrieves the sum of the values of the pixels in the model over the N pixels of the model. Note that to retrieve this result type, the sums must be saved by using MpatSetSearchParameter() with M_SAVE_SUMS set to M_ENABLE . (summarize)
<input type="checkbox"/> M_SUM_MM	Retrieves the sum of the squares of the values of the pixels in the model over the N pixels of the model. Note that to retrieve this result type, the sums must be saved by using MpatSetSearchParameter() with M_SAVE_SUMS set to M_ENABLE . (summarize)

UserArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type MIL_DOUBLE
- M_NULL
- MIL_DOUBLE

Specifies the address of the one-dimensional array in which to write the specified results. The array must be big enough to hold the number of results indicated by [MpatGetNumber\(\)](#).

Set this parameter to [M_NULL](#) if you are setting the [ResultType](#) parameter to [M_INTERACTIVE](#) or if you are putting the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatInquire

Synopsis

Inquire about the pattern matching model or the result buffer parameter setting.

Syntax

```
MIL_INT MpatInquire(
    MIL_ID PatId,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function returns information about the specified model or result buffer. It is useful to determine, for example, the size of a model that has been restored from disk and its position in the model's source image.

The position ([M_ORIGINAL_X](#) and [M_ORIGINAL_Y](#)) can be directly compared with search results (by [MpatFindModel\(\)](#) or [MpatFindMultipleModel\(\)](#)), to calculate the displacement of target images relative to the model's source image.

Parameters

PatId

Specifies the identifier of the model or result buffer for which to read the information.

InquireType

Specifies the parameter about which to inquire. For a model identifier, this parameter can be set to one of the values below.

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

For a model identifier	
Value	Description
<input type="checkbox"/> M_ACCEPTANCE_THRESHOLD +	Returns the minimum acceptable match score to be considered as an occurrence of the model. (summarize) <div><i>UserVarPtr info</i> Return values: Please see the AcceptanceThreshold parameter of MpatSetAcceptance(). (details)</div>
<input type="checkbox"/> M_ALLOC_OFFSET_X +	Returns the X-offset of model's top-left corner relative to the top-left corner of the model's source image (set using MpatAlloc...()). (summarize)
<input type="checkbox"/> M_ALLOC_OFFSET_Y +	Returns the Y-offset of model's top-left corner relative to the top-left corner of the model's source image (set using MpatAlloc...()). (summarize)
<input type="checkbox"/> M_ALLOC_SIZE_X +	Returns the model width (set using MpatAlloc...()). (summarize)
<input type="checkbox"/> M_ALLOC_SIZE_Y +	Returns the model height (set using MpatAlloc...()). (summarize)
<input type="checkbox"/> M_ALLOC_TYPE +	Returns the model type (set using MpatAlloc...()). (summarize)
<input type="checkbox"/> M_CENTER_X +	Returns the X-offset of the model's reference position relative to the top-left corner of model. (summarize)

	<i>UserVarPtr info</i> Return values: Please see the OffX parameter of MpatSetCenter() . (details)
M_CENTER_Y +	Returns the Y-offset of the model's reference position relative to the top-left corner of model. (summarize)
	<i>UserVarPtr info</i> Return values: Please see the OffY parameter of MpatSetCenter() . (details)
M_CERTAINTY_THRESHOLD +	Returns the match score at which an occurrence is assumed, without looking for better matches elsewhere in the image. (summarize)
	<i>UserVarPtr info</i> Return values: Please see the CertaintyThreshold parameter of MpatSetCertainty() . (details)
M_COARSE_SEARCH_ACCEPTANCE +	Returns the minimum acceptable match score at all levels except the last level, to be considered as an occurrence of the model. (summarize)
	<i>UserVarPtr info</i> Return values: 1.0 to 100.0; M_DEFAULT; (details)
M_EXTRA_CANDIDATES +	Returns the number of extra candidates to consider as possible candidates. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
M_FAST_FIND +	Returns whether forcing or preventing fast peak finding is used. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
M_FIRST_LEVEL +	Returns the resolution level for the initial stage of the search. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 7; M_DEFAULT; (details)
M_LAST_LEVEL +	Returns the resolution level for the final stage of the search. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 7; M_DEFAULT; (details)
M_MIN_SPACING_X +	Returns the minimum spacing (in X) between two models in order for them to be considered distinct. (summarize)
	<i>UserVarPtr info</i> Return values: 1.0 to 100.0; (details)
M_MIN_SPACING_Y +	Returns the minimum spacing (in Y) between two models in order for them to be considered distinct. (summarize)
	<i>UserVarPtr info</i> Return values: 1.0 to 100.0; (details)
M_MODEL_STEP +	Returns whether all or every second model pixel is used in the high-resolution stage of the search. (summarize)
	<i>UserVarPtr info</i> Return values: 1; 2; M_DEFAULT; (details)
M_NUMBER_OF_OCCURRENCES +	Returns the number of model occurrences for which to search in the target image. (summarize)

	<i>UserVarPtr info</i> Return values: 1 <= Value <= 1000; M_ALL; (details)
<input type="checkbox"/> M_ORIGINAL_X +	Returns the x-offset of the model's reference position relative to the top-left corner of the model's source image (takes into account the MpatSetCenter() setting).
<input type="checkbox"/> M_ORIGINAL_Y +	Returns the y-offset of the model's reference position relative to the top-left corner of the model's source image (takes into account the MpatSetCenter() setting).
<input type="checkbox"/> M_POSITION_ACCURACY +	Returns the search position accuracy. (summarize)
	<i>UserVarPtr info</i> Return values: M_HIGH; M_LOW; M_MEDIUM; (details)
<input type="checkbox"/> M_POSITION_START_X +	Returns the x-coordinate of search region origin within target image. (summarize)
	<i>UserVarPtr info</i> Return values: M_ALL; M_NO_CHANGE; Value; (details)
<input type="checkbox"/> M_POSITION_START_Y +	Returns the y-coordinate of search region origin within target image. (summarize)
	<i>UserVarPtr info</i> Return values: M_ALL; M_NO_CHANGE; Value; (details)
<input type="checkbox"/> M_POSITION_UNCERTAINTY_X +	Returns the search region width. (summarize)
	<i>UserVarPtr info</i> Return values: M_ALL; M_NO_CHANGE; Value; (details)
<input type="checkbox"/> M_POSITION_UNCERTAINTY_Y +	Returns the search region height. (summarize)
	<i>UserVarPtr info</i> Return values: M_ALL; M_NO_CHANGE; Value; (details)
<input type="checkbox"/> M_PREPROCESSED +	Returns whether or not the model is preprocessed. (summarize)
	<i>UserVarPtr info</i> Return values: 0 The model is not preprocessed. 1 The model is preprocessed.
<input type="checkbox"/> M_PROC_FIRST_LEVEL +	Returns the default resolution level for the initial stage (lowest resolution) of the search. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 7 The lowest resolution level.
<input type="checkbox"/> M_PROC_LAST_LEVEL +	Returns the default level for the final stage (highest resolution) of the search. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 7 The highest resolution level.
<input type="checkbox"/> M_SEARCH_ANGLE +	Returns the value of the initial search angle. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 360.0; (details)
<input type="checkbox"/> M_SEARCH_ANGLE_ACCURACY +	Returns the angular accuracy. (summarize)

	<i>UserVarPtr info</i> Return values: 0.1 to 180.0; M_DISABLE; (details)
<input type="checkbox"/> M_SEARCH_ANGLE_DELTA_NEG +	Returns the difference that determines the lower limit of the search angle's range. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 180.0; (details)
<input type="checkbox"/> M_SEARCH_ANGLE_DELTA_POS +	Returns the difference that determines the upper limit of the search angle's range. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 180.0; (details)
<input type="checkbox"/> M_SEARCH_ANGLE_INTERPOLATION_MODE +	Returns the interpolation mode. (summarize)
	<i>UserVarPtr info</i> Return values: M_BICUBIC; M_BILINEAR; M_NEAREST_NEIGHBOR; (details)
<input type="checkbox"/> M_SEARCH_ANGLE_MODE +	Returns the state of angular search mode. (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SEARCH_ANGLE_TOLERANCE +	Returns the full range of degrees within which the pattern in the target image can be rotated from a model at a specific angle and still meet the acceptance level. (summarize)
	<i>UserVarPtr info</i> Return values: 0.1 to 180.0; (details)
<input type="checkbox"/> M_SPEED +	Returns the model search speed. (summarize)
	<i>UserVarPtr info</i> Return values: M_HIGH; M_LOW; M_MEDIUM; M_VERY_HIGH; M_VERY_LOW; (details)

For a result buffer identifier, this parameter can be set to one of the values below.

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For a result buffer identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NUMBER_OF_ENTRIES +	Returns the number of entries allocated in the result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)
<input type="checkbox"/> M_SAVE_SUMS +	Returns whether the values of the sums used to compute the normalized correlation function for each model occurrence are saved. (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_TARGET_CACHING +	Returns whether the pyramidal representation of the buffer is kept in the result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)

For inquiring the system on which the model or result buffer is allocated, set this parameter to the following value.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

● For inquiring the system on which the model or result buffer is allocated		
Value	Description	
M_OWNER_SYSTEM +	Returns the system on which the model or result buffer is allocated. (summarize)	
		<i>UserVarPtr info</i> Return values: MIL system identifier; M_DEFAULT_HOST; (details)

Combination constants for any of the possible values of the [InquireType](#) parameter

You can add one of the following values to the above-mentioned values to cast the requested information to the required data type.

● For casting information to the required data type.		
Value	Description	
M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . (summarize)	
		<i>UserVarPtr info</i> Data type: double
M_TYPE_LONG	Casts the requested information to a <i>long</i> . (summarize)	
		<i>UserVarPtr info</i> Data type: long
M_TYPE_MIL_DOUBLE	Casts the requested information to a <i>MIL_DOUBLE</i> . (summarize)	
		<i>UserVarPtr info</i> Data type: MIL_DOUBLE
M_TYPE_MIL_INT	Casts the requested information to a <i>MIL_INT</i> . (summarize)	
		<i>UserVarPtr info</i> Data type: MIL_INT
M_TYPE_MIL_INT32	Casts the requested information to a <i>MIL_INT32</i> . (summarize)	
		<i>UserVarPtr info</i> Data type: MIL_INT32
M_TYPE_MIL_INT64	Casts the requested information to a <i>MIL_INT64</i> . (summarize)	
		<i>UserVarPtr info</i> Data type: MIL_INT64

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- double
- long
- MIL_DOUBLE
- MIL_INT

- MIL_INT32
- MIL_INT64

Specifies the address in which to write the requested information. Since this function also returns the requested information, you can set this parameter to M_NULL.

Return value

The returned value is the requested system information cast to *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatPreprocModel

Synopsis

Preprocess a pattern matching model.

Syntax

```
void MpatPreprocModel(
    MIL_ID TypicalImageBufId,
    MIL_ID ModelId,
    MIL_INT Mode
)
```

Description

This function preprocesses the specified model. It trains the system to search for the model in the most efficient manner (optionally within a specified typical image). The procedure is potentially quite lengthy (up to several seconds).

Call this function after all search parameters have been set. When you save, the model's preprocessing changes are stored with the model. Upon restoration, the model need not be preprocessed.

Note that if some of the model's search parameters are changed after a call to **MpatPreprocModel()**, the model must be preprocessed again. To inquire if your model is in a preprocessed state, use [MpatInquire\(\)](#) with **M_PREPROCESSED**.

Parameters

- TypicalImageBufId
- Specifies the identifier of a typical target image. The specified typical image will be used to refine and adapt the model to search on this typical background. You should only specify an image buffer if the model will always appear on such a background; otherwise, set this parameter to **M_NULL**.
- ModelId
- Specifies the identifier of the model to preprocess.
- Mode
- Specifies the preprocessing mode. Set this parameter to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatRestore

Synopsis

Restore a pattern matching model from disk.

Syntax

```
MIL_ID MpatRestore(  
    MIL_ID SystemId,  
    MIL_CONST_TEXT_PTR Filename,  
    MIL_ID *ModelIdPtr  
)
```

Description

This function restores a model that was previously saved to a file, using [MpatSave\(\)](#). If the model was preprocessed before saving, you do not need to preprocess it again.

This function also restores all the model's search parameters that were in effect when the model was saved.

Parameters

SystemId

Specifies the system on which to restore the model.

This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> Value	Specifies a valid system identifier, which you have allocated using the MsysAlloc() function.

Filename

Specifies the name and path of the file from which to restore the model. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

● For specifying the file name and path	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)
	<div>Parameters</div> <div>FileName</div> <div>Specifies the drive, directory, and name of the file.</div>
	<div>[This is only applicable to Windows]</div> <div>Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.</div>
<input type="checkbox"/> M_INTERACTIVE	

ModelIdPtr

Specifies the address of the variable in which to write the model identifier. Since the **MpatRestore()** function also returns the model identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the model identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSave

Synopsis

Save a pattern matching model to disk.

Syntax

```
void MpatSave(  
    MIL_CONST_TEXT_PTR FileName,  
    MIL_ID ModelId  
)
```

Description

This function saves all the information about the previously allocated model to disk, including all of the model's current search parameters values and any effects of preprocessing. Later, this information can be reloaded, using [MpatRestore\(\)](#).

Parameters

FileName

Specifies the name and path of the file in which to save the model. It is recommended that you use the MOD file extension for easier use with other Matrox Imaging software products. The function handles (internally) the opening and closing of the file. If the file already exists, it will be overwritten.

This parameter can be set to one of the following values:

For specifying the file name and path	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)
	<div>Parameters</div>
	<div>FileName</div> <div>Specifies the drive, directory, and name of the file.</div>
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.

ModelId

Specifies the identifier of the model to save.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetAcceptance

Synopsis

Set the acceptance level of a model.

Syntax

```
void MpatSetAcceptance(
    MIL_ID ModelId,
    MIL_DOUBLE AcceptanceThreshold
)
```

Description

This function sets the acceptance level for a match made with the specified model when it is sought in an image. If the correlation (match score) between the image and the model is less than this level, it is not considered a match (an occurrence). The default acceptance is set when the model is allocated.

Parameters

ModelId

Specifies the identifier of the model for which to change the acceptance threshold search parameter.

AcceptanceThreshold

Specifies the acceptance level, as a percentage. A perfect match is 100%, no correlation is 0%. The match score depends on the image quality. You should experiment to decide what is a typical match score for your application.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetAccuracy

Synopsis

Set the positional accuracy of a model.

Syntax

```
void MpatSetAccuracy(  
    MIL_ID ModelId,  
    MIL_INT Accuracy  
)
```

Description

This function sets the specified model's search parameter for positional accuracy and the complexity of the image. You can enhance speed performance by selecting a lower positional accuracy. Also, whenever the positional accuracy of a model changes, the effect of any preprocessing is undone. Therefore, if the accuracy is changed, call [MpatPreprocModel\(\)](#) before searching for the model.

The positional accuracy is also slightly affected by the search speed ([MpatSetSpeed\(\)](#)).

Parameters

ModelId
Specifies the identifier of the model for which to change the positional accuracy search parameter.

Accuracy
Sets the required positional accuracy for the search.

This parameter must be set to one of the values below. Note, the precision achieved is dependent on the quality of the model and the image (the tolerances listed below are typical for high-quality, low-noise images).

For specifying the positional accuracy of the search	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_HIGH	Specifies a high accuracy (typically ± 0.05 pixels). (summarize)
<input type="checkbox"/> M_LOW	Specifies a low accuracy (typically ± 0.20 pixels). Note that when using this setting, the match scores can be slightly lower than usual. If a precise match score is important to you, use at least medium accuracy. (summarize)
<input type="checkbox"/> M_MEDIUM	Specifies a medium accuracy (typically ± 0.10 pixels). (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetAngle

Synopsis

Set the angular search parameters of a model.

Syntax

```
void MpatSetAngle(
    MIL_ID ModelId,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function sets the specified model's angular search parameters. Also, whenever the angular search parameters of a model change, the effect of any preprocessing of the model is undone. Therefore, if the search angle is changed, call [MpatPreprocModel\(\)](#) before searching for the model.

Parameters

- ModelId
- Specifies the identifier of the model for which to change the angular search parameter. This can be any type of model except a model created using [MpatAllocRotatedModel\(\)](#).
- ControlType
- Specifies the type of control to set.
- By default, the [ControlType M_SEARCH_ANGLE_MODE](#) is disabled and the search is done at 0° (no rotation). When enabled, the model is searched at angle, respecting all search control parameters.
- See the [Parameter associations](#) section for possible values.
- ControlValue
- Specifies the value to which to set the control.
- See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For specifying the range of angles to cover in the search](#)
- [For specifying the accuracy, interpolation, and the tolerance of the search](#)

The values below determine the range of angles to cover in the search operation; that is, ([M_SEARCH_ANGLE](#) - [M_SEARCH_ANGLE_DELTA_NEG](#)) to ([M_SEARCH_ANGLE](#) + [M_SEARCH_ANGLE_DELTA_POS](#)) inclusively, starting at an angle close to [M_SEARCH_ANGLE](#).

For specifying the range of angles to cover in the search	
ControlType	Description
ControlValue	
M_SEARCH_ANGLE	Sets the starting angle value for the search. (summarize)

<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> 0.0 to 360.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_SEARCH_ANGLE_DELTA_NEG	Sets the maximum negative delta for the search. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> 0.0 to 180.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_SEARCH_ANGLE_DELTA_POS	Sets the maximum positive delta for the search. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0. (summarize)
<input type="checkbox"/> 0.0 to 180.0	Specifies the angle, in degrees.
<input type="checkbox"/> M_SEARCH_ANGLE_MODE	Sets whether an angular search is enabled. Angular searches are not supported when using <code>MpatFindMultipleModel()</code> with <code>M_FIND_BEST_MODELS</code> . (summarize)
<input type="checkbox"/> M_DISABLE	Disables an angular search. This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Enables an angular search.

The values below handle the accuracy, the type of interpolation, and the tolerance of the search.

For specifying the accuracy, interpolation, and the tolerance of the search		
<input type="checkbox"/> ControlType	Description	
ControlValue		
<input type="checkbox"/> M_SEARCH_ANGLE_ACCURACY	Sets the required precision of the resulting angle. (summarize)	
<input type="checkbox"/> M_DEFAULT	Same as <code>M_DISABLE</code> .	
<input type="checkbox"/> 0.1 to 180.0	Specifies the angle accuracy, in degrees.	
<input type="checkbox"/> M_DISABLE	Specifies that the angular accuracy equals the tolerance.	
<input type="checkbox"/> M_SEARCH_ANGLE_INTERPOLATION_MODE	Sets the type of interpolation used. (summarize)	
<input type="checkbox"/> M_DEFAULT	Same as <code>M_NEAREST_NEIGHBOR</code> .	
<input type="checkbox"/> M_BICUBIC	Specifies that bicubic interpolation is used when rotating the model.	
<input type="checkbox"/> M_BILINEAR	Specifies that bilinear interpolation is used when rotating the model.	
<input type="checkbox"/> M_NEAREST_NEIGHBOR	Specifies that nearest neighbor interpolation is used when rotating the model.	
<input type="checkbox"/> M_SEARCH_ANGLE_TOLERANCE	Sets the full range of degrees within which the pattern in the target image can be rotated from a model at a specific angle and still meet the acceptance level. (summarize)	
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 5.0. (summarize)	
<input type="checkbox"/> 0.1 to 180.0	Specifies that the given angle is the range of degrees within which the pattern in the target image can be rotated from a model at a specific angle and still meet the acceptance level. For example, if a model can tolerate the target image being offset from its search angle by $\pm 2.5^\circ$, specify 5° . The specified tolerance determines the step angle. Note that very small tolerance values should only be used within small angular ranges, otherwise application performance can be adversely affected.	

	(summarize)
--	-----------------------------

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetCenter

Synopsis

Set the reference position of a model.

Syntax

```
void MpatSetCenter(
    MIL_ID ModelId,
    MIL_DOUBLE OffX,
    MIL_DOUBLE OffY
)
```

Description

This function sets the reference position of the specified model. The default reference position is equal to ((SizeX - 1) / 2.0, (SizeY - 1) / 2.0) (relative to the model origin). All positional search results are based on this defined model's reference position, relative to the top-left corner of the image in which the search is performed.

Parameters

- ModelId
- Specifies the identifier of the model for which to change the reference position.
- OffX
- Specifies the X-offset of the new model's reference position relative to the model origin. Note that the reference position need not be in the model.
- OffY
- Specifies the Y-offset of the new model's reference position relative to the model origin. Note that the reference position need not be in the model.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetCertainty

Synopsis

Set the certainty level of a model.

Syntax

```
void MpatSetCertainty(
    MIL_ID ModelId,
    MIL_DOUBLE CertaintyThreshold
)
```

Description

This function sets the certainty level for a match made with the specified model when it is sought in an image. If the correlation (match score) between the image and the model is greater than or equal to the certainty level, a match is assumed without looking elsewhere in the image for a better match. The default certainty level is set when the model is allocated (**MpatAlloc...Q**).

Parameters

ModelId

Specifies the identifier of the model for which to change the certainty threshold search parameter.

CertaintyThreshold

Specifies the certainty level, as a percentage. If you set the certainty level too high (close to 100%), you slow down the search because you force the search algorithm to check the whole search region for the best match. A good certainty level is slightly lower than the expected score, so that the search can finish as soon as the match is found. However, if you set the certainty level too low, false matches might be found. Note that the certainty level must be greater than or equal to the acceptance level ([MpatSetAcceptance\(\)](#)).

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetDontCare

Synopsis

Set the "don't care" pixels in a model.

Syntax

```
void MpatSetDontCare(
    MIL_ID ModelId,
    MIL_ID ImageBufId,
    MIL_INT OffX,
    MIL_INT OffY,
    MIL_INT Value
)
```

Description

This function sets pixels in the specified model to a "don't care" state. The "don't care" pixels of the model will not be considered when searching for occurrences of the model in the target image ([MpatFindModel\(\)](#) or [MpatFindMultipleModel\(\)](#)). To determine which model pixels to set, a region of the specified image buffer, starting at the specified offset and equal in size to the model, is compared with the specified "don't care" value. If an image pixel is equal to this value, the corresponding pixel in the model is set to "don't care".

Note, each time this function is called, a new set of "don't care" pixels is assigned to the specified model, superseding any existing set. Therefore, multiple calls to this function do not have a cumulative effect. Also, whenever the "don't care" pixels in a model change, the effect of any preprocessing of the model is undone. Therefore, if the "don't care" pixels are changed, call [MpatPreprocModel\(\)](#) before searching for the model.

This function does not support a [M_NORMALIZED](#) + [M_CIRCULAR_OVERSCAN](#) type of model.

Parameters

ModelId

Specifies the identifier of the model in which to set "don't care" pixels.

ImageBufId

Specifies the identifier of the image buffer used to identify which pixels in the model will be set to "don't care". This buffer must be at least as large as the model.

OffX

Specifies the X-offset from the upper-left corner of the specified image buffer to the upper-left corner of the pixel-value comparison area. The size of the comparison area is determined by the size of the model.

OffY

Specifies the Y-offset from the upper-left corner of the specified image buffer to the upper-left corner of the pixel-value comparison area. The size of the comparison area is determined by the size of the model.

Value

Specifies the pixel value in the image buffer that determine the corresponding "don't care" pixels.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetNumber

Synopsis

Set the expected number of occurrences of a model.

Syntax

```
void MpatSetNumber(
    MIL_ID ModelId,
    MIL_INT NbOccurrences
)
```

Description

This function sets the number of occurrences of a model for which to search when using [MpatFindModel\(\)](#) or [MpatFindMultipleModel\(\)](#). Occurrences with match scores that are greater than or equal to the certainty level (set with [MpatSetCertainty\(\)](#)) are returned, up to the number specified in [NbOccurrences](#). If such occurrences are fewer than the specified number, the remaining occurrences returned are the best of those that are greater than or equal to the acceptance level (set with [MpatSetAcceptance\(\)](#)). The default is 1 (that is, find only the first occurrence with a match score that is greater than or equal to the certainty level).

Parameters

ModelId

Specifies the identifier of the model for which to set the expected number of occurrences.

NbOccurrences

Specifies the maximum number of occurrences for which to look in the target image. The number of occurrences should be less than or equal to the number of result buffer entries or some results might be lost.

This parameter must be set to one of the values below.

● For specifying the maximim number of occurrences	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 1 <= Value <= 1000	Specifies the maximum number of occurrences for which to look in the target image.
<input type="checkbox"/> M_ALL	Returns all matches that are greater than or equal to the acceptance level.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetPosition

Synopsis

Set the search region of a model.

Syntax

```
void MpatSetPosition(
    MIL_ID ModelId,
    MIL_INT OffX,
    MIL_INT OffY,
    MIL_INT SizeX,
    MIL_INT SizeY
)
```

Description

This function sets the specified model's search region. It limits the area in the target image in which to find the reference position of the model (set with [MpatSetCenter\(\)](#)), and consequently increases the search speed. This function is useful when the model's reference position is expected to be found in a certain area. You can change the search region at any time (for example, when tracking an object, the search region could be changed before each call to [MpatFindModel\(\)](#) or [MpatFindMultipleModel\(\)](#)).

If you have redefined the model's reference position (with [MpatSetCenter\(\)](#)), make sure that the search region defined by [MpatSetPosition\(\)](#) covers this new reference position and takes into account the angular search range of the model.

Always use this function, rather than a child buffer, to restrict the search region. Other methods result in edge effects, and if the search region is as small as or smaller than the model, other methods produce invalid results.

To change some of the search parameters set with this function without affecting others, specify **M_NO_CHANGE** for those you do not want to change. For example, you can change [OffX](#) and [OffY](#) to the required values and keep [SizeX](#) and [SizeY](#) as is by setting them to **M_NO_CHANGE**.

To reset the search region size to the full image, set [OffX](#) and [OffY](#) to zero (0), and [SizeX](#) and [SizeY](#) to **M_ALL**.

Note that it is valid to specify a search region ([SizeX](#) and [SizeY](#)) that is smaller than the search model since you are setting the area in which to find the model's reference position.

Parameters

ModelId

Specifies the identifier of the model for which to change the search region.

OffX

Specifies the X-coordinate of the top-left corner of the search region in the target image.

● For specifying the X-coordinate of the search region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL	Resets the coordinate to the X-coordinate of the top-left corner of the target image to allow a search of the full image.
<input type="checkbox"/> M_NO_CHANGE	Keeps the previous value of this coordinate as set by MpatSetCenter() .
<input type="checkbox"/> Value	Sets the value of this coordinate to a number.

OffY

Specifies the Y-coordinate of the top-left corner of the search region in the target image.



● For specifying the Y-coordinate of the search region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL	Resets the coordinate to the Y-coordinate of the top left corner of the target image to allow a search of the full image.
<input type="checkbox"/> M_NO_CHANGE	Keeps the previous value of this coordinate as set by MpatSetCenter() .
<input type="checkbox"/> Value	Sets the value of this coordinate to a number.

SizeX

Specifies the width of the search region.

● For specifying the width of the search region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL	Resets the width of the search to that of the full image.
<input type="checkbox"/> M_NO_CHANGE	Keeps the previously set value.
<input type="checkbox"/> Value	Sets the value of this size to a number.

SizeY

Specifies the height of the search region.

● For specifying the height of the search region	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL	Resets the height of the search to that of the full image.
<input type="checkbox"/> M_NO_CHANGE	Keeps the previously set value.
<input type="checkbox"/> Value	Sets the value of this size to a number.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetSearchParameter

Synopsis

Set the internal search parameters of a model.

Syntax

```
void MpatSetSearchParameter(  
    MIL_ID PatId,  
    MIL_INT Parameter,  
    MIL_DOUBLE Value  
)
```

Description

This function is for advanced users of the pattern matching module. It need not be used under most circumstances because the default settings usually provide the best results for search operations. However, if the default values do not satisfy the requirements of your application, this function can be used to set the model's internal search parameters. These internal search parameters are normally derived from the speed and accuracy settings (see [MpatSetSpeed\(\)](#) and [MpatSetAccuracy\(\)](#)), but this function gives the experienced user precise control over them. To gain a clearer understanding of how the search algorithm works, see [Chapter 7: Pattern matching](#).

Note that if some of the model's search parameters are changed after a call to [MpatPreprocModel\(\)](#), the model must be preprocessed again. To inquire if your model is in a preprocessed state, use the [MpatInquire\(\)](#) function with [M_PREPROCESSED](#).

Note that all of the model's internal parameters, except [M_COARSE_SEARCH_ACCEPTANCE](#), are saved and restored with the model, just like the other parameters such as search region, speed, and accuracy.

Parameters

- PatId**
Specifies the identifier of the model whose search parameter to change, or of the result buffer.
- Parameter**
Specifies the search parameter to set.
See the [Parameter associations](#) section for possible values.
- Value**
Specifies the value to which to set the specified parameter.
See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **Parameter** and **Value** parameters are described in the following tables:

- [For a model identifier](#)
- [For a result buffer identifier](#)

The values below are for a model identifier.

For a model identifier	
Parameter	Description
Value	

<input type="checkbox"/> M_ALL	Sets all search parameters automatically from the current speed and accuracy settings. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies all search parameters.
<input type="checkbox"/> M_ALLOC_OFFSET_X	Sets the model's allocation X-offset. (summarize)
<input type="checkbox"/> Value	Specifies the X-offset. This might be useful when the original value is lost after rotating a model. You can use MpatInquire() to inquire about the current X-offset. This value can be any integer. The default value is 0. (summarize)
<input type="checkbox"/> M_ALLOC_OFFSET_Y	Sets the model's allocation Y-offset. (summarize)
<input type="checkbox"/> Value	Specifies the Y-offset. This might be useful when the original value is lost after rotating the model. You can use MpatInquire() to inquire about the current Y-offset. This value can be any integer. The default value is 0. (summarize)
<input type="checkbox"/> M_COARSE_SEARCH_ACCEPTANCE	Sets the coarse search acceptance threshold level to use for rejecting candidate model peaks at low resolution levels. (summarize)
<input type="checkbox"/> M_DEFAULT	Determines the coarse search acceptance threshold level automatically.
<input type="checkbox"/> 1.0 to 100.0	Specifies the coarse search acceptance threshold level. This can speed up the search when some of the matches you request do not reach the certainty threshold, or when you request more matches than are really present in the image. The coarse search acceptance level should usually be set much lower than the search acceptance level. For example, a good level to set it to is about 20% to 30%. Note that if it is too low, you will not see any increase in speed. However, if it is too high, you risk rejecting real match peaks. Note that the coarse search acceptance threshold level is not saved and restored with the model. So, if you always want to use your own coarse search acceptance level, you should set it each time after restoring your model and before calling MpatFindModel() . Otherwise, the default coarse search acceptance level is used. (summarize)
<input type="checkbox"/> M_EXTRA_CANDIDATES	Sets the number of extra candidates to consider. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> Value	Specifies the number of extra candidates. This value can be any integer. Normally, the search algorithm considers only a limited number of (best) scores as possible candidates to match when proceeding at the most sub-sampled stage. This parameter allows you to add robustness to the algorithm, by considering more candidates, without compromising too heavily on search speed. (summarize)
<input type="checkbox"/> M_FAST_FIND	Sets whether to use fast peak finding. (summarize)
<input type="checkbox"/> M_DEFAULT	Preprocessing decides if fast peak finding is appropriate.
<input type="checkbox"/> M_DISABLE	Specifies no fast peak finding. The initial search (at the resolution level determined by M_FIRST_LEVEL) computes the correlation at every position in the search region. This guarantees that the biggest match peak will be found, and that it will be investigated first. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies fast peak finding. The search algorithm attempts to find the peaks without checking every point. This is safe in most cases, but can cause matches to be missed for models that produce very narrow peaks. (summarize)

<input type="checkbox"/> M_FIRST_LEVEL	<p>Sets the resolution level for the initial stage (lowest resolution) of the search.</p> <p>Note that level 0 is the original target image and each higher level is half the size (and resolution) of the previous one. If the specified level is not supported by the search algorithm, the highest possible level will be used. A higher first level speeds up the initial search but makes it less reliable, because the model might not retain enough distinctive features at such a low resolution. Use MpatInquire() with M_PROC_FIRST_LEVEL to inquire about the default value of the lowest resolution level.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the first level automatically.
<input type="checkbox"/> 0 to 7	Specifies the first level.
<input type="checkbox"/> M_LAST_LEVEL	<p>Sets the resolution level for the final stage (highest resolution) of the search.</p> <p>Note that if the specified level is not supported by the search algorithm, the highest acceptable level will be used. Search score can also be less reliable for levels above 0, depending on the characteristics of the model. Use MpatInquire() with M_PROC_LAST_LEVEL to inquire about the default value of the highest resolution level.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the last level automatically.
<input type="checkbox"/> 0 to 7	Specifies the last level.
<input type="checkbox"/> M_MIN_SPACING_X	<p>Sets the minimum spacing (in X) between two models in order for them to be considered distinct.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Defaults to 75% of the model width (SizeX of MpatAlloc...()).
<input type="checkbox"/> 1.0 to 100.0	Specifies the minimum spacing.
<input type="checkbox"/> M_MIN_SPACING_Y	<p>Sets the minimum spacing (in Y) between two models in order for them to be considered distinct.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Defaults to 75% of the model height (SizeY of MpatAlloc...()).
<input type="checkbox"/> 1.0 to 100.0	Specifies the minimum spacing.
<input type="checkbox"/> M_MODEL_STEP	<p>Sets whether all or every second pixel in the model is used in the correlation during the high resolution stage of the search.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the model step automatically.
<input type="checkbox"/> 1	Specifies all model pixels.
<input type="checkbox"/> 2	<p>Specifies every second model pixel (in both the X and Y directions).</p> <p>This speeds up the last (high resolution) stage of the search, particularly for large models. The match score can be affected if the model has many fine features, but will tend not to be affected if the model has mainly coarse features.</p> <p>(summarize)</p>

The values below are for a result buffer identifier.

For a result buffer identifier	
Parameter	Description
Value	
<input type="checkbox"/> M_SAVE_SUMS	<p>Sets whether to save the values of the sums used to compute the normalized correlation function for each model occurrence.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DISABLE	<p>Specifies that the sums are not saved.</p> <p>This is the default value.</p> <p>(summarize)</p>
<input type="checkbox"/> M_ENABLE	Specifies that the sums are saved.

<input type="checkbox"/> M_TARGET_CACHING	Sets whether the pyramidal representation of the buffer is called or kept in the result buffer. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies that the pyramidal representation of the buffer is generated each time MpatFindModel() or MpatFindMultipleModel() is called. This is the default value. (summarize)
<input type="checkbox"/> M_ENABLE	Specifies that the pyramidal representation of the buffer (generated when MpatFindModel() or MpatFindMultipleModel() is called) is kept in the result buffer. This pyramidal representation is re-used by consecutive calls to MpatFindModel() and MpatFindMultipleModel() as long as the same result buffer is used and the image, search region, and model size are not modified. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatSetSpeed

Synopsis

Set search speed of a model.

Syntax

```
void MpatSetSpeed(
    MIL_ID ModelId,
    MIL_INT SpeedFactor
)
```

Description

This function specifies the required search speed when using [MpatFindModel\(\)](#) or [MpatFindMultipleModel\(\)](#). At a higher speed, the search takes all reasonable short cuts; therefore, the search should be performed faster than at lower settings. Generally, the high speed setting should be used for better quality images or when using a simple model. Note, the high speed setting reduces positional accuracy very slightly. Try the low speed settings only if your image quality is particularly poor and you have encountered problems at higher speeds. Also, whenever the search speed parameter setting of a model changes, the effect of any preprocessing of the model is undone. Therefore, if the search speed is changed, call [MpatPreprocModel\(\)](#) before searching for the model.

Parameters

ModelId

Specifies the identifier of the model for which to change the speed parameter.

SpeedFactor

Specifies the search speed. Set this parameter to one of the values below.

For specifying the search speed	
Value	Description
M_HIGH	Specifies a high search speed.
M_LOW	Specifies a low search speed.
M_MEDIUM	Specifies a medium search speed.
M_VERY_HIGH	Specifies a very high search speed.
M_VERY_LOW	Specifies a very low search speed.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

MpatStream

Synopsis

Load, restore, or save a pattern matching model from/to a file or a memory stream.

Syntax

```
void MpatStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *ModelIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load or restore a pattern matching model from a file or memory stream. All of the pattern matching model settings that were in effect when the pattern matching model was saved will be restored. A loaded or restored pattern matching model is not preprocessed, therefore you must call [MpatPreprocModel\(\)](#) before performing a search with [MpatFindModel\(\)](#) or [MpatFindMultipleModel\(\)](#).

This function can also save a pattern matching model to a file or memory stream. All information about the previously allocated pattern matching model is saved, including all of the model's current search settings. However, any effects of preprocessing are not saved.

To inquire the number of bytes necessary to save a pattern matching model to memory stream, you should first call this function (**MpatStream()**) with [M_INQUIRE_SIZE_BYTE](#).

The content saved to memory stream is equivalent to the content saved to file. In addition, any file saved with [MpatSave\(\)](#) is equivalent to a file saved using this function.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using **MpatStream()**, you can choose to save a backwards-compatible version of the pattern matching model, which will work using a version of MIL that is up to one major release older than the current version (depending on which version is specified). For example, if you allocate a pattern matching model using MIL 9.0 and save it to version 8.0, you can restore this context on a computer where MIL 8.0 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore pattern matching models saved using MIL version 9.0 or above. Settings that do not exist in the lower version will be filled with default values when the pattern matching model is loaded or restored.

Parameters

MemPtrOrFileName

Specifies the file or memory stream.

● For specifying the file or memory stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving a pattern matching model to a file, use the MOD file extension. The function handles (internally) the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p> <p>To open or save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p>
	<div>Parameters</div>

	<div>FileName</div> <div>Specifies the drive, directory, and name of the file.</div>
<div><div></div>M_INTERACTIVE</div>	<div>[<i>This is only applicable to Windows</i>]</div> <div>Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE.</div>
<div><div></div>M_NULL</div>	<div>Specifies to ignore this parameter. This parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation.</div> <div>(summarize)</div>
<div><div></div>MemPtr</div>	<div>Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY. The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (MpatStream()) first with M_INQUIRE_SIZE_BYTE.</div> <div>(summarize)</div>

SystemId

Specifies the system on which to restore the pattern matching model. For **M_INQUIRE_SIZE_BYTE**, **M_LOAD**, and **M_SAVE**, **SystemId** is ignored and should be set to **M_NULL**. For an **M_RESTORE** operation, this parameter should be set to one of the following values:

For specifying the system on which to restore the pattern matching model	
Value	Description
M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform on the pattern matching model. This parameter must be set to one of the following values:

For specifying the operation	
Value	Description
M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save a pattern matching model to memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
M_LOAD	Loads the content of a specified file or memory stream into a previously allocated pattern matching model.
M_RESTORE	Restores a pattern matching model from a file or memory stream and assigns it an identifier.
M_SAVE	Saves a pattern matching model to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the pattern matching model. This parameter must be set to one of the following values:

For specifying the type of stream																											
Value	Description	corona-II (a)	igge vision (c)	gpu processing (d)	helios ea/xa (e)	helios ecl/xd (f)	helios ed/xd (g)	helios ed/xd (h)	iris (i)	met-II/cl (j)	met-II/dlg (k)	met-II/fmc (l)	met-II/std (m)	morphis (n)	nexis (p)	odyssey ea/xa (q)	odyssey ed/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ed/xd (u)	solos gige (v)	solos gige (w)					
M_FILE	Specifies a file stream.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v				
M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. Note that this value is not supported on remote systems. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v				

[illegible]

Version

Specifies the MIL version of the pattern matching model. This parameter must be set to one of the following values.

For specifying the MIL version	
Value	Description
M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL. For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ModelIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the pattern matching model.

For `M_INQUIRE_SIZE_BYTE` and `M_SAVE` operations, `ModelIdPtr` specifies the address of the variable from which to read the pattern matching model identifier.

For an **M_LOAD** operation, **ModelIdPtr** specifies the address of the variable from which to read the identifier of the pattern matching model where the file or memory stream content will be loaded.

For an **M_RESTORE** operation, **ModelIdPtr** specifies the address in which to return the identifier of the restored pattern matching model. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the pattern matching model, in bytes. If the size is not required, you can set this parameter to `M_NULL`.

Note that the size of a pattern matching model will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milpat.lib.
DLL	Requires mil.dll; milpat.dll.

Mreg functions

Synopsis

The functions prefixed with Mreg make up the Registration module. The Registration module allows you to find the transformations that optimally position images in the same coordinate system; this process is referred to as image registration and the common coordinate system is referred to as the global coordinate system. The module uses the rough location of the images to determine how they overlap. It then optimizes the match in the overlapping regions and calculates the transformation needed to achieve this match. The MIL Registration module can use these results to perform other operations. These operations include mosaicing and transforming coordinates between two of the following coordinate systems: the global coordinate system, any registered image's coordinate system, or the mosaic coordinate system.

Functions

- [MregAlloc](#)
- [MregAllocResult](#)
- [MregCalculate](#)
- [MregControl](#)
- [MregDraw](#)
- [MregFree](#)
- [MregGetResult](#)
- [MregInquire](#)
- [MregRestore](#)
- [MregSave](#)
- [MregSetLocation](#)
- [MregStream](#)
- [MregTransformCoordinate](#)
- [MregTransformCoordinateList](#)
- [MregTransformImage](#)

MregAlloc

Synopsis

Allocate a registration context.

Syntax

```
MIL_ID MregAlloc(
    MIL_ID SystemId,
    MIL_INT RegistrationType,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr
)
```

Description

This function allocates a registration context on the specified system. A registration context contains all the information needed to perform the [MregCalculate\(\)](#) operation, including global registration settings and settings for each registration element.

When you allocate a registration context, it is defined with a default number of registration elements (256). You can add or remove registration elements using [MregControl\(\)](#) with [M_NUMBER_OF_ELEMENTS](#).

When the registration context is no longer required, you should release its memory, using [MregFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the context. This parameter should be set to one of the following values:

For specifying the system	
Value	Description
M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

RegistrationType

Specifies the type of registration context. This parameter defines the algorithm to use for the registration. This parameter should be set to the value below:

For specifying the type of registration context	
Value	Description
M_CORRELATION	Uses normalized grayscale correlation to optimize the match in the images' overlapping regions during registration.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the registration context identifier. Since the **MregAlloc()** function also returns the registration context identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the registration context identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregAllocResult

Synopsis

Allocate a registration result buffer.

Syntax

```
MIL_ID MregAllocResult(  
    MIL_ID SystemId,  
    MIL_INT ControlFlag,  
    MIL_ID *ResultIdPtr  
)
```

Description

This function allocates a registration result buffer, on the specified system, to store results obtained from an [MregCalculate\(\)](#) operation. When the registration result buffer is no longer required, release its memory, using [MregFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the result buffer. This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> Value	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ResultIdPtr

Specifies the address of the variable in which to write the registration result buffer identifier. Since **MregAllocResult()** also returns the registration result buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregCalculate

Synopsis

Perform the registration of the input images using the rough locations specified by the registration elements.

Syntax

```
void MregCalculate(
    MIL_ID ContextId,
    const MIL_ID *ImageArrayPtr,
    MIL_ID RegResultId,
    MIL_INT NumImages,
    MIL_INT ControlFlag
)
```

Description

This function performs the registration of the input images by finding the transformations that optimally position them in a global coordinate system. The function optimizes the match in the overlapping region between each image and its reference image and then calculates the transformation needed to achieve this match. Once the transformation that maps each image into its reference image's coordinate system is calculated, it is converted so that it maps the image into the global coordinate system.

You might not want to perform the optimization step of the registration calculation when the precise location of one or more images is known. In this case, you have two options. You can replace the image for which you want to skip the optimization step with **M_NULL** in the image array or you can disable **M_OPTIMIZE_LOCATION** in the image's registration element using [MregControl\(\)](#). For more information, see the [Skipping the optimization step](#) subsection in the [Customizing your registration settings](#) section in [Chapter 10: Registration](#).

Every image in the specified image array is associated with the registration element and registration result element of the same index. For example, the first image is associated with the first registration element and the first registration result element. Therefore, it is important that the number of images supplied in the array be less than or equal to the number of registration elements.

The results of the registration calculation can be retrieved using [MregGetResult\(\)](#). These results can be used for other applications. You can create a mosaic from the images with [MregTransformImage\(\)](#) or use [MregTransformCoordinate\(\)](#) or [MregTransformCoordinateList\(\)](#) to transform positions between two of the following coordinate systems: the global coordinate system, any registered image's coordinate system, or the mosaic's coordinate system.

Parameters

ContextId

Specifies the context to use for the registration. The registration context must have been previously allocated on the required system using [MregAlloc\(\)](#).

ImageArrayPtr

Specifies the array of images to register.

RegResultId

Specifies the identifier of the registration result buffer in which to write the results of the registration. The result buffer must have been previously allocated on the required system using [MregAllocResult\(\)](#).

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

The registration result buffer ([RegResultId](#)) must be allocated on the same system as the registration context ([ContextId](#)). If it is not, an error will occur.

NumImages

Specifies the number of images in the specified image array. Note that if you replace an image by **M_NULL**, you should still include it when determining the value of [NumImages](#).

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregControl

Synopsis

Control a registration context, registration element setting, or registration result buffer.

Syntax

```
void MregControl(
    MIL_ID ContextOrResultId,
    MIL_INT Index,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function sets the specified control for either a registration context itself, one (or all) of the registration elements contained therein, or a registration result buffer. For registration contexts and registration elements, these settings control the execution of [MregSetLocation\(\)](#) and [MregCalculate\(\)](#) operations. For registration result buffers, these settings control the execution of [MregDraw\(\)](#) and [MregTransformImage\(\)](#) operations.

All the control type settings can be inquired using [MregInquire\(\)](#).

Parameters

ContextOrResultId

Specifies the registration context or registration result buffer whose settings you want to modify. The registration context or the registration result buffer must have been previously allocated on the system using [MregAlloc\(\)](#) or [MregAllocResult\(\)](#), respectively.

Index

Specifies that a registration context, an individual registration element, or a registration result buffer is controlled. Set this parameter to one of the following values:

For the registration context, registration element, or registration result buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default value. If a registration context is specified, same as M_ALL . If a registration result buffer is specified, same as M_GENERAL . (summarize)
<input type="checkbox"/> M_ALL	Applies the specified control setting to all registration elements, if a registration context is specified.
<input type="checkbox"/> M_CONTEXT	Controls a general setting of a registration context, if one is specified.
<input type="checkbox"/> M_GENERAL	Controls a general setting of a registration result buffer, if one is specified.
<input type="checkbox"/> Value	Specifies the index of the individual registration element to control, if a registration context is specified.

ControlType

Specifies the setting to change.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the setting's new value.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For M_CONTEXT](#)
- [For registration elements \(one or all\)](#)
- [For M_GENERAL](#)

The possible [ControlType](#) and corresponding [ControlValue](#) settings for [M_CONTEXT](#) are:

For M_CONTEXT		
ControlType	Description	
ControlValue		
M_ACCURACY	Sets the accuracy of the registration calculation. Accuracy depends on the image's dynamic range, sharpness, and noise. The best accuracy is achieved for well-contrasted noise-free images. (summarize)	
M_DEFAULT	Same as M_HIGH .	
M_HIGH	Specifies high accuracy. Registration will be calculated with subpixel accuracy. (summarize)	
M_LOW	Specifies low accuracy. Registration will be calculated with pixel accuracy. (summarize)	
M_LOCATION_DELTA	Sets the maximum displacement that will be applied to any pixel in the images during registration, relative to its initial location set using MregSetLocation() . With a larger maximum displacement, you can align the images even if the location that was set using MregSetLocation() was not precise. However, this can lead to a longer calculation time. (summarize)	
M_DEFAULT	Specifies the default value. The default value is 5 percent. (summarize)	
0<Value<=100.0	Specifies the maximum displacement as a percentage. The displacement is expressed as a percentage of the biggest dimension of the images to align. (summarize)	
M_MIN_OVERLAP	Sets the minimum overlap that should exist between an image and its reference image so that the registration calculation can optimize the match in their overlapping region. If the minimum overlap is not met, the optimization step will not be performed for this image. Instead, the transformation that was set using MregSetLocation() becomes the optimal transformation. The image's registration result element will indicate that the registration operation failed for this image. (summarize)	
M_DEFAULT	Specifies the default value. The default value is 20.0. (summarize)	
0<Value<=100.0	Specifies the minimum overlap as a percentage. The overlap is expressed as a percentage of the smallest of the two images. (summarize)	
M_NUMBER_OF_ELEMENTS	Sets the number of registration elements in the registration context. If you reduce the number of registration elements, the settings of the ones that remain are kept unchanged. If you increase the number of registration elements, the new ones are initialized with the default settings. You should have at least as many registration elements as the number of images that you want to register. (summarize)	
M_DEFAULT	Specifies the default value. The default value is 256. (summarize)	
1<=Value<8192	Specifies the number of registration elements.	
M_SCORE_TYPE	Sets the type of score calculated during registration. This calculated score can be retrieved after registration using MregGetResult() with M_SCORE . (summarize)	
M_CORRELATION	Specifies to calculate the score based on the normalized grayscale correlation in the overlapped region.	

<input type="checkbox"/> M_NONE	Specifies that no score is calculated; it is set to 100%. This is the default value. (summarize)
<input type="checkbox"/> M_TRANSFORMATION_TYPE	Sets the type of transformation that the registration calculation will use to optimize the match in the images' overlapping regions. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_TRANSLATION .
<input type="checkbox"/> M_PERSPECTIVE	Specifies that a perspective warping can be performed to optimize the match in the overlapping regions.
<input type="checkbox"/> M_TRANSLATION	Specifies that a translation can be performed to optimize the match in the overlapping regions.
<input type="checkbox"/> M_TRANSLATION_ROTATION	Specifies that a translation and a rotation can be performed to optimize the match in the overlapping regions.
<input type="checkbox"/> M_TRANSLATION_ROTATION_SCALE	Specifies that a translation, a rotation, and a scale operation can be performed to optimize the match in the overlapping regions.

The possible [ControlType](#) and corresponding [ControlValue](#) settings for registration elements (all or a specified one) are:

For registration elements (one or all)	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_OPTIMIZE_LOCATION	Sets whether to perform the optimization step of the registration calculation for the registration element's image. If you choose not to perform the optimization calculation, the transformation that you set with MregSetLocation() becomes the optimal transformation for this image. When MregCalculate() is called, the transformation is converted so that it maps the image into the global coordinate system instead of into its reference image's coordinate system. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ENABLE .
<input type="checkbox"/> M_DISABLE	Disables the optimization calculation.
<input type="checkbox"/> M_ENABLE	Enables the optimization calculation.
<input type="checkbox"/> M_REFERENCE_X	Sets the X-offset between the origin of the registration element's image and the center of the image's top-left pixel. Note that the origin of an image need not lie within the image. This is an advanced control type and changing the origin of an image's coordinate system will affect various aspects of the registration process. In MregSetLocation() , you set the rough location of an image with respect to its reference image's coordinate system. Similarly, the position of your mosaic in the destination image buffer depends on the mosaic's reference coordinate system (M_MOSAIC_STATIC_INDEX in MregControl()), which can be the coordinate system of a particular image. Finally, you can convert coordinates to and from an image's coordinate system, with MregTransformCoordinate() and MregTransformCoordinateList() . If you modify the origin of an image's coordinate system, you must keep the origin's new location in mind when performing any of these operations. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies that the default X-offset will be used. The default value is 0.0. (summarize)
<input type="checkbox"/> M_CENTER_ELEMENT	Sets the X-offset to be the center of the image.
<input type="checkbox"/> Value	Specifies the X-offset, in pixels. The value can be specified in subpixel accuracy. (summarize)
<input type="checkbox"/> M_REFERENCE_Y	Sets the Y-offset between the origin of the registration element's image and the center of the top-left pixel. Note that the origin of an image need not lie within it. This is an advanced control type and changing the origin of an image's coordinate system will affect various aspects of the registration process. In MregSetLocation() , you set the rough location of an image with respect to its reference image's coordinate system. Similarly, the position of your mosaic in the destination image buffer depends on the mosaic's reference coordinate system (M_MOSAIC_STATIC_INDEX in MregControl()), which can be the coordinate system of a particular image. Finally, you can convert coordinates to and from an image's coordinate system, with MregTransformCoordinate() and MregTransformCoordinateList() . If you modify the origin of an image's coordinate system, you must keep the origin's new location in mind when performing any of these operations. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies that the default Y-offset will be used.

	The default value is 0.0. (summarize)
<input type="checkbox"/> M_CENTER_ELEMENT	Sets the Y-offset to be the center of the image.
<input type="checkbox"/> Value	Specifies the Y-offset, in pixels. The value can be specified in subpixel accuracy. (summarize)

The possible [ControlType](#) and corresponding [ControlValue](#) settings for [M_GENERAL](#) are:

For M_GENERAL	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_X	Sets the X-coordinate in the global coordinate system that will be the starting point of the drawing region. This point will appear in the top-left corner of the destination image buffer. For more information, see MregDraw() . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the relative X-offset, in pixels.
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_Y	Sets the Y-coordinate in the global coordinate system that will be the starting point of the drawing region. This point will appear in the top-left corner of the destination image buffer. For more information, see MregDraw() . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.0 pixels. (summarize)
<input type="checkbox"/> Value	Specifies the relative Y-offset, in pixels.
<input type="checkbox"/> M_DRAW_SCALE_X	Sets the scale factor in the X-direction. This value is used when performing zoomed drawings of results. For more information, see MregDraw() . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the scale factor in the X-direction.
<input type="checkbox"/> M_DRAW_SCALE_Y	Sets the scale factor in the Y-direction. This value is used when performing zoomed drawings of results. For more information, see MregDraw() . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> Value > 0	Specifies the scale factor in the Y-direction.
<input type="checkbox"/> M_MOSAIC_COMPOSITION	Sets which image's pixel values to use when composing the mosaic and two or more images overlap. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_LAST_IMAGE .
<input type="checkbox"/> M_AVERAGE_IMAGE	Specifies to use the average value of the images' pixels in the overlapping region.
<input type="checkbox"/> M_FIRST_IMAGE	Specifies to use the pixels of the image associated with the registration result element with the lowest index.
<input type="checkbox"/> M_FUSION_IMAGE	Specifies to fuse the images by progressively blending overlapping pixels. That is, overlapping pixel values are modified (blended) to form a transitional portion. Blending is based on the distance between each pixel and the edges of the images in the mosaic. (summarize)
<input type="checkbox"/> M_LAST_IMAGE	Specifies to use the pixels of the image associated with the registration result element with the highest index.
<input type="checkbox"/> M_SUPER_RESOLUTION	Specifies to use a super-resolution algorithm to create the mosaic. For best results, the image alignment should be accurate. To perform registration calculations with subpixel accuracy, call MregControl() with M_ACCURACY set to M_HIGH . The source images should be of good quality with clear, distinct features in the overlapping region. To customize the super-resolution process, set the point-spread function, its radius, and the degree of smoothness to use, with M_SR_PSF_TYPE ,

	M_SR_PSF_RADIUS , and M_SR_SMOOTHNESS respectively. (summarize)
<input type="checkbox"/> M_MOSAIC_OFFSET_X	Sets the X-offset between the origin of the coordinate system used to compose the mosaic and the left side of the destination image buffer. The coordinate system used to compose the mosaic is the one specified with M_MOSAIC_STATIC_INDEX . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ALIGN_LEFT .
<input type="checkbox"/> M_ALIGN_LEFT	Specifies that the X-offset (in pixels) will be calculated such that the left-most part of the mosaic will be aligned with the left side of the destination image buffer.
<input type="checkbox"/> Value	Specifies the X-offset in pixels.
<input type="checkbox"/> M_MOSAIC_OFFSET_Y	Sets the Y-offset between the origin of the coordinate system used to compose the mosaic and the top of the destination image buffer. The coordinate system used to compose the mosaic is the one specified with M_MOSAIC_STATIC_INDEX . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ALIGN_TOP .
<input type="checkbox"/> M_ALIGN_TOP	Specifies that the Y-offset (in pixels) will be calculated such that the top-most part of the mosaic will be aligned with the top of the destination image buffer.
<input type="checkbox"/> Value	Specifies the Y-offset in pixels.
<input type="checkbox"/> M_MOSAIC_SCALE	Sets the scale factor to apply to the images before composition of the mosaic. A value greater than 1.0 produces an enlarged mosaic, while a value less than 1.0 produces a reduced one. (summarize)
<input type="checkbox"/> $0.0 < \text{Value} \leq 10.0$	Specifies the scale factor. The default value is 1.0. (summarize)
<input type="checkbox"/> M_MOSAIC_STATIC_INDEX	Sets the coordinate system relative to which the mosaic will be composed. This can be a registered image's coordinate system or the global coordinate system. If you select an image's coordinate system, that image will appear upright in the mosaic, and the other images will be oriented relative to it. By default, the origin of the mosaic will appear at the top-left corner of the destination image buffer. You can adjust the horizontal and vertical offset of the origin using M_MOSAIC_OFFSET_X and M_MOSAIC_OFFSET_Y . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. By default, the mosaic will be composed with respect to the coordinate system of the image associated with registration result element 0. (summarize)
<input type="checkbox"/> $0 \leq \text{Value} \leq \text{NumberOfElements}-1$	Specifies the index of the registration result element whose image's coordinate system will be used as the reference coordinate system.
<input type="checkbox"/> M_ALL	Specifies that the coordinate system will be chosen such that the minimum change is done on all images during the mosaic composition. This setting is only available when composing a mosaic with only two images. (summarize)
<input type="checkbox"/> M_REGISTRATION_GLOBAL	Specifies that the mosaic will be composed with respect to the global coordinate system.
<input type="checkbox"/> M_SR_PSF_RADIUS	Sets the radius of the M_CIRCULAR and M_GAUSSIAN point-spread functions (PSF). For M_GAUSSIAN , the radius corresponds to the standard deviation. This control type is only taken into account when M_MOSAIC_COMPOSITION is set to M_SUPER_RESOLUTION and M_SR_PSF_TYPE is not set to M_DISABLE . (summarize)
<input type="checkbox"/> Value ≥ 0.0	Specifies the radius of the PSF, in pixel units at the scale of the source images. The default value is 0.5 pixels. (summarize)
<input type="checkbox"/> M_SR_PSF_TYPE	Sets the type of point-spread function (PSF) to use during super-resolution calculations to model the blurring in source images. The point-spread function describes how a small point of light in the world is blurred by the acquisition setup (lens and CCD). Use M_SR_PSF_RADIUS to set the radius of the PSF. The value of this control type is only taken into account when M_MOSAIC_COMPOSITION is set to M_SUPER_RESOLUTION . (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_GAUSSIAN .

<input type="checkbox"/> M_CIRCULAR	Specifies to assume a PSF that models blurring of single points of light into uniform circles in the image.
<input type="checkbox"/> M_DISABLE	Specifies that no PSF should be assumed during super-resolution calculations.
<input type="checkbox"/> M_GAUSSIAN	Specifies to assume a PSF that models blurring of single points of light into radially symmetric gaussian functions in the image.
<input type="checkbox"/> M_SR_SMOOTHNESS	<p>Sets the smoothness value to use during super-resolution calculations.</p> <p>Use a high smoothness value to reduce local variations in the mosaic image. This is useful to prevent artifacts caused by noise in the source images.</p> <p>Use a low smoothness value when there is little noise in the source image. This is useful to preserve the most detail in your mosaic.</p> <p>This control type is only taken into account when M_MOSAIC_COMPOSITION is set to M_SUPER_RESOLUTION. (summarize)</p>
<input type="checkbox"/> 0.0 <= Value <= 100.0	<p>Specifies the smoothness value.</p> <p>The default value is 50.0. (summarize)</p>

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregDraw

Synopsis

Draw specific features of the registration results in the destination image buffer.

Syntax

```
void MregDraw(
    MIL_ID GraphContId,
    MIL_ID RegResultId,
    MIL_ID DestImageId,
    MIL_INT Operation,
    MIL_INT Index,
    MIL_INT ControlFlag
)
```

Description

This function draws specific features of the registration results in the destination image buffer.

To draw these results, the required information must be available in the registration result buffer.

Parameters

GraphContId

Specifies the graphics context to use when drawing. This parameter must be set to one of the following values:

For specifying the graphics context	
Value	Description
M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

RegResultId

Specifies the identifier of the registration result buffer from which to extract the results to draw. The registration result buffer must have been previously allocated on the required system using [MregAllocResult\(\)](#).

DestImageId

Specifies the identifier of the destination image buffer in which to draw. The destination image can be any supported MIL image. By drawing into a display's overlay buffer, you can also choose to annotate an image non-destructively.

Operation

Specifies the type of operation to perform.

For specifying the operation	
Value	Description
M_DRAW_BOX	Draws the bounding box of the image associated to the specified registration element, at its position, angle, and scale. The position, angle, and scale are determined by the registration. (summarize)

Index

Specifies the registration element that is associated to the image whose features you want to draw. This parameter must be set to one of the values below.

● For specifying the registration element	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> 0 to M_NUMBER_OF_ELEMENTS-1	Specifies the registration element's index.
<input type="checkbox"/> M_ALL	Specifies all registration elements.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregFree

Synopsis

Free a registration context or a registration result buffer.

Syntax

```
void MregFree(  
    MIL_ID ContextOrResultId  
)
```

Description

This function deletes the specified registration context or registration result buffer, and releases any memory associated with it.

All registration contexts and registration result buffers allocated on a particular system must be freed before the system can be freed.

Parameter

ContextOrResultId

Specifies the identifier of the registration context or registration result buffer to free. These must have been successfully allocated (with [MregAlloc\(\)](#) or [MregAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregGetResult

Synopsis

Get the specified type of result(s) from a registration result buffer.

Syntax

```
void MregGetResult(
    MIL_ID RegResultId,
    MIL_INT ResultIndex,
    MIL_INT ResultType,
    void *ResultArrayPtr
)
```

Description

This function retrieves the result(s) of the specified type from a registration result buffer. Results are only available after calling [MregCalculate\(\)](#).

Parameters

RegResultId

Specifies the identifier of the registration result buffer from which to retrieve results.

ResultIndex

Specifies which results to get. This parameter can be set to one of the following values:

● For retrieving results	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> 0 to M_NUMBER_OF_ELEMENTS-1	Specifies the registration result element's index.
<input type="checkbox"/> M_ALL	Retrieves the results of the specified type for all registration result elements.
<input type="checkbox"/> M_GENERAL	Retrieves a result relating to the entire registration result buffer.

ResultType

Specifies the type of result to retrieve.

When retrieving a general result or the results for one or all registration result elements, the [ResultType](#) parameter can be set to one of the following values:

Unless otherwise specified, for one registration result element or [M_GENERAL](#), the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type [MIL_DOUBLE](#) with a size equal to 1 . For [M_ALL](#), the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type [MIL_DOUBLE](#) with a size equal to [MregInquire\(\)](#) with [M_NUMBER_OF_ELEMENTS](#).

● For general results or registration result element results	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_RESULT +	Returns whether the registration calculation (MregCalculate()) was successful. When using M_GENERAL , and the calculation of one registration result element fails, M_RESULT returns a fail as well. (summarize)

	<i>ResultArrayPtr info</i> Return values: M_FAILURE Returned if the calculate operation was a failure. M_SUCCESS Returned if the calculate operation was successful.
<input type="checkbox"/> M_SCORE +	Retrieves the score of the registration calculation (MregCalculate()). To determine the type of score to calculate, use MregControl() with M_SCORE_TYPE . When using M_GENERAL , the score is an average of the scores of all result elements together. (summarize)
	<i>ResultArrayPtr info</i> Return values: 0.0 to 100.0 The lower the value, the less optimal the alignment. For example, 0.0 equals no alignment and 100.0 equals a perfect alignment. Note that image noise can also adversely affect the score.

When retrieving a general result, the [ResultType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to 1 .

● For general results	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_MOSAIC_OFFSET_X +	Retrieves the horizontal offset that will exist between the origin of the mosaic's coordinate system and the left side of the destination image buffer, when you compose the mosaic.
<input type="checkbox"/> M_MOSAIC_OFFSET_Y +	Retrieves the vertical offset that will exist between the origin of the mosaic's coordinate system and the top of the destination image buffer, when you compose the mosaic.
<input type="checkbox"/> M_MOSAIC_SIZE_X +	Retrieves the width that the mosaic will have when it is composed. This value can be used to allocate an image buffer large enough to hold the entire mosaic composed by MregTransformImage() . (summarize)
<input type="checkbox"/> M_MOSAIC_SIZE_Y +	Retrieves the height that the mosaic will have when it is composed. This value can be used to allocate an image buffer large enough to hold the entire mosaic composed by MregTransformImage() . (summarize)

When retrieving the result of one or all the registration result elements, the [ResultType](#) parameter can be set to one of the following values:

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to 1 for one registration result element and `M_NUMBER_OF_ELEMENTS` for `M_ALL` .

● For results (one or all registration result elements)																								
<div><input type="checkbox"/> Value</div>	Description	corona-II (a)	cronosplus (b)	glige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ed/Xd (f)	helios ed/Xd (g)	ilee 1394 iicc (h)	iris (i)	met-II /d (j)	met-II /mc (k)	met-II /dig (k)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ed/Xd (r)	odyssey ed/Xd (s)	solios ea/Xa (t)	solios ec/Xd (u)	solios gige (v)	vio (w)
<div><input type="checkbox"/> M_POSITION_X +</div>	Retrieves the X-coordinate of the registration result element image's origin in the global/reference coordinate system.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><input type="checkbox"/> M_POSITION_Y +</div>	Retrieves the Y-coordinate of the registration result element image's origin in the global/reference coordinate system.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><input type="checkbox"/> M_REVERSE_TRANSFORMATION_MATRIX +</div>	Retrieves the 9 elements of the 3x3 transformation matrix that performs a reverse transformation. The transformation matrix can transform any position in the specified image (x_s, y_s) with a position in the global/reference coordinate system (x_g, y_g). (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

<input type="checkbox"/> M_TRANSFORMED_UL_X +	Retrieves the X-coordinate of the upper left corner of the specified registration result element's image. This coordinate is with respect to the global/reference coordinate system. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_TRANSFORMED_UL_Y +	Retrieves the Y-coordinate of the upper left corner of the specified registration result element's image. This coordinate is with respect to the global/reference coordinate system. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_TRANSFORMED_UR_X +	Retrieves the X-coordinate of the upper right corner of the specified registration result element's image. This coordinate is with respect to the global/reference coordinate system. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_TRANSFORMED_UR_Y +	Retrieves the Y-coordinate of the upper right corner of the specified registration result element's image. This coordinate is with respect to the global/reference coordinate system. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Combination constants for the values listed in [For results \(one or all registration result elements\)](#)

You can add one of the following values to the above-mentioned values to get the requested results with respect to the required coordinate system.

● For specifying the coordinate system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_REFERENCE	Specifies the results with respect to the reference coordinate system. The reference coordinate system is associated with the registration element that was specified in the Target parameter of MregSetLocation() . (summarize)
<input type="checkbox"/> M_REGISTRATION_GLOBAL	Specifies the results with respect to the global coordinate system. This is the default value. (summarize)

Combination constants for [the values listed in](#) all tables

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

● For specifying the data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_DOUBLE	<p>Casts the requested results to a <i>double</i>. This is the default value. (summarize)</p> <div> <i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type double Array size: Depends on the result being cast. Note: When multiple results. • Data type: double Note: When a single result. </div>
<input type="checkbox"/> M_TYPE_LONG	<p>Casts the requested results to a <i>long</i>. (summarize)</p> <div> <i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type long Array size: Depends on the result being cast. Note: When multiple results. • Data type: long Note: When a single result. </div>

<input type="checkbox"/> M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_DOUBLE Array size: Depends on the result being cast. Note: When multiple results. • Data type: MIL_DOUBLE Note: When a single result.
<input type="checkbox"/> M_TYPE_MIL_ID	<p>Casts the requested results to a <i>MIL_ID</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_ID Array size: Depends on the result being cast. Note: When multiple results. • Data type: MIL_ID Note: When a single result.
<input type="checkbox"/> M_TYPE_MIL_INT	<p>Casts the requested results to a <i>MIL_INT</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT Array size: Depends on the result being cast. Note: When multiple results. • Data type: MIL_INT Note: When a single result.
<input type="checkbox"/> M_TYPE_MIL_INT32	<p>Casts the requested results to a <i>MIL_INT32</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT32 Array size: Depends on the result being cast. Note: When multiple results. • Data type: MIL_INT32 Note: When a single result.
<input type="checkbox"/> M_TYPE_MIL_INT64	<p>Casts the requested results to a <i>MIL_INT64</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type MIL_INT64 Array size: Depends on the result being cast. Note: When multiple results. • Data type: MIL_INT64 Note: When a single result.

ResultArrayPtr

<p>Accepts the address of one of the following (see above for specifics on which is expected):</p> <ul style="list-style-type: none"> • array of type double • array of type long • array of type MIL_DOUBLE • array of type MIL_ID • array of type MIL_INT • array of type MIL_INT32 • array of type MIL_INT64 • double • long • MIL_DOUBLE • MIL_ID • MIL_INT • MIL_INT32 • MIL_INT64

Specifies the address of the array in which to write the requested information.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregInquire

Synopsis

Inquire information about a registration context, a specified registration element, or a registration result buffer.

Syntax

```
MIL_INT MregInquire(  
    MIL_ID ContextOrResultId,  
    MIL_INT Index,  
    MIL_INT InquireType,  
    void *UserVarPtr  
)
```

Description

This function inquires information about a specified registration context, the registration elements contained therein, or a specified registration result buffer.

Note that for a registration result buffer, this function only retrieves information about result buffer settings (set with [MregControl\(\)](#), for example). To retrieve results from the registration result buffer, use [MregGetResult\(\)](#).

Parameters

ContextOrResultId

Specifies either the registration context or registration result buffer about which to inquire information. Both the registration context and the registration result buffer must have been previously allocated on the system using [MregAlloc\(\)](#) or [MregAllocResult\(\)](#), respectively.

Index

Specifies that information will be inquired about a registration context, an individual registration element, or a registration result buffer. This parameter should be set to one of the following values:

● For specifying a registration context, registration element, or registration result buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default value. If a registration context is specified, this parameter inquires information about registration element 0. If a result buffer is specified, same as M_GENERAL . (summarize)
<input type="checkbox"/> M_CONTEXT	Inquires information about a registration context, if one is specified.
<input type="checkbox"/> M_GENERAL	Inquires general information about the registration result buffer, if one is specified.
<input type="checkbox"/> Value	Inquires information about a registration element, if a registration context is specified. Set the Index parameter to the index of the required registration element. (summarize)

InquireType

Specifies the setting about which to inquire.

For [M_CONTEXT](#) or [M_GENERAL](#), the [InquireType](#) parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For M_CONTEXT or M_GENERAL	

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_NUMBER_OF_ELEMENTS +	Returns the number of registration elements in the registration context or the number of registration result elements in the result buffer. (summarize)
	<i>UserVarPtr info</i> Return values: 1<=Value<8192; M_DEFAULT; (details)
<input type="checkbox"/> M_OWNER_SYSTEM +	Returns the identifier of the system on which the context was allocated.

For **M_CONTEXT**, the **InquireType** parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For M_CONTEXT	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ACCURACY +	Returns the accuracy of the registration calculation. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_HIGH; M_LOW; (details)
<input type="checkbox"/> M_LOCATION_DELTA +	Returns the maximum displacement that will be applied to any pixel in the image during registration, relative to its initial location. (summarize)
	<i>UserVarPtr info</i> Return values: 0<Value<=100.0; M_DEFAULT; (details)
<input type="checkbox"/> M_MIN_OVERLAP +	Returns the minimum overlap that should exist between an image and its reference image so that the registration calculation can optimize the match in their overlapping region. (summarize)
	<i>UserVarPtr info</i> Return values: 0<Value<=100.0; M_DEFAULT; (details)
<input type="checkbox"/> M_SCORE_TYPE +	Returns the type of score calculated during registration. This calculated score can be retrieved after registration using MregGetResult with M_SCORE. (summarize)
	<i>UserVarPtr info</i> Return values: M_CORRELATION; M_NONE; (details)
<input type="checkbox"/> M_TRANSFORMATION_TYPE +	Returns the type of transformation that the registration calculation will use to optimize the match in the images' overlapping regions. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_PERSPECTIVE; M_TRANSLATION; M_TRANSLATION_ROTATION; M_TRANSLATION_ROTATION_SCALE; (details)

For one or all registration elements, the **InquireType** parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of a *MIL_DOUBLE*.

● For registration elements	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_OPTIMIZE_LOCATION +	Returns whether the optimization step of the registration calculation will be performed. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_REFERENCE_X +	Returns the X-offset between the origin of the registration element's image and the center of the image's top-left pixel.

	(summarize)
	<i>UserVarPtr info</i> Return values: M_CENTER_ELEMENT; M_DEFAULT; Value; (details)
<input type="checkbox"/> M_REFERENCE_Y +	Returns the Y-offset between the origin of the registration element's image and the center of the image's top-left pixel. (summarize)
	<i>UserVarPtr info</i> Return values: M_CENTER_ELEMENT; M_DEFAULT; Value; (details)
<input type="checkbox"/> M_SET_LOCATION_PARAM_1 +	Returns the first attribute of the transformation used to set the rough location of the registration element's image. If the first attribute was set to a MIL identifier, this identifier cannot be returned; M_NULL will be returned instead. This is the case when the rough location was set using MregSetLocation() with, for example, M_COPY_REG_CONTEXT or M_COPY_REG_RESULT. (summarize)
	<i>UserVarPtr info</i> Return values: Please see the Param1 parameter of MregSetLocation() . (details)
<input type="checkbox"/> M_SET_LOCATION_PARAM_2 +	Returns the second attribute of the transformation used to set the rough location of the registration element's image. (summarize)
	<i>UserVarPtr info</i> Return values: Please see the Param2 parameter of MregSetLocation() . (details)
<input type="checkbox"/> M_SET_LOCATION_PARAM_3 +	Returns the third attribute of the transformation used to set the rough location of the registration element's image. (summarize)
	<i>UserVarPtr info</i> Return values: Please see the Param3 parameter of MregSetLocation() . (details)
<input type="checkbox"/> M_SET_LOCATION_PARAM_4 +	Returns the fourth attribute of the transformation used to set the rough location of the registration element's image. (summarize)
	<i>UserVarPtr info</i> Return values: Please see the Param4 parameter of MregSetLocation() . (details)
<input type="checkbox"/> M_SET_LOCATION_PARAM_TYPE +	Returns the type of transformation used to set the rough location of the registration element's image. (summarize)
	<i>UserVarPtr info</i> Return values: M_COPY_REG_CONTEXT; M_COPY_REG_RESULT; M_POSITION_XY; M_POSITION_XY_ANGLE; M_WARP_POLYNOMIAL; M_WARP_4_CORNER; M_WARP_4_CORNER_REVERSE;
<input type="checkbox"/> M_SET_LOCATION_TARGET +	Returns the reference of the registration element's image. (summarize)
	<i>UserVarPtr info</i> Return values: 0 <= Value <= Number of input images-1; M_DEFAULT; M_NEXT; M_PREVIOUS; M_REGISTRATION_GLOBAL; (details)

For [M_GENERAL](#), the [InquireType](#) parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a [MIL_DOUBLE](#).

● For M_GENERAL	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_X +	Returns the X-coordinate, in the global coordinate system, that will be the starting point of the drawing region. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)

<input type="checkbox"/> M_DRAW_RELATIVE_ORIGIN_Y +	<p>Returns the Y-coordinate, in the global coordinate system, that will be the starting point of the drawing region. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value; (details)</p>
<input type="checkbox"/> M_DRAW_SCALE_X +	<p>Returns the scale factor, in the X-direction, to perform zoomed drawings of results. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)</p>
<input type="checkbox"/> M_DRAW_SCALE_Y +	<p>Returns the scale factor, in the Y-direction, to perform zoomed drawings of results. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value > 0; (details)</p>
<input type="checkbox"/> M_MOSAIC_COMPOSITION +	<p>Returns which image's pixel values to use when composing the mosaic and two or more images overlap. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_AVERAGE_IMAGE; M_DEFAULT; M_FIRST_IMAGE; M_FUSION_IMAGE; M_LAST_IMAGE; M_SUPER_RESOLUTION; (details)</p>
<input type="checkbox"/> M_MOSAIC_OFFSET_X +	<p>Returns the X-offset between the origin of the coordinate system used to compose the mosaic and the left side of the destination image buffer. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_ALIGN_LEFT; M_DEFAULT; Value; (details)</p>
<input type="checkbox"/> M_MOSAIC_OFFSET_Y +	<p>Returns the Y-offset between the origin of the coordinate system used to compose the mosaic and the top of the destination image buffer. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_ALIGN_TOP; M_DEFAULT; Value; (details)</p>
<input type="checkbox"/> M_MOSAIC_SCALE +	<p>Returns the scaling factor used on images while creating the mosaic. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 < Value <= 10.0; (details)</p>
<input type="checkbox"/> M_MOSAIC_STATIC_INDEX +	<p>Returns the element whose coordinate system is used when generating the mosaic. If the coordinate system of one of the images is used, the index of its corresponding registration element is returned. If the coordinate system which minimizes changes to the images or the global coordinate system is used, M_ALL or M_REGISTRATION_GLOBAL will be returned, respectively. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0 <= Value <= NumberOfElements-1; M_ALL; M_DEFAULT; M_REGISTRATION_GLOBAL; (details)</p>
<input type="checkbox"/> M_SR_PSF_RADIUS +	<p>Returns the radius of the M_CIRCULAR or M_GAUSSIAN point-spread functions (PSF). For M_GAUSSIAN, the radius corresponds to the standard deviation. (summarize)</p> <p><i>UserVarPtr info</i> Return values: Value >= 0.0; (details)</p>
<input type="checkbox"/> M_SR_PSF_TYPE +	<p>Returns the type of point-spread function (PSF) to use during super-resolution calculations to model the blurring in source images. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_CIRCULAR; M_DEFAULT; M_DISABLE; M_GAUSSIAN; (details)</p>
<input type="checkbox"/> M_SR_SMOOTHNESS +	<p>Returns the smoothness value used during super-resolution calculations. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 <= Value <= 100.0; (details)</p>

Combination constants for any of the possible values of the [InquireType](#) parameter

You can add one of the following values to the above-mentioned values to cast the requested information to a required data type.

● For specifying the data type	
☐ Value	Description
☐ M_TYPE_DOUBLE	Casts the requested information to a <i>double</i> . (summarize)
	<i>UserVarPtr info</i> Data type: double
☐ M_TYPE_LONG	Casts the requested information to a <i>long</i> . (summarize)
	<i>UserVarPtr info</i> Data type: long
☐ M_TYPE_MIL_DOUBLE	Casts the requested information to a <i>MIL_DOUBLE</i> . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE
☐ M_TYPE_MIL_ID	Casts the requested information to a <i>MIL_ID</i> . Note that M_TYPE_MIL_ID should only be used with M_OWNER_SYSTEM . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_ID
☐ M_TYPE_MIL_INT	Casts the requested information to a <i>MIL_INT</i> . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_INT
☐ M_TYPE_MIL_INT32	Casts the requested information to a <i>MIL_INT32</i> . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_INT32
☐ M_TYPE_MIL_INT64	Casts the requested information to a <i>MIL_INT64</i> . (summarize)
	<i>UserVarPtr info</i> Data type: MIL_INT64

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- double
- long
- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT32
- MIL_INT64

Specifies the address in which to write the requested information. Since the **MregInquire()** function also returns the requested information, you can set this parameter to **M_NULL**.

Return value

The return value is the required information, cast to a *MIL_INT*.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregRestore

Synopsis

Restore a registration context or result buffer from disk.

Syntax

```
MIL_ID MregRestore(
    MIL_CONST_TEXT_PTR FileName,
    MIL_ID SystemId,
    MIL_INT ControlFlag,
    MIL_ID *ContextOrResultIdPtr
)
```

Description

This function restores a registration context or result buffer that was previously saved to a file, using [MregSave\(\)](#) or [MregStream\(\)](#). This function restores all of the registration context's or result buffer's settings that were in effect when it was saved.

Parameters

FileName

Specifies the name and path of the file from which to restore the registration context. The function handles (internally) the opening and closing of the file.

● For specifying the file name and path		
☐ Value	Description	
☐ MIL_TEXT(MIL_TEXT_PTR FileName)	Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile". (summarize)	
		Parameters
	FileName	Specifies the drive, directory, and name of the file.
☐ M_INTERACTIVE	[This is only applicable to Windows] Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.	

SystemId

Specifies the system on which to restore the registration context.

This parameter should be set to one of the following values:

● For specifying the system identifier		
☐ Value	Description	
☐ M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.	
☐ MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .	

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextOrResultIdPtr

Specifies the address of the variable in which to write the registration context or result buffer identifier. Since the **MregRestore()** function also returns the registration context or result buffer identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the registration context or result buffer identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregSave

Synopsis

Save a registration context or the contents of a registration result buffer to a file.

Syntax

```
void MregSave(  
    MIL_CONST_TEXT_PTR FileName,  
    MIL_ID ContextOrResultId,  
    MIL_INT ControlFlag  
)
```

Description

This function saves all the information about the previously allocated registration context or registration result buffer to disk. This information can be reloaded, using [MregRestore\(\)](#) or [MregStream\(\)](#). Note that saving the result buffer is useful if you want the reuse the registration results.

Parameters

FileName

Specifies the name and path of the file in which to save the registration context or the contents of the registration result buffer. It is recommended that you use the MRE file extension for easier use with other Matrox Imaging software products. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following values:

For specifying the file name and path					
Value	Description				
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p> <table><tr><th colspan="2">Parameters</th></tr><tr><td>FileName</td><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters		FileName	Specifies the drive, directory, and name of the file.
Parameters					
FileName	Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</p>				

ContextOrResultId

Specifies either the identifier of the registration context or the registration result buffer to save.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregSetLocation

Synopsis

Sets the rough location of the specified registration element's image with respect to another registration element's image or the global coordinate system.

Syntax

```
void MregSetLocation(
    MIL_ID ContextId,
    MIL_INT Index,
    MIL_INT Target,
    MIL_INT ParamType,
    MIL_DOUBLE Param1,
    MIL_DOUBLE Param2,
    MIL_DOUBLE Param3,
    MIL_DOUBLE Param4,
    MIL_INT ControlFlag
)
```

Description

This function sets the rough location of the specified registration element's image with respect to another registration element's image or the global coordinate system. The [MregCalculate\(\)](#) function uses this location to find the transformations that optimally position the images in the global coordinate system.

You can explicitly specify the rough location or you can copy it from a registration element of another registration context or a result element of a registration result buffer. When copying the rough location, you can use the same reference index as the element from which you are copying the rough location. Alternatively, you can specify a different reference index.

The more precise the information, the faster the registration calculation will be. If you know the exact position of the image, you can bypass the optimization step of the calculation by either disabling the [MregControl\(\)](#) [M_OPTIMIZE_LOCATION](#) control type or replacing the image in the array by [M_NULL](#) (see the [Skipping the optimization step](#) subsection in the [Customizing your registration settings](#) section in [Chapter 10: Registration](#)). In this case, the transformation that you specify using [MregSetLocation\(\)](#) becomes the optimal transformation. Upon calling [MregCalculate\(\)](#), this transformation will be converted so that it maps the image into the global coordinate system without first converting it into its reference image's coordinate system.

Some restrictions apply when selecting the image's reference. The settings cannot be circularly defined. If you take a specific image and look at its reference, and then at its reference image's reference, and continue through the series of images linked in this manner, the first image must never appear as another image's reference. Furthermore, there should only be one image with its reference set to the global coordinate system.

Parameters

ContextId

Specifies the registration context of the registration element to affect. The registration context must have been previously allocated on the required system using [MregAlloc\(\)](#).

Index

Specifies the index of the registration element. This parameter can be set to one of the following:

For specifying the index	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 0 <= Value <= Number of input images-1	Specifies the index of the registration element.
<input type="checkbox"/> M_ALL	Specifies that all of the registration elements will have their location set.

Target

Specifies the reference coordinate system of the registration element's image.

● For specifying the reference	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_PREVIOUS .
<input type="checkbox"/> 0 <= Value <= Number of input images-1	Specifies the index of the registration element of the reference image.
<input type="checkbox"/> M_COPY	Uses the same index as the reference of the element from which positional information is being copied. This setting is only available if the ParamType parameter is set to either M_COPY_REG_CONTEXT or M_COPY_REG_RESULT . (summarize)
<input type="checkbox"/> M_NEXT	Sets the reference to the image associated with the registration element whose index follows the specified registration element's index. When the registration element's image is the last one in the input sequence, the reference is set to M_REGISTRATION_GLOBAL , the global coordinate system. (summarize)
<input type="checkbox"/> M_PREVIOUS	Sets the reference to the image associated with the registration element whose index precedes the specified registration element's index. When the registration element has an index 0, the reference is set to M_REGISTRATION_GLOBAL , the global coordinate system. (summarize)
<input type="checkbox"/> M_REGISTRATION_GLOBAL	Specifies that the reference is the global coordinate system.
<input type="checkbox"/> M_UNCHANGED	Specifies that only the rough locations are changed.

ParamType

Specifies the type of transformation to use to set the location of the registration element's image.

See the [Parameter associations](#) section for possible values.

Param1

Specifies an attribute of the transformation. Its definition depends on the value of [ParamType](#).

See the [Parameter associations](#) section for possible values.

Param2

Specifies an attribute of the transformation. Its definition depends on the value of [ParamType](#).

See the [Parameter associations](#) section for possible values.

Param3

Specifies an attribute of the transformation. Its definition depends on the value of [ParamType](#).

See the [Parameter associations](#) section for possible values.

Param4

Specifies an attribute of the transformation. Its definition depends on the value of [ParamType](#).

See the [Parameter associations](#) section for possible values.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Parameter associations

Possible values for the **ParamType**, **Param1**, **Param2**, **Param3**, and **Param4** parameters are described in the table below.

- [For specifying the transformation type](#)

Note that any unused parameters should be set to **M_DEFAULT**.

For specifying the transformation type	
ParamType	Description
Param1	
Param2	
Param3	
Param4	
M_COPY_REG_CONTEXT	Specifies that the rough locations are copied from another registration context. (summarize)
Param1	Specifies the identifier of the registration context from which to copy the transformation settings. (summarize)
Param2	Specifies the registration element index from which to copy the transformation settings. (summarize)
M_DEFAULT	Specifies the default index. The default index is the same as the one specified for the Index parameter. (summarize)
0 <= Value <= Number of input images-1	Specifies the registration element's index.
M_COPY_REG_RESULT	Specifies that the transformation between two registration result elements is copied. (summarize)
Param1	Specifies the identifier of the registration result from which to copy the transformations. (summarize)
Param2	Specifies the registration result element's index from which to copy the transformation settings. (summarize)
M_DEFAULT	Specifies the index that is specified in the Index parameter.
0 <= Value <= Number of input images-1	Specifies the registration result element's index.
Param3	Specifies the registration result element's index of the target transformation element. (summarize)
M_DEFAULT	Specifies that the target registration results element's index is the same as the one that is specified in the Target parameter.
M_REGISTRATION_GLOBAL	Specifies that the target registration result element's reference is the global coordinate system.
Value >= 0	Specifies the target registration result element's index.
M_POSITION_XY	Sets the coordinates of the origin of the registration element's image in the coordinate system of its reference. Note that when the reference is an image, the specified coordinates need not lie within the reference image. (summarize)
Param1	Sets the X-coordinate of the image's origin in its reference coordinate system. (summarize)
Value	Specifies an X-coordinate.
Param2	Sets the Y-coordinate of the image's origin in its reference coordinate system. (summarize)
Value	Specifies a Y-coordinate.
M_POSITION_XY_ANGLE	Sets the coordinates of the origin and the angle of the registration element's image in the coordinate system of its reference. Note that when the reference is an image, the specified coordinates need not lie within the reference image. (summarize)
Param1	Sets the X-coordinate of the image's origin in its reference coordinate system. (summarize)
Value	Specifies an X-coordinate.

<input type="checkbox"/> Param2	Sets the Y-coordinate of the image's origin in its reference coordinate system. (summarize)
<input type="checkbox"/> Value	Specifies a Y-coordinate.
<input type="checkbox"/> Param3	Sets the angle of the image in its reference coordinate system. The angle is measured in degrees in counter-clockwise direction. (summarize)
<input type="checkbox"/> Value	Specifies an angle.
<input type="checkbox"/> M_WARP_4_CORNER	Sets the transformation that transforms an arbitrary quadrilateral in the current image's coordinate system into a rectangle in its reference coordinate system. You specify the transformation by supplying 4 points in the current image and the corresponding 4 points in the reference image. (summarize)
<input type="checkbox"/> Param1	Specifies the MIL identifier of the buffer to use to store the coordinates of the 8 points. This buffer should have an M_ARRAY attribute. For more information on the buffer and its contents, see MgenWarpParameter() . (summarize)
<input type="checkbox"/> M_WARP_4_CORNER_REVERSE	Sets the transformation that transforms a rectangle in the current image's coordinate system into an arbitrary quadrilateral in its reference coordinate system. You specify the transformation by supplying 4 points in the current image and the corresponding 4 points in the reference system. (summarize)
<input type="checkbox"/> Param1	Specifies the MIL identifier of the buffer to use to store the coordinates of the 8 points needed to warp the image. This buffer should have an M_ARRAY attribute. For more information on the buffer and its contents, see MgenWarpParameter() . (summarize)
<input type="checkbox"/> M_WARP_POLYNOMIAL	Sets the transformation matrix that transforms points in the reference coordinate system into points in the current image's coordinate system. (summarize)
<input type="checkbox"/> Param1	Specifies the MIL identifier of the buffer containing the transformation matrix. For more information on the buffer and its contents, see MimWarp() . (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregStream

Synopsis

Load, restore, or save a registration context or result buffer from/to a file or memory stream.

Syntax

```
void MregStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *ContextOrResultIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load, restore, or save a registration context and/or registration result buffer from/to a file or memory stream.

To inquire the number of bytes necessary to save a registration context or result buffer to a memory stream, you should call this function (**MregStream()**) first with **M_INQUIRE_SIZE_BYTE**.

Note that the content saved to memory stream is equivalent to the content saved to file. In addition, any file saved with **MregSave()** is equivalent to a file saved using this function.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using **MregStream()**, you can choose to save a backwards-compatible version of the registration context, which will work using a version of MIL that is older than the current version (depending on which version is specified). For example, if you allocate a registration context using MIL 9.0 and save it to version 8.0 Processing Pack 4, you can restore this context on a computer where MIL 8.0 Processing Pack 4 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore registration contexts saved using MIL version 8.0 Processing Pack 4 or above. Settings that do not exist in the lower version will be filled with default values when the registration context is loaded or restored.

Parameters

MemPtrOrFileName
Specifies the file or memory stream.

● For specifying the name and path of a file or memory stream

<div><div></div>Value</div>	Description
<div><div><div><div></div></div><div>MIL_TEXT(MIL_TEXT_PTR <i>FileName</i>)</div></div></div>	<div><p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving a registration context to a file, use the MRE file extension. The function internally handles the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p><p>To open or save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile".</p><div><div>(summarize)</div></div></div> <div><div><div></div></div><div><div>Parameters</div><div><div>FileName</div><div>Specifies the drive, directory, and name of the file.</div></div></div></div>
<div><div><div></div></div><div>M_INTERACTIVE</div></div>	<div>[<i>This is only applicable to Windows</i>]</div>

	Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. The parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (MregStream()) first with M_INQUIRE_SIZE_BYTE . (summarize)

SystemId

Specifies the system on which to restore the registration context or result buffer. For **M_INQUIRE_SIZE_BYTE**, **M_LOAD**, and **M_SAVE**, this parameter is ignored and should be set to **M_NULL**. For an **M_RESTORE** operation, this parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform. This parameter must be set to one of the following values:

● For specifying the type of operation to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save a registration context or result buffer to a memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated registration context or result buffer.
<input type="checkbox"/> M_RESTORE	Restores a registration context or result buffer from a file or memory stream and assigns it an identifier.
<input type="checkbox"/> M_SAVE	Saves a registration context or result buffer to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the registration context or result buffer. This parameter must be set to one of the following values:

● For specifying the type of stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream.
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)

Version

Specifies the MIL version of the registration context. This parameter must be set to one of the following values.

● For specifying the MIL version	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL.

	For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
<input type="checkbox"/> M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Specifies the function's control flag. For any operation on registration contexts or result buffers, this parameter must be set to **M_DEFAULT**.

ContextOrResultIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the registration context or the registration result buffer.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, **ContextOrResultIdPtr** specifies the address of the variable from which to read the registration context result buffer identifier.

For an [M_LOAD](#) operation, **ContextOrResultIdPtr** specifies the address of the variable from which to read the registration context or result buffer identifier where the file or memory stream content will be loaded.

For an [M_RESTORE](#) operation, **ContextOrResultIdPtr** specifies the address of the variable in which to return the restored registration context or result buffer identifier. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the registration context, in bytes.

If the size is not required, you can set this parameter to **M_NULL**.

Note that the size of the registration context or result buffer will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregTransformCoordinate

Synopsis

Convert a pair of coordinates between two of the following coordinate systems: the global coordinate system, any registered image's coordinate system, and the mosaic's coordinate system.

Syntax

```
void MregTransformCoordinate(
    MIL_INT RegResultId,
    MIL_INT Source,
    MIL_INT Destination,
    MIL_DOUBLE X,
    MIL_DOUBLE Y,
    MIL_DOUBLE *ResXPtr,
    MIL_DOUBLE *ResYPtr,
    MIL_INT ControlFlag
)
```

Description

This function converts a pair of coordinates between two of the following coordinate systems: the global coordinate system, any registered image's coordinate system, and the mosaic's coordinate system.

Parameters

RegResultId

Specifies the registration result buffer that contains the information that will be used to transform the coordinates. The registration result buffer must have been previously allocated on the required system using [MregAllocResult\(\)](#) or restored from a file using [MregRestore\(\)](#).

Source


Specifies the source coordinate system. This parameter must be set to one of the following values:

For the source coordinate system	
Value	Description
<input type="checkbox"/> M_MOSAIC	Specifies to use the coordinate system relative to which the mosaic will be composed. This coordinate system was specified using MregControl() with M_MOSAIC_STATIC_INDEX . (summarize)
<input type="checkbox"/> M_REGISTRATION_GLOBAL	Specifies the global coordinate system.
<input type="checkbox"/> Value	Specifies the index of the registration result element associated with the image whose coordinate system to use.

Destination

Specifies the destination coordinate system. This parameter must be set to one of the following values:

For the destination coordinate system	
Value	Description
<input type="checkbox"/> M_MOSAIC	Specifies to use the coordinate system relative to which the mosaic will be composed. This coordinate system was specified using MregControl() with M_MOSAIC_STATIC_INDEX . (summarize)
<input type="checkbox"/> M_REGISTRATION_GLOBAL	Specifies the global coordinate system.

 Value	Specifies the index of the registration result element associated with the image whose coordinate system to use.
-----------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------

X

Specifies the X-coordinate in the source coordinate system.

Y

Specifies the Y-coordinate in the source coordinate system.

ResXPtr

Specifies the address in which to write the resulting X-coordinate. This coordinate is given in the coordinate system that is specified in the [Destination](#) parameter.

ResYPtr

Specifies the address in which to write the resulting Y-coordinate. This coordinate is given in the coordinate system that is specified in the [Destination](#) parameter.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregTransformCoordinateList

Synopsis

Convert a list of coordinates between two of the following coordinate systems: the global coordinate system, any registered image's coordinate system, and the mosaic's coordinate system.

Syntax

```
void MregTransformCoordinateList(
    MIL_INT RegResultId,
    MIL_INT Source,
    MIL_INT Destination,
    MIL_INT NumPoints,
    const MIL_DOUBLE *SrcArrayXPtr,
    const MIL_DOUBLE *SrcArrayYPtr,
    MIL_DOUBLE *DesArrayXPtr,
    MIL_DOUBLE *DesArrayYPtr,
    MIL_INT ControlFlag
)
```

Description

Convert a list of coordinates between two of the following coordinate systems: the global coordinate system, any registered image's coordinate system, and the mosaic's coordinate system.

Parameters

RegResultId

Specifies the registration result buffer that contains the information that will be used to transform the coordinates. The registration result buffer must have been previously allocated on the required system using [MregAllocResult\(\)](#) or restored from a file using [MregRestore\(\)](#).

Source



Specifies the source coordinate system. This parameter must be set to one of the following values:

For the source coordinate system	
Value	Description
<input type="checkbox"/> M_MOSAIC	Specifies to use the coordinate system relative to which the mosaic will be composed. This coordinate system was specified using MregControl() with M_MOSAIC_STATIC_INDEX . (summarize)
<input type="checkbox"/> M_REGISTRATION_GLOBAL	Specifies the global coordinate system.
<input type="checkbox"/> Value	Specifies the index of the registration result element associated with the image whose coordinate system to use.

Destination

Specifies the destination coordinate system. This parameter must be set to one of the following values:

For the destination coordinate system	
Value	Description
<input type="checkbox"/> M_MOSAIC	Specifies to use the coordinate system relative to which the mosaic will be composed. This coordinate system was specified using MregControl() with M_MOSAIC_STATIC_INDEX . (summarize)

 M_REGISTRATION_GLOBAL	Specifies the global coordinate system.
 Value	Specifies the index of the registration result element associated with the image whose coordinate system to use.

NumPoints

Specifies the number of points in the coordinate list.

SrcArrayXPtr

Specifies the address of the array of the X-coordinates in the source coordinate system.

SrcArrayYPtr

Specifies the address of the array of the Y-coordinates in the source coordinate system.

DesArrayXPtr

Specifies the address in which to write the resulting X-coordinate. This coordinate is given in the coordinate system that is specified in the [Destination](#) parameter.

DesArrayYPtr

Specifies the address in which to write the resulting Y-coordinate. This coordinate is given in the coordinate system that is specified in the [Destination](#) parameter.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

MregTransformImage

Synopsis

Composes a mosaic from the specified source images and stores it in the destination image buffer.

Syntax

```
void MregTransformImage(
    MIL_ID RegResultId,
    const MIL_ID *ImageArrayPtr,
    MIL_ID DestImageBufId,
    MIL_INT NumImages,
    MIL_INT InterpolationMode,
    MIL_INT ControlFlag
)
```

Description

This function composes a mosaic from the specified source images and stores it in the destination image buffer. The source images are transformed according to the results, stored in the registration result buffer, that were calculated during the [MregCalculate\(\)](#) operation.

Each image in the image array is associated with the registration result element with the same index (for example, the third image in the array is associated with the third registration result element). Note that for the image to appear at the correct location in the mosaic, its index in the array must match the index of the registration result element containing the image's transformation.

It is possible to use some images and omit others when composing the mosaic. In this case, replace the images that you want to omit by **M_NULL** in the image array. It can be useful to omit images from a mosaic if you want to add images to a preexisting mosaic, for example. In this case, replace all the images in the array by **M_NULL**, except the ones to add, and call **MregTransformImage()** with the preexisting mosaic passed in [DestImageBufId](#).

When composing the mosaic, you can control how it will appear in the destination image buffer. You can specify which image to place first, using [M_MOSAIC_STATIC_INDEX](#). This image will be placed upright and will not be scaled. The other images in the mosaic will be positioned relative to this image's coordinate system. In addition, you can also adjust the vertical and horizontal offsets between the origin of the mosaic's coordinate system and the origin of the destination image buffer, using [MregControl\(\)](#) with [M_MOSAIC_OFFSET_X](#) and [M_MOSAIC_OFFSET_Y](#).

Parameters

- RegResultId
- Specifies the registration result buffer to use to perform the mosaicing operation. The registration result buffer must have been previously allocated or restored on the required system using [MregAllocResult\(\)](#) or [MregRestore\(\)](#), respectively.
- ImageArrayPtr
- Specifies the array of image identifiers. To omit images from the mosaic, replace their identifiers with **M_NULL** in the array.
- DestImageBufId
- Specifies the destination image buffer that will contain the mosaic after the **MregTransformImage()** operation is performed.
- NumImages
- Specifies the number of images in the image array.
- InterpolationMode
- Specifies the interpolation mode. This parameter must be set to one of the values below.

● For specifying the interpolation mode

<div><div></div>Value</div>	Description
<div><div></div>M_DEFAULT +</div>	Same as M_NEAREST_NEIGHBOR + M_OVERSCAN_ENABLE .
<div><div></div>M_BICUBIC +</div>	Specifies bicubic interpolation. Saturation is performed according to the type of the destination buffer. (summarize)
<div><div></div>M_BILINEAR +</div>	Specifies bilinear interpolation. No saturation is performed. (summarize)
<div><div></div>M_NEAREST_NEIGHBOR +</div>	Specifies nearest neighbor interpolation. No saturation is performed. (summarize)

Combination constants for any of the possible values of the [InterpolationMode](#) parameter

You can add one of the following values to the above-mentioned values to specify the type of overscan to use for a destination pixel when its associated point falls outside the source buffers.

● For overscan	
<div><div></div>Value</div>	Description
<div><div></div>M_OVERSCAN_CLEAR</div>	Sets the destination pixel to 0.
<div><div></div>M_OVERSCAN_DISABLE</div>	Leaves the destination pixel as is.
<div><div></div>M_OVERSCAN_ENABLE</div>	Uses pixels from the source buffer's ancestor buffer. If the source buffer is not a child buffer or if the point falls outside all of the ancestor buffers, leave the destination pixel as is. This is the default value. (summarize)

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milreg.lib.
DLL	Requires mil.dll; milreg.dll.

Mstr functions

Synopsis

The functions prefixed with Mstr make up the String Reader module. The String Reader module performs character recognition using feature-based technology. The module is invariant to changes in scale, aspect ratio, and contrast. String Reader also offers considerable tolerance towards variations in perspective and angle. In addition, String Reader supports multiple user-defined grammar rules and multi-font definitions in a single context, and features a powerful Graphical User Interface (GUI) for easy font definition. String Reader also offers an expert function, allowing you to quickly determine why your application is not reading the strings you expect. All this makes the String Reader module the ideal tool for writing seemingly complex applications, such as Automatic Number Plate Recognition (ANPR) programs.

Functions

- [MstrAlloc](#)
- [MstrAllocResult](#)
- [MstrControl](#)
- [MstrDraw](#)
- [MstrEditFont](#)
- [MstrExpert](#)
- [MstrFree](#)
- [MstrGetResult](#)
- [MstrInquire](#)
- [MstrPreprocess](#)
- [MstrRead](#)
- [MstrRestore](#)
- [MstrSave](#)
- [MstrSetConstraint](#)
- [MstrStream](#)

MstrAlloc

Synopsis

Allocates a String Reader context.

Syntax

```
MIL_ID MstrAlloc(  
    MIL_ID SystemId,  
    MIL_INT ContextType,  
    MIL_INT ControlFlag,  
    MIL_ID *ObjectIdPtr  
)
```

Description

This function allocates a String Reader context on the specified system. A String Reader context contains the read parameters, the list of fonts, and the list of string models. When the String Reader context is no longer required, you should release its memory, using [MstrFree\(\)](#).

Note that you cannot allocate a fontless context. You must restore one of the predefined fontless context that is distributed with MIL using [MstrRestore\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the String Reader context.

This parameter should be set to one of the following values:

For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ContextType

Specifies the type of String Reader context. This parameter must be set to the value below.

For specifying the context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FONT_BASED	Specifies a font-based context.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ObjectIdPtr

Specifies the address of the variable in which to write the String Reader context identifier. Since the **MstrAlloc()** function also returns the String Reader context identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the String Reader context identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrAllocResult

Synopsis

Allocate a String Reader result buffer.

Syntax

```
MIL_ID MstrAllocResult(
    MIL_ID SystemId,
    MIL_INT ControlFlag,
    MIL_ID *ObjectIdPtr
)
```

Description

This function allocates a String Reader result buffer, on the specified system, to store results obtained from an [MstrRead\(\)](#) operation. When the String Reader result buffer is no longer required, release its memory, using [MstrFree\(\)](#).

Parameters

SystemId

Specifies the system on which to allocate the String Reader result buffer.

This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ObjectIdPtr

Specifies the address of the variable in which to write the String Reader result buffer identifier. Since the **MstrAllocResult()** function also returns the String Reader result buffer identifier, you can set this parameter to **M_NULL**.

Return value

The returned value is the result buffer identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrControl

Synopsis

Control a String Reader context, a specific string model, or a specific font.

Syntax

```
void MstrControl(
    MIL_ID ContextId,
    MIL_INT Index,
    MIL_INT ControlType,
    MIL_DOUBLE ControlValue
)
```

Description

This function sets the specified control for a String Reader context, or for one (or all) of the specified string models or fonts contained therein. These settings control the execution of [MstrRead\(\)](#) operations. All the control type settings can be inquired using [MstrInquire\(\)](#).

Changing certain control type settings might require preprocessing the String Reader context again, using [MstrPreprocess\(\)](#). To know if the String Reader context needs to be preprocessed, call [MstrInquire\(\)](#) with [M_PREPROCESSED](#).

By setting the [ControlType](#) parameter to [M_INTERACTIVE](#), you can set the control types interactively.

Unless otherwise specified, the control type settings are applicable to both font-based and fontless contexts.

Parameters

ContextId

Specifies the String Reader context whose settings to modify. The String Reader context must have been previously allocated on the system using [MstrAlloc\(\)](#) or [MstrAllocResult\(\)](#), respectively.

Index

Specifies that the String Reader context, a specific font, or a specific string model is controlled. Set this parameter to one of the following values:

● For a String Reader context, specific font, or specific string model				
☐ Value	Description			
☐ M_DEFAULT	Same as M_CONTEXT .			
☐ M_FONT_INDEX(MIL_INT FontIndex)	Specifies to apply the control settings to a font. Note that this value is only available for a font-based context. (summarize)			
	Parameters			
	FontIndex This parameter specifies the index of the font. You can set this parameter to one of the following:			
	<table><tr><td>Value >= 0</td><td>Specifies the index of the individual font to which to apply the control settings. User labels cannot be used as indices; to retrieve the index of a font from its user label, use MstrInquire() with M_FONT_INDEX_FROM_LABEL.</td></tr><tr><td>M_ALL</td><td>Specifies to apply the control settings to all fonts.</td></tr></table>	Value >= 0	Specifies the index of the individual font to which to apply the control settings. User labels cannot be used as indices; to retrieve the index of a font from its user label, use MstrInquire() with M_FONT_INDEX_FROM_LABEL .	M_ALL
Value >= 0	Specifies the index of the individual font to which to apply the control settings. User labels cannot be used as indices; to retrieve the index of a font from its user label, use MstrInquire() with M_FONT_INDEX_FROM_LABEL .			
M_ALL	Specifies to apply the control settings to all fonts.			
☐ M_STRING_INDEX(MIL_INT StringIndex)	Specifies to apply the control settings to a string. (summarize)			

	<i>Parameters</i>	
	<i>StringIndex</i>	
	This parameter specifies the index of the string. You can set this parameter to one of the following:	
	Value ≥ 0	Specifies the index of the individual string to which to apply the control settings. User labels cannot be used as indices; to retrieve the index of a string from its user label, use MstrInquire() with M_STRING_INDEX_FROM_LABEL .
<input type="checkbox"/> M_CONTEXT	M_ALL	Specifies to apply the control settings to all strings. Note that to apply control settings to all strings, the ControlType and ControlValue parameters must be supported by all the string models.
	Specifies that a general setting of a String Reader context will be controlled.	

ControlType

Specifies the setting to change.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the setting's new value.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For the context or a specific string](#)
- [For the context](#)
- [For a specific string](#)
- [For a specific font](#)
- [For a font-based or a fontless context](#)

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings can be specified for either the String Reader context ([M_CONTEXT](#)) or each specific string in the String Reader context ([M_STRING_INDEX\(\)](#)):

For the context or a specific string	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens a dialog box that allows you to edit the control types interactively.</p> <p>For M_CONTEXT, this setting opens a dialog box that allows you to edit the control types of the specified String Reader context.</p> <p>For a string model, this setting opens a dialog box that allows you to edit the string model's control types interactively. Once the dialog box is opened, you can change between string models interactively using the Index field.</p> <p>For a font, this setting opens a dialog box that allows you to edit the font's control types interactively. Once the dialog box is opened, you can change between fonts interactively using the Index field.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.
<input type="checkbox"/> M_STRING_NUMBER	Sets either the maximum (total) number of all strings that can be read with a String Reader context (M_CONTEXT), or the maximum number of a specific string (M_STRING_INDEX()) that can be read with a String Reader context.
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.

	(summarize)
<input type="checkbox"/> M_ALL	Specifies that all strings will be read. Note that this setting can increase the read time; always set M_STRING_NUMBER to a specific number whenever possible. (summarize)
<input type="checkbox"/> Value	Specifies the maximum number of strings that can be read. For M_CONTEXT , this value specifies the maximum (total) number of strings that can be read. Any value greater than or equal to 1 can be set. For M_STRING_INDEX() , this value specifies the maximum number of a specific string model that can be read. Any value greater than or equal to 0 can be set. (summarize)

The possible [ControlType](#) and corresponding [ControlValue](#) parameter settings for the String Reader context ([M_CONTEXT](#)) are:

For the context	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_DISABLE_CHAR	Disables the specified character in the fontless context. Only uppercase characters and numbers can be disabled. The String Reader module will not try to match found characters with disabled characters. Note that this control type is only available for a fontless context. (summarize)
<input type="checkbox"/> ASCII value	Specifies the ASCII value of the character to disable.
<input type="checkbox"/> M_ALL	Specifies that no characters will be read.
<input type="checkbox"/> M_ENABLE_CHAR	Enables the specified character in the fontless context. Only uppercase characters and numbers can be enabled. The string reader module will only try to identify enabled characters. Note that this control type is only available for a fontless context. (summarize)
<input type="checkbox"/> ASCII value	Specifies the ASCII value of the character.
<input type="checkbox"/> M_ALL	Specifies that all characters will be read. This is the default value. (summarize)
<input type="checkbox"/> M_ENCODING	Sets the type of character encoding used by the String Reader context. All functions that accept or return a string of characters as a parameter use character encoding to determine the type of the pointer. The type of character encoding is also used as the default result type (MstrGetResult()) for a string of characters. The character encoding cannot be changed when there are characters in the context whose ASCII character values lie outside the range [0, 127]. This applies to characters which are in fonts and constraints. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ASCII .
<input type="checkbox"/> M_ASCII	Specifies an 8-bit ASCII standard. This corresponds to the <i>char</i> data type. (summarize)
<input type="checkbox"/> M_UNICODE	Specifies a 16-bit Unicode standard. This corresponds to the <i>short</i> data type. (summarize)
<input type="checkbox"/> M_FONT_ADD	Adds an empty font to the String Reader context. The maximum number of fonts in a String Reader context is 255. To add characters to or to edit the font, use MstrEditFont() .

	Note that this control type is only available for a font-based context. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_USER_DEFINED .
<input type="checkbox"/> M_USER_DEFINED	Specifies an empty font. You must use MstrEditFont() to add characters to it. (summarize)
<input checked="" type="checkbox"/> M_FONT_DELETE	Deletes a font from the String Reader context. Note that this control type is only available for a font-based context. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> M_ALL	Specifies that all fonts will be deleted.
<input type="checkbox"/> Value > = 0	Specifies the index of the font to delete.
<input checked="" type="checkbox"/> M_MAX_CHAR_SIZE_X	Sets the width (X-size) of the widest enabled character in the fontless context. Specify the width that the character has when it has a height of M_REF_CHAR_SIZE_Y . If the read algorithm encounters a character with a height other than M_REF_CHAR_SIZE_Y , the maximum width will be interpolated. Note that this control type is only available for a fontless context. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 65 pixels. (summarize)
<input type="checkbox"/> 0 to 65536	Specifies the maximum character width in pixels.
<input checked="" type="checkbox"/> M_MIN_CHAR_SIZE_X	Sets the width (X-size) of the narrowest enabled character in the fontless context. Specify the width that the character has when it has a height of M_REF_CHAR_SIZE_Y . If the read algorithm encounters a character with a height other than M_REF_CHAR_SIZE_Y , the maximum width will be interpolated. Note that this control type is only available for a fontless context. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 6 pixels. (summarize)
<input type="checkbox"/> 0 to 65536	Specifies the minimum character width in pixels.
<input checked="" type="checkbox"/> M_MINIMUM_CONTRAST	Sets the minimum contrast between a character in the target image and its background, in order for the character to be read by MstrRead() . Increasing the minimum contrast will typically speed up the read operation, particularly in presence of noise; however, if the minimum contrast is higher than the contrast of an actual character, that character will be lost. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 15. (summarize)
<input type="checkbox"/> 1 to 255	Specifies the minimum contrast.
<input checked="" type="checkbox"/> M_REF_CHAR_SIZE_Y	Sets the height which will be the reference for M_MIN_CHAR_SIZE_X , M_MAX_CHAR_SIZE_X , and M_REF_CHAR_THICKNESS . The reference height is the height at which you specify the width of the narrowest and widest characters, as well as their thickness, with M_MIN_CHAR_SIZE_X , M_MAX_CHAR_SIZE_X , and M_REF_CHAR_THICKNESS , respectively. Note that this control type is only available for a fontless context. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 46 pixels. (summarize)
<input type="checkbox"/> 0 to 65536	Specifies the height in pixels.
<input checked="" type="checkbox"/> M_REF_CHAR_THICKNESS	Sets the thickness (stroke width) of enabled characters in the fontless context. Specify the thickness of characters when they have a height of

	<p>M_REF_CHAR_SIZE_Y.</p> <p>If the read algorithm encounters a character with a height other than M_REF_CHAR_SIZE_Y, the maximum width will be interpolated.</p> <p>Note that this control type is only available for a fontless context. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 6 pixels. (summarize)
<input type="checkbox"/> 0 to 65536	Specifies the thickness in pixels.
<input type="checkbox"/> M_SEARCH_CHAR_ANGLE	<p>Sets whether to perform calculations specific to angular-range search strategies, for characters.</p> <p>The read algorithm is naturally robust to variations in a character's angle. Therefore, a character might still be read correctly, even if M_SEARCH_CHAR_ANGLE is disabled. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Disables calculations specific to angular-range search strategies.
<input type="checkbox"/> M_ENABLE	Enables calculations specific to angular-range search strategies.
<input type="checkbox"/> M_SEARCH_SKEW_ANGLE	<p>Sets whether to perform calculations specific to angular-range skew search strategies, for characters.</p> <p>The read algorithm is naturally robust to variations in a character's skew. Therefore, a character might still be read correctly, even if M_SEARCH_SKEW_ANGLE is disabled. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .
<input type="checkbox"/> M_DISABLE	Disables calculations specific to angular-range skew search strategies.
<input type="checkbox"/> M_ENABLE	Enables calculations specific to angular-range skew search strategies.
<input type="checkbox"/> M_SEARCH_STRING_ANGLE	<p>Sets whether to search for the string angle. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_ENABLE .
<input type="checkbox"/> M_DISABLE	Disables the search for the string angle.
<input type="checkbox"/> M_ENABLE	Enables the search for the string angle.
<input type="checkbox"/> M_SPACE_CHARACTER	<p>Sets the character value to use as a space character within the formatted string or text that will be read.</p> <p>M_SPACE_CHARACTER applies only when getting the formatted string (MstrGetResult() with M_FORMATTED_STRING) or the formatted text (MstrGetResult() with M_TEXT). A space character is inserted in the string each time the distance between two adjacent characters exceeds the space width, at the scale of the string.</p> <p>The space character must be set before the string is read. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is numerical ASCII code 32. (summarize)
<input type="checkbox"/> M_NONE	Specifies no space character.
<input type="checkbox"/> Value	Specifies the space character, as its numerical code (ASCII or Unicode). For example, ASCII 10 results in a new line character, ASCII 32 results in a space character. Note that the character value must be cast to a <i>MIL_DOUBLE</i> . (summarize)
<input type="checkbox"/> M_SPEED	<p>Sets the search and read speed of the MstrRead() operation. Note that increasing the speed can decrease the robustness of MstrRead(). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_MEDIUM .
<input type="checkbox"/> M_HIGH	Sets the speed to high.
<input type="checkbox"/> M_MEDIUM	Sets the speed to medium.

<input type="checkbox"/> M_STOP_EXPERT	Stops the current string expert operation. Note that you can only perform this operation from another thread of higher priority. (summarize)
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.
<input type="checkbox"/> M_STOP_READ	Stops the current string read operation. Note that you can only perform this operation from another thread of higher priority. (summarize)
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.
<input type="checkbox"/> M_STRING_ADD	Adds a string to the String Reader context. The maximum number of strings in a String Reader context is 255. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_USER_DEFINED .
<input type="checkbox"/> M_USER_DEFINED	Specifies a user-defined string. Initially, a user-defined string does not have any constraints, does not use any checksum error detection, and all its parameters are set to their default value. (summarize)
<input type="checkbox"/> M_STRING_DELETE	Deletes a string from the String Reader context. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> M_ALL	Specifies that all strings will be deleted.
<input type="checkbox"/> Value > = 0	Specifies the index of the string to delete.
<input type="checkbox"/> M_STRING_SEPARATOR	Sets the character value to use as a string separator within the formatted text that will be read. M_STRING_SEPARATOR applies only when getting the formatted text (MstrGetResult() with M_TEXT). A string separator is inserted between each string in the text; that is, each time the distance between two adjacent characters is greater than the space size. The space size depends on the space width (at the scale of the string) and the maximum number of consecutive spaces. The string separator character must be set before the string is read. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is numerical ASCII code 10. (summarize)
<input type="checkbox"/> M_NONE	Specifies no separator character.
<input type="checkbox"/> Value	Specifies the separator character, as its numerical code (ASCII or Unicode). For example, ASCII 10 results in a new line character, ASCII 32 results in a space character. Note that the character value must be cast to a <i>MIL_DOUBLE</i> . (summarize)
<input type="checkbox"/> M_THICKEN_CHAR	Sets the number of character thickening iterations. Each iteration enlarges the thickness of the target character. This is useful for some types of fonts (for example, dotted characters). You should choose a value large enough for the intra character dots to connect but not too large to connect separate characters together. Note that the characters included in the font should not be dotted. Instead a thickened version should be included. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0. (summarize)
<input type="checkbox"/> 0 to 100	Specifies the number of character thickening iterations.
<input type="checkbox"/> M_THRESHOLD_MODE	Sets the threshold method of the character blob extraction. (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_LOCAL_WITH_RESEGMENTATION .

<input type="checkbox"/> M_LOCAL	Specifies local thresholding that uses only one threshold step that is automatically computed by the module. This threshold method can be used in various contrast situations and can improve the read operation's speed, compared to M_LOCAL_WITH_RESEGMENTATION . (summarize)
<input type="checkbox"/> M_LOCAL_WITH_RESEGMENTATION	Specifies local thresholding that uses multiple threshold steps and split and merge techniques. This threshold method allows the module to extract character blobs in low contrast and in complex situations such as merged or broken characters. (summarize)
<input type="checkbox"/> M_USER_DEFINED	Specifies global thresholding that uses the threshold value set with M_THRESHOLD_VALUE . This threshold method improves the read operation's speed, compared to M_LOCAL_WITH_RESEGMENTATION and M_LOCAL . (summarize)
<input type="checkbox"/> M_THRESHOLD_VALUE	Sets the global threshold value when M_THRESHOLD_MODE is set to M_USER_DEFINED (otherwise it is ignored). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_AUTO_COMPUTE .
<input type="checkbox"/> 0 to 255	Specifies the threshold value.
<input type="checkbox"/> M_AUTO_COMPUTE	Computes automatically the global threshold value.
<input type="checkbox"/> M_TIMEOUT	Sets the maximum read time for MstrRead() , in msec. If the read operation times out, the strings read up to that point will be returned as a result. Note that the actual maximum read time might vary slightly from the time that you set. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 2000.0 msec. (summarize)
<input type="checkbox"/> M_DISABLE	Specifies an infinite amount of read time.
<input type="checkbox"/> Value > = 0	Specifies the maximum read time, in msec.

The possible [ControlType](#) and corresponding [ControlValue](#) parameter settings for a specific string ([M_STRING_INDEX\(\)](#)) are:

For a specific string	
<input type="checkbox"/> ControlType	Description
ControlValue	
<input type="checkbox"/> M_CHAR_ACCEPTANCE	Sets the acceptance level for the character score. The character's score quantifies the similarity between the character in the target and the character in the font; it also quantifies the similarity between the character in the target and the other characters of the string in the target. To be accepted as a string result, the character's score must be greater than or equal to M_CHAR_ACCEPTANCE . Scores can be retrieved with MstrGetResult() . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 50.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies an acceptable score, as a percentage.
<input type="checkbox"/> M_CHAR_ASPECT_RATIO_MAX_FACTOR	Sets the factor used to determine the maximum acceptable aspect ratio for the characters in the string. This value is relative to the found string aspect ratio. That is, $UpperLimit = FoundStringAspectRatio \times M_CHAR_ASPECT_RATIO_MAX_FACTOR$. This value sets the maximum possible aspect ratio variation for the characters in the string. Characters with an aspect ratio outside the aspect ratio range will not be returned as results. Note that this control type is only available for a font-based context. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.1. (summarize)
<input type="checkbox"/> 1.0 to 2.0	Specifies the factor that determines the maximum aspect ratio of the characters in the string.

<input type="checkbox"/> M_CHAR_ASPECT_RATIO_MIN_FACTOR	<p>Sets the factor used to determine the minimum acceptable aspect ratio for the characters in the string. This value is relative to the found string aspect ratio. That is, $LowerLimit = FoundStringAspectRatio \times M_CHAR_ASPECT_RATIO_MIN_FACTOR$.</p> <p>This value sets the minimum possible aspect ratio variation for the characters in the string. Characters with an aspect ratio outside the aspect ratio range will not be returned as results.</p> <p>Note that this control type is only available for a font-based context. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.9. (summarize)
<input type="checkbox"/> 0.5 to 1.0	Specifies the factor that determines the minimum aspect ratio of the characters in the string.
<input type="checkbox"/> M_CHAR_HOMOGENEITY_ACCEPTANCE	<p>Sets the acceptance level for the character's homogeneity score.</p> <p>The character's homogeneity score quantifies the similarity between the character in the target and the other characters of the string in the target.</p> <p>To be accepted as a string result, the character's homogeneity score must be greater than or equal to M_CHAR_HOMOGENEITY_ACCEPTANCE.</p> <p>Scores can be retrieved with MstrGetResult(). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 50.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies an acceptable score, as a percentage.
<input type="checkbox"/> M_CHAR_MAX_BASELINE_DEVIATION	<p>Sets the maximum baseline deviation that all characters defined in the font can have in the target image.</p> <p>A character's maximum baseline deviation is the maximum tolerable baseline deviation that a character in a string can have before it is rejected from the string. For more information on character baseline deviations, see the Character's maximum baseline deviation subsection in the Degrees of freedom section in Chapter 12: String Reader.</p> <p>Note that for non-punctuation characters, String Reader calculates the baseline deviation, as a percentage of each individual character's height (Y-size). For example, a baseline deviation of 10 means that a character's baseline deviates from the string's baseline by 10% of the character's height (Y-size). For punctuation characters (M_PUNCTUATION), String Reader calculates the baseline deviation as a percentage of the height of the character with the greatest Y-size in the font. For example, a baseline deviation of 10 means that the punctuation character's baseline deviates from the string's baseline by 10% of the height of the tallest character in the font. This is done because the Y-size of punctuation characters is typically too small to calculate the baseline deviation as a percentage of the character's height. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 10. (summarize)
<input type="checkbox"/> 0 to 100	Specifies the character's maximum baseline deviation within the string model.
<input type="checkbox"/> M_CHAR_SCALE_MAX_FACTOR	<p>Sets the factor used to determine the upper limit (maximum permitted scale) of the scale range for the characters in the string. This value is relative to the found string scale. That is, $UpperLimit = FoundStringScale \times M_CHAR_SCALE_MAX_FACTOR$.</p> <p>This value sets the possible increasing scale variation for the characters in the string. Characters with a scale outside the scale range will not be returned as results. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.1. (summarize)
<input type="checkbox"/> 1.0 to 2.0	Specifies the factor that determines the maximum scale of the characters in the string.
<input type="checkbox"/> M_CHAR_SCALE_MIN_FACTOR	<p>Sets the factor used to determine the lower limit (minimum permitted scale) of the scale range for the characters in the string. This value is relative to the found string scale. That is, $LowerLimit = FoundStringScale \times M_CHAR_SCALE_MIN_FACTOR$.</p> <p>This value sets the possible decreasing scale variation for the characters in the string. Characters with a scale outside the scale range will not be returned as results. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.9. (summarize)

<input type="checkbox"/> 0.5 to 1.0	Specifies the factor that determines the minimum scale of the characters in the string.
<input type="checkbox"/> M_CHAR_SIMILARITY_ACCEPTANCE	<p>Sets the acceptance level for the character's similarity score.</p> <p>The character's similarity score quantifies the similarity between the character in the target and the character in the font.</p> <p>To be accepted as a string result, the character's similarity score must be greater than or equal to M_CHAR_SIMILARITY_ACCEPTANCE.</p> <p>Scores can be retrieved with MstrGetResult(). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 50.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies an acceptable score, as a percentage.
<input type="checkbox"/> M_FOREGROUND_VALUE	<p>Sets the foreground color of the string. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_FOREGROUND_BLACK .
<input type="checkbox"/> M_FOREGROUND_BLACK	Specifies that black is the foreground color of the string.
<input type="checkbox"/> M_FOREGROUND_BLACK_OR_WHITE	Specifies that the foreground color of the string can be either black or white.
<input type="checkbox"/> M_FOREGROUND_WHITE	Specifies that white is the foreground color of the string.
<input type="checkbox"/> M_SPACE_MAX_CONSECUTIVE	<p>Sets the maximum number of consecutive space characters allowed in the string.</p> <p>Together with the space width (MstrControl() with M_SPACE_WIDTH), M_SPACE_MAX_CONSECUTIVE determines the maximum distance between 2 adjacent characters of the same string. If the distance between two characters is greater than the maximum distance, then they are considered to be part of 2 different strings. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 3. (summarize)
<input type="checkbox"/> Value > = 0	Specifies the maximum number of consecutive space characters.
<input type="checkbox"/> M_STRING_ACCEPTANCE	<p>Sets the acceptance level for the string score.</p> <p>To be accepted as a string result, both the string's score and the string target score must be greater than or equal to their respective acceptance levels.</p> <p>The string score is the average score of the characters in the string.</p> <p>Scores can be retrieved with MstrGetResult(). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 50.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies an acceptable score, as a percentage.
<input type="checkbox"/> M_STRING_ANGLE	<p>Sets the nominal angle of the string. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 0.0°.</p> <p>M_STRING_ANGLE can only be set to M_DEFAULT. (summarize)</p>
<input type="checkbox"/> M_STRING_ANGLE_DELTA_NEG	<p>Sets the lower limit of the string's angular range, relative to the nominal angle (MstrControl() with M_STRING_ANGLE). That is, <i>LowerLimit</i> = M_STRING_ANGLE - M_STRING_ANGLE_DELTA_NEG.</p> <p>This value sets the possible clockwise angular variation of the string. Strings with angles outside the angular range will not be returned as results. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 10.0 °. (summarize)

<input type="checkbox"/> 0.0 to 10.0	Specifies the lower limit of the angular range, in degrees.
<input type="checkbox"/> M_STRING_ANGLE_DELTA_POS	<p>Sets the upper limit of the angular range, relative to the nominal angle (MstrControl() with M_STRING_ANGLE). That is, $UpperLimit = M_STRING_ANGLE + M_STRING_ANGLE_DELTA_POS$.</p> <p>This value sets the possible counter-clockwise angular variation of the string. Strings with angles outside the angular range will not be returned as results. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 10.0 °. (summarize)
<input type="checkbox"/> 0.0 to 10.0	Specifies the upper limit of the angular range, in degrees.
<input type="checkbox"/> M_STRING_ASPECT_RATIO	<p>Sets the nominal aspect ratio of the string.</p> <p>The aspect ratio of the string is the median aspect ratio of all the characters in the string, while the aspect ratio of a character is the ratio of its scale in X by its scale in Y. This indicates how the width and the height of the characters in the string vary relative to the characters in the font.</p> <p>Note that this control type is only available for a font-based context. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> 0.5 to 2.0	Specifies the pixel width/pixel height of the target.
<input type="checkbox"/> M_STRING_ASPECT_RATIO_MAX_FACTOR	<p>Sets the factor used to determine the upper limit of the string's aspect ratio. This value is relative to the nominal aspect ratio (MstrControl() with M_STRING_ASPECT_RATIO). That is, $UpperLimit = M_STRING_ASPECT_RATIO \times M_STRING_ASPECT_RATIO_MAX_FACTOR$.</p> <p>This value sets the possible increasing aspect ratio variation for the string. Strings with an aspect ratio outside the aspect ratio range will not be returned as results.</p> <p>Note that this control type is only available for a font-based context. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.25. (summarize)
<input type="checkbox"/> 1.0 to 2.0	Specifies the factor that determines the maximum aspect ratio of the string.
<input type="checkbox"/> M_STRING_ASPECT_RATIO_MIN_FACTOR	<p>Sets the factor used to determine the lower limit of the string's aspect ratio. This value is relative to the nominal aspect ratio (MstrControl() with M_STRING_ASPECT_RATIO). That is, $LowerLimit = M_STRING_ASPECT_RATIO \times M_STRING_ASPECT_RATIO_MIN_FACTOR$.</p> <p>This value sets the possible decreasing aspect ratio variation for the string. Strings with an aspect ratio outside the aspect ratio range will not be returned as results.</p> <p>Note that this control type is only available for a font-based context. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.8. (summarize)
<input type="checkbox"/> 0.5 to 1.0	Specifies the factor that determines the minimum aspect ratio of the string.
<input type="checkbox"/> M_STRING_CERTAINTY	<p>Sets the certainty level for the string score.</p> <p>If both the string score and the string target score of a string candidate are above their respective certainty levels, the candidate is immediately considered a string result, without reading the rest of the target image for a string with higher scores (provided the specified number of strings to read has been read).</p> <p>Scores can be retrieved with MstrGetResult(). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 70.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies the certainty level for the string score, as a percentage.
<input type="checkbox"/> M_STRING_SCALE	Sets the nominal scale of the string.

	The scale of the string is the median scale in the X-direction of all the characters in the string. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1.0. (summarize)
<input type="checkbox"/> 0.25 to 4.0	Specifies the value of the scale.
<input checked="" type="checkbox"/> M_STRING_SCALE_MAX_FACTOR	Sets the factor used to determine the upper limit (maximum permitted scale) of the string's scale range. This value is relative to the nominal scale (MstrControl () with M_STRING_SCALE). That is, <i>UpperLimit</i> = M_STRING_SCALE x M_STRING_SCALE_MAX_FACTOR . This value sets the possible increasing scale variation for the string. Strings with a scale outside the scale range will not be returned as results. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 2.0. (summarize)
<input type="checkbox"/> 1.0 to 2.0	Specifies the factor that determines the minimum scale of the string.
<input checked="" type="checkbox"/> M_STRING_SCALE_MIN_FACTOR	Sets the factor used to determine the lower limit (minimum permitted scale) of the string's scale range. This value is relative to the nominal scale (MstrControl () with M_STRING_SCALE). That is, <i>LowerLimit</i> = M_STRING_SCALE x M_STRING_SCALE_MIN_FACTOR . This value sets the possible decreasing scale variation for the string. Strings with a scale outside the scale range will not be returned as results. (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 0.5. (summarize)
<input type="checkbox"/> 0.5 to 1.0	Specifies the factor that determines the minimum scale of the string.
<input checked="" type="checkbox"/> M_STRING_SIZE_MAX	Sets the maximum string size (number of characters) of the string model. The space characters are not counted in the string size. The maximum string size must be greater than or equal to the minimum string size (MstrControl () with M_STRING_SIZE_MIN). (summarize)
<input type="checkbox"/> M_DEFAULT	Same as M_INFINITE .
<input type="checkbox"/> M_INFINITE	Specifies no maximum string size.
<input type="checkbox"/> Value > = 1	Specifies the maximum string size.
<input checked="" type="checkbox"/> M_STRING_SIZE_MIN	Sets the minimum string size (number of characters) of the string model. The space characters are not counted in the string size. The minimum string size must be lower than or equal to the maximum string size (MstrControl () with M_STRING_SIZE_MAX). (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 1. (summarize)
<input type="checkbox"/> Value > = 1	Specifies the minimum string size.
<input checked="" type="checkbox"/> M_STRING_TARGET_ACCEPTANCE	Sets the acceptance level for the string target score. To be accepted as a string result, both the string's score and the string target score must be greater than or equal to their respective acceptance levels. The string target score is computed from the unread characters that are in the region of the string in the target image. These otherwise readable characters remain unread due to user-specified constraints, such as scale and grammar restrictions. The more unread characters there are, the lower the target score will be. Scores can be retrieved with MstrGetResult() . (summarize)
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 50.0 percent. (summarize)
<input type="checkbox"/> 0.0 to 100.0	Specifies an acceptable string target score, as a percentage.
<input checked="" type="checkbox"/> M_STRING_TARGET_CERTAINTY	Sets the certainty level for the string target score.

	<p>If both the string score and the string target score of a string candidate are above their respective certainty levels, the candidate is immediately considered a string result, without reading the rest of the target image for a string with higher scores (provided the specified number of strings to read has been read).</p> <p>Scores can be retrieved with MstrGetResult(). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 70.0 percent. (summarize)</p>
<input type="checkbox"/> 0.0 to 100.0	<p>Specifies the certainty level for the string target score, as a percentage.</p>
<input type="checkbox"/> M_STRING_USER_LABEL	<p>Sets a unique user-defined label for the specified string model. This label can be used as a means of identifying your string model, independently from its index, in the String Reader context. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Same as M_NO_LABEL.</p>
<input type="checkbox"/> M_NO_LABEL	<p>Specifies that no user label is associated with the string model.</p>
<input type="checkbox"/> Value	<p>Specifies the user label of the string model. The value must be an integer that is not associated as a label with any other string model in the String Reader context. Since the same label cannot be associated with multiple string models, you cannot pass a user label if you use MstrControl() with M_STRING_INDEX() set to M_ALL. (summarize)</p>

The possible [ControlType](#) and corresponding [ControlValue](#) parameter settings for a specific font ([M_FONT_INDEX\(\)](#)) are:

● For a specific font	
<input type="checkbox"/> ControlType	Description
<input type="checkbox"/> ControlValue	
<input type="checkbox"/> M_DRAW_BOX_MARGIN_X	<p>Sets the margin in the X-direction, between the character box and the drawing box.</p> <p>The character box is the bounding box of the character. The drawing box is the box used for drawing purposes. For more information, see MstrDraw(). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 3. (summarize)</p>
<input type="checkbox"/> Value > = 0	<p>Specifies the horizontal margin, in pixels.</p>
<input type="checkbox"/> M_DRAW_BOX_MARGIN_Y	<p>Sets the margin in the Y-direction, between the character box and the drawing box.</p> <p>The character box is the bounding box of the character. The drawing box is the box used for drawing purposes. For more information, see MstrDraw(). (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Specifies the default value. The default value is 3. (summarize)</p>
<input type="checkbox"/> Value > = 0	<p>Specifies the vertical margin, in pixels.</p>
<input type="checkbox"/> M_FONT_USER_LABEL	<p>Sets a unique user-defined label for the specified font. This label can be used as a means of identifying your font, independently from its index, in the String Reader context. (summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Same as M_NO_LABEL.</p>
<input type="checkbox"/> M_NO_LABEL	<p>Specifies that no user label is associated with the font.</p>
<input type="checkbox"/> Value	<p>Specifies the user label of the font. The value must be an integer that is not associated as a label with any other font in the String Reader context. Since the same label cannot be associated with multiple fonts, you cannot pass a user label if you use MstrControl() with M_FONT_INDEX() set to M_ALL. (summarize)</p>

The following [ControlType](#) and corresponding [ControlValue](#) parameter settings can be specified for either a font-based context or for a fontless context.

Note that the [Index](#) parameter must be set to [M_CONTEXT](#) for a font-based context or to [M_FONT_INDEX](#) for a fontless context.

For a font-based or a fontless context	
ControlType	Description
ControlValue	
<input type="checkbox"/> M_SPACE_WIDTH	<p>Sets the width of the space character of the font.</p> <p>Note that the space character is not like other characters in the font. For example, you cannot set it as a constraint (MstrSetConstraint()), and it is not counted in the string size (MstrControl() with M_STRING_SIZE_MIN and M_STRING_SIZE_MAX). For more information, see MstrGetResult() with M_STRING_SIZE, M_STRING, M_FORMATTED_STRING_SIZE and M_FORMATTED_STRING.</p> <p>Together with the maximum number of consecutive space characters allowed in the string (MstrControl() with M_SPACE_MAX_CONSECUTIVE), M_SPACE_WIDTH determines the maximum distance between 2 adjacent characters of the same string. If the distance between two characters is greater than the maximum distance, then they are considered to be part of 2 different strings. M_SPACE_WIDTH also determines how to convert the distance into space characters, when getting formatted string results (MstrGetResult() with M_FORMATTED_STRING).</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	Same as M_MEAN_CHAR_WIDTH .
<input type="checkbox"/> M_INFINITE	<p>Specifies that the width of the space character is infinite.</p> <p>In this case, all characters on the same line are always part of the same string. Also, there will never be a space character in the read string since the distance can never be large enough to constitute a space character.</p> <p>(summarize)</p>
<input type="checkbox"/> M_MAX_CHAR_WIDTH	Specifies that the width of the space character is equal to the maximum character X-size of the font.
<input type="checkbox"/> M_MEAN_CHAR_WIDTH	Specifies that the width of the space character is equal to the average character X-size of the font.
<input type="checkbox"/> M_MIN_CHAR_WIDTH	Specifies that the width of the space character is equal to the minimum character X-size of the font.
<input type="checkbox"/> M_QUARTER_MAX_CHAR_WIDTH	<p>Specifies that the width of the space character is equal to a quarter of the maximum character width of the font.</p> <p>M_QUARTER_MAX_CHAR_WIDTH is the closest to the typical space width of a standard font.</p> <p>(summarize)</p>
<input type="checkbox"/> Value > = 1	Specifies the width of the space character, in pixels.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrDraw

Synopsis

Draw specific features of the String Reader context or String Reader results.

Syntax

```
void MstrDraw(
    MIL_ID GraphContId,
    MIL_ID ContextOrResultId,
    MIL_ID DestImageId,
    MIL_INT Operation,
    MIL_INT Index,
    const void *CharList,
    MIL_INT ControlFlag
)
```

Description

This function draws specific features of the String Reader context or String Reader results in the destination image buffer.

After each drawing operation, the optimal image size that was needed for that operation is updated in [M_DRAW_LAST_SIZE_X](#) and [M_DRAW_LAST_SIZE_Y](#) ([MstrInquire\(\)](#)). When **M_NULL** is passed as the destination image ([DestImageId](#) parameter), only these size constants are updated; that is, the drawing operation is not performed. This can be useful if, for example, you only want to get the optimal image buffer size for the drawing operation.

Parameters

GraphContId

Specifies the graphics context to use when drawing. This parameter must be set to one of the following values:

● For specifying the graphics context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies that the default graphics context of the current MIL application is used.
<input type="checkbox"/> MIL graphics context identifier	Specifies a valid graphics context identifier, which you have allocated using MgraAlloc() .

ContextOrResultId

Specifies the String Reader context or String Reader result buffer from which to extract the features to draw. The String Reader context or result buffer must have been previously allocated on the system using [MstrAlloc\(\)](#) or [MstrAllocResult\(\)](#), respectively.

DestImageId

Specifies the identifier of the destination image buffer in which to draw. The buffer can be any supported MIL image buffer or **M_NULL**. By drawing into its display's overlay buffer, you can also annotate an image non-destructively.

When this parameter is set to **M_NULL**, only the size constants are updated ([M_DRAW_LAST_SIZE_X](#) and [M_DRAW_LAST_SIZE_Y](#) with [MstrInquire\(\)](#)); that is, the drawing operation is not performed.

Operation

Specifies the type of operation to perform.

The following [Operation](#) parameter value can only be set for a String Reader context.

● For a font-based context	

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_CHAR	<p>Draws a character representation of the font in the destination image. Characters are drawn from left to right, and from top to bottom. Characters that fall outside the destination image are clipped.</p> <p>The characters are drawn according to their draw box margin, set with M_DRAW_BOX_MARGIN_X and M_DRAW_BOX_MARGIN_Y in MstrControl(). This margin will determine the amount of white space between each character that is drawn.</p> <p>Note that this operation is only available if using a font-based context.</p> <p>(summarize)</p>

The following [Operation](#) parameter values can only be set for a String Reader result buffer. Note that these values can be added together to draw multiple features at once.

Unless otherwise specified, you can use these operations if results were obtained using either font-based and fontless contexts.

When applicable, features are always drawn at the location read in the target with the correct angle, scale, and aspect ratio.

● For a result buffer	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DRAW_MIL_FONT_STRING	<p>Draws all the characters of the String Reader result(s) under the bottom left corner of the string bounding box (M_DRAW_STRING_BOX). The font associated with the graphics context is used to draw the characters.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DRAW_STRING	<p>Draws all the characters of String Reader result(s).</p> <p>Note that this operation is only available if the results were obtained using a font-based context.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DRAW_STRING_BOX	<p>Draws a box around the string that is read in the target. Note that the bounding box of the string is the bounding box of all the string's characters, including the draw margin (set with M_DRAW_BOX_MARGIN_X and M_DRAW_BOX_MARGIN_Y in MstrControl()).</p> <p>(summarize)</p>
<input type="checkbox"/> M_DRAW_STRING_CHAR_BOX	<p>Draws all the character boxes of the String Reader result(s). The character box drawn is the bounding box of the character plus the draw margin (set with M_DRAW_BOX_MARGIN_X and M_DRAW_BOX_MARGIN_Y in MstrControl()).</p> <p>(summarize)</p>
<input type="checkbox"/> M_DRAW_STRING_CHAR_POSITION	<p>Draws a cross at the center of gravity of each string's character.</p>
<input type="checkbox"/> M_DRAW_STRING_CONTOUR	<p>Draws the contour of all the characters of the result(s) of the string(s) read.</p> <p>Note that this operation is only available if the results were obtained using a font-based context.</p> <p>(summarize)</p>

Index

Specifies the index of the font feature or result feature to draw.

Set this parameter to one of the following values:

● For specifying the font or result	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	<p>Draws the default features for either fonts or results.</p> <p>For a String Reader context, the default value is the same as M_FONT_INDEX().</p> <p>For a String Reader result, the default value is the same as M_ALL.</p> <p>(summarize)</p>
<input type="checkbox"/> M_FONT_INDEX(MIL_INT <i>FontIndex</i>)	<p>Draws the features of a specific font.</p> <p>Note that this operation is only available if using a font-based context.</p> <p>(summarize)</p>
	<i>Parameters</i>
	<i>FontIndex</i>

	This parameter specifies the index of the font. Set this parameter to the following:	
	Value >= 0	Specifies a specific font.
	M_ALL	Specifies all fonts.
<input type="checkbox"/> M_ALL	Draws the features of all results.	
<input type="checkbox"/> Value >= 0	Draws the features of a specific result.	

CharList

Specifies an explicit list of valid characters to draw, at the specified position. This is an optional, null-terminated string.

This parameter is not needed when drawing results and should be set to **M_NULL**.

Characters will be drawn in the order given in the [CharList](#) parameter. To draw the characters on multiple lines, a space character must be inserted to separate the strings included in the null-terminated terminated string.

The array of strings must be of the right type for the encoding scheme selected ([MstrControl\(\)](#) with [M_ENCODING](#)). At preprocessing time, each character in an explicit character list must exist in at least one of the fonts of the constraint.

ControlFlag

Specifies where in the destination image buffer to draw.

For a String Reader result, this parameter must be set to [M_DEFAULT](#).

For a String Reader context, set this parameter to one of the following values:

● For specifying where to draw	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Draws features from the top-left corner of the destination image buffer.
<input type="checkbox"/> M_ORIGINAL	Draws characters offset from their original position in the definition image, for user-defined characters (set using M_USER_DEFINED in MstrEditFont()). Note that they are all offset to 0 for other types of characters (for example, TrueType). By drawing user-defined characters immediately after they are defined, you can ascertain where the characters are located in the definition image (for example, in the overlay on top of the definition image). For more information, refer to M_DEFINITION_OFFSET_X and M_DEFINITION_OFFSET_Y in MstrInquire() . (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrEditFont

Synopsis

Edit a specified font.

Syntax

```
void MstrEditFont(
    MIL_ID ContextId,
    MIL_INT FontIndex,
    MIL_INT Operation,
    MIL_INT OperationMode,
    MIL_INT Param1,
    const void *Param2,
    const void *Param3
)
```

Description

This function allows you to edit a specified font. For example, you can add or remove characters from a font, as well as normalize a character in a font. Use [MstrControl\(\)](#) with [M_FONT_ADD](#) to add a font to the context.

Note that this function is only available for a font-based context.

Parameters

ContextId

Specifies the String Reader context that contains the font to edit. The String Reader context must have been previously allocated on the required system using [MstrAlloc\(\)](#).

FontIndex

Specifies the index of the font to edit.

● For specifying the font to edit	
☐ Value	Description
☐ M_FONT_INDEX(MIL_INT FontIndex)	Specifies the font to which to apply the edit settings. (summarize)
	Parameters
	FontIndex This parameter specifies the index of the font to edit. You can set this parameter to the following:
	Value >= 0 Specifies the index of the individual font.

Operation

Specifies the operation to be performed, or opens an interactive dialog box.

See the [Parameter associations](#) section for possible values.

OperationMode

Specifies the mode of the operation.

See the [Parameter associations](#) section for possible values.

Param1

Specifies a value that is dependent on the operation and mode chosen.

See the [Parameter associations](#) section for possible values.

Param2

Specifies a value that is dependent on the operation and mode chosen.

See the [Parameter associations](#) section for possible values.

Param3

Specifies a value that is dependent on the operation and mode chosen.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **Operation**, **OperationMode**, **Param1**, **Param2**, and **Param3** parameters are described in the table below.

- [For performing the operation](#)

To specify the operation to perform, set the **Operation**, **OperationMode**, **Param1**, **Param2**, and **Param3** parameters to the following values:

For performing the operation	
Operation	Description
OperationMode	
Param1	
Param2	
Param3	
M_CHAR_ADD	Adds a character to the font. (summarize)
M_SYSTEM_FONT +	Specifies that characters from a given system font (for example, TrueType and Postscript) will be added. (summarize)
Param1	Specifies the size of the characters to add. (summarize)
Value > 6	Specifies the size, in points.
Param2	Specifies an optional string containing all of the characters to add. Two characters in one font cannot have the same value. If you try to add a character with a value that already exists in a font, the existing character will be replaced by the newly added one, unless you use M_NO_OVERWRITE . For more information, see the combination constants tables below. (summarize)
M_NULL	Specifies that the standard characters of the font will be added. That is, A to Z, a to z, and 0 to 9. (summarize)
String	Specifies the string. This must be a null-terminated string. The string array must be of the right type for the encoding scheme selected (MstrControl() with M_ENCODING). (summarize)
Param3	Specifies the system font's file name.

	(summarize)
<input type="checkbox"/> System font file name	<p>Specifies the string containing the font's file name (for example, "TrueType" and "Postscript"). This must be a null-terminated string.</p> <p>The string array must be of type <code>MIL_TEXT_CHAR</code>.</p> <p>(summarize)</p>
<input type="checkbox"/> M_USER_DEFINED +	<p>Specifies that user-defined characters will be added from a given image to the specified font.</p> <p>(summarize)</p>
<input type="checkbox"/> Param1	<p>Specifies the identifier of the image containing the characters to add.</p> <p>(summarize)</p>
<input type="checkbox"/> Image identifier	<p>Specifies the image ID. The characters to add in the image must all be approximately the same size. You cannot add a character that is smaller than 6x6 pixels. Note that the size of the characters is automatically determined from the image.</p> <p>The characters in the image are taken from left to right and from top to bottom and are associated to the corresponding characters given in the character list. Each character in the image must be entirely connected (except for the accentuated characters) and should not be merged with other characters or other image objects.</p> <p>It is best to provide a character definition image that is of good quality, and that ideally contains only the characters that you want to add to the font (though this is not mandatory). If all the characters in the string you want to add cannot be found in the image, an error is returned. Even if the operation succeeds, you should draw the characters of the font to ensure that every character is defined as expected, using <code>MstrDraw()</code>.</p> <p>(summarize)</p>
<input type="checkbox"/> Param2	<p>Specifies a string containing all of the characters to add. Unlike <code>M_SYSTEM_FONT</code>, this setting is not optional.</p> <p>(summarize)</p>
<input type="checkbox"/> String	<p>Specifies the string. This must be a null-terminated string.</p> <p>The string array must be of the right type for the encoding scheme selected (<code>MstrControl()</code> with <code>M_ENCODING</code>).</p> <p>Two characters in one font cannot have the same value. If you try to add a character with a value that already exists in a font, the existing character will be replaced by the newly added one, unless you use <code>M_NO_OVERWRITE</code>. For more information, see the combination constants tables below.</p> <p>For a multi-string definition image, the space character (ASCII 32) must be inserted to separate the strings included in this null-terminated string.</p> <p>You should use multi-strings when one (or more) of the following situations is encountered: the characters are not all on the same line, the contrast is clearly different between characters, there is clearly a horizontal space between the characters (for example, ABC DEF).</p> <p>(summarize)</p>
<input type="checkbox"/> Param3	<p>This parameter must be set to <code>M_NULL</code>.</p> <p>(summarize)</p>
<input type="checkbox"/> M_CHAR_BASELINE	<p>Sets the baseline of the characters in the font.</p> <p>(summarize)</p>
<input type="checkbox"/> M_DEFAULT	<p>Implements the default behavior.</p> <p>By default, for system fonts (<code>M_SYSTEM_FONT</code>), the baseline will be taken from that font's file.</p> <p>By default, for user-defined fonts (<code>M_USER_DEFINED</code>), the baseline will be set to <code>M_AUTO_COMPUTE</code>, which automatically computes the baseline to an appropriate value.</p> <p>When using a font to read a string model, the baseline of all characters in the string model are expected to be aligned together within a certain degree of tolerance (set using <code>MstrControl()</code> with <code>M_CHAR_MAX_BASELINE_DEVIATION</code>), otherwise the string will not be read.</p> <p>(summarize)</p>
<input type="checkbox"/> Param1	<p>Specifies the baseline.</p> <p>(summarize)</p>
<input type="checkbox"/> -1000 to 1000	<p>Specifies the baseline value, as a percentage of the character's height.</p> <p>The position of the baseline in a character is defined as a percentage of the height of the character. That is, 0 refers to the bottom of character, 100 refers to the top of the character, and 50 refers to the middle of character. For example, in "ABCD", you would typically set the baseline to 0 for all characters, while in "pobq", you would typically set the baseline to 0 for "ob" and approximately 25 for "pq". Note that you can set a baseline value that falls outside the limits of the character's height (Y-size).</p> <p>(summarize)</p>

<input type="checkbox"/> M_AUTO_COMPUTE	Specifies that the baseline will be automatically computed to an appropriate value.
<input type="checkbox"/> M_NONE	Specifies no baseline.
<input type="checkbox"/> Param2	Specifies an optional string containing all the characters for which you want to set a baseline. The string array must be of the right type for the encoding scheme selected (MstrControl() with M_ENCODING). (summarize)
<input type="checkbox"/> M_NULL	Specifies that the baseline will be set for all the characters in the font.
<input type="checkbox"/> String	Specifies the string. This must be a null-terminated string. (summarize)
<input type="checkbox"/> Param3	This parameter must be set to M_NULL . (summarize)
<input type="checkbox"/> M_CHAR_DELETE	Deletes characters from the font. (summarize)
<input type="checkbox"/> M_DEFAULT	Implements the default behavior. (summarize)
<input type="checkbox"/> Param1	This parameter must be set to M_DEFAULT . (summarize)
<input type="checkbox"/> Param2	Specifies an optional string containing all the characters to delete. The string array must be of the right type for the encoding scheme selected (MstrControl() with M_ENCODING). (summarize)
<input type="checkbox"/> M_NULL	Specifies that all the characters in the font will be deleted.
<input type="checkbox"/> String	Specifies the string. This must be a null-terminated string. (summarize)
<input type="checkbox"/> Param3	This parameter must be set to M_NULL . (summarize)
<input type="checkbox"/> M_CHAR_NORMALIZE	Normalizes the characters of the font. (summarize)
<input type="checkbox"/> M_SIZE_X	Specifies that the characters of the font are normalized by taking the X-size as the reference. That is, all the specified characters will be uniformly scaled to a given X-size. Note that the characters' Y-size is adjusted to maintain the same aspect ratio. (summarize)
<input type="checkbox"/> Param1	Specifies the X-size. (summarize)
<input type="checkbox"/> Value >= 8	Specifies the size, in pixels.
<input type="checkbox"/> Param2	Specifies an optional string containing all the characters to normalize. The string array must be of the right type for the encoding scheme selected (MstrControl() with M_ENCODING). (summarize)
<input type="checkbox"/> M_NULL	Specifies that all the characters in the font will be normalized a given X-size.
<input type="checkbox"/> String	Specifies the string. This must be a null-terminated string. (summarize)
<input type="checkbox"/> Param3	This parameter must be set to M_NULL . (summarize)
<input type="checkbox"/> M_SIZE_Y	Specifies that the characters of the font are normalized by taking the Y-size as the reference. That is, all the specified characters will be uniformly scaled to a given Y-size. Note that the characters' X-size is adjusted to maintain the same aspect ratio. (summarize)
<input type="checkbox"/> Param1	Specifies the Y-size. (summarize)

<input type="checkbox"/> Value >= 8	Specifies the size, in pixels.
<input type="checkbox"/> Param2	Specifies an optional string containing all the characters to normalize. The string array must be of the right type for the encoding scheme selected (MstrControl() with M_ENCODING). (summarize)
<input type="checkbox"/> M_NULL	Specifies that all the characters in the font will be normalized a given Y-size.
<input type="checkbox"/> String	Specifies the string. This must be a null-terminated string. (summarize)
<input type="checkbox"/> Param3	This parameter must be set to M_NULL . (summarize)
<input type="checkbox"/> M_CHAR_SORT	Sorts the characters of the font. (summarize)
<input type="checkbox"/> M_ASCENDING	Specifies that the characters will be sorted, according to their character values, in ascending order. (summarize)
<input type="checkbox"/> Param1	This parameter must be set to M_DEFAULT . (summarize)
<input type="checkbox"/> Param2	Specifies an optional string containing all the characters to sort. The string array must be of the right type for the encoding scheme selected (MstrControl() with M_ENCODING). (summarize)
<input type="checkbox"/> M_NULL	Specifies that all the characters in the font will be sorted in ascending order.
<input type="checkbox"/> String	Specifies the string. This must be a null-terminated string. (summarize)
<input type="checkbox"/> Param3	This parameter must be set to M_NULL . (summarize)
<input type="checkbox"/> M_DESCENDING	Specifies that the characters will be sorted, according to their character values, in descending order. (summarize)
<input type="checkbox"/> Param1	This parameter must be set to M_NULL . (summarize)
<input type="checkbox"/> Param2	Specifies an optional string containing all the characters to sort. The string array must be of the right type for the encoding scheme selected (MstrControl() with M_ENCODING). (summarize)
<input type="checkbox"/> M_NULL	Specifies that all the characters in the font will be sorted in descending order.
<input type="checkbox"/> String	Specifies the string. This must be a null-terminated string. (summarize)
<input type="checkbox"/> Param3	This parameter must be set to M_NULL . (summarize)
<input type="checkbox"/> M_CHAR_TYPE	Sets the type of the characters in the font. (summarize)
<input type="checkbox"/> M_DEFAULT	Implements the default behavior. (summarize)
<input type="checkbox"/> Param1	Specifies the characters' type. (summarize)
<input type="checkbox"/> M_AUTO_COMPUTE	Specifies that the characters' type will be computed automatically. The type will be set to either M_REGULAR or M_PUNCTUATION , based on the character's shape and numerical code. (summarize)
<input type="checkbox"/> M_PUNCTUATION	Specifies punctuation type characters. Punctuation characters can typically be categorized as characters that are neither letters nor numbers, such as the hyphen ('-'). Note that to be considered part of the string, a punctuation character must fall within the range of at least one regular character's Y-size.

	(summarize)
<input type="checkbox"/> M_REGULAR	Specifies regular type characters. Regular characters can typically be categorized as letters and numbers. Note that a string is formed by a linear sequence of regular characters. (summarize)
<input type="checkbox"/> Param2	Specifies an optional string containing all the characters for which to set the type. The string array must be of the right type for the encoding scheme selected (MstrControl() with M_ENCODING). (summarize)
<input type="checkbox"/> M_NULL	Specifies that the type will be set for all the characters in the font.
<input type="checkbox"/> String	Specifies the string. This must be a null-terminated string. (summarize)
<input type="checkbox"/> Param3	This parameter must be set to M_NULL . (summarize)
<input type="checkbox"/> M_THICKEN_CHAR	Thickens the characters of the font. The characters of the font should always be the thickened version, so use this Operation if the font has dotted characters. (summarize)
<input type="checkbox"/> M_DEFAULT	Implements the default behavior. (summarize)
<input type="checkbox"/> Param1	Specifies the number of character thickening iterations. Each iteration enlarges the thickness of the target character of the font. This is useful for some types of fonts (for example, dotted characters). You should choose a value large enough for the intra character dots to connect. (summarize)
<input type="checkbox"/> 0 to 100	Specifies the number of iterations.
<input type="checkbox"/> Param2	Specifies an optional string containing all the characters to normalize. The string array must be of the right type for the encoding scheme selected (MstrControl() with M_ENCODING). (summarize)
<input type="checkbox"/> M_NULL	Specifies to normalize all the characters of the font.
<input type="checkbox"/> String	Specifies the string. This must be a null-terminated string. (summarize)
<input type="checkbox"/> Param3	This parameter must be set to M_NULL . (summarize)

Combination constant for [M_USER_DEFINED](#) (of [M_CHAR_ADD](#));

You can add the following value to the above-mentioned value to specify that a single user-defined character is added to the font.

🔹 For adding a single user-defined character	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SINGLE	<p>Specifies that a single user-defined character will be added from a given image in the specified font.</p> <p>If you use M_USER_DEFINED + M_SINGLE, the whole image is taken as a single character. This allows you to define special characters, which are not necessarily connected. The image is associated to the character specified by the Param2 parameter.</p> <p>It is best to provide a character definition image that is of good quality. Note that if the character you want to add already exists in the font, the new character will replace the old one. You cannot add a character that is smaller than 6x6 pixels. By default, user-defined characters have no baseline.</p> <p>If you use M_USER_DEFINED + M_SINGLE, set the following parameters accordingly:</p> <ul style="list-style-type: none">● Param1: Sets the identifier of the image containing the character to add.

- **Param2**: Sets the pointer of the character to add. The character must be of the right type for the encoding scheme selected ([MstrControl\(\)](#) with [M_ENCODING](#)).
- **Param3**: Must be set to **M_NULL**.

(summarize)

Combination constants for [M_USER_DEFINED](#) (of [M_CHAR_ADD](#));

You can add one of the following values to the above-mentioned value to set the foreground of the characters in the definition image when adding user-defined characters to a font.

Note that when a font is defined, it has no foreground. When performing a read operation ([MstrRead\(\)](#)), the foreground the font is read with is set using [M_FOREGROUND_VALUE](#) in [MstrControl\(\)](#). In this case (when adding user-defined characters to a font), you are setting the foreground for the font in the definition image.

● For setting the foreground of the characters

<div><input type="checkbox"/> Value</div>	Description
<div><input type="checkbox"/> M_FOREGROUND_BLACK</div>	<div>Specifies that black is the foreground color, for the definition image.</div> <div>This is the default value.</div> <div>(summarize)</div>
<div><input type="checkbox"/> M_FOREGROUND_WHITE</div>	<div>Specifies that white is the foreground color, for the definition image.</div>

Combination constant for [M_SYSTEM_FONT](#) (of [M_CHAR_ADD](#)); [M_USER_DEFINED](#) (of [M_CHAR_ADD](#));

You can add the following value to the above-mentioned values to specify that characters will not be added to the font if they already exist.

● For not overwriting characters

<div><input type="checkbox"/> Value</div>	Description
<div><input type="checkbox"/> M_NO_OVERWRITE</div>	<div>Specifies that the characters previously added to the font will not be overwritten.</div> <div>This is the default value.</div> <div>(summarize)</div>

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrExpert

Synopsis

Makes a diagnosis of the String Reader context and reports configuration problems.

Syntax

```
void MstrExpert(
    MIL_ID ContextId,
    MIL_ID *ImageArrayPtr,
    void *TargetStringArrayPtr,
    MIL_INT NbImages,
    MIL_ID ResultId,
    MIL_INT ControlFlag
)
```

Description

This function analyzes the consistency of a String Reader context using a given image and a target string. Retrieve results using [MstrGetResult\(\)](#); this function returns error and/or warning codes for problems found. Use [MstrInquire\(\)](#) to retrieve the meaning associated with these codes.

When the diagnosis of the String Reader context does not cause a general error (see [MstrGetResult\(\)](#) with [M_GENERAL](#)), the context is ready to be used to read the image provided to **MstrExpert()**. You can call **MstrExpert()** multiple times to find and correct problems in the context.

By setting the **ControlFlag** parameter to [M_INTERACTIVE](#), you can start the String Expert analysis and view the results interactively.

Note that this function is only available for a font-based context.

Parameters

ContextId

Specifies the identifier of the String Reader context. The String Reader context must have been previously allocated on the required system using [MstrAlloc\(\)](#).

ImageArrayPtr

Specifies the identifier of the image to analyze. Note that you cannot provide more than one image.

TargetStringArrayPtr

Specifies the identifier of the string to locate. This string represents the expected strings to locate in each image analyzed using [MstrRead\(\)](#). The string array must be of the right type for the encoding scheme selected ([MstrControl\(\)](#) with [M_ENCODING](#)). Only one string is supported.

NbImages

Specifies the number of images to analyze. This parameter must be set to 1.

ResultId

Specifies the identifier of the String Reader result buffer in which to save the results of **MstrExpert()**.

ControlFlag

Specifies the type of analysis to perform with **MstrExpert()**.

To specify the type of analysis to perform, set this parameter to one of the following values. Except for [M_DEFAULT](#), the following values can be added together.

● For specifying the type of analysis to perform	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies to report errors and warnings.
<input type="checkbox"/> M_ERRORS	Specifies to report errors.
<input type="checkbox"/> M_WARNINGS	Specifies to report warnings.

To specify an interactive type of analysis, set this parameter to the following value.

● For specifying an interactive type of analysis	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens an interactive dialog box that allows you to read a string and view the results interactively.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrFree

Synopsis

Free a String Reader context or a String Reader result buffer.

Syntax

```
void MstrFree(
    MIL_ID ObjectId
)
```

Description

This function deletes the specified String Reader context or String Reader result buffer identifier, and releases any memory associated with it.

All String Reader contexts and all String Reader result buffers allocated on a particular system must be freed before the system can be freed.

Parameter

ObjectId
Specifies the identifier of the String Reader context or String Reader result buffer to free. These must have been successfully allocated (with [MstrAlloc\(\)](#) or [MstrAllocResult\(\)](#)) prior to calling this function.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrGetResult

Synopsis

Get the specified type of result(s) from a String Reader result buffer.

Syntax

```
void MstrGetResult(
    MIL_ID ResultId,
    MIL_INT Index,
    MIL_INT ResultType,
    void *ResultArrayPtr
)
```

Description

This function retrieves the result(s) of the specified type from a String Reader result buffer. Results are only available after calling [MstrRead\(\)](#).

When retrieving results, you will need to know the required type and size of the array in which to store the results. Typically, the default data type is *MIL_DOUBLE*. For certain results (such as, [M_TEXT](#), [M_FORMATTED_STRING](#), and [M_STRING](#)), the default type of the required array depends on the selected encoding scheme ([MstrControl\(\)](#) with [M_ENCODING](#)). You can specify any data type by explicitly adding it to the result. The size of the array depends on the result type.

All positional results are relative to the center of the top-left pixel in the target.

By setting the [ResultType](#) parameter to [M_INTERACTIVE](#), you can view the results interactively.

To decrease result retrieval time, especially for remote systems, you can retrieve different types of results from the result buffer all at once. To do so, set [ResultArrayPtr](#) to **M_NULL**. When this parameter is set to **M_NULL**, the results are not retrieved, but the result request is instead entered into a local queue associated with the given result buffer identifier. All the queued requests are processed when you call [MstrGetResult\(\)](#) again and you pass an array to the parameter [ResultArrayPtr](#). You must ensure that the array is sufficiently large to hold the results associated with the current and previously queued requests. Once the results are retrieved, the queue associated with the result buffer is cleared.

Unless otherwise specified, the value settings are applicable to both font based and fontless contexts.

Parameters

ResultId

Specifies the identifier of the String Reader result buffer from which to retrieve results.

Index

Specifies where to get results. This parameter can be set to one of the following:

● For specifying where to get results	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Same as M_ALL .
<input type="checkbox"/> 0 to M_STRING_NUMBER - 1	Specifies the index of the string from which to get results.
<input type="checkbox"/> M_ALL	Specifies that the results of all strings will be retrieved.
<input type="checkbox"/> M_GENERAL	Specifies that the results relating to the entire String Reader context will be retrieved.

ResultType

Specifies the type of result to retrieve or opens an interactive dialog box that displays the results stored in the result buffer, interactively.

To display the results currently stored in the result buffer in an interactive dialog box, select the following.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter `M_NULL`.

● For displaying the results	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	<p>[<i>This is only applicable to Windows</i>]</p> <p>Opens a dialog box containing a spreadsheet that displays the types of results stored in the result buffer. The results are displayed in separate columns. If there are queued requests, only those results are displayed.</p> <p>(summarize)</p>

When retrieving a general result (`M_GENERAL`), the `ResultType` parameter can be set to one of the following values.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a `MIL_DOUBLE`.

● For retrieving general results					
<input type="checkbox"/> Value	Description				
<input type="checkbox"/> M_CHAR_NUMBER +	Returns the total number of characters read (all strings together).				
<input type="checkbox"/> M_CONTEXT_ID +	<p>Returns the identifier of the String Reader context used by MstrRead() to obtain the results in the result buffer. Note that this identifier might not be valid if the String Reader context has been freed using MstrFree().</p> <p>(summarize)</p>				
<input type="checkbox"/> M_STRING_NUMBER +	Returns the total number of strings read.				
<input type="checkbox"/> M_TEXT +	<p>Retrieves the entire text. This includes all string characters, the inserted space characters, the string separators, and the null-terminated character. Strings are ordered in the natural Latin-based reading order.</p> <p>A space character is inserted in the text each time the distance between two adjacent characters exceeds the space width, at the scale of the string. The inserted space character can be set using MstrControl() with M_SPACE_CHARACTER. By default, the inserted space character is a space (ASCII code 32).</p> <p>A string separator is inserted between each string in the text; that is, each time the distance between two adjacent characters is greater than the space size, which depends on the space width (at the scale of the string) and the maximum number of consecutive spaces. The string separator can be set using MstrControl() with M_STRING_SEPARATOR. By default, the string separator is a new line (ASCII code 10).</p> <p>(summarize)</p> <div> <p><i>ResultArrayPtr Info</i></p> <ul style="list-style-type: none"> • Data type: array of type char Array size: This array must be of size M_TEXT_SIZE. Note: The default data type is either <i>char</i> or <i>short</i>, depending on the encoding scheme selected, using MstrControl() with M_ENCODING. • Data type: array of type short Array size: This array must be of size M_TEXT_SIZE. Note: The default data type is either <i>char</i> or <i>short</i>, depending on the encoding scheme selected, using MstrControl() with M_ENCODING. </div>				
<input type="checkbox"/> M_TEXT_ALLOC_SIZE_BYTE +	<p>Returns the number of bytes required to retrieve M_TEXT result(s). This includes all string characters, the inserted space characters, the string separators, the null-terminated character, and takes the encoding bit size into account (MstrControl() with M_ENCODING). Note that the size returned is always the number of bytes, not the number of characters.</p> <p>(summarize)</p>				
<input type="checkbox"/> M_TEXT_SIZE +	<p>Returns the total number of characters in the text. This includes all strings characters, the inserted space characters, and the string separators. Note that this does not include the null-terminated character. The string separator can be controlled using MstrControl() with M_STRING_SEPARATOR. The inserted space characters can be controlled using MstrControl() with M_SPACE_CHARACTER.</p> <p>(summarize)</p>				
<input type="checkbox"/> M_TIMEOUT_END +	<p>Returns whether the timeout limit was reached. You can set the timeout limit using MstrControl() with M_TIMEOUT.</p> <p>(summarize)</p> <div> <p><i>ResultArrayPtr Info</i></p> <p>Return values:</p> <table> <tr> <td><code>M_FALSE</code></td><td>Specifies that the timeout limit was not reached.</td></tr> <tr> <td><code>M_TRUE</code></td><td>Specifies that the timeout limit was reached.</td></tr> </table> </div>	<code>M_FALSE</code>	Specifies that the timeout limit was not reached.	<code>M_TRUE</code>	Specifies that the timeout limit was reached.
<code>M_FALSE</code>	Specifies that the timeout limit was not reached.				
<code>M_TRUE</code>	Specifies that the timeout limit was reached.				

When retrieving individual string results, the [ResultType](#) parameter can be set to one of the following values. Results can be retrieved for a specific string, or for all strings ([M_ALL](#)), unless otherwise specified. Note that strings are ordered in the natural Latin-based reading order.

Unless otherwise specified, the following values require that you pass the [ResultArrayPtr](#) parameter the address of a `MIL_DOUBLE` (when retrieving a specific string) or the address of an array of type `MIL_DOUBLE` with a size equal to the number of strings read. This number can be obtained using [MstrGetResult\(\)](#) with [M_STRING_NUMBER](#) (when retrieving all results).

● For individual strings	
☐ Value	Description
☐ M_CHAR_ANGLE +	Returns the angle of the characters in the string, in degrees. The angle is the same for all the characters in the string; different angles for individual characters within a string cannot be obtained. The characters' angle is not necessarily equal to the string's angle. (summarize)
☐ M_FOREGROUND_VALUE +	Returns the foreground value of the string read.
☐ M_FORMATTED_STRING +	Returns the formatted string. This includes all strings characters, the inserted space characters and the null-terminated character. A space character is inserted in the formatted string each time the distance between two adjacent characters exceeds the space width, at the scale of the string. The inserted space character can be set using MstrControl() with M_SPACE_CHARACTER . By default, the inserted space character is a space (ASCII code 32). (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type char Array size: This array must be of size M_FORMATTED_STRING_SIZE. Note: The default data type is either <i>char</i> or <i>short</i>, depending on the encoding scheme selected, using MstrControl() with M_ENCODING. • Data type: array of type short Array size: This array must be of size M_FORMATTED_STRING_SIZE. Note: The default data type is either <i>char</i> or <i>short</i>, depending on the encoding scheme selected, using MstrControl() with M_ENCODING.
☐ M_FORMATTED_STRING_SIZE +	Returns the number of characters in the formatted string. This includes string characters and the inserted space characters. This does not include the null-terminated character. A space character is inserted in the formatted string each time the distance between 2 adjacent characters exceeds the M_SPACE_WIDTH value, which you can set with MstrControl() . Note that the inserted space characters can be controlled using MstrControl() with M_SPACE_CHARACTER . (summarize)
☐ M_SKEW_ANGLE +	Returns the angle of the characters' skew in the string, in degrees. The angle is the same for all the characters in the string; different angles for individual characters within a string cannot be obtained. (summarize)
☐ M_STRING +	Returns the null-terminated string that was located during the read operation. The string does not contain any space characters or string separators. Refer to M_CHAR_VALUE to retrieve all characters of all strings without the null-terminated character. Refer to M_FORMATTED_STRING or M_TEXT to retrieve all strings with their space characters and/or string separators. (summarize)
	<i>ResultArrayPtr info</i> <ul style="list-style-type: none"> • Data type: array of type char Array size: This array must be of size M_STRING_ALLOC_SIZE_BYTE. Note: The default data type is either <i>char</i> or <i>short</i>, depending on the encoding scheme selected, using MstrControl() with M_ENCODING. • Data type: array of type short Array size: This array must be of size M_STRING_ALLOC_SIZE_BYTE. Note: The default data type is either <i>char</i> or <i>short</i>, depending on the encoding scheme selected, using MstrControl() with M_ENCODING.
☐ M_STRING_ALLOC_SIZE_BYTE +	Returns the number of bytes required to retrieve M_STRING result(s). This includes the null-terminated character, and takes the encoding bit size into account (MstrControl() with M_ENCODING). M_STRING_ALLOC_SIZE_BYTE is typically used when retrieving all results (M_ALL), and will return an array of string size byte (1 element for each string). Note that the size is always the number of bytes, not the number of characters. (summarize)
☐ M_STRING_ANGLE +	Returns the angle of the string, in degrees.
☐ M_STRING_ASPECT_RATIO +	Returns the aspect ratio of the string. The aspect ratio of the string is the median aspect ratio of all the characters in the string. Note that this control type is only available for a font-based context. (summarize)
☐ M_STRING_BOX_BL_X +	Returns the X-position of the bottom-left corner of the bounding box of the string.
☐ M_STRING_BOX_BL_Y +	Returns the Y-position of the bottom-left corner of the bounding box of the string.
☐ M_STRING_BOX_BR_X +	Returns the X-position of the bottom-right corner of the bounding box of the string.

<input type="checkbox"/> M_STRING_BOX_BR_Y +	Returns the Y-position of the bottom-right corner of the bounding box of the string.
<input type="checkbox"/> M_STRING_BOX_UL_X +	Returns the X-position of the top-left corner of the bounding box of the string.
<input type="checkbox"/> M_STRING_BOX_UL_Y +	Returns the Y-position of the top-left corner of the bounding box of the string.
<input type="checkbox"/> M_STRING_BOX_UR_X +	Returns the X-position of the top-right corner of the bounding box of the string.
<input type="checkbox"/> M_STRING_BOX_UR_Y +	Returns the Y-position of the top-right corner of the bounding box of the string.
<input type="checkbox"/> M_STRING_CHAR_SCORE_MAX +	Returns the maximum character score of the string.
<input type="checkbox"/> M_STRING_CHAR_SCORE_MIN +	Returns the minimum character score of the string.
<input type="checkbox"/> M_STRING_MODEL_INDEX +	Returns the index of the string model read.
<input type="checkbox"/> M_STRING_MODEL_USER_LABEL +	Returns the user label of the string model read.
<input type="checkbox"/> M_STRING_POSITION_X +	Returns the X-position of the string. The position of the string is the position of its first character. For more information, see M_CHAR_POSITION_X . (summarize)
<input type="checkbox"/> M_STRING_POSITION_Y +	Returns the Y-position of the string. The position of the string is the position of its first character. For more information, see M_CHAR_POSITION_Y . (summarize)
<input type="checkbox"/> M_STRING_SCALE +	Returns the scale of the string. The scale of the string is the median scale in the X-direction of all the characters in the string. (summarize)
<input type="checkbox"/> M_STRING_SCORE +	Returns the string score calculated during the read operation for the entire string. The string score is the average score of the characters in the string. (summarize)
<input type="checkbox"/> M_STRING_SIZE +	Returns the size of the string. This does not include the null-terminated character. Space characters are not counted in the string size. (summarize)
<input type="checkbox"/> M_STRING_TARGET_SCORE +	Returns the string target score calculated during the read operation for the entire string. The string target score is computed from the unread characters that are in the region of the string in the target image. These otherwise readable characters remain unread due to user-specified constraints, such as scale and grammar restrictions. The more unread characters there are, the lower the target score will be. (summarize)

When retrieving results for individual characters of a string, the [ResultType](#) parameter can be set to one of the following values. Results can be returned for characters of a specific string or for characters of all ([M_ALL](#)) strings. However, an array of results is always returned. Note that characters in the string are ordered by their index position in the string.

Unless otherwise specified, when retrieving a specific string, the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to the individual string. This number can be obtained using [MstrGetResult\(\)](#) with [M_STRING_SIZE](#). When retrieving all strings, the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to all the strings. This number can be obtained using [MstrGetResult\(\)](#) with [M_CHAR_NUMBER](#).

● For individual characters of a string	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CHAR_ASPECT_RATIO +	Returns the aspect ratio of the character. The aspect ratio of the character is the ratio of its scale in the X-direction and in the Y-direction. Note that this control type is only available for a font-based context. (summarize)
<input type="checkbox"/> M_CHAR_BASELINE_DEVIATION +	Returns the baseline deviation of every read character within the string. This represents the deviation of the character's baseline from the string's baseline. For non-punctuation characters, this value is a percentage of the character's height (Y-size). For punctuation characters, this value is a percentage of the height of the tallest character within the font. For more information, refer to MstrControl() with M_CHAR_MAX_BASELINE_DEVIATION and the Character's maximum baseline deviation subsection in the Degrees of freedom section in Chapter 12: String Reader . Note that if the character's baseline is set to M_NONE , its deviation will always be returned as '0'. Note that this control type is only available for a font-based context. (summarize)
<input type="checkbox"/> M_CHAR_BOX_BL_X +	Returns the X-position of the bottom-left corner of the bounding box of each character.
<input type="checkbox"/> M_CHAR_BOX_BL_Y +	Returns the Y-position of the bottom-left corner of the bounding box of each character.
<input type="checkbox"/> M_CHAR_BOX_BR_X +	Returns the X-position of the bottom-right corner of the bounding box of each character.
<input type="checkbox"/> M_CHAR_BOX_BR_Y +	Returns the Y-position of the bottom-right corner of the bounding box of each character.

M_CHAR_BOX_UL_X +	Returns the X-position of the top-left corner of the bounding box of each character.
M_CHAR_BOX_UL_Y +	Returns the Y-position of the top-left corner of the bounding box of each character.
M_CHAR_BOX_UR_X +	Returns the X-position of the top-right corner of the bounding box of each character.
M_CHAR_BOX_UR_Y +	Returns the Y-position of the top-right corner of the bounding box of each character.
M_CHAR_CONSECUTIVE_SPACE +	Returns the number of consecutive spaces that can be inserted between this character and the following one in the string. For more information, refer to M_SPACE_WIDTH and M_SPACE_MAX_CONSECUTIVE in MstrControl() . (summarize)
M_CHAR_FONT +	Returns the index of the font of each individual character within the string(s). The index returned is that of the character's font at the moment the MstrRead() operation was performed. Note that this control type is only available for a font-based context. (summarize)
M_CHAR_FONT_USER_LABEL +	Returns the user label of the font of each individual character within the string(s). Note that this control type is only available for a font-based context. (summarize)
M_CHAR_HOMOGENEITY_SCORE +	Returns the homogeneity score of each individual character within the string(s). The character's homogeneity score quantifies the similarity between the character in the target and the other characters of the string in the target. (summarize)
M_CHAR_INDEX +	Returns the index of the character in the font for each individual character within the string(s). The index returned is that of the index of the character in its font at the moment the MstrRead() operation was performed. Note that this control type is only available for a font-based context. (summarize)
M_CHAR_POSITION_X +	Returns the X-position of the character. The position of a character is the position of its center of gravity. (summarize)
M_CHAR_POSITION_Y +	Returns the Y-position of the character. The position of a character is the position of its center of gravity. (summarize)
M_CHAR_SCALE +	Returns the scale of the character. The scale of the character is its scale in the X-direction. (summarize)
M_CHAR_SCORE +	Returns the score of each individual character within the string(s). The character's score quantifies the similarity between the character in the target and the character in the font; it also quantifies the similarity between the character in the target and the other characters of the string in the target. (summarize)
M_CHAR_SIMILARITY_SCORE +	Returns the similarity score of each individual character within the string(s). The character's similarity score quantifies the similarity between the character in the target and the character in the font. (summarize)
M_CHAR_VALUE +	Returns the value of each character read. (summarize)
	<p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type char Array size: This array must be of size M_STRING_SIZE (for one string) or M_CHAR_NUMBER (for all strings). Note: The default data type is either <i>char</i> or <i>short</i>, depending on the encoding scheme selected, using MstrControl() with M_ENCODING. • Data type: array of type short Array size: This array must be of size M_STRING_SIZE (for one string) or M_CHAR_NUMBER (for all strings). Note: The default data type is either <i>char</i> or <i>short</i>, depending on the encoding scheme selected, using MstrControl() with M_ENCODING.

When retrieving transformation coefficient results for individual characters of a string, the [ResultType](#) parameter can be set to one of the following values.

Transformation coefficients allow you to transform any point from the character's source image (the image from which the characters were defined) to its corresponding position in the target image (forward), or from the read position in the target image to its corresponding position in the character's source image (reverse). This is done using the following equations:

$$x_d = A x_s + B y_s + C.$$

$$y_d = D x_s + E y_s + F.$$

where A , B , C , D , E , and F are the transformation coefficients (forward or reverse); x_s and y_s specify the source coordinates (character image for forward transformation or target image for reverse transformation); and,

x_d and y_d specify the destination coordinates (target image for forward transformation or character image for reverse transformation).

Unless otherwise specified, when retrieving a specific string, the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to the individual string. This number can be obtained using [MstrGetResult\(\)](#) with [M_STRING_SIZE](#). When retrieving all strings, the following values require that you pass the [ResultArrayPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to all the strings. This number can be obtained using [MstrGetResult\(\)](#) with [M_CHAR_NUMBER](#).

● For retrieving transformation coefficient results for individual characters of a string with a fontbased context	
Value	Description
M_A_FORWARD +	Returns the forward transformation coefficient A.
M_A_REVERSE +	Returns the reverse transformation coefficient A.
M_B_FORWARD +	Returns the forward transformation coefficient B.
M_B_REVERSE +	Returns the reverse transformation coefficient B.
M_C_FORWARD +	Returns the forward transformation coefficient C.
M_C_REVERSE +	Returns the reverse transformation coefficient C.
M_D_FORWARD +	Returns the forward transformation coefficient D.
M_D_REVERSE +	Returns the reverse transformation coefficient D.
M_E_FORWARD +	Returns the forward transformation coefficient E.
M_E_REVERSE +	Returns the reverse transformation coefficient E.
M_F_FORWARD +	Returns the forward transformation coefficient F.
M_F_REVERSE +	Returns the reverse transformation coefficient F.

When retrieving errors or warnings reported by the last call to [MstrExpert\(\)](#), the [ResultType](#) parameter can be set to one of the following values.

To retrieve general errors or warnings (relating to the entire String Reader context), the [Index](#) parameter must be set to [M_GENERAL](#). To retrieve errors or warnings for a specific string, or for all strings, the [Index](#) parameter must be set to the string's index, or to [M_ALL](#).

The following values require that you pass the value listed in the data-type area of the specified parameter.

● For retrieving errors or warnings reported by the last call to MstrExpert()	
Value	Description
M_REPORT_ERRORS +	<p>Returns a list of all the errors reported by the last call to MstrExpert(). To inquire the string associated with an error, use MstrInquire() with M_REPORT_STRING. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type <code>MIL_INT</code> Array size: M_REPORT_NUMBER_OF_ERRORS Note: when retrieving general errors, using MstrGetResult() with M_GENERAL • Data type: array of type <code>MIL_INT</code> Array size: M_REPORT_NUMBER_OF_ERRORS Note: when retrieving errors for a specific string index • Data type: array of type <code>MIL_INT</code> Array size: the sum of the elements of the array retrieved by M_REPORT_NUMBER_OF_ERRORS Note: when retrieving errors for all strings
M_REPORT_NUMBER_OF_ERRORS +	<p>Returns the number of errors reported by the last call to MstrExpert(). (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> • Data type: <code>MIL_DOUBLE</code> Note: when retrieving general errors, using MstrGetResult() with M_GENERAL • Data type: <code>MIL_DOUBLE</code> Note: when retrieving errors for a specific string index

	<ul style="list-style-type: none"> Data type: array of type MIL_DOUBLE Array size: MstrInquire() with M_NUMBER_OF_STRING_MODELS Note: when retrieving errors for all strings
<input type="checkbox"/> M_REPORT_NUMBER_OF_WARNINGS +	<p>Returns the number of warnings reported by the last call to MstrExpert(). (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> Data type: MIL_DOUBLE Note: when retrieving warnings, using MstrGetResult() with M_GENERAL Data type: MIL_DOUBLE Note: when retrieving warnings for a specific string index Data type: array of type MIL_DOUBLE Array size: MstrInquire() with M_NUMBER_OF_STRING_MODELS Note: when retrieving warnings for all strings
<input type="checkbox"/> M_REPORT_WARNINGS +	<p>Returns a list of all the warnings reported by the last call to MstrExpert(). To inquire the string associated with a warning, use MstrInquire() with M_REPORT_STRING. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> Data type: array of type MIL_INT Array size: M_REPORT_NUMBER_OF_WARNINGS Note: when retrieving general warnings, using MstrGetResult() with M_GENERAL Data type: array of type MIL_INT Array size: M_REPORT_NUMBER_OF_WARNINGS Note: when retrieving warnings for a specific string index Data type: array of type MIL_INT Array size: the sum of the elements of the array retrieved by M_REPORT_NUMBER_OF_WARNINGS Note: when retrieving warnings for all strings

Combination constant for [the values listed in](#) all tables **except For displaying the results**

You can add the following value to the above-mentioned values to determine whether a result is available and/or supported.

● For determining if results are available					
<input type="checkbox"/> Value	Description				
<input type="checkbox"/> M_AVAILABLE	<p>Returns whether a result is available for a general result, string model, or font. (summarize)</p> <p><i>ResultType info</i></p> <p>Return values:</p> <table> <tr> <td>M_NULL</td><td>The result is not available to be retrieved.</td></tr> <tr> <td>Value != 0</td><td>The result is available to be retrieved.</td></tr> </table>	M_NULL	The result is not available to be retrieved.	Value != 0	The result is available to be retrieved.
M_NULL	The result is not available to be retrieved.				
Value != 0	The result is available to be retrieved.				

Combination constants for [the values listed in](#) all tables **except For displaying the results, For retrieving errors or warnings reported by the last call to MstrExpert()**

You can add one of the following values to the above-mentioned values to cast the requested results to a required data type.

● For casting the result to a required data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_CHAR	<p>Casts the requested results to a <i>char</i>. (summarize)</p> <p><i>ResultArrayPtr info</i></p> <ul style="list-style-type: none"> Data type: char Data type: array of type char Array size: Size depends on the contents of the array.

<div> <div></div> <div>M_TYPE_DOUBLE</div> </div>	<div> <div> Casts the requested results to a <i>double</i>. This is the default value. (summarize) </div> <div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none"> • Data type: double • Data type: array of type double <div>Array size: Size depends on the contents of the array.</div> </div> </div>
<div> <div></div> <div>M_TYPE_FLOAT</div> </div>	<div> <div> Casts the requested results to a <i>float</i>. (summarize) </div> <div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none"> • Data type: float • Data type: array of type float <div>Array size: Size depends on the contents of the array.</div> </div> </div>
<div> <div></div> <div>M_TYPE_LONG</div> </div>	<div> <div> Casts the requested results to a <i>long</i>. (summarize) </div> <div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none"> • Data type: long • Data type: array of type long <div>Array size: Size depends on the contents of the array.</div> </div> </div>
<div> <div></div> <div>M_TYPE_MIL_DOUBLE</div> </div>	<div> <div> Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize) </div> <div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none"> • Data type: MIL_DOUBLE • Data type: array of type MIL_DOUBLE <div>Array size: Size depends on the contents of the array.</div> </div> </div>
<div> <div></div> <div>M_TYPE_MIL_ID</div> </div>	<div> <div> Casts the requested results to a <i>MIL_ID</i>. (summarize) </div> <div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none"> • Data type: MIL_ID • Data type: array of type MIL_ID <div>Array size: Size depends on the contents of the array.</div> </div> </div>
<div> <div></div> <div>M_TYPE_MIL_INT</div> </div>	<div> <div> Casts the requested results to a <i>MIL_INT</i>. (summarize) </div> <div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none"> • Data type: MIL_INT • Data type: array of type MIL_INT <div>Array size: Size depends on the contents of the array.</div> </div> </div>
<div> <div></div> <div>M_TYPE_MIL_INT32</div> </div>	<div> <div> Casts the requested results to a <i>MIL_INT32</i>. (summarize) </div> <div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none"> • Data type: MIL_INT32 • Data type: array of type MIL_INT32 <div>Array size: Size depends on the contents of the array.</div> </div> </div>
<div> <div></div> <div>M_TYPE_MIL_INT64</div> </div>	<div> <div> Casts the requested results to a <i>MIL_INT64</i>. (summarize) </div> <div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none"> • Data type: MIL_INT64 • Data type: array of type MIL_INT64 <div>Array size: Size depends on the contents of the array.</div> </div> </div>
<div> <div></div> <div>M_TYPE_SHORT</div> </div>	<div> <div> Casts the requested results to a <i>short</i>. (summarize) </div> <div> <div>ResultArrayPtr info</div> <ul style="list-style-type: none"> • Data type: short </div> </div>

		<ul style="list-style-type: none">• Data type: array of type short Array size: Size depends on the contents of the array.
<div><div></div>M_TYPE_TEXT_CHAR</div>		Casts the requested results to a <i>MIL_TEXT_CHAR</i> . (summarize)
		<i>ResultArrayPtr info</i> <ul style="list-style-type: none">• Data type: MIL_TEXT_CHAR• Data type: array of type MIL_TEXT_CHAR Array size: Size depends on the contents of the array.

ResultArrayPtr

Accepts the address of one of the following (see above for specifics on which is expected): <ul style="list-style-type: none">• array of type char• array of type double• array of type float• array of type long• array of type MIL_DOUBLE• array of type MIL_ID• array of type MIL_INT• array of type MIL_INT32• array of type MIL_INT64• array of type MIL_TEXT_CHAR• array of type short• char• double• float• long• M_NULL• MIL_DOUBLE• MIL_ID• MIL_INT• MIL_INT32• MIL_INT64• MIL_TEXT_CHAR• short

Specifies the address in which to write results.

Set this parameter to **M_NULL** if you are setting the [ResultType](#) parameter to **M_INTERACTIVE** or if you are putting the result request on the local queue of the result buffer.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrInquire

Synopsis

Inquire information about a specified String Reader context, string model, font, user label, or report generated by the last call to [MstrExpert\(\)](#).

Syntax

```
MIL_INT MstrInquire(
    MIL_ID ContextId,
    MIL_INT Index,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires information about a String Reader context, the string models or fonts contained therein, a user label, or the report generated by the last call to [MstrExpert\(\)](#).

If the inquired setting is set to **M_DEFAULT**, **MstrInquire()** will return **M_DEFAULT**, rather than the actual default value. To inquire the setting's default value, add **M_DEFAULT** to the inquire type.

By setting the [InquireType](#) parameter to **M_INTERACTIVE**, you can view the setting of all the inquire types interactively.

Unless otherwise specified, the inquire type settings are applicable to both font-based and fontless contexts.

Parameters

ContextId

Specifies the String Reader context about which to inquire information. The String Reader context must have been previously allocated on the required system using [MstrAlloc\(\)](#).

Index

Specifies that information will be inquired about the String Reader context, a specific font, a specific string model, a user label, or a report generated by the last call to [MstrExpert\(\)](#). Set this parameter to one of the following values:

● For a String Reader context, specific font, specific string model, user label, or a report generated by the last call to MstrExpert().	
☐ Value	Description
☐ M_DEFAULT	Same as M_CONTEXT .
☐ M_FONT_INDEX(MIL_INT FontIndex)	Specifies the index of the font about which to inquire. Note that this macro is only available for a font-based context. (summarize)
	Parameters
	FontIndex This parameter specifies the index of the font. You can set this parameter to the following:
	Value >= 0 Specifies the index of the individual font about which to inquire. To retrieve the index of a font from its user label, set the Index parameter of MstrInquire() to the user label and set the inquire type to M_FONT_INDEX_FROM_LABEL .
☐ M_STRING_INDEX(MIL_INT StringIndex)	Specifies the index of the string about which to inquire. (summarize)
	Parameters

	<div><div><div>StringIndex</div><div>This parameter specifies the index of the string. You can set this parameter to the following:</div></div><div><div>Value >= 0</div><div>Specifies the index of the individual string about which you want to inquire. To retrieve the index of a string from its user label, set the Index parameter of MstrInquire() to the user label and set the inquire type to M_STRING_INDEX_FROM_LABEL.</div></div></div>
<input type="checkbox"/> Error or warning value	Specifies the error or warning code about which to inquire. Typically, these codes are generated by the last call to MstrExpert() , and retrieved using MstrGetResult() with either M_REPORT_ERRORS or M_REPORT_WARNINGS . (summarize)
<input type="checkbox"/> M_CONTEXT	Specifies to inquire information about a general setting of a String Reader context.
<input type="checkbox"/> User label	Specifies the label of the font or string about which to inquire, assigned using MstrControl() . Set InquireType to either M_FONT_INDEX_FROM_LABEL or M_STRING_INDEX_FROM_LABEL to retrieve the font or string index associated with the user label, respectively. (summarize)

InquireType

Specifies the type of setting about which to inquire.

For a String Reader context (**M_CONTEXT**), or a specific string in the String Reader context (**M_STRING_INDEX()**), the **InquireType** parameter can be set to the following:

Unless otherwise specified, the following values require that you pass the **UserVarPtr** parameter **M_NULL**.

● For opening an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	<div>[This is only applicable to Windows] Opens a read-only dialog box that displays the setting of the inquire types. For M_CONTEXT, opens a read-only dialog box that displays the setting of all the inquire types of the specified String Reader context. For a string model, opens a read-only dialog box that displays the setting of the string model's inquire types. Once the dialog box is opened, you can change between string models using the Index field. For a font, opens a read-only dialog box that displays the setting of the font's inquire types. Once the dialog box is opened, you can change between fonts using the Index field. (summarize)</div>

For a String Reader context or a specific string in the String Reader context the **InquireType** parameter can be set to the following.

Unless otherwise specified, the following values require that you pass the **UserVarPtr** parameter the address of a **MIL_DOUBLE**.

● For a context or an individual string	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_STRING_NUMBER +	<div>Returns either the maximum (total) number of all strings that can be read with a String Reader context (M_CONTEXT), or the maximum number of a specific string (M_STRING_INDEX()) that can be read with a String Reader context. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values: M_ALL; M_DEFAULT; Value; (details)</div></div>

For a fontless context (**M_CONTEXT**), or font-based context (**M_FONT_INDEX()**), the **InquireType** parameter can be set to the following:

Unless otherwise specified, the following values require that you pass the **UserVarPtr** parameter the address of a **MIL_DOUBLE**.

● For a context or a specific font	
<input type="checkbox"/> Value	Description

M_CHAR_VALUE +	Returns an array containing the character value for each character. Unless M_TYPE_LONG or M_TYPE_DOUBLE is explicitly added to M_CHAR_VALUE , the string array must be of the right type for the encoding scheme selected (MstrInquire() with M_ENCODING). (summarize)
M_NUMBER_OF_CHARS +	Returns the number of characters in the font-based or fontless context. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE
M_SPACE_WIDTH +	Returns the width of the space character of the font or fontless context. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE Return values: M_DEFAULT; M_INFINITE; M_MAX_CHAR_WIDTH; M_MEAN_CHAR_WIDTH; M_MIN_CHAR_WIDTH; M_QUARTER_MAX_CHAR_WIDTH; Value > = 1; (details)
M_SPACE_WIDTH_VALUE +	Returns the actual width value of the space character of the font or fontless context, in pixels. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE

For a String Reader context ([M_CONTEXT](#)), the [InquireType](#) parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_DOUBLE.

● For M_CONTEXT	
Value	Description
M_CHAR_STATUS +	Returns whether a character is searched when performing a read operation. Note that this inquire type is only available for a fontless context. (summarize)
	<i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: The size of the array must be large enough to contain the status for every character in the fontless context. This number can be retrieved with M_NUMBER_OF_CHARS . Return values: M_DISABLE The character is not searched when performing a read operation. M_ENABLE The character is searched when performing a read operation.
M_CONTEXT_TYPE +	Returns the type of String Reader context. (summarize)
	<i>UserVarPtr info</i> Return values: M_FONTLESS A fontless context. M_FONT_BASED A font-based context.
M_DRAW_LAST_SIZE_X +	Returns the width of the image buffer needed by the last call to MstrDraw() . This allows you to allocate a buffer of optimal size, for the required drawing operation. Note that when you call MstrDraw() for the first time, you can pass M_NULL as the destination image to get the size needed for the required drawing operation. For more information, see MstrDraw() . (summarize)
M_DRAW_LAST_SIZE_Y +	Returns the height of the image buffer needed by the last call to MstrDraw() . This allows you to allocate a buffer of optimal size, for the required drawing operation. Note that when you call MstrDraw() for the first time, you can pass M_NULL as the destination image to get the size needed for the required drawing operation. For more information, see MstrDraw() .

	(summarize)
<input type="checkbox"/> M_ENCODING +	<p>Returns the type of character encoding used by the String Reader context.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: M_ASCII; M_DEFAULT; M_UNICODE; (details) </div>
<input type="checkbox"/> M_MAX_CHAR_SIZE_X +	<p>Returns the width (X-size) of the widest, enabled character in the fontless context when it has a height of M_REF_CHAR_SIZE_Y. Note that this inquire type is only available for a fontless context.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: 0 to 65536; M_DEFAULT; (details) </div>
<input type="checkbox"/> M_MIN_CHAR_SIZE_X +	<p>Returns the width (X-size) of the narrowest, enabled character in the fontless context when it has a height of M_REF_CHAR_SIZE_Y. Note that this inquire type is only available for a fontless context.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: 0 to 65536; M_DEFAULT; (details) </div>
<input type="checkbox"/> M_MINIMUM_CONTRAST +	<p>Returns the minimum contrast between a character in the target image and its background, in order for the character to be read by MstrRead().</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: 1 to 255; M_DEFAULT; (details) </div>
<input type="checkbox"/> M_MODIFICATION_COUNT +	<p>Returns the current value of the modification counter. The modification counter is increased by one each time settings for the context are modified. Although you cannot identify the modification counter's contents, you can compare them throughout your application to know if the context has been altered. If the modification counter has changed you can, for example, prompt the user to save before closing the application.</p> <p>(summarize)</p>
<input type="checkbox"/> M_NUMBER_OF_FONTS +	<p>Returns the number of fonts in the context.</p> <p>Note that this inquire type is only available for a font-based context.</p> <p>(summarize)</p>
<input type="checkbox"/> M_NUMBER_OF_STRING_MODELS +	Returns the number of string models in the context.
<input type="checkbox"/> M_OWNER_SYSTEM +	Returns the identifier of the system on which the context was allocated.
<input type="checkbox"/> M_PREPROCESSED +	<p>Returns whether the String Reader context is preprocessed. This will indicate whether preprocessing (using MstrPreprocess()) is needed to prepare the context for use.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: Value != 0 The context is preprocessed; you should not preprocess again. 0 The context is not preprocessed; you should preprocess again. </div>
<input type="checkbox"/> M_REF_CHAR_SIZE_Y +	<p>Returns the reference height.</p> <p>Note that this inquire type is only available for a fontless context.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: 0 to 65536; M_DEFAULT; (details) </div>
<input type="checkbox"/> M_REF_CHAR_THICKNESS +	<p>Returns the stroke width of characters in the fontless context when they have a height of M_REF_CHAR_SIZE_Y. Note that this inquire type is only available for a fontless context.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Return values: 0 to 65536; M_DEFAULT; (details) </div>
<input type="checkbox"/> M_SEARCH_CHAR_ANGLE +	<p>Returns whether strategies specific to angular-range will be calculated.</p> <p>(summarize)</p>

	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SEARCH_SKEW_ANGLE +	Returns whether strategies specific to angular-range skew will be calculated. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SEARCH_STRING_ANGLE +	Returns whether to search for the string angle. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; M_ENABLE; (details)
<input type="checkbox"/> M_SPACE_CHARACTER +	Returns the character value to use as a space character within the formatted text that will be read. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_NONE; Value; (details)
<input type="checkbox"/> M_SPEED +	Returns the search/read speed. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_HIGH; M_MEDIUM; (details)
<input type="checkbox"/> M_STRING_SEPARATOR +	Returns the character value to use as a string separator within the formatted text that will be read. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_NONE; Value; (details)
<input type="checkbox"/> M_THICKEN_CHAR +	Returns the number of character thickening iterations. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 100; M_DEFAULT; (details)
<input type="checkbox"/> M_THRESHOLD_MODE +	Returns the threshold method of the character blob extraction. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_LOCAL; M_LOCAL_WITH_RESEGMENTATION; M_USER_DEFINED; (details)
<input type="checkbox"/> M_THRESHOLD_VALUE +	Returns the global threshold value when M_THRESHOLD_MODE is set to M_USER_DEFINED (otherwise it is ignored). (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 255; M_AUTO_COMPUTE; M_DEFAULT; (details)
<input type="checkbox"/> M_TIMEOUT +	Returns the maximum read time for MstrRead() , in msec. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_DISABLE; Value > = 0; (details)

For a specific string ([M_STRING_INDEX\(\)](#)) in the String Reader context, the [InquireType](#) parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

● For an individual string	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CHAR_ACCEPTANCE +	Returns the acceptance level for the character score. (summarize)

	<i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)
<input type="checkbox"/> M_CHAR_ASPECT_RATIO_MAX_FACTOR +	Returns the factor used to determine the upper limit of the aspect ratio for the characters in the string. Note that this inquire type is only available for a font-based context. (summarize)
	<i>UserVarPtr info</i> Return values: 1.0 to 2.0; M_DEFAULT; (details)
<input type="checkbox"/> M_CHAR_ASPECT_RATIO_MIN_FACTOR +	Returns the factor used to determine the lower limit of the aspect ratio for the characters in the string. Note that this inquire type is only available for a font-based context. (summarize)
	<i>UserVarPtr info</i> Return values: 0.5 to 1.0; M_DEFAULT; (details)
<input type="checkbox"/> M_CHAR_HOMOGENEITY_ACCEPTANCE +	Returns the acceptance level for the character's homogeneity score. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)
<input type="checkbox"/> M_CHAR_MAX_BASELINE_DEVIATION +	Returns the maximum deviation that a character can have from the string's baseline. (summarize)
	<i>UserVarPtr info</i> Return values: 0 to 100; M_DEFAULT; (details)
<input type="checkbox"/> M_CHAR_SCALE_MAX_FACTOR +	Returns the factor used to determine the upper limit (maximum permitted scale) of the scale range for the characters in the string. (summarize)
	<i>UserVarPtr info</i> Return values: 1.0 to 2.0; M_DEFAULT; (details)
<input type="checkbox"/> M_CHAR_SCALE_MIN_FACTOR +	Returns the factor used to determine the lower limit (minimum permitted scale) of the scale range for the characters in the string. (summarize)
	<i>UserVarPtr info</i> Return values: 0.5 to 1.0; M_DEFAULT; (details)
<input type="checkbox"/> M_CHAR_SIMILARITY_ACCEPTANCE +	Returns the acceptance level for the character's similarity score. (summarize)
	<i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)
<input type="checkbox"/> M_DEFAULT_CONSTRAINT_FONT +	Returns the default valid font index. This information can also be inquired using MstrInquire() with M_DEFAULT_CONSTRAINT_TYPE ; however, it is encoded with other constant values (the one that was applied by default to all characters of the string model when no explicit constraint has been set to a specific position or if M_DEFAULT has been set to a specific position). M_DEFAULT_CONSTRAINT_FONT always returns a single value. Note that this inquire type is only available for a font-based context. (summarize)
<input type="checkbox"/> M_DEFAULT_CONSTRAINT_TYPE +	Returns the default constraint type. Returns the default constraint type (the one that was applied by default to all characters of the string model when no explicit constraint has been set to a specific position or if M_DEFAULT has been set to a specific position). M_DEFAULT_CONSTRAINT_TYPE always returns a single value. (summarize)
<input type="checkbox"/> M_FOREGROUND_VALUE +	Returns the foreground color of the string. (summarize)
	<i>UserVarPtr info</i> Return values: M_DEFAULT; M_FOREGROUND_BLACK; M_FOREGROUND_BLACK_OR_WHITE; M_FOREGROUND_WHITE; (details)
<input type="checkbox"/> M_NUMBER_OF_CONSTRAINTS +	Returns the number of constraints (set in MstrControl()) that have been set to a value other than M_DEFAULT (non-default constraints).
<input type="checkbox"/> M_SPACE_MAX_CONSECUTIVE +	Returns the maximum number of consecutive space characters allowed in the string.

	(summarize)
	<div> <div>UserVarPtr info</div> <div>Return values: M_DEFAULT; Value > = 0; (details)</div> </div>
<input type="checkbox"/> M_STRING_ACCEPTANCE +	<div> <div>Returns the acceptance level for the string score.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 100.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_STRING_ANGLE +	<div> <div>Returns the nominal angle of the string.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_STRING_ANGLE_DELTA_NEG +	<div> <div>Returns the lower limit of the string's angular range, relative to the nominal angle (MstrControl() with M_STRING_ANGLE).</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 10.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_STRING_ANGLE_DELTA_POS +	<div> <div>Returns the upper limit of the string's angular range, relative to the nominal angle (MstrControl() with M_STRING_ANGLE).</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 10.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_STRING_ASPECT_RATIO +	<div> <div>Returns the nominal aspect ratio of the string.</div> <div>Note that this inquire type is only available for a font-based context.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 0.5 to 2.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_STRING_ASPECT_RATIO_MAX_FACTOR +	<div> <div>Returns the factor used to determine the upper limit of the string's aspect ratio. This value is relative to the nominal aspect ratio (MstrControl() with M_STRING_ASPECT_RATIO).</div> <div>Note that this inquire type is only available for a font-based context.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 1.0 to 2.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_STRING_ASPECT_RATIO_MIN_FACTOR +	<div> <div>Returns the factor used to determine the lower limit of the string's aspect ratio. This value is relative to the nominal aspect ratio (MstrControl() with M_STRING_ASPECT_RATIO).</div> <div>Note that this inquire type is only available for a font-based context.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 0.5 to 1.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_STRING_CERTAINTY +	<div> <div>Returns the certainty level for the string score.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 0.0 to 100.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_STRING_SCALE +	<div> <div>Returns the nominal scale of the string.</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> <div>Return values: 0.25 to 4.0; M_DEFAULT; (details)</div> </div>
<input type="checkbox"/> M_STRING_SCALE_MAX_FACTOR +	<div> <div>Returns the factor used to determine the upper limit (maximum permitted scale) of the string's scale range. This value is relative to the nominal scale (MstrControl() with M_STRING_SCALE).</div> <div>(summarize)</div> </div>
	<div> <div>UserVarPtr info</div> </div>

	Return values: 1.0 to 2.0; M_DEFAULT; (details)
<input type="checkbox"/> M_STRING_SCALE_MIN_FACTOR +	<p>Returns the factor used to determine the lower limit (minimum permitted scale) of the string's scale range. This value is relative to the nominal scale (MstrControl() with M_STRING_SCALE). (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.5 to 1.0; M_DEFAULT; (details)</p>
<input type="checkbox"/> M_STRING_SIZE_MAX +	<p>Returns the maximum string size (number of characters) of the string model. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_INFINITE; Value > = 1; (details)</p>
<input type="checkbox"/> M_STRING_SIZE_MIN +	<p>Returns the minimum string size (number of characters) of the string model. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; Value > = 1; (details)</p>
<input type="checkbox"/> M_STRING_TARGET_ACCEPTANCE +	<p>Returns the acceptance level for the string target score. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>
<input type="checkbox"/> M_STRING_TARGET_CERTAINTY +	<p>Returns the certainty level for the string target score. (summarize)</p> <p><i>UserVarPtr info</i> Return values: 0.0 to 100.0; M_DEFAULT; (details)</p>
<input type="checkbox"/> M_STRING_TYPE +	<p>Returns the type of the string model added. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_USER_DEFINED; (details)</p>
<input type="checkbox"/> M_STRING_USER_LABEL +	<p>Returns the string model's user-defined label. (summarize)</p> <p><i>UserVarPtr info</i> Return values: M_DEFAULT; M_NO_LABEL; Value; (details)</p>

For a specific string ([M_STRING_INDEX\(\)](#)) in the String Reader context, the [InquireType](#) parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of an array of type `MIL_DOUBLE` with a size equal to [M_NUMBER_OF_CONSTRAINTS](#) results.

● For an individual string when specifying constraints	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_CONSTRAINT +	<p>Returns the constraint string at the specified index.</p> <p>Unless M_TYPE_LONG or M_TYPE_DOUBLE is explicitly added to M_CONSTRAINT, the string array type corresponds to the encoding scheme selected with M_ENCODING. The string returned is null-terminated.</p> <p>You must specify a combination value from the table below. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: array of type char Array size: M_CONSTRAINT_ALLOC_SIZE characters. Note: If M_ENCODING is set to M_ASCII. • Data type: array of type short Array size: M_CONSTRAINT_ALLOC_SIZE characters. Note: If M_ENCODING is set to M_UNICODE

<input type="checkbox"/> M_CONSTRAINT_ALLOC_SIZE +	Returns an array containing the allocation size needed for each non-default constraint string. The allocation size of the constraint string includes the terminating 0 of the string. (summarize)
<input type="checkbox"/> M_CONSTRAINT_FONT +	Returns an array containing the valid font index of each non-default constraint in the string model. This information can also be found using MstrInquire() with M_CONSTRAINT_TYPE , but it is encoded with other values. You can inquire the position in the string model that this constraint is applied to using MstrInquire() with M_CONSTRAINT_POSITION . Note that this inquire type is only available for a font-based context. (summarize)
<input type="checkbox"/> M_CONSTRAINT_POSITION +	Returns an array containing the position of each non-default constraint in the string model (0, 255). The default constraint is therefore applied to all positions not returned in this array. (summarize)
<input type="checkbox"/> M_CONSTRAINT_TYPE +	Returns an array containing the type of each non-default constraint in the string model (for example, M_LETTER , M_DIGIT). You can inquire the position in the string model that this constraint is applied to using MstrInquire() with M_CONSTRAINT_POSITION . (summarize)
<input type="checkbox"/> M_DEFAULT_CONSTRAINT +	<p>Returns the default constraint string (the one that was applied by default to all characters of the string model when no explicit constraint has been set to a specific position or if M_DEFAULT has been set to a specific position).</p> <p>The string array must be large enough to contain M_DEFAULT_CONSTRAINT_ALLOC_SIZE characters.</p> <p>Unless M_TYPE_LONG or M_TYPE_DOUBLE is explicitly added to M_DEFAULT_CONSTRAINT, the string array type corresponds to the encoding scheme selected with M_ENCODING. The string returned is null-terminated.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: array of type char Array size: M_CONSTRAINT_ALLOC_SIZE characters. Note: If M_ENCODING is set to M_ASCII • Data type: array of type short Array size: M_CONSTRAINT_ALLOC_SIZE characters. Note: If M_ENCODING is set to M_UNICODE </div>
<input type="checkbox"/> M_DEFAULT_CONSTRAINT_ALLOC_SIZE +	<p>Returns the allocation size needed to get the default constraint string. The allocation size includes the null terminator of the default string.</p> <p>M_DEFAULT_CONSTRAINT_ALLOC_SIZE always returns a single value.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Data type: MIL_DOUBLE </div>

Combination constant for **M_CONSTRAINT**;

You must add the following value to the above-mentioned value to get the constraint string at the specified index.

● For M_CONSTRAINT to specify the index	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> 0 to (M_NUMBER_OF_CONSTRAINTS - 1)	<p>Returns the constraint string at the specified index.</p> <p>The index you provide is the index of the constraint in the list of non-default constraints, and not its actual position in the string model. You can inquire the position in the string model that this constraint string applies to by using MstrInquire() with M_CONSTRAINT_POSITION.</p> <p>(summarize)</p> <div> <i>UserVarPtr info</i> Data type: array of type MIL_DOUBLE Array size: M_CONSTRAINT_ALLOC_SIZE characters. </div>

For a specific font (**M_FONT_INDEX()**) in the String Reader context, the **InquireType** parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the *UserVarPtr* parameter the address of an array of type MIL_DOUBLE with a size equal to **M_NUMBER_OF_CHARS** results .

● For an individual font	
<input type="checkbox"/> Value	Description

<input type="checkbox"/> M_CHAR_BASELINE +	Returns an array containing the baseline value for each character. For more information, see MstrEditFont() . (summarize)
<input type="checkbox"/> M_CHAR_SIZE_X +	Returns an array containing the X-size of each character in the font, in pixels.
<input type="checkbox"/> M_CHAR_SIZE_Y +	Returns an array containing the Y-size of each character in the font, in pixels.
<input type="checkbox"/> M_CHAR_THICKNESS +	Returns an array containing the average thickness of each character in the font, in pixels.
<input type="checkbox"/> M_CHAR_TYPE +	Returns an array containing the type of each character in the font. The type of a character can be M_AUTO_COMPUTE , M_REGULAR , or M_PUNCTUATION . For more information, see M_CHAR_TYPE in MstrEditFont() . (summarize)
<input type="checkbox"/> M_CHAR_TYPE_VALUE +	Returns an array containing the type value of each character in the font. When the value of M_CHAR_TYPE is M_AUTO_COMPUTE , M_CHAR_TYPE_VALUE contains the auto-computed value (M_REGULAR or M_PUNCTUATION). For more information, see M_CHAR_TYPE in MstrEditFont() . (summarize)
<input type="checkbox"/> M_DEFINITION_OFFSET_X +	Returns an array containing the X-offset definition of each character in the font, in pixels. The X-offset definition is the offset of the character in the character definition image for user defined characters (MstrControl() with M_USER_DEFINED). It is M_NONE for a system font (MstrEditFont() with M_SYSTEM_FONT). (summarize)
<input type="checkbox"/> M_DEFINITION_OFFSET_Y +	Returns an array containing the Y-offset definition of each character in the font, in pixels. The Y-offset definition is the offset of the character in the character definition image for user defined characters (MstrControl() with M_USER_DEFINED). It is M_NONE for a system font (MstrEditFont() with M_SYSTEM_FONT). (summarize)
<input type="checkbox"/> M_DRAW_BOX_MARGIN_X +	Returns the margin in the X-direction, between the character box and the drawing box. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE Return values: M_DEFAULT; Value > = 0; (details)
<input type="checkbox"/> M_DRAW_BOX_MARGIN_Y +	Returns the margin in the Y-direction, between the character box and the drawing box. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE Return values: M_DEFAULT; Value > = 0; (details)
<input type="checkbox"/> M_FONT_TYPE +	Returns the type of font added. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE Return values: M_DEFAULT; M_USER_DEFINED; (details)
<input type="checkbox"/> M_FONT_USER_LABEL +	Returns the font's user-defined label. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE Return values: M_DEFAULT; M_NO_LABEL; Value; (details)

Combination constants for [the values listed in](#) all tables **except** For opening an interactive dialog box, For a report generated by the last call to [MstrExpert\(\)](#), For user label inquiries

You can add one of the following values to the above-mentioned values to cast the requested results to the required data type.

● For specifying the data type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_TYPE_CHAR	Casts the requested results to a <i>char</i> . (summarize)
	<i>UserVarPtr info</i>

	<ul style="list-style-type: none"> • Data type: char • Data type: array of type char <p>Array size: Size depends on the contents of the array.</p>
 M_TYPE_DOUBLE	<p>Casts the requested results to a <i>double</i>. This is the default value. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: double • Data type: array of type double <p>Array size: Size depends on the contents of the array.</p>
 M_TYPE_FLOAT	<p>Casts the requested results to a <i>float</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: float • Data type: array of type float <p>Array size: Size depends on the contents of the array.</p>
 M_TYPE_LONG	<p>Casts the requested results to a <i>long</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: long • Data type: array of type long <p>Array size: Size depends on the contents of the array.</p>
 M_TYPE_MIL_DOUBLE	<p>Casts the requested results to a <i>MIL_DOUBLE</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: MIL_DOUBLE • Data type: array of type MIL_DOUBLE <p>Array size: Size depends on the contents of the array.</p>
 M_TYPE_MIL_ID	<p>Casts the requested results to a <i>MIL_ID</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: MIL_ID • Data type: array of type MIL_ID <p>Array size: Size depends on the contents of the array.</p>
 M_TYPE_MIL_INT	<p>Casts the requested results to a <i>MIL_INT</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: MIL_INT • Data type: array of type MIL_INT <p>Array size: Size depends on the contents of the array.</p>
 M_TYPE_MIL_INT32	<p>Casts the requested results to a <i>MIL_INT32</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: MIL_INT32 • Data type: array of type MIL_INT32 <p>Array size: Size depends on the contents of the array.</p>
 M_TYPE_MIL_INT64	<p>Casts the requested results to a <i>MIL_INT64</i>. (summarize)</p> <p><i>UserVarPtr info</i></p> <ul style="list-style-type: none"> • Data type: MIL_INT64 • Data type: array of type MIL_INT64 <p>Array size: Size depends on the contents of the array.</p>
 M_TYPE_SHORT	<p>Casts the requested results to a <i>short</i>.</p>

	(summarize)
	<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: short • Data type: array of type short • Array size: Size depends on the contents of the array.
<input type="checkbox"/> M_TYPE_TEXT_CHAR	Casts the requested results to a <i>MIL_TEXT_CHAR</i> . (summarize)
	<i>UserVarPtr info</i> <ul style="list-style-type: none"> • Data type: MIL_TEXT_CHAR • Data type: array of type MIL_TEXT_CHAR • Array size: Size depends on the contents of the array.

Combination constant for [the values listed in](#) all tables **except** **For opening an interactive dialog box**, **For a report generated by the last call to MstrExpert()**, **For user label inquires**

You can add the following value to the above-mentioned values to determine the default value of an inquire type.

● For the default value of an inquire type	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Returns the actual default value of the specified inquire type. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE

For a report generated by the last call to [MstrExpert\(\)](#), the [InquireType](#) parameter can be set to the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of an array of type *MIL_TEXT_CHAR* with a size equal to *M_MAX_REPORT_STRING_LENGTH*.

● For a report generated by the last call to MstrExpert()	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_REPORT_STRING +	Returns the string associated with an error or warning report.

Combination constant for [the values listed in](#) all tables **except** **For opening an interactive dialog box**, **For user label inquires**

You can add the following value to the above-mentioned values to determine whether an inquire type is supported.

See below.

● For inquiring what is supported	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_SUPPORTED	Returns whether the specified inquire type is supported for the String Reader context. (summarize)
	<i>UserVarPtr info</i> Data type: MIL_DOUBLE Return values: M_NULL Inquire type is not supported. Value != 0 Inquire type is supported.

For user label inquiries, the [InquireType](#) parameter can be set to one of the following:

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a `MIL_DOUBLE`.

For user label inquiries	
Value	Description
M_FONT_INDEX_FROM_LABEL	Returns the font index associated with a font user label if the user label is used. Note that this inquire type is only available for a font-based context. (summarize)
	UserVarPtr info
	Return values: M_INVALID Invalid user label. This is returned if the specified font label is not associated with a font. Value Font index associated with the specified user label.
M_STRING_INDEX_FROM_LABEL	Returns the string model index associated with a string model user label if the user label is used. (summarize)
	UserVarPtr info
	Return values: M_INVALID Invalid user label. This is returned if the specified string model label is not associated with a string model. Value String model index associated with the specified user label.

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type char
- array of type double
- array of type float
- array of type long
- array of type MIL_DOUBLE
- array of type MIL_ID
- array of type MIL_INT
- array of type MIL_INT32
- array of type MIL_INT64
- array of type MIL_TEXT_CHAR
- array of type short
- char
- double
- float
- long
- M_NULL
- MIL_DOUBLE
- MIL_ID
- MIL_INT
- MIL_INT32
- MIL_INT64
- MIL_TEXT_CHAR
- short

Specifies the address in which to write the requested information. Since the `MstrInquire()` function also returns the requested information, you can set this parameter to `M_NULL` (for non-array results only).

Return value

The return value is the required information, cast to a `MIL_INT`.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrPreprocess

Synopsis

Preprocess a String Reader context.

Syntax

```
void MstrPreprocess(
    MIL_ID ContextId,
    MIL_INT ControlFlag
)
```

Description

This function prepares the specified String Reader context, font, and string model to be read. Essentially, **MstrPreprocess()** makes the context ready for the read operation.

This function must be called before the first call to [MstrRead\(\)](#). Changes to control settings, the String Reader context, one of its font or string models often require you to re-preprocess the context. To inquire if you need to preprocess or re-preprocess the context, use [MstrInquire\(\)](#) with [M_PREPROCESSED](#).

Note that in a typical application, **MstrPreprocess()** is called once. Also note that, to preprocess a String Reader context without an error, the context must have at least one font, one character, and one string.

When you save the String Reader context, the preprocessing changes are not saved. Upon restoration, you must preprocess the context again.

Parameters

- ContextId
- Specifies the String Reader context to preprocess. The String Reader context must have been previously allocated on the system using [MstrAlloc\(\)](#).
- ControlFlag
- Specifies whether to preprocess the String Reader context. Set this parameter to one of the following values:

● For specifying whether to preprocess the context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Preprocesses the String Reader context.
<input type="checkbox"/> M_RESET	Un-preprocesses the String Reader context. Un-preprocessing the String Reader context can be useful if you want to conserve system memory within an application and preserve String Reader context settings. (summarize)

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrRead

Synopsis

Read strings from a target image.

Syntax

```
void MstrRead(
    MIL_ID ContextId,
    MIL_ID TargetImageId,
    MIL_ID ResultId
)
```

Description

This function reads one or more strings from the specified target image using the specified String Reader context. All specifications, such as the number of strings to read in the target image, the speed and/or robustness parameters, the constraints on the characters in the string, and the font used, depend on the settings of the String Reader context, which can be set with [MstrControl\(\)](#), [MstrSetConstraint\(\)](#) and [MstrEditFont\(\)](#). All settings are taken into account by **MstrRead()**, and results are stored in the specified result buffer. Results can be read from the result buffer using the [MstrGetResult\(\)](#) function.

For a string to be read, it must not only meet all of the restrictions placed on it by the String Reader context, but it must also obey the basic rules of a string. That is, it must be a linear sequence of regular characters.

The string read will be ordered in the natural Latin-based reading order. The position of a string is the position of the center of the box of its first character.

The target image must be 1-band 8-bit unsigned, and cannot exceed a maximum size of 65536 x 65536 pixels.

Note that before performing a read operation, the String Reader context must have at least one font, one character, and one string; it must also be preprocessed ([MstrPreprocess\(\)](#)).

Parameters

ContextId

Specifies the String Reader context to use for the read operation. The String Reader context must have been previously allocated on the required system using [MstrAlloc\(\)](#), and preprocessed using [MstrPreprocess\(\)](#).

TargetImageId

Specifies the target image in which to read the strings.

ResultId

Specifies the result buffer in which to write the results of the read operation.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrRestore

Synopsis

Restore a String Reader context from disk.

Syntax

```
MIL_ID MstrRestore(
    MIL_CONST_TEXT_PTR Filename,
    MIL_ID SystemId,
    MIL_INT ControlFlag,
    MIL_ID *ContextIdPtr
)
```

Description

This function restores a String Reader context that was previously saved to a file, using [MstrSave\(\)](#) or [MstrStream\(\)](#). This function restores all of the String Reader context's settings (including font and string model settings) that were in effect when the String Reader context was saved. A restored String Reader context is not preprocessed; therefore, you must call [MstrPreprocess\(\)](#) before performing a read with [MstrRead\(\)](#).

To use a fontless context, you can to restore one of the predefined context files located under the "\Matrox Imaging\contexts\" MIL installation folder. The String Reader module comes with three predefined fontless contexts:

- "FONTLESS_ANPR.msr". A generic context useful to read licence plates.
- "FONTLESS_EUROPEAN_ANPR.msr". A context useful to read European licence plates.
- "FONTLESS_MACHINE_PRINT.msr". A special context that reads machine printed characters in Arial, Ocr-B, or other sans-serif fonts.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

Note that restoring a fontless context is not supported.

Parameters

Filename

Specifies the name and path of the file from which to restore the String Reader context. The function handles (internally) the opening and closing of the file.

This parameter can be set to one of the following:

For specifying the file name and path	
Value	Description
<div>MIL_TEXT(<div>MIL_TEXT_PTR FileName</div>)</div>	<div>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile". To open the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile". (summarize)</div> <div><div>Parameters</div><div>FileName</div><div>Specifies the drive, directory, and name of the file.</div></div>
M_INTERACTIVE	<div>[This is only applicable to Windows]</div> <div>Opens the File Open dialog box from which you can interactively specify the drive, directory, and name of the file.</div>

SystemId

Specifies the system on which to restore the String Reader context.

This parameter should be set to one of the following values:

● For specifying the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ContextIdPtr

Specifies the address of the variable in which to write the String Reader context identifier. Since this function also returns the String Reader context identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the String Reader context identifier. If allocation fails, **M_NULL** is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrSave

Synopsis

Save a String Reader context to a file.

Syntax

```
void MstrSave(
    MIL_CONST_TEXT_PTR FileName,
    MIL_ID ContextId,
    MIL_INT ControlFlag
)
```

Description

This function saves all the information about the previously allocated String Reader context to disk, including all of the individual font and string settings. This information can be reloaded, using [MstrRestore\(\)](#) or [MstrStream\(\)](#).

When you save the String Reader context, the preprocessing changes are not saved. Upon restoration, you must preprocess the context again.

Parameters

FileName

Specifies the name and path of the file in which to save the String Reader context. It is recommended that you use the MSR file extension for easier use with other Matrox Imaging software products. The function internally handles the opening and closing of this file. If this file already exists, it will be overwritten.

This parameter can be set to one of the following values:

For specifying the file name and path					
Value	Description				
<input type="checkbox"/> MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile".</p> <p>To save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote://", for example: "remote:///C:\mydirectory\myfile".</p> <p>(summarize)</p> <table><tr><th colspan="2">Parameters</th></tr><tr><td>FileName</td><td>Specifies the drive, directory, and name of the file.</td></tr></table>	Parameters		FileName	Specifies the drive, directory, and name of the file.
Parameters					
FileName	Specifies the drive, directory, and name of the file.				
<input type="checkbox"/> M_INTERACTIVE	<p>[This is only applicable to Windows]</p> <p>Opens the File Save As dialog box from which you can interactively specify the drive, directory, and name of the file.</p>				

ContextId

Specifies the identifier of the String Reader context to save.

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.

DLL	Requires mil.dll; milstr.dll.
-----	-------------------------------

MstrSetConstraint

Synopsis

Set character constraints.

Syntax

```
void MstrSetConstraint(
    MIL_ID ContextId,
    MIL_INT StringIndex,
    MIL_INT CharPos,
    MIL_INT ConstraintType,
    const void *CharList
)
```

Description

This function specifies the constraint to apply to a specified character of a specified string or of all strings.

By setting the **ConstraintType** parameter to [M_INTERACTIVE](#), you can set the constraints interactively. Each constraint must include at least one character existing in one of the selected fonts. For example, if the constraint is [M_DIGIT](#) then at least one of the fonts must contain a digit.

Parameters

ContextId

Specifies the String Reader context with which to associate the constraint. The String Reader context must have been previously allocated on the required system using [MstrAlloc\(\)](#).

StringIndex

Specifies the string(s) in the String Reader context to affect.

Note that a new string can be added to the String Reader context using [MstrControl\(\)](#) with [M_STRING_ADD](#).

This parameter should be set to the following value:

For specifying the string index			
Value		Description	
<div><input type="checkbox"/> <code>M_STRING_INDEX(</code> MIL_INT <i>StringIndex</i>)</div>		Specifies that the constraint will be applied to a string or to all strings. (summarize)	
		Parameters	
		StringIndex	
		This parameter specifies the index of the string. You can set this parameter to one of the following:	
		Value >= 0	Specifies the index of the individual string to which the constraint will be applied.
		M_ALL	Specifies that the constraint will be applied to all strings.

CharPos

Specifies the character position in the string for which to set the constraint.

A constraint can be set for any position, even a position greater than the maximum number of characters in the string. Constraints set at a position greater than the minimum number of characters in the string are ignored (when the

string read does not contain a character at that position).

This parameter should be set to one of the following values:

● For specifying the character position	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Sets the default constraint of the string model. The default constraint of the string model is used when no specific constraint is set to a given position. (summarize)
<input type="checkbox"/> 0 to 255	Sets the character position in the string to which to apply the constraint.

ConstraintType

Specifies the type of constraint to set or opens an interactive dialog box that allows you to edit the constraints interactively. Unless otherwise specified, the constants listed below can be added together.

By default, all constraints apply to any font in the context. For example, `M_LETTER + M_DIGIT` means any letter and any digit of any font is acceptable. To restrict the constraint to a specific font, see the combination value below.

To set or edit the constraints in an interactive dialog box, select the following.

● For opening an interactive dialog box	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INTERACTIVE	<i>[This is only applicable to Windows]</i> Opens a dialog box that allows you to set or edit all the constraints of the specified string interactively. In this case, the CharPos parameter specifies the first character listed in the dialog box. Use the scroll bar to access the other characters in the string. You must set the CharList parameter to M_NULL . (summarize)

Unless otherwise specified, the ConstraintType constants listed below can be added together.

● For specifying the constraint	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT +	Sets the constraint to the default constraint. For a specific character position (the CharPos parameter is set to an individual value), the default is the same as the default constraint of the string model. For the default character position (the CharPos parameter is set to M_DEFAULT), the default is the same as M_ANY . With the exception of combination values, M_DEFAULT cannot be added to any other constraints. (summarize)
<input type="checkbox"/> M_ANY +	Sets the constraint to any character. With the exception of combination values, M_ANY cannot be added to any other constraints. (summarize)
<input type="checkbox"/> M_DIGIT +	Sets the constraint to any digit (0 to 9).
<input type="checkbox"/> M_LETTER +	Sets the constraint to any letter (a to z and A to Z). Note that M_LETTER does not differentiate between upper and lower cases. (summarize)

Combination constants for [M_LETTER](#);



You can add one of the following values to the above-mentioned value to set the letter case of the constraint.

● For specifying the letter case of the constraint	

 Value	Description
 M_LOWERCASE	Sets the constraint to any lowercase letter (a to z).
 M_UPPERCASE	Sets the constraint to any uppercase letter (A to Z).

Combination constant for the values listed in [For specifying the constraint](#)

You can add the following value to the above-mentioned values to set the constraint to a specific font.

● For restricting the constraint		
 Value	Description	
 M_FONT_INDEX(MIL_INT <i>FontIndex</i>)	Specifies that the constraint will be restricted to a specific font. Note that this value is only available for a fontbased context. (summarize)	
	Parameters	
	FontIndex This parameter specifies the index of the font. You can set this parameter to one of the following:	
	<table><tr><td>0 to 255</td><td>Specifies the index of the individual font from which the constraint is taken. When setting the constraint, the font index can be the index of a font that does not currently exist. However, if the font still does not exist at preprocessing time, an error will be logged.</td></tr></table>	0 to 255
0 to 255	Specifies the index of the individual font from which the constraint is taken. When setting the constraint, the font index can be the index of a font that does not currently exist. However, if the font still does not exist at preprocessing time, an error will be logged.	

CharList

Specifies an explicit list of valid characters, at the specified position. This is an optional, null-terminated string.

All the specified characters must be compatible with the **ConstraintType** parameter definition. Each character in the character list must be unique; no character repetition is allowed in the **CharList** parameter.

The string array must be of the right type for the encoding scheme selected ([MstrInquire\(\)](#) with [M_ENCODING](#)).

Set this parameter to **M_NULL** if you are setting the **ConstraintType** parameter to [M_INTERACTIVE](#) or if not specifying an explicit list.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

MstrStream

Synopsis

Load, restore, or save a String Reader context from/to a file or a memory stream.

Syntax

```
void MstrStream(
    MIL_TEXT_PTR MemPtrOrFileName,
    MIL_ID SystemId,
    MIL_INT Operation,
    MIL_INT StreamType,
    MIL_DOUBLE Version,
    MIL_INT ControlFlag,
    MIL_ID *ObjectIdPtr,
    MIL_INT *SizeByteVarPtr
)
```

Description

This function can load or restore a String Reader context from a file or memory stream. All of the String Reader context settings that were in effect when the String Reader context was saved will be restored. A loaded or restored String Reader context is not preprocessed, therefore you must call [MstrPreprocess\(\)](#) before performing a read with [MstrRead\(\)](#).

Moreover, this function can also save a String Reader context to a file or memory stream. All information about the previously allocated String Reader context is saved, including all of the individual font and string settings. However, preprocessing changes are not saved.

To inquire the number of bytes necessary to save a String Reader context to memory stream, you should first call this function ([MstrStream\(\)](#)) with [M_INQUIRE_SIZE_BYTE](#).

The content saved to memory stream is equivalent to the content saved to file. In addition, any file saved with [MstrSave\(\)](#) is equivalent to a file saved using this function.

You can use this and other MIL stream functions, for example, to save all required MIL objects, as well as any other custom data, for your application to a memory stream. Once in a memory stream, you can write the stream to a single file or transfer it over a network. You are responsible for concatenating the streams and for saving the stream to file.

Using [MstrStream\(\)](#), you can choose to save a backwards-compatible version of the String Reader context, which will work using a version of MIL that is up to one major release older than the current version (depending on which version is specified). For example, if you allocate a String Reader context using MIL 9.0 and save it to version 8.0, you can restore this context on a computer where MIL 8.0 is installed. However, all settings and features unique to the higher version will be ignored when restored using the lower version. Besides saving backwards-compatible versions, you can also load or restore String Reader contexts saved using MIL version 8.0 or above. Settings that do not exist in the lower version will be filled with default values when the String Reader context is loaded or restored.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]
Note that streaming a fontless context is not supported.

Parameters

MemPtrOrFileName

Specifies the file or memory stream.

● For specifying the file or memory stream	
☐ Value	Description
☐ MIL_TEXT(MIL_TEXT_PTR FileName)	<p>Specifies the drive, directory, and name of the file, for example: "C:\mydirectory\myfile", when the StreamType parameter is set to M_FILE. For easier use with other Matrox Imaging software products, when saving a String Reader context to a file, use the MSR file extension. The function handles (internally) the opening and closing of the file. If the file already exists, it will be overwritten when M_SAVE is performed.</p> <p>To open or save the file on the remote computer (under Distributed MIL), prefix the specified filename string with "remote:///", for example: "remote:///C:\mydirectory\myfile".</p>

	(summarize)						
	<table> <tr> <td></td><td><i>Parameters</i></td></tr> <tr> <td></td><td><i>FileName</i></td></tr> <tr> <td></td><td>Specifies the drive, directory, and name of the file.</td></tr> </table>		<i>Parameters</i>		<i>FileName</i>		Specifies the drive, directory, and name of the file.
	<i>Parameters</i>						
	<i>FileName</i>						
	Specifies the drive, directory, and name of the file.						
<input type="checkbox"/> M_INTERACTIVE	[<i>This is only applicable to Windows</i>] Opens a dialog box from which you can interactively specify the drive, directory, and name of the file, when the StreamType parameter is set to M_FILE .						
<input type="checkbox"/> M_NULL	Specifies to ignore this parameter. This parameter must be set to M_NULL when performing an M_INQUIRE_SIZE_BYTE operation. (summarize)						
<input type="checkbox"/> MemPtr	Specifies the address of the block of memory, when the StreamType parameter is set to M_MEMORY . The designated block must be large enough to stream the entire object. To ascertain the required size, call this function (MstrStream()) first with M_INQUIRE_SIZE_BYTE . (summarize)						

SystemId

Specifies the system on which to restore the String Reader context. For [M_INQUIRE_SIZE_BYTE](#), [M_LOAD](#), and [M_SAVE](#), **SystemId** is ignored and should be set to **M_NULL**. For an [M_RESTORE](#) operation, this parameter should be set to one of the following values:

● For specifying the system on which to restore the String Reader context	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

Operation

Specifies the operation to perform on the String Reader context. This parameter must be set to one of the following values:

● For specifying the operation	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_INQUIRE_SIZE_BYTE	Inquires the number of bytes required to save a String Reader context to memory stream. This operation is not supported when the StreamType parameter is set to M_FILE . (summarize)
<input type="checkbox"/> M_LOAD	Loads the content of a specified file or memory stream into a previously allocated String Reader context.
<input type="checkbox"/> M_RESTORE	Restores a String Reader context from a file or memory stream and assigns it an identifier.
<input type="checkbox"/> M_SAVE	Saves a String Reader context to a specified file or memory stream.

StreamType

Specifies the type of stream in which to store/from which to restore the String Reader context. This parameter must be set to one of the following values:

● For specifying the type of stream	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_FILE	Specifies a file stream.
<input type="checkbox"/> M_MEMORY	Specifies a memory stream. You are responsible for allocating a block of memory for the stream. (summarize)

Version

Specifies the MIL version of the String Reader context. This parameter must be set to one of the following values.

● For specifying the MIL version	
☐ Value	Description
☐ M_DEFAULT	Specifies the default version. For an M_SAVE or M_INQUIRE_SIZE_BYTE operation, it sets the version to current version of MIL. For an M_LOAD or M_RESTORE operation, it reads the file or stream for the version information. This is the only possible setting for these operations. (summarize)
☐ M_PROC_VERSION_80	Sets the version to MIL 8.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
☐ M_PROC_VERSION_80_PP1	Sets the version to MIL 8.0 Processing Pack 1. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
☐ M_PROC_VERSION_80_PP2	Sets the version to MIL 8.0 Processing Pack 2. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
☐ M_PROC_VERSION_80_PP3	Sets the version to MIL 8.0 Processing Pack 3. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
☐ M_PROC_VERSION_80_PP4	Sets the version to MIL 8.0 Processing Pack 4. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)
☐ M_PROC_VERSION_90	Sets the version to MIL 9.0. This option is available only for M_INQUIRE_SIZE_BYTE and M_SAVE operations. (summarize)

ControlFlag

Reserved for future expansion and must be set to **M_DEFAULT**.

ObjectIdPtr

Specifies the address of the variable in which to write or from which to read the identifier of the String Reader context.

For [M_INQUIRE_SIZE_BYTE](#) and [M_SAVE](#) operations, **ObjectIdPtr** specifies the address of the variable from which to read the String Reader context identifier.

For an [M_LOAD](#) operation, **ObjectIdPtr** specifies the address of the variable from which to read the identifier of the String Reader context where the file or memory stream content will be loaded.

For an [M_RESTORE](#) operation, **ObjectIdPtr** specifies the address in which to return the identifier of the restored String Reader context. If the operation is not successful, **M_NULL** is returned.

SizeByteVarPtr

Specifies the address of the variable in which to write the size of the String Reader context, in bytes. If the size is not required, you can set this parameter to **M_NULL**.

Note that the size of a String Reader context will vary depending on the MIL version specified.

Compilation information

Header	Include mil.h.
Library	Use mil.lib; milstr.lib.
DLL	Requires mil.dll; milstr.dll.

Msys functions

Synopsis

The functions prefixed with Msys make up the System Allocation and Inquiry module. The System Allocation and Inquiry module supports the allocation and inquire of systems. The system control represents a physical board, most commonly an imaging frame grabber. The system control also allows you to access the graphics controller and Host CPU. The system control is used to specify which physical device you wish to access, and allows some system-wide settings to be set. Once the system control is set up, you can add other components to your application, such as a digitizer control, to control specific aspects of the device.

Functions

- [MsysAlloc](#)
- [MsysControl](#)
- [MsysFree](#)
- [MsysGetHookInfo](#)
- [MsysHookFunction](#)
- [MsysInquire](#)

MsysAlloc

Synopsis

Allocate a MIL system.

Syntax

```
MIL_ID MsysAlloc(
    MIL_CONST_TEXT_PTR SystemDescriptor,
    MIL_INT SystemNum,
    MIL_INT InitFlag,
    MIL_ID *SystemIdPtr
)
```

Description

This function allocates a MIL system so that it can be used by subsequent MIL functions. This function can allocate a board-type system, which consists of a Matrox imaging board (or third-party board that is compatible with ActiveMIL), the Host CPU and memory, and any available graphics controller. Alternatively, this function can allocate a Host-type system, which consists of the Host CPU and memory, and any available graphics controller. Upon execution of this function, MIL ensures that it can open communication with the hardware associated with the system before allocating it, and generates an error if it cannot.

A system must be allocated before any buffers, displays, or digitizer can be allocated on it. Before allocating a system, an application must be allocated, using [MappAlloc\(\)](#) or [MappAllocDefault\(\)](#). To use the default system, you must allocate it using [M_SYSTEM_DEFAULT](#).

Note, upon allocation of an application, a default Host system is automatically allocated. Rather than using **MsysAlloc()** to allocate a Host system, you can use this default Host system, by specifying **M_DEFAULT_HOST** wherever a Host system identifier is required.

When you no longer need a particular system, free it using `MsysFree()`.

[Matrox Odyssey eA/XA; Matrox Odyssey eCL/XCL; Matrox Odyssey eD/XD]

When you have several imaging boards in one cluster, you must allocate the boards in that cluster in sequential order, starting with the first board. However, when boards are not connected together, they can be allocated in any order.

Parameters

SystemDescriptor

Specifies the type of system to allocate. Set this parameter to one of the following values:

For specifying the type of system to allocate																																					
Value	Description														corona-II (a)	giga vision (b)	cronosplus (c)	gpu processing (d)	hellios ea/Xa (e)	hellios eel/Xcd (f)	hellios ed/Xd (g)	ieee 1394 i1dc (h)	met-II /cl (i)	met-II /dmg (k)	met-II /mc (l)	met-II /std (m)	morphis qxt (o)	nextis (p)	odyssey ea/Xa (q)	odyssey eel/Xcd (r)	odyssey ed/Xd (s)	solios ea/Xa (t)	solios eel/Xcd (u)	solios gige (v)	wio (w)		
MIL_TEXT(MIL_TEXT_PTR DMILRemoteSys)	Allocates a DMIL remote system.														a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	To allocate a DMIL remote system on a computer, that computer must have a valid MIL installation.																																				
	Parameters																																				
	DMILRemoteSys A string that specifies the remote computer's name or IP address, port number, and MIL system type.														a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Performs the same initialization as M_COMPLETE .
<input type="checkbox"/> M_COMPLETE	Performs a complete initialization of the system: initialize the system to its default state and download any required resident software. At least one complete initialization is necessary after you power-up your system. (summarize)
<input type="checkbox"/> M_PARTIAL	Initializes the system with its default state, but do not download any resident software.

SystemIdPtr

Specifies the address of the variable in which to write the system identifier. Since the **MsysAlloc()** function also returns the system identifier, you can set this parameter to **M_NULL**. If allocation fails, **M_NULL** is written as the identifier.

Return value

The returned value is the system identifier if the allocation is successful. If allocation fails, **M_NULL** is returned.

Remark

- If you are creating a DLL that includes a call to **MsysAlloc()**, ensure that the call is not made from the **DllMain()** function, because **MsysAlloc()** might load a required DLL and you cannot load a DLL from **DllMain()**. If necessary, call **MsysAlloc()** from an initialization function in your DLL instead.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MsysControl

Synopsis

Control a system setting.

Syntax

```
void MsysControl(
    MIL_ID SystemId,
    MIL_INT ControlType,
    MIL_INT ControlValue
)
```

Description

This function allows you to control the specified system setting.

To inquire the current value of a particular system setting, use [MsysInquire\(\)](#).

Parameters

SystemId

Specifies the system identifier. This parameter should be set to one of the following values:

● For the system identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
<input type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .

ControlType

Specifies the type of setting to control.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the new value to assign to the system setting specified by the **ControlType** parameter.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- [For controlling auxiliary signals](#)
- [For system settings](#)
- [For MPEG compression streams](#)
- [For UART settings](#)
- [For user-defined signals](#)
- [For Watchdog settings](#)

The following control types allow you to set the mode and the purpose of an auxiliary signal. To control specific types of signals, refer to the table [For user-defined signals](#) below and to the tables [For controlling the settings to grab using an exposure](#) and [For controlling the settings to grab using a trigger](#) in `MdigControl()`.

For controlling auxiliary signals																	
ControlType	Description																
ControlValue																	
<input type="checkbox"/> M_AUX_SIGNAL_SOURCE +	Sets the type of signal to be routed to a auxiliary output signal, or I/O signal set to output. Note that a user-defined signal used as either a trigger or exposure source cannot also be used as a user-defined signal. (summarize)																
<input type="checkbox"/> M_DEFAULT	Same as M_USER_BIT .																
<input type="checkbox"/> M_GRAB_EXPOSURE +	Specifies that an exposure signal is routed. Note that to use the exposure signal of a specific timer, this value must be specified in combination with either M_TIMER1 or M_TIMER2 . (summarize)																
<input type="checkbox"/> M_GRAB_TRIGGER	Specifies that a trigger signal will be routed.																
<input type="checkbox"/> M_USER_BIT	Specifies that a user-defined output signal is routed.																
		corona-II (a)	gige vision (c)	gige vision (d)	heliios ea/xa (e)	heliios ed/xd (g)	heliios ed/xd (h)	iris (i)	met-II /dig (k)	met-II /dmc (l)	met-II /std (m)	morphis qxt (o)	odyssey ea/xa (q)	odyssey ed/xd (s)	odyssey ed/xd (t)	solios ea/xa (u)	vio (w)

Combination constants for [M_GRAB_EXPOSURE](#) (of [M_AUX_SIGNAL_SOURCE](#));

You must add one of the following values to the above-mentioned value to set the timer to use.

For specifying which timer to use																	
Value	Description																
<input type="checkbox"/> M_TIMER1	Specifies that the exposure signal for timer 1 will be used.																
<input type="checkbox"/> M_TIMER2	Specifies that the exposure signal for timer 2 will be used.																
		corona-II (a)	gige vision (c)	gige vision (d)	heliios ea/xa (e)	heliios ed/xd (f)	heliios ed/xd (g)	iris (i)	met-II /dig (k)	met-II /dmc (l)	met-II /std (m)	morphis qxt (o)	odyssey ea/xa (q)	odyssey ed/xd (s)	odyssey ed/xd (t)	solios ea/xa (u)	vio (w)

The following control types allow you to control the system settings.

For system settings																	
ControlType	Description																
ControlValue																	
<input type="checkbox"/> M_1394_POWER +	Sets the functional state of the specified port. Note that this value is only available for Matrox 4Sight-M 1394b adapter board.																
		corona-II (a)	gige vision (c)	gige vision (d)	heliios ea/xa (e)	heliios ed/xd (f)	heliios ed/xd (g)	iris (i)	met-II /dig (k)	met-II /dmc (l)	met-II /std (m)	morphis qxt (o)	odyssey ea/xa (q)	odyssey ed/xd (s)	odyssey ed/xd (t)	solios ea/xa (u)	vio (w)

The following control types and control values specify the settings for MPEG compression streams. To specify a particular stream, see the combination value below.

For MPEG compression streams																									
ControlType	Description	corona-II (a)	chromplus (b)	gige vision (c)	gpu processing (d)	hellios ea/Xa (e)	hellios ec/Xd (f)	hellios ed/Xd (g)	hellios ee/Xd (h)	hls (i)	met-II /dl (j)	met-II /dlq (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nextis (p)	odyssey ea/Xa (q)	odyssey ec/Xd (r)	odyssey ed/Xd (s)	solios ea/Xa (t)	solios ed/Xd (u)	solios gige (v)	vio (w)	
ControlValue																									
M_STREAM_BIT_RATE +	<div>[For essential MIL-Lite information, see remarks.]</div> <div>Sets the encoding bit rate of the specified stream, in kbps. Used for M_CONSTANT and M_VARIABLE_MAX bit rate modes. (summarize)</div>															o									
M_DEFAULT	Specifies the default value. The default value is 600 kbps. (summarize)															o									
192 to 4000	Range of acceptable bit rate values.															o									
M_STREAM_BIT_RATE_MODE +	<div>[For essential MIL-Lite information, see remarks.]</div> <div>Sets the mode for the encoding bit rate of the specified stream. (summarize)</div>															o									
M_DEFAULT	Same as M_VARIABLE_MAX .															o									
M_CONSTANT	Specifies that the encoding bit rate of the specified stream is constant. The encoding bit rate is set using M_STREAM_BIT_RATE . (summarize)															o									
M_VARIABLE	Specifies that the encoding bit rate of the specified stream is variable. The encoding bit rate is set using M_STREAM_Q_PARAMETER as a fixed quantization parameter. (summarize)															o									
M_VARIABLE_MAX	Specifies that the encoding bit rate of the specified stream is variable. The maximum encoding bit rate is set using M_STREAM_BIT_RATE . (summarize)															o									
M_STREAM_CONTROL +	<div>[For essential MIL-Lite information, see remarks.]</div> <div>Sets the state of the specified MPEG compression stream. (summarize)</div>															o									
M_CLOSE	Closes a stream and frees the associated resources. Closing a stream will reset all compression parameters. (summarize)															o									
M_OPEN + M_MPEG4	Opens an MPEG compression stream on that specific system.															o									
M_STREAM_ENCODING_MODE +	<div>[For essential MIL-Lite information, see remarks.]</div> <div>Sets the interlaced encoding mode for the specified stream. Note that this parameter is used only in D1 size. (summarize)</div>															o									
M_DEFAULT	Same as M_DEINTERLACE .															o									
M_DEINTERLACE	Specifies a deinterlaced encoding mode.															o									
M_INTERLACE	Specifies an interlaced encoding mode.															o									
M_PROGRESSIVE	Specifies a progressinve encoding mode.															o									
M_STREAM_GROUP_OF_PICTURE_SIZE +	<div>[For essential MIL-Lite information, see remarks.]</div> <div>Sets the interval between intra-coded frames (I-frames) for the specified stream.</div>															o									

<input type="checkbox"/> 8	Specifies that the data length is 8 bits.				e	f	g					n					t	u	
<input type="checkbox"/> M_UART_FREE +	Stops all MIL UART operations and frees all UART resources used by MIL. (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> M_DEFAULT	Specifies the default behavior.				e	f	g					n					t	u	
<input type="checkbox"/> M_UART_PARITY +	Sets whether character data is sent or received with a parity bit and how the parity bit is set. The parity bit is an extra data bit (0 or 1) that is added to each character for error checking purposes. (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> M_DEFAULT	Same as M_DISABLE .				e	f	g					n					t	u	
<input type="checkbox"/> M_DISABLE	Species that no extra bit is added (no parity).				e	f	g					n					t	u	
<input type="checkbox"/> M_EVEN	Specifies that the number of 1's will be even.				e	f	g					n					t	u	
<input type="checkbox"/> M_ODD	Specifies that the number of 1's will be odd.				e	f	g					n					t	u	
<input type="checkbox"/> M_UART_READ_CHAR +	Reads one character from the UART input buffer. If the input buffer is empty, the function will wait for the amount of time specified by M_UART_TIMEOUT . If a time-out occurs, the '?' character will be returned. (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> Value	Specifies the address of the variable in which to save the character read from the UART.				e	f	g					n					t	u	
<input type="checkbox"/> M_UART_READ_STRING +	Reads a string of incoming data from the UART. The number of characters to read can be specified with M_UART_READ_STRING_LENGTH or M_UART_STRING_DELIMITER . M_UART_TIMEOUT specifies the maximum time to wait between each byte when reading incoming data. Use MsysInquire() with M_UART_BYTES_READ to wait for the read operation to complete and retrieve the actual number of bytes read. (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> Value	Specifies the address of the character array in which to save the string read from the UART. The size of this array must be set to the same value as the M_UART_READ_STRING_MAXIMUM_LENGTH control type. (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> M_UART_READ_STRING_LENGTH +	Sets the length of the string to read using M_UART_READ_STRING . (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> M_DEFAULT	Specifies the use of M_UART_STRING_DELIMITER to delineate the end of the string to read.				e	f	g					n					t	u	
<input type="checkbox"/> Value	Specifies the string length, in bytes.				e	f	g					n					t	u	
<input type="checkbox"/> M_UART_READ_STRING_MAXIMUM_LENGTH +	Sets the maximum length of the string to read. This prevents global protection faults from happening when using the M_UART_READ_STRING control type. (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> Value	Specifies the maximum length of the string, in bytes.				e	f	g					n					t	u	
<input type="checkbox"/> M_UART_SPEED +	Sets the baud rate of the UART. (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> M_DEFAULT	Specifies the default value. The default value is 14400. (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> 300; 600; 1200; 1800; 2400; 3600; 4800; 7200; 9600; 14400; 19200; 28800; 38400; 57600; 76800; 115200; 128000; 256000	Specifies the baud rate of the UART.				e	f	g					n					t	u	
<input type="checkbox"/> M_UART_STOP_BITS +	Sets the number of extra data bit(s) (1 or 2) that are added to each character to indicate the end of the character. (summarize)				e	f	g					n					t	u	
<input type="checkbox"/> M_DEFAULT	Same as 1 .				e	f	g					n					t	u	
<input type="checkbox"/> 1	Specifies that there is 1 stop bit.				e	f	g					n					t	u	
<input type="checkbox"/> 2	Specifies that there are 2 stop bits.				e	f	g					n					t	u	

	M_DEVn	Specifies the device to control, where <i>n</i> can be from 0 and 15.			e	f	g					n						t	u	
	<i>Board specific</i>																			
	There is 1 UART available on Matrox Morphis.											n								
	Note that there are no UARTs available on Matrox Morphis PC/104-Plus.																			
	There are 4 UARTs available.					e		g												
	There are 2 UARTs available on Matrox Helios eCL/XCL dual-Base, 1 on Matrox Helios eCL/XCL single-Full.						f													
	There are 4 UARTs available on Matrox Solios eA/XA Quad, 2 on Matrox Solios eA/XA dual, and 1 on Matrox Solios eA/XA single.																	t		
	There are 2 UARTs are available on Matrox Solios eCL/XCL dual-Base/single-Medium operating in dual-Base mode, and 1 is available on Matrox Solios eCL/XCL-B, eCL/XCL-F, and eCL/XCL dual-Base/single-Medium operating in single-Medium mode.																		u	

The following values allow you to control user-defined signals.

[Matrox 4Sight]

Note that the following are available on Matrox 4Sight-M, even without any additional Matrox imaging board(s), and on Matrox CronosPlus, Matrox Iris, Matrox Nexis, and Matrox Morphis.

[Matrox Morphis]

User-defined signals are not supported on Matrox Morphis PC/104-Plus. Use the user-defined signals of Matrox 4Sight-M instead.

[Matrox IEEE 1394 IIDC driver]

These control types are only supported on Matrox 4Sight-M 1394b; they are not supported on third-party IEEE 1394 IIDC-compliant network boards. In addition, these control types are only available on the Matrox Imaging board, and not on the camera.

Note that for other Matrox imaging boards that have user-defined signals, but are not supported with the constants below, see [MdigControl\(\)](#).

For user-defined signals																								
ControlType	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	ilee 1394 iIdc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios ec/xd (u)	solios gige (v)	vio (w)
ControlValue																								
<input type="checkbox"/> M_USER_BIT_IN_ACTIVE_LEVEL	Sets the active level for all user-defined input signals. This allows you to define whether MsysInquire() with M_USER_BIT_VALUE_.... should return M_ON when the signal is high or when it is low. (summarize)									i							p							
<input type="checkbox"/> M_HIGH	Specifies that a user-defined input signal is active when it is high (maximum signal level).									i							p							
<input type="checkbox"/> M_LOW	Specifies that a user-defined input signal is active when it is low (minimum signal level). This is the default value. (summarize)									i							p							
<input type="checkbox"/> M_USER_BIT_INTERRUPT_MODE	Sets the type of functional-state change upon which to generate an interrupt. Note that this only applies to user-defined input signals.	b							h	i					n	o	p							

[illegible]

Note that in this case, the combination constant can only be set to a value.

This is the default value.

[\(summarize\)](#)

Combination constants for the values listed in [For controlling auxiliary signals](#); and for the following values: [M_USER_BIT_INTERRUPT_STATE](#); [M_USER_BIT_MODE](#); [M_USER_BIT_VALUE_OUT](#);

You must add one or more of the following values to the above-mentioned values to set the user-defined signal(s) to affect.

The following values are available for Matrox 4Sight-M, Matrox CronosPlus, Matrox IEEE 1394 IIDC, Matrox Iris, Matrox Morphis, and/or Matrox Morphis QxT. See the Board Specific Notes for a listing of available user-defined signals and their corresponding numbers in MIL.

For specifying which user-defined signal(s) to affect		corona-II (a)	gige vision (c)	gige vision (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	helios ee/xd (h)	ieee 1394 i1dc (i)	iris (j)	met-II /dlg (k)	met-II /dmc (l)	met-II /std (m)	morpheus (n)	morpheus qxt (o)	nexus (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ec/xd (u)	solos gige (v)	vio (w)
Value	Description																						
<input type="checkbox"/> <code>M_BIT_MASK(MIL_INT Bit_encoded_value)</code>	Specifies a bit-encoded mask value to identify the group of user-defined signals to be affected. Note that this value can only added to M_USER_BIT_VALUE_OUT . (summarize)	b							i					n	o	p							
	<i>Parameters</i>																						
	<i>Bit_encoded_value</i> Enabled bits in the mask value identify which user-defined signals to affect. Note that this value is only available when using a bit-encoded control value. (summarize)	b							i					n	o	p							
<input type="checkbox"/> 0 <= Value	Specifies a specific user-defined signal. Note that in this case, the control value cannot be set to a bit-encoded value. (summarize)	b							h	i				n	o	p							

The following control types and control values specify the settings for the Watchdog.

The Watchdog is not supported on Matrox Morphis PC/104-Plus.

For Watchdog settings		corona-II (a)	gige vision (c)	gige vision (d)	helios ea/xa (e)	helios ec/xd (f)	helios ed/xd (g)	helios ee/xd (h)	ieee 1394 i1dc (i)	iris (j)	met-II /dlg (k)	met-II /dmc (l)	met-II /std (m)	morpheus (n)	morpheus qxt (o)	nexus (p)	odyssey ea/xa (q)	odyssey ec/xd (r)	odyssey ed/xd (s)	solos ea/xa (t)	solos ec/xd (u)	solos gige (v)	vio (w)
ControlType	Description																						
<input type="checkbox"/> <code>M_WATCHDOG_MODE</code>	Sets the current state of the Watchdog. (summarize)													n	o								
<input type="checkbox"/> <code>M_DEFAULT</code>	Same as M_DISABLE .													n	o								
<input type="checkbox"/> <code>M_DISABLE</code>	Disables the Watchdog.													n	o								
<input type="checkbox"/> <code>M_ENABLE</code>	Enables the Watchdog. The application should be ready to reset the Watchdog and all timeouts (M_WATCHDOG_REBOOT_TIMEOUT & M_WATCHDOG_TIMEOUT) should be programmed before enabling the Watchdog.													n	o								

- *[Matrox-4Sight]*
Note that **M_USER_BIT...** control types are available on Matrox 4Sight-M even without any additional Matrox imaging boards, and on Matrox CronosPlus, Matrox IEEE 1394 IIDC, Matrox Iris, Matrox Morphis, and/or Matrox Morphis QxT, unless otherwise specified. For more information on user-defined signals, refer to the [User-defined signals for Matrox 4Sight-M](#) section in [Chapter 16: Matrox 4Sight-M](#) section of the MIL Board-specific Notes. To see the information regarding the **M_USER_BIT...** control types if you have a Matrox 4Sight-M, select Matrox Morphis as your board type in the [Welcome](#) section in [About Matrox MIL help](#).
- *[MIL-Lite]*
Note that MIL-Lite compression support is reliant upon the presence of Matrox compression acceleration hardware. The compression algorithm supported depends on the hardware. If you don't have Matrox compression acceleration hardware, you can still perform compression under MIL-Lite with the compression/decompression runtime license package installed.

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MsysFree

Synopsis

Free a system.

Syntax

```
void MsysFree(  
    MIL_ID SystemId  
)
```

Description

This function deallocates a system previously allocated with [MsysAlloc\(\)](#).

Prior to freeing a system, ensure that all buffers, displays, and digitizers allocated on the system are freed.

Parameter

SystemId

Specifies the identifier of the system to free.

Remark

- If you are creating a DLL that includes a call to **MsysFree()**, ensure that the call is not made from the **DllMain()** function, because **MsysFree()** might unload any DLL loaded with [MsysAlloc\(\)](#) and you cannot unload a DLL from **DllMain()**. If necessary, call **MsysFree()** from a clean-up function in your DLL instead.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MsysGetHookInfo

Synopsis

Get information about a hook event.

Syntax

```
MIL_INT MsysGetHookInfo(
    MIL_ID SystemId,
    MIL_ID EventId,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function allows you to get information about the event that caused the hook function to be called. **MsysGetHookInfo()** should only be called within the scope of a system hook-handler function (see [MsysHookFunction\(\)](#)).

Note that functions hooked to an event execute on a distinct thread. This permits the functions to run asynchronously from the operation that fired the event and from functions hooked to other events. Although there is a small queue to permit a certain amount of overlap, hooked functions should not take longer to execute than the period in which two of their associated events can occur. You cannot determine the instance of the event that fired the function, and even if this were possible, this information would generally not be very useful. Typically, a hooked function performs the minimum number of operations required and, if necessary, performs longer processes by launching other threads.

Parameters

SystemId

Specifies the identifier of the system.

For the system identifier																	
<input checked="" type="checkbox"/> Value	Description	corona-II (a)	crotopus (b)	glige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ec/Xcl (f)	helios ed/Xd (g)	leee 1394 i/dc (h)	iris (i)	met-II /cl (j)	met-II /dig (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nextis (p)
<input type="checkbox"/> M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.	b							h	i				m	n	o	p
<input checked="" type="checkbox"/> MIL system identifier	Specifies a valid system identifier, previously allocated using MsysAlloc() .	b							h	i				m	n	o	p

EventId

Specifies the system event identifier received by the hook-handler function (see [MsysHookFunction\(\)](#)).

InquireType

Specifies the type of information about which to inquire.

If the hook-handler function was called with a **HookType** parameter of the [MsysHookFunction\(\)](#) set to **M_USER_BIT_CHANGE**, set the **InquireType** parameter to the value below.

Unless otherwise specified, the following values require that you pass the **UserVarPtr** parameter the address of a **MIL_INT**.

For inquiring information																	
<input checked="" type="checkbox"/> Value	Description	corona-	crotopus	glige vis	gpu prc	helios e	helios e	helios e	leee 13	iris (i)	met-II	met-II	met-II	met-II	morphi	morphi	nextis (f)

[illegible]

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- MIL_INT

Specifies the address in which to write the requested information.

Return value

Returns null (**M_NULL**) on success, and returns a non-null (!**M_NULL**) value on failure, without logging any errors in the application.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MsysHookFunction

Synopsis

Hook a function to a system event.

Syntax

```
void MsysHookFunction(
    MIL_ID SystemId,
    MIL_INT HookType,
    MIL_SYS_HOOK_FUNCTION_PTR HookHandlerPtr,
    void *UserDataPtr
)
```

Description

This function allows you to attach or detach a user-defined function to a specified system event. Once a hook-handler function is defined and hooked to an event, it is automatically called when the event occurs.

Note that functions hooked to an event execute on a distinct thread. This permits the functions to run asynchronously from the operation that fired the event and from functions hooked to other events. Although there is a small queue to permit a certain amount of overlap, hooked functions should not take longer to execute than the period in which two of their associated events can occur. You cannot determine the instance of the event that fired the function. Typically, a hooked function performs the minimum number of operations required and, if necessary, performs longer processes by launching other threads.

You can hook more than one function to an event by making separate calls to `MsysHookFunction` for each function that you want to hook. MIL automatically chains and keeps an internal list of all these hooked functions. When a function is hooked, this new function is added to the end of the list. When the event happens, all user-defined functions in the list will be executed in the same order that they were hooked to the event. You can also remove any function from the list; in this case, MIL preserves the order of the remaining functions in the list.

Parameters

SystemId

Specifies the identifier of the system on which to hook a function.

For specifying the system identifier																				
Value	Description	corona-II (a)	giga vision (c)	gpu processing (d)	helios ea/xa (e)	helios ed/xd (f)	helios ed/xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /d1g (j)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	odyssey ea/xa (q)	odyssey ed/xd (s)	odyssey ed/xd (t)	solios ea/xa (u)	solios ed/xd (v)	vio (w)
MIL system identifier	Specifies a valid system identifier, previously allocated using the MsysAlloc() . Note that this is only available on an allocated Matrox CronosPlus, Matrox Morphis, Matrox Helios, or Matrox Solios system. (summarize)	b			e	f	g	h	i				n	o	p			t	u	

HookType

Specifies the system event to which to hook the function. This parameter can be set to one of the following values.

<div> <div></div> <div>For specifying the system event to hook</div> </div> <div> <input type="checkbox"/> Value </div>		Description
<div> <div></div> <div>solios glge (v)</div> </div> <div> <div></div> <div>solios ea/xa (t)</div> </div> <div> <div></div> <div>solios eci/xd (l)</div> </div> <div> <div></div> <div>odyssey ea/xa</div> </div> <div> <div></div> <div>odyssey ed/xd</div> </div> <div> <div></div> <div>odyssey eci/xc</div> </div> <div> <div></div> <div>odyssey ea/xa</div> </div> <div> <div></div> <div>next (p)</div> </div> <div> <div></div> <div>morphis opt (o)</div> </div> <div> <div></div> <div>morphis (n)</div> </div> <div> <div></div> <div>met-II /std (m)</div> </div> <div> <div></div> <div>met-II /mc (l)</div> </div> <div> <div></div> <div>met-II /dkg (k)</div> </div> <div> <div></div> <div>met-II /cl (l)</div> </div> <div> <div></div> <div>lrs (l)</div> </div> <div> <div></div> <div>lee 1394 /lhc</div> </div> <div> <div></div> <div>helios ed/xd (g)</div> </div> <div> <div></div> <div>helios ec/xd (t)</div> </div> <div> <div></div> <div>helios ea/xa (e)</div> </div> <div> <div></div> <div>gpu processing</div> </div> <div> <div></div> <div>glge vision (c)</div> </div> <div> <div></div> <div>crnosplus (b)</div> </div> <div> <div></div> <div>corona-II (a)</div> </div>		

You can add the following value to the above-mentioned values to specify that the function should be unhooked.

For specifying that the function should be unhooked																								
Value	Description	corona-II (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	helios ea/Xa (e)	helios ec/Xd (f)	helios ed/Xd (g)	ieee 1394 i1dc (h)	iris (i)	met-II /cl (j)	met-II /dlg (k)	met-II /mc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/Xa (q)	odyssey ec/Xd (r)	odyssey ed/Xd (s)	solios ea/Xa (t)	solios ec/Xd (u)	solios gige (v)	vio (w)
M_UNHOOK	Unhooks the specified function, if hooked to an event.		b			e	f	g	h	i					n	o	p				t	u		

HookHandlerPtr

Specifies the address of the function that should be called when the specified event occurs. The hook-handler function must be declared as follows:

```
MIL_INT MFTYPE HookHandler(
    MIL_INT HookType,
    MIL_ID EventId,
    void *UserDataPtr
)
Parameters:
    HookType
        Type of system event hooked.

    EventId
        Event identifier to pass to MsysGetHookInfo\(\) when inquiring about the hooked event.

    UserDataPtr
        Specifies the user data pointer passed to MsysHookFunction().
```

Upon successful completion, the hook-handler function should return **M_NULL**. Note, *MIL_SYS_HOOK_FUNCTION_PTR*, *MFTYPE*, and *MPTYPE* are reserved MIL predefined types for functions and data pointers.

UserDataPtr

Specifies the address of the user data that you want to make available to the hook-handler function. This address is passed to the hook-handler function, through its **UserDataPtr** parameter, when the specified event occurs. Set this parameter to **M_NULL** if not used.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MsysInquire

Synopsis

Inquire about a system setting.

Syntax

```
MIL_INT MsysInquire(
    MIL_ID SystemId,
    MIL_INT InquireType,
    void *UserVarPtr
)
```

Description

This function inquires about the specified system setting.

Note that you can use `MsysControl()` to control specific system settings.

Parameters

SystemId

Specifies the system identifier. This parameter should be set to one of the following values:

For the system identifier	
Value	Description
M_DEFAULT_HOST	Specifies the default Host system of the current MIL application.
MIL system identifier	Specifies a valid system identifier, which you have allocated using the MsysAlloc() function.

InquireType

Specifies the type of system setting about which to inquire. This parameter can be set to one of the following values:

The following inquire types allow you to inquire the mode and the purpose of an auxiliary signal. To inquire specific types of signals, refer to the table [For user-defined signals](#) below and to the tables [For controlling the settings to grab using an exposure](#) and [For controlling the settings to grab using a trigger](#) in `MdigInquire()`.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

For inquiring auxiliary signals	
Value	Description
	<div>solios gige (v) solios ec/xcl (u) solios ea/xa (t) odyssey ed/xd (s) odyssey ec/xcl (r) odyssey ea/xa (q) nexis (p) morphis qxt (o) morphis (n) met-II /std (m) met-II /mc (l) met-II /dlg (k) met-II /d (j) iris (i) leee 1394 iildc (h) hellios ed/xd (g) hellios ec/xcl (f) hellios ea/xa (e) gpu processing (d) gige vision (c) cronosplus (b) corona-II (a)</div>
M_AUX_SIGNAL_SOURCE +	<div>Returns the source of the specified auxiliary signal. You must specify a combination value from the table below. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values: M_GRAB_EXPOSURE; M_GRAB_TRIGGER; M_USER_BIT; (details)</div></div>

M_CRONOSPLUS	Matrox CronosPlus.
M_GIGE_VISION	Matrox GigE Vision.
M_HELIOS + M_XA + M_QHA	Matrox Helios eA/XA quad-independent.
M_HELIOS + M_XCL + M_DBCL	Matrox Helios eCL/XCL dual-Base.
M_HELIOS + M_XCL + M_SFCL	Matrox Helios eCL/XCL single-Full.
M_HELIOS + M_XD + M_QD	Matrox Helios digital.
M_IEEE_1394_IIDC	Matrox IEEE 1394 IIDC driver.
M_IRIS + M_a300b	Matrox Iris 300, where <i>a</i> represents the series name and <i>b</i> represents the specific model. Matrox Iris 300 comes in two series named P and E. There are eight models of the P300 and E300 available. They are: monochrome (no letter), the C (color), CR (color remote head), H (high-speed), HM (high-speed microhead), HR (high-speed remote head), HMR (high-speed microhead remote), and R (remote head).
M_IRIS + M_a700b	Matrox Iris 700, where <i>a</i> represents the series name and <i>b</i> represents the specific model. The Matrox Iris 700 comes in two series named P and E. There are six models of the P700 and E700 available. They are: monochrome (no letter), the R (remote head), WR (wafer reader remote head), and W (wafer reader).
M_IRIS + M_a1200b	Matrox Iris 1200, where <i>a</i> represents the series name and <i>b</i> represents the specific model. The Matrox Iris 1200 comes in two series named P and E. There are two models of the P1200 and E1200 available. They are: monochrome (no letter) and R (remote head).
M_METEOR_II_CL	Matrox Meteor-II /Camera Link.
M_METEOR_II_DIG	Matrox Meteor-II /Digital.
M_METEOR_II_MC	Matrox Meteor-II /Multi-channel.
M_METEOR_II_STD	Matrox Meteor-II /Standard.
M_MORPHIS + M_IO + M_J2K	Matrox Morphis with the I/O module and the JPEG2000 compression accelerator.
M_MORPHIS + M_J2K	Matrox Morphis PC/104-Plus with the JPEG2000 compression accelerator.
M_MORPHIS + M_2VD	Matrox Morphis PC/104-Plus with two video decoders.
M_MORPHIS + M_2VD + M_IO	Matrox Morphis with two video decoders and the I/O module.
M_MORPHIS + M_2VD + M_IO + M_J2K	Matrox Morphis with two video decoders, the I/O module, and the JPEG2000 compression accelerator.
M_MORPHIS + M_2VD + M_J2K	Matrox Morphis PC/104-Plus with two video decoders and the JPEG2000 compression accelerator.

M_MORPHISQXT + M_4VD + M_IO	Matrox Morphis QxT with four video decoders and the I/O module.
M_MORPHISQXT + M_16VD + M_IO	Matrox Morphis QxT analog color/monochrome x4 PCIe™ frame grabber with 16 video decoders.
M_MORPHISQXT + M_4VD + M_IO + M_COMPRESSION	Matrox Morphis QxT with four video decoders and an integrated JPEG2000 accelerator.
M_MORPHISQXT + M_16VD + M_IO + M_COMPRESSION	Matrox Morphis QxT analog color/monochrome x4 PCIe™ frame grabber with 16 video decoders and an integrated MPEG-4 video encoder.
M_MORPHISQXT + M_4VD + M_IO + M_COMPRESSION + M_AUDIO_MODULE	Matrox Morphis QxT with four video decoders, an integrated JPEG2000 accelerator, and an add-on module supporting 16 audio inputs.
M_MORPHISQXT + M_16VD + M_IO + M_COMPRESSION + M_AUDIO_MODULE	Matrox Morphis QxT analog color/monochrome x4 PCIe™ frame grabber with 16 video decoders, an integrated MPEG-4 video encoder, and an add-on module supporting 16 audio inputs.
M_NEXIS	Matrox Nexis S-series camera.
M_ODYSSEY + M_XA + M_QHA	Matrox Odyssey eA/XA quad-independent.
M_ODYSSEY + M_XCL + M_DBCL	Matrox Odyssey eCL/XCL dual-Base.
M_ODYSSEY + M_XCL + M_SFCL	Matrox Odyssey eCL/XCL single-Full.
M_ODYSSEY + M_XD + M_QD	Matrox Odyssey digital.
M_ODYSSEY + M_XPRO	Matrox Odyssey Xpro.
M_ODYSSEY + M_XPRO + M_DBCL	Matrox Odyssey Xpro with a Camera Link dual-Base module.
M_ODYSSEY + M_XPRO + M_QD	Matrox Odyssey Xpro with a Digital module.
M_ODYSSEY + M_XPRO + M_QHA	Matrox Odyssey Xpro with a quad-independent Analog module.
M_ODYSSEY + M_XPRO + M_SFCL	Matrox Odyssey Xpro with a Camera Link single-Full module.
M_ODYSSEY + M_XPRO_PLUS + M_DBCL + M_PF	Matrox Odyssey Xpro+ with a Camera Link dual-Base module and a processing FPGA.
M_ODYSSEY + M_XPRO_PLUS + M_PF	Matrox Odyssey Xpro+ with a processing FPGA.
M_ODYSSEY + M_XPRO_PLUS + M_QD + M_PF	Matrox Odyssey Xpro+ with a Digital module and a processing FPGA.
M_ODYSSEY + M_XPRO_PLUS + M_QHA + M_PF	Matrox Odyssey Xpro+ with a quad-independent Analog module and a processing FPGA.
M_ODYSSEY + M_XPRO_PLUS + M_SFCL + M_PF	Matrox Odyssey Xpro+ with a Camera Link single-Full module and a processing FPGA.
M_SOLIOS + M_XA + M_DA	Matrox Solios eA/XA dual analog.
M_SOLIOS + M_XA + M_QA	Matrox Solios eA/XA quad analog.

[illegible]

<div><div></div><div>M_CURRENT_THREAD_ID</div></div>	<div>Returns the identifier of the current system thread. This identifier can be used with the thread module. (summarize)</div> <div><div>UserVarPtr info</div><div>Data type: MIL_ID</div></div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_DCF_SUPPORTED</div></div>	<div>Returns whether the system supports downloadable digitizer configuration format (<i>DCF</i>) files.</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_DEFAULT_PITCH_BYTE</div></div>	<div>Returns the factor MIL uses to set the pitch, in bytes, of the buffers allocated on the system.</div>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_DIGITAL_MODULE_PRESENT</div></div>	<div>Returns whether the companion digital-input board is present or not. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values:</div><div><div>M_FALSE</div><div>Digital module is not present.</div></div><div><div>M_TRUE</div><div>Digital module is present.</div></div></div>	a																						
<div><div></div><div>M_DIGITIZER_NUM</div></div>	<div>Returns the total number of possible independent acquisition paths on the system. Note that this is not the same as the total number of allocated acquisition paths on the system. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values:</div><div><div>0 <= Value</div><div>The number independent acquisition paths available.</div></div><div><div>Board specific</div><div>When dealing with a cluster of boards featuring Matrox Odyssey Xpro, the total number of independent acquisition paths includes those of the frame grabber modules attached to any Matrox Odyssey Xpro board (or PMC board) in the cluster. The total number can be from 0 to 31. Use MsysInquire() with M_DIGITIZER_TYPE + M_DEVn to return the type of frame grabber module to which an independent acquisition path belongs.</div></div><div>Returns the total number of GigEVision cameras discovered.</div></div> <td>a</td> <td>b</td> <td>c</td> <td>d</td> <td>e</td> <td>f</td> <td>g</td> <td>h</td> <td>i</td> <td>j</td> <td>k</td> <td>l</td> <td>m</td> <td>n</td> <td>o</td> <td>p</td> <td>q</td> <td>r</td> <td>s</td> <td>t</td> <td>u</td> <td>v</td> <td>w</td>	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div><div>M_DIGITIZER_TYPE +</div></div>	<div>Returns the type of frame grabber(s) available to allocate a digitizer on the system. When dealing with a cluster of boards featuring Matrox Odyssey Xpro, the total number of acquisition paths available includes those of the frame grabber modules attached to any Matrox Odyssey Xpro board (or PMC board) in the cluster. This total number can be from 0 to 31. Use MsysInquire() with M_DIGITIZER_NUM to return the total number of frame grabber modules in the cluster. (summarize)</div> <div><div>UserVarPtr info</div><div>Return values:</div><div><div>M_DBCL</div><div>Matrox Odyssey XCL dual-Base Camera Link.</div></div><div><div>M_NEXIS + M_S700T</div><div>Matrox Nexis S-Series 700 camera.</div></div><div><div>M_NEXIS + M_S1200T</div><div>Matrox Nexis S-Series 1200 camera.</div></div><div><div>M_NEXIS + M_S300a</div><div>Matrox Nexis S-Series 300 camera, where a represents represents the specific model. The Matrox Nexis S-series 300 camera comes in 4 models; T (monochrome), CT (color), HT (high-speed), and HM (microhead).</div></div><div><div>M_NULL</div><div>No camera is connected to your Matrox Nexis.</div></div><div><div>M_QD</div><div>Matrox Odyssey XD.</div></div></div>			c														p	q	r	s			v

[illegible]

[illegible]

M_OWNER_APPLICATION	Returns the MIL identifier of the application on which the system has been allocated. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	UserVarPtr Info Data type: MIL_ID																							
M_PROCESSING_SYSTEM_TYPE	Returns the type of the system (on-board or Host) used to process allocated buffers. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	UserVarPtr Info Return values: M_SYSTEM_HOST_TYPE Host. M_SYSTEM_IRIS Matrox Iris. M_SYSTEM_ODYSSEY_TYPE Matrox Odyssey.																							
M_PROCESSOR_NUM	Returns the number of processors (CPUs) available on the allocated Matrox imaging board. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	UserVarPtr Info Return values: 0 <= Value The number of processors available.																							
	Board specific																							
	Note that, when dealing with a Matrox Odyssey Xpro+ board with the Processing FPGA, the FPGA will not register as a processor. To find out what type of Matrox Odyssey you are using, use M_BOARD_TYPE .																		q	r	s			
	Note that, when dealing with a Matrox Solios eCL/XCL with the optional Processing FPGA, the FPGA will register as a processor. To find out what type of Matrox Solios you are using, use M_BOARD_TYPE .																				t	u	v	
M_SERIAL_NUMBER	Returns the serial number of the Matrox Imaging board, as a string. (summarize)		b			e	f	g		i	j	k			n	o	p	q	r	s	t	u	v	w
	UserVarPtr Info Data type: array of type MIL_TEXT_CHAR Array size: To determine the size of the array required to store the serial number, use MsysInquire() with M_SERIAL_NUMBER_SIZE .																							
M_SERIAL_NUMBER_SIZE	Returns the length of the string returned by M_SERIAL_NUMBER .		b			e	f	g		i	j	k			n	o	p	q	r	s	t	u	v	w
M_SHARED_MEMORY_FREE	Returns the total amount of free on-board shared memory, in bytes.					e	f											q	r	s	t	u	v	w
M_STREAM_BIT_RATE +	Returns the encoding bit rate of the specified stream, in kbps. You must specify a combination value from the table below . (summarize)															o								
	UserVarPtr Info Return values: 192 to 4000; (details)																							
M_STREAM_BIT_RATE_MODE +	Returns the mode for the encoding bit rate of the specified stream. You must specify a combination value from the table below . (summarize)															o								
	UserVarPtr Info Return values: M_CONSTANT; M_VARIABLE; M_VARIABLE_MAX; (details)																							
M_STREAM_CONTROL +	Returns the state of the specified MPEG compression stream. You must specify a combination value from the table below . (summarize)															o								
	UserVarPtr Info Return values: M_CLOSE; M_OPEN + M_MPEG4; (details)																							
M_STREAM_ENCODING_MODE +	Returns the interlaced encoding mode for the specified stream. You must specify a combination value from the table below .															o								

[illegible]

[illegible]

The following inquire types and inquire values specify the settings for the Watchdog.

The Watchdog is not supported on Matrox Morphis PC/104-Plus.

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

For Watchdog settings																								
Value	Description	corona-ii (a)	cronosplus (b)	gige vision (c)	gpu processing (d)	hellios ea/xa (e)	hellios ed/xcl (f)	hellios ed/xcd (g)	ieee 1394 i8dc (h)	iris (i)	met-ii /dlg (k)	met-ii /cd (l)	met-ii /mc (l)	met-ii /std (m)	morphis (n)	morphis qxt (o)	nexts (p)	odyssey ea/xa (q)	odyssey ed/xcl (r)	odyssey ed/xcd (s)	sollos ea/xa (t)	sollos acd/xcl (u)	sollos gige (v)	vio (w)
M_WATCHDOG_MODE	Returns the current state of the Watchdog. (summarize)														n	o								
	<i>UserVarPtr info</i> Return values: M_DISABLE; M_ENABLE; (details)																							
M_WATCHDOG_PRESENT	Returns whether Watchdog circuitry is present. (summarize)														n	o								
	<i>UserVarPtr info</i> Return values: M_FALSE Watchdog circuitry is not present. M_TRUE Watchdog circuitry is present.																							
M_WATCHDOG_REBOOT_TIMEOUT	Returns the timeout value of the Watchdog's reboot timer, in msec. (summarize)														n	o								
	<i>UserVarPtr info</i> Return values: Please see MsysControl() with M_WATCHDOG_REBOOT_TIMEOUT . (details)																							
M_WATCHDOG_RESET_COUNTER	Returns the number of times the Watchdog has rebooted your computer. (summarize)														n	o								
	<i>UserVarPtr info</i> Return values: Value The number of Watchdog reboots.																							
M_WATCHDOG_TIMEOUT	Returns the timeout value of the Watchdog's main timer, in msec. (summarize)														n	o								
	<i>UserVarPtr info</i> Return values: Please see MsysControl() with M_WATCHDOG_TIMEOUT . (details)																							
M_WATCHDOG_WARNING_TIME	Returns the timeout value of the Watchdog's warning timer, in msec. (summarize)														n	o								

UserVarPtr info
Return values: M_INFINITE; Value; ([details](#))

The following values allow you to inquire user-defined signals.

[Matrox 4Sight]

Note that the following are available on Matrox 4Sight-M, even without any additional Matrox Imaging board(s), and on Matrox Iris, Matrox Nexis, Matrox Morphis, Matrox Morphis QxT, and Matrox CronosPlus.

[Matrox Morphis]

User-defined signals are not supported on Matrox Morphis PC/104-Plus. Use the user-defined signals of Matrox 4Sight-M instead.

[Matrox IEEE 1394 IIDC driver]

These inquire types are only supported on Matrox 4Sight-M 1394b; they are not supported on third-party IEEE 1394 IIDC-compliant network boards. In addition, these inquire types are only available on the Matrox Imaging board, and not on the camera.

Note that for other Matrox imaging boards that are not supported with the constants below, see [MdigInquire\(\)](#).

Unless otherwise specified, the following values require that you pass the [UserVarPtr](#) parameter the address of a MIL_INT.

For user-defined signals																									
Value	Description	corona-II (a)	cromoplus (b)	giga vision (c)	gpu processing (d)	hellios ea/xa (e)	hellios eci/xci (f)	hellios ed/xd (g)	ieee 1394 iidec (h)	iris (i)	met-II /ci (j)	met-II /dlig (k)	met-II /fmc (l)	met-II /std (m)	morphis (n)	morphis qxt (o)	nexis (p)	odyssey ea/xa (q)	odyssey eci/xd (r)	odyssey ed/xd (s)	solios ea/xa (t)	solios eci/xd (u)	solios gige (v)	va (w)	
M_USER_BIT_COUNT_IN	Returns the number of user-defined signals that can be used for input. This count includes the number of input and I/O user-defined signals. (summarize) <div><div>UserVarPtr info</div><div>Return values:</div><div>ValueNumber of user-defined signals that can be used for input.</div></div>	b							i						n	o									
M_USER_BIT_COUNT_OUT	Returns the number of user-defined signals that can be used for output. This count includes the number of output and I/O user-defined signals. (summarize) <div><div>UserVarPtr info</div><div>Return values:</div><div>ValueNumber of user-defined signals that can be used for output.</div></div>	b						h	i						n	o									
M_USER_BIT_IN_ACTIVE_LEVEL	Returns the active level for all user-defined input signals. (summarize) <div><div>UserVarPtr info</div><div>Return values: M_HIGH; M_LOW; (details)</div></div>								i								p								
M_USER_BIT_INTERRUPT_MODE	Returns the type of functional-state change upon which to generate an interrupt. Note that this only applies to user-defined input signals. (summarize) <div><div>UserVarPtr info</div><div>Return values: M_EDGE_FALLING; M_EDGE_RISING; (details)</div><div>Board specific</div></div>	b						h	i						n	o	p								

[illegible]

UserVarPtr

Accepts the address of one of the following (see above for specifics on which is expected):

- array of type MIL_TEXT_CHAR
- MIL_ID
- MIL_INT

Specifies the address in which to write the requested information. When **MsysInquire()** also returns the requested information, you can set this parameter to **M_NULL**.

Return value

The returned value is the requested information, usually cast to a *MIL_INT*. For inquire types which return a value cast to *MIL_TEXT_CHAR*, the returned value is **M_NULL**.

Remarks

- [\[matrox-4sight\]](#)
Note that **M_USER_BIT**... inquire types are available on Matrox 4Sight-M even without any additional Matrox imaging boards, and on Matrox CronosPlus, Matrox IEEE 1394 IIDC, Matrox Iris, Matrox Morphis, and/or Matrox Morphis QxT, unless otherwise specified. For more information on user-defined signals, refer to
- the [User-defined signals for Matrox 4Sight-M](#) section in [Chapter 16: Matrox 4Sight-M](#) section of the MIL Board-specific Notes. To see the information regarding the **M_USER_BIT**... inquire types if you have a Matrox 4Sight-M, select Matrox Morphis as your board type in the the [Welcome](#) section in [About Matrox MIL help](#).

Examples

The following example uses `M_BOARD_TYPE`. The returned value is then masked so that only the board type is returned.

```

/* To return only the main board type, and not the sub-board types (for example M_XCL,
M_FAST, or M_SDI), mask the return value with M_BOARD_TYPE_MASK.
Note that when dealing with the Matrox Corona II with Digital module
(M_CORONA_II_WITH_DIG_MODULE), using M_BOARD_TYPE_MASK will return
M_CORONA_II because M_CORONA_II_WITH_DIG_MODULE is the same as
M_CORONA_II + M_DIGITAL_MODULE). */

/* Call MsysInquire with M_BOARD_TYPE to inquire the full board type. */
MsysInquire(MilSystem, M_BOARD_TYPE, &BoardType);

/* Use M_BOARD_TYPE_MASK to verify the main board type. */
if ((BoardType & M_BOARD_TYPE_MASK) == M_SOLIOS)
{
/* Perform a Matrox Solios-specific task. */
}

```

[Matrox Iris; Matrox Nexis]

The following example uses `M_USER_BIT_IN_ACTIVE_LEVEL`.

```
/* A useful technique for MIL application to detect whether a new Matrox Iris has been
added is to inquire M_USER_BIT_IN_ACTIVE_LEVEL while temporarily disabling
error printing. */

unsigned long ActiveLevelInquire = M_LOW;
```



```
MapControl(M_ERROR,M_PRINT_DISABLE);

/*  The ActiveLevelInquire variable is set to M_HIGH when a new Matrox Iris
    is found. Otherwise, the ActiveLevelInquire remains in its default state (M_LOW). */
MsysInquire(MilSystem,M_USER_BIT_IN_ACTIVE_LEVEL,&ActiveLevelInquire);
MapControl(M_ERROR,M_PRINT_ENABLE);
```

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

Mthr functions

Synopsis

The functions prefixed with Mthr make up the Thread module. The Thread module supports allocation of MIL thread contexts and synchronization events. Using the functions of the Thread module, you can also control the created MIL thread contexts and events, inquire about various settings, and synchronize execution of multiple threads.

Functions

- [MthrAlloc](#)
- [MthrControl](#)
- [MthrFree](#)
- [MthrInquire](#)
- [MthrWait](#)
- [MthrWaitMultiple](#)

MthrAlloc

Synopsis

Allocate a MIL thread context, event, or mutex.

Syntax

```
MIL_ID MthrAlloc(  
    MIL_ID SystemId,  
    MIL_INT ObjectType,  
    MIL_INT ControlFlag,  
    MTHREADFCTPTR ThreadFctPtr,  
    void *UserPtr,  
    MIL_ID *ThreadEventOrMutexId  
)
```

Description

This function allocates a MIL thread context, event, or mutex.

Threads are function streams, used to ensure sequential execution of operations within the same thread, while allowing simultaneous yet independent execution of operations in other threads. **MthrAlloc()** can allocate threads using two different methods:

- With the first method ([M_THREAD](#)), **MthrAlloc()** creates a MIL thread context for the new thread, and allows you to specify a pointer to a function that will be executed by the thread. When a thread contains a function call whose target processor is an on-board processor that supports multi-threading, MIL automatically creates a corresponding thread on that system's on-board processor. The functions of the Thread module allow you to synchronize threads running on the Host and/or various MIL systems.
- With the second method, **MthrAlloc()** creates a selectable thread ([M_SELECTABLE_THREAD](#)). Selectable threads are threads executed on an on-board processor that supports multi-threading but can be controlled from a single corresponding thread on the Host. Use [MthrControl\(\)](#) with [M_THREAD_SELECT](#) to send MIL functions to be executed by a selectable thread.

For more information on these two types of threads, see the [Multi-threading](#) section in [Chapter 25: Multi-processing, multi-core, and multi-threading](#).

A MIL event is a synchronization marker that can signal the completion of a required set of functions in one thread to other threads. **MthrAlloc()** can allocate MIL events, or map a new MIL event to an existing MIL event.

A mutex is a mutual exclusion object that allows threads to synchronize access to shared resources. Once a mutex is allocated on the specified system, you can lock and unlock critical sections of code. Locking a critical section of code ensures that no two threads can access the same data at the same time. To lock a MIL mutex, you must call [MthrControl\(\)](#) with [M_LOCK](#) or [M_LOCK_TRY](#) immediately preceding the section of code to lock. If you lock a mutex, you must unlock it at the end of the critical section of code using [MthrControl\(\)](#) with [M_UNLOCK](#). Once you are finished using the mutex object, free it using [MthrFree\(\)](#).

To control or inquire about a MIL thread context, event, or mutex, use the [MthrControl\(\)](#) or [MthrInquire\(\)](#) function, respectively. To synchronize the execution of threads, use [MthrWait\(\)](#).

Parameters

SystemId

Specifies the identifier of the system on which to allocate a MIL thread context, event, or mutex.

This parameter should be set to one of the following values:

● For specifying the system	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_ALL	Specifies that the identifier of the allocated MIL thread context or event can be recognized by any multi-thread system allocated in your application. You cannot specify this value when allocating a MIL mutex. (summarize)
<input type="checkbox"/> MIL system identifier	Specifies the MIL identifier of the required system, previously allocated using MsysAlloc() .

ObjectType

Specifies the type of object to allocate. This parameter should be set to one of the following values:

● For specifying the type of object to allocate	
☐ Value	Description
☐ M_EVENT	Allocates a new MIL synchronization event on the specified system.
☐ M_EVENT_CREATE	Creates a new MIL synchronization event on the specified system, which maps to the MIL event specified by the ControlFlag parameter and is in the same state. The created event allows for multiple, individual on-board processors to change and read the state of the same event. (summarize)
☐ M_MUTEX	Allocates a MIL mutex on the specified system. Note that you cannot allocate a mutex when the SystemId parameter is set to M_ALL . (summarize)
☐ M_SELECTABLE_THREAD	Allocates a selectable thread on the specified multi-threaded system. This allows you to synchronize the execution of functions on an on-board processor without creating a Host thread. Note that you cannot allocate selectable threads when the SystemId parameter is set to M_ALL . Use MthrControl() with M_THREAD_SELECT to select the on-board thread to which to send the functions. (summarize)
☐ M_THREAD	Allocates a thread and associates it with a MIL thread context. If the target processor is an on-board processor of a system that supports multi-threading (such as Matrox Odyssey), MIL automatically creates, and eventually terminates, an on-board thread for each thread that sends commands to the board. (summarize)

ControlFlag

Specifies the initialization state of the MIL thread context, event, mutex, or the identifier of an existing MIL event. This parameter should be set to one of the following values:

● For specifying the initiation state	
☐ Value	Description
☐ M_DEFAULT	Specifies the default initialization state. This is the only setting available for threads and mutexes. For events, M_DEFAULT is the same as M_NOT_SIGNED + M_AUTO_RESET . (summarize)
☐ M_NOT_SIGNED +	Initializes the event as not signaled. You must specify a combination value from the table below . (summarize)
☐ M_SIGNED +	Initializes the event as signaled. You must specify a combination value from the table below . (summarize)
☐ MIL event identifier	Specifies the identifier of the user-allocated MIL event to which to map the new M_EVENT_CREATE event. The specified user-allocated MIL event should not be one that is recognized by multiple multi-thread systems (SystemId parameter set to M_ALL). (summarize)

Combination constants for [M_SIGNED](#); [M_NOT_SIGNED](#);

You must add one of the following values to the above-mentioned values to set how the event is reset.

● For M_SIGNED or M_NOT_SIGNED	
☐ Value	Description
☐ M_AUTO_RESET	Specifies that the event is reset automatically. The state of this type of event is automatically reset to M_NOT_SIGNED upon the return of a call to MthrWait() with M_EVENT_SYNCHRONIZE or M_EVENT_WAIT . Events that are reset automatically are useful in applications where only one thread waits on a specific event. (summarize)

<input type="checkbox"/> M_MANUAL_RESET	<div>Specifies that the event is reset manually.</div> <div>The state of this type of event remains unchanged until a call to <code>MthrControl()</code> with <code>M_EVENT_SET</code> is issued.</div> <div>Events that are reset manually are useful when multiple threads wait on a specific event.</div> <div>(summarize)</div>
-----------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

ThreadFctPtr

Specifies the address of the function that will be executed by the new thread (`M_THREAD`). For all other objects, this parameter should be set to `M_NULL`.

The function must include calls to all functions that you consider as being part of one thread, and be declared as follows:

```
MIL_UINT32 MFTYPE FunctionToCall(  
    void *UserDataPtr  
)  
Parameters:  
  
    UserDataPtr  
  
    Specifies a pointer to user data that is passed to the UserPtr parameter of MthrAlloc().
```

UserPtr

Specifies the address of user data necessary to execute the `FunctionToCall()` function of the new thread (`M_THREAD`). For all other objects, this parameter should be set to `M_NULL`.

ThreadEventOrMutexId

Specifies the address of the variable in which to write the MIL identifier of the MIL thread context, event, or mutex. Since the `MthrAlloc()` function also returns the requested information, you can set this parameter to `M_NULL`.

Return value

The returned value is the identifier of the MIL thread, event, or mutex if the allocation is successful. If allocation fails, `M_NULL` is returned.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MthrControl

Synopsis

Control a MIL thread context, MIL event, or MIL mutex setting.

Syntax

```
void MthrControl(  
    MIL_ID ThreadEventOrMutexId,  
    MIL_INT ControlType,  
    MIL_DOUBLE ControlValue  
)
```

Description

This function controls a MIL thread context, MIL event, or MIL mutex setting. Most of these control type settings can be inquired using [MthrInquire\(\)](#).

Parameters

ThreadEventOrMutexId

Specifies the identifier of a user-allocated MIL thread context, event, or mutex, allocated using [MthrAlloc\(\)](#). This parameter can also be set to the MIL identifier of a system.

For specifying the identifier of a user-allocated MIL thread context, event, or mutex	
Value	Description
M_DEFAULT	Specifies the default MIL thread context identifier associated with the current Host thread.
MIL thread context/event/mutex/system identifier	Specifies the identifier of a user-allocated MIL thread context, event, or mutex (MthrAlloc()), or a valid system identifier (MsysAlloc()). When the parameter is set to the MIL identifier of a system, the function controls the current thread on the particular system. (summarize)

ControlType

Specifies the type of setting to control.

See the [Parameter associations](#) section for possible values.

ControlValue

Specifies the new value to assign to the setting specified by the **ControlType** parameter.

See the [Parameter associations](#) section for possible values.

Parameter associations

Possible values for the **ControlType** and **ControlValue** parameters are described in the following tables:

- For thread contexts
- For controlling MIL events
- For controlling a MIL mutex

The following **ControlType** and corresponding **ControlValue** settings can be specified to control MIL thread contexts:

For thread contexts

ControlType	Description	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
ControlValue																								
<div><div></div>M_ACCELERATOR</div>	Sets whether the thread uses hardware acceleration. Hardware acceleration speeds up the execution of certain functions in a thread. Use MsysInquire() with M_ACCELERATOR_PRESENT to learn if your Matrox imaging board has an accelerator. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_DISABLE</div>	Disables hardware acceleration.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_ENABLE</div>	Enables hardware acceleration. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_BUS_MASTER_COPY_MODE</div>	Sets the synchronization mode for copy operations when they are driven by your imaging board (bus master). In this case, the current setting for M_THREAD_MODE does not effect the synchronization mode. (summarize) <div><div>Board specific</div><div>The setting of this control is only meaningful when either M_BUS_MASTER_COPY_FROM_HOST or M_BUS_MASTER_COPY_TO_HOST (MsysControl()) is set to M_ENABLE; when set to M_DISABLE, the synchronization mode is always synchronous.</div></div>					e	f	g			j	k			n	o		q	r	s	t	u	v	
<div><div></div>M_DEFAULT</div>	Same as M_SYNCHRONOUS .					e	f	g			j	k			n	o		q	r	s	t	u	v	
<div><div></div>M_ASYNCHRONOUS</div>	Specifies that when possible, control will be returned to the imaging board (bus master) immediately after a copy operation is launched.					e	f	g			j	k			n	o		q	r	s	t	u	v	
<div><div></div>M_SYNCHRONOUS</div>	Specifies that the execution of a copy operation must be completed before returning control to the imaging board (bus master).					e	f	g			j	k			n	o		q	r	s	t	u	v	
<div><div></div>M_MP_MAX_CORES</div>	Sets the maximum number of CPU cores to use for the thread when multi-core processing is enabled using MappControl() or MthrControl() with M_MP_USE . To set a different number of CPU cores for all threads, use MappControl() with M_MP_MAX_CORES_PER_THREAD . As long as the sum of the maximum number of CPU cores available for each thread does not exceed the number of CPU cores available to your MIL application, MIL tries to ensure that the threads use different cores. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_DEFAULT</div>	Specifies the default value. The default value is determined by MappControl() with M_MP_MAX_CORES_PER_THREAD . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>2 <= Value <= 65535</div>	Specifies the maximum number of CPU cores to use for the thread. Although you can specify a specific number of CPU cores, the effective number of CPU cores is limited by the absolute maximum number of CPU cores that any thread can use (as set using MappControl() with M_MP_MAX_CORES_PER_THREAD), any limits imposed by your operating system, and the physical number of CPU cores in your computer. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_MP_USE</div>	Sets whether multi-core processing (MP) is enabled for the thread. This control type overrides MappControl() with M_MP_USE . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_DEFAULT</div>	Specifies the default value. The default value is determined by MappControl() with M_MP_USE . (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<div><div></div>M_DISABLE</div>	Disables multi-core processing.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

<input type="checkbox"/> M_ENABLE	Enables multi-core processing.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_THREAD_COMMANDS_ABORT	Cancels all calls queued in the thread. The remainder of the application is executed. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_DEFAULT	Implements the default behavior.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_THREAD_MODE	Sets the synchronization mode for the execution of the thread. Works in parallel with MsysControl() ; if either function's M_THREAD_MODE controls are set to M_SYNCHRONOUS , threads will run synchronously. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
	<i>Board specific</i>																							
	Note that the specified synchronization mode does not affect copy operations. Use M_BUS_MASTER_COPY_MODE to specify the required mode for copying.					e	f	g			j	k			n	o		q	r	s	t	u	v	
<input type="checkbox"/> M_DEFAULT	Same as M_ASYNCHRONOUS .	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_ASYNCHRONOUS	Specifies that control will be returned to the Host immediately after a MIL function is sent to the processor of a system (when the system and function allow an immediate return).	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_SYNCHRONOUS	Specifies that the execution of a MIL function sent to the processor of a system must be completed (execution terminated) before returning control to the Host.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_THREAD_PRIORITY	Sets the priority status of the thread. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> 1 to 15	Specifies the priority status of the thread executed on a Matrox Odyssey on-board processor. 1 specifies the lowest priority. (summarize)																		q	r	s			
<input type="checkbox"/> M_ABOVE_NORMAL	Specifies that the thread is above normal priority. Only time critical threads will be executed before it. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_BELOW_NORMAL	Specifies that the thread is below normal priority. Threads of normal, above normal, and time critical priority will be executed before it. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_IDLE	Specifies that the thread is idle. The thread will remain idle until its priority status is changed. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_LOWEST	Specifies that the thread is of the lowest priority. All other non-idle threads will be executed before it. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_NORMAL	Specifies that the thread is of normal priority. Threads of above normal and time critical priority will be executed before it. This is the default value. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_TIME_CRITICAL	Specifies that the thread is time critical. Time critical threads will be executed before all other threads that are not time critical. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
<input type="checkbox"/> M_THREAD_SELECT	Sets the selectable thread, specified by the ThreadEventOrMutexId parameter, as the destination for subsequent MIL functions. This allows MIL functions to be executed on systems with on-board processors. Before calling MthrControl() with M_THREAD_SELECT you should inquire and save the identifier of the on-board thread that is by default associated with the Host thread, using MsysInquire() with M_CURRENT_THREAD_ID . This allows you to return control to this on-board thread. If the ThreadEventOrMutexId parameter is set to an identifier of a thread that is not selectable, changing the settings of this control will generate an error. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

<input type="checkbox"/> M_DEFAULT	Implements the default behavior.	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
------------------------------------	----------------------------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

The following **ControlType** and corresponding **ControlValue** setting can be specified to control MIL events:

For controlling MIL events		
<div><input type="checkbox"/> ControlType</div>		Description
	ControlValue	
<div><input type="checkbox"/> M_EVENT_SET</div>		Sets an event to the specified state. (summarize)
<div><input type="checkbox"/> M_NOT_SINGALED</div>		Specifies to set the event to the not signaled state.
<div><input type="checkbox"/> M_SINGALED</div>		Specifies to set the event to the signaled state.

The following **ControlType** and corresponding **ControlValue** setting can be specified to control a MIL mutex:

For controlling a MIL mutex	
ControlType	Description
ControlValue	
M_LOCK	<p>Forces the current thread to wait until the specified MIL mutex is available and then locks it. Locking the mutex blocks all other threads from accessing the current critical section of code.</p> <p>Note that the current thread can lock the same mutex several times without an error occurring. However, the current thread must unlock (M_UNLOCK) the mutex as many times as it was locked. For example, if the current thread previously locked the mutex twice, the mutex must be unlocked twice after the critical section of code has completed.</p> <p>(summarize)</p>
M_DEFAULT	Specifies the default behavior.
M_LOCK_TRY	<p>Locks the specified MIL mutex if it is currently unlocked. Locking the mutex blocks all other threads from accessing the current critical section of code.</p> <p>If the mutex is locked by another thread, M_LOCK_TRY does not force the thread to wait for the mutex to become unlocked; the thread continues executing without executing the critical section of code protected by the mutex.</p> <p>To determine whether the current thread has successfully locked the mutex, use MthrInquire() with M_LOCK_TRY.</p> <p>(summarize)</p>
M_DEFAULT	Implements the default behavior.
M_UNLOCK	<p>Unlocks the specified mutex.</p> <p>(summarize)</p>
M_DEFAULT	Implements the default behavior.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MthrFree

Synopsis

Free a MIL thread context, event, or mutex.

Syntax

```
void MthrFree (
    MIL_ID ThreadEventorMutexId
)
```

Description

This function deallocates a MIL thread context, event, or mutex previously allocated with [MthrAlloc\(\)](#).

In a given thread, **MthrFree()** must be the last MIL function called; no other MIL function can be executed in the thread after a call to this function.

Parameter

ThreadEventorMutexId

Specifies the identifier of the MIL thread context/event/mutex to free.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MthrInquire

Synopsis

Inquire about a MIL thread context, event, or mutex setting.

Syntax

```
MIL_INT MthrInquire (
    MIL_ID ThreadEvenOrMutexId,
    MIL_INT InquireType,
    void *InquireValue
)
```

Description

This function inquires about the specified MIL thread context, event setting, or mutex setting.

Parameters

ThreadEvenOrMutexId

Specifies the identifier of a user-allocated MIL thread context, event, or mutex, allocated using [MthrAlloc\(\)](#). This parameter should be set to one of the following values:

● For a user-allocated thread context or event	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default MIL thread context identifier associated with the current Host thread.
<input type="checkbox"/> MIL thread context/event/mutex/system identifier	Specifies the identifier of a user-allocated MIL thread context, event, mutex (MthrAlloc()), or a valid system identifier (MsysAlloc()). When the parameter is set to the MIL identifier of a system, the function inquires about the current thread on the particular system. (summarize)

InquireType

Specifies the type of information about which to inquire. This parameter can be set to one of the values below. See [MthrControl\(\)](#) for more information about these values.

The following [InquireType](#) settings can be specified to inquire about MIL thread contexts.

Unless otherwise specified, the following values require that you pass the [InquireValue](#) parameter the address of a MIL_INT.

● For thread contexts																											
<input type="checkbox"/> Value	Description	vivo (w) solos gige (v) solos ed/xcd (u) solos ea/xa (t) odyssey ed/xcd (s) odyssey ed/xcd (r) odyssey ea/xa (q) nexis (p) morphis qxt (o) morphis (n) met-II /std (m) met-II /mc (l) met-II /dig (k) met-II /d (j) iris (i) leee 1394 ldc (h) helios ed/xcd (g) helios ed/xcd (f) helios ea/xa (e) gpu processing (d) gige vision (c) cromoplus (b) corona II (a)																									
<input type="checkbox"/> M_ACCELERATOR	Returns whether the thread uses hardware acceleration. (summarize)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			
	<i>InquireValue info</i> Return values: M_DISABLE; M_ENABLE; (details)																										
<input type="checkbox"/> M_BUS_MASTER_COPY_MODE	Returns the synchronization mode for copy operations when they are driven by your					e	f	g			j	k			n	o		q	r	s	t	u	v				

[illegible]

The following `InquireType` settings can be specified to inquire about MIL events.

Unless otherwise specified, the following values require that you pass the [InquireValue](#) parameter the address of a MIL_INT.

For MIL events	
Value	Description
M_EVENT_MODE	Returns the reset mode of the event. (summarize)
	<div><i>InquireValue info</i></div> <div>Return values: M_AUTO_RESET; M_MANUAL_RESET; (details)</div>
M_EVENT_STATE	Returns the state of the event. (summarize)
	<div><i>InquireValue info</i></div> <div>Return values: M_NOT_SINGALED; M_SINGALED; (details)</div>

The following `InquireType` settings can be specified to inquire about MIL mutexes.

Unless otherwise specified, the following values require that you pass the [InquireValue](#) parameter the address of a MIL_INT.

For mutexes	
	co
	gl
	gp
	ha
	ha
	he
	ir
	m
	m
	m
	m
	m
	ne
	od
	od
	od
	so
	so
	vi

MthrWait

Synopsis

Perform a wait operation on a MIL thread or event.

Syntax

```
MIL_INT MthrWait (
    MIL_ID ThreadOrEventId,
    MIL_INT WaitOption,
    MIL_INT *State
)
```

Description

This function allows you to synchronize the execution of threads by forcing the current thread to wait for the completion of the specified thread or the change of state of the specified event.

Parameters

ThreadOrEventId

Sets the identifier of the MIL thread context or event with which to be synchronized. This parameter can also be set to the MIL identifier of a system.

This parameter should be set to one of the following values:

● For specifying the thread context or event identifier	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_DEFAULT	Specifies the default MIL thread context identifier associated with the current Host thread.
<input type="checkbox"/> MIL thread context/event/system identifier	Specifies the identifier of a user-allocated MIL thread context or event (MthrAlloc()), or a valid system identifier (MsysAlloc()).

WaitOption

Specifies the wait option. The following wait options are available for threads.

● For specifying the wait option for threads	
<input type="checkbox"/> Value	Description
<input type="checkbox"/> M_THREAD_END_WAIT +	Forces the current thread to wait for the end (the death) of the specified thread. The thread currently running cannot be the thread for which you are waiting. This option can only be used for a thread allocated using MthrAlloc() with M_THREAD . If the ThreadOrEventId parameter is set to a MIL system identifier, using this wait option results in an error. (summarize)
<input type="checkbox"/> M_THREAD_WAIT +	Forces the current thread to wait for the completion of all functions that are not asynchronous grab commands, in the specified thread's command queue. If the ThreadOrEventId parameter was set to a MIL system identifier, the current thread waits for the completion of the current thread on the specified system. (summarize)

Combination constant for the values listed in [For specifying the wait option for threads](#)

You can add the following value to the above-mentioned values to set the time interval after which a thread is considered to be timed out.

● For threads	

Value	Description										
<code>M_THREAD_TIMEOUT(MIL_INT ThreadTimeout)</code>	<p>Specifies the time interval after which a thread is considered to be timed out. (summarize)</p> <table> <tr> <td colspan="2"><i>Parameters</i></td></tr> <tr> <td colspan="2"><i>ThreadTimeout</i> Sets the required timeout interval, in msec. Set this parameter to the following:</td></tr> <tr> <td>0</td><td>Same as M_INFINITE.</td></tr> <tr> <td>M_INFINITE</td><td>Specifies an infinitely long timeout interval.</td></tr> <tr> <td>Value > 0</td><td>Sets the timeout interval to a user-defined value, in msec.</td></tr> </table>	<i>Parameters</i>		<i>ThreadTimeout</i> Sets the required timeout interval, in msec. Set this parameter to the following:		0	Same as M_INFINITE .	M_INFINITE	Specifies an infinitely long timeout interval.	Value > 0	Sets the timeout interval to a user-defined value, in msec.
<i>Parameters</i>											
<i>ThreadTimeout</i> Sets the required timeout interval, in msec. Set this parameter to the following:											
0	Same as M_INFINITE .										
M_INFINITE	Specifies an infinitely long timeout interval.										
Value > 0	Sets the timeout interval to a user-defined value, in msec.										

The following wait options are available for events.

For specifying the wait option for events	
Value	Description
<code>M_EVENT_SYNCHRONIZE +</code>	<p>Allows the current thread to continue with execution while forcing its corresponding on-board thread, located on the same board as the specified event, to wait for the specified event to be in an M_SINGALED state.</p> <p>If the specified event is allocated on a board that supports asynchronous calls, MIL issues the wait command to that board and allows the current thread to proceed executing while its corresponding thread on that board is waiting for the specified event to change to a signaled state. Any subsequent calls issued on that on-board thread are executed only after the event is signaled. If the specified event is not allocated on a board that supports asynchronous calls, M_EVENT_SYNCHRONIZE acts exactly like M_EVENT_WAIT.</p> <p>If the event is of the type that is reset automatically (M_AUTO_RESET), the state of the event is reset to M_NOT_SINGALED, after the wait operation. (summarize)</p>
<code>M_EVENT_WAIT +</code>	<p>Forces the current thread to wait for the specified event to be in an M_SINGALED state or for the event to be timed out. The current thread will not proceed with execution until the specified event has changed to a signaled state or timed out.</p> <p>If the event is of the type that is reset automatically (M_AUTO_RESET), after the wait operation, the state of the event is reset to M_NOT_SINGALED. (summarize)</p>

Combination constant for the values listed in [For specifying the wait option for events](#)

You can add the following value to the above-mentioned values to set the time interval after which an event is considered to be timed out.

For events											
Value	Description										
<code>M_EVENT_TIMEOUT(MIL_INT EventTimeout)</code>	<p>Specifies the time interval after which an event is considered to be timed out. (summarize)</p> <table> <tr> <td colspan="2"><i>Parameters</i></td></tr> <tr> <td colspan="2"><i>EventTimeout</i> Sets the required timeout interval, in msec. Set this parameter to the following:</td></tr> <tr> <td>0</td><td>Same as M_INFINITE.</td></tr> <tr> <td>M_INFINITE</td><td>Specifies an infinitely long timeout interval.</td></tr> <tr> <td>Value > 0</td><td>Sets the timeout interval to a user-defined value, in msec.</td></tr> </table>	<i>Parameters</i>		<i>EventTimeout</i> Sets the required timeout interval, in msec. Set this parameter to the following:		0	Same as M_INFINITE .	M_INFINITE	Specifies an infinitely long timeout interval.	Value > 0	Sets the timeout interval to a user-defined value, in msec.
<i>Parameters</i>											
<i>EventTimeout</i> Sets the required timeout interval, in msec. Set this parameter to the following:											
0	Same as M_INFINITE .										
M_INFINITE	Specifies an infinitely long timeout interval.										
Value > 0	Sets the timeout interval to a user-defined value, in msec.										

State

Specifies the address of the variable in which to write the state of the specified thread or event. For **M_THREAD_END_WAIT**, **M_THREAD_WAIT**, and **M_EVENT_WAIT** the state is **M_SINGALED** if the specified thread has successfully completed or if the state of the specified event has changed, or **M_TIMEOUT** if the thread or event on which the current thread was waiting timed out. For **M_EVENT_SYNCHRONIZE**, the state is **M_UNKNOWN**.

Since the **MthrWait()** function also returns the requested information, you can set this parameter to **M_NULL**.

Return value

The returned value for **M_THREAD_END_WAIT**, **M_THREAD_WAIT** is **M_SINGALED**, and **M_EVENT_WAIT** if the specified thread has successfully completed or if the state of the specified event has changed, or **M_TIMEOUT** if the thread or event on which the current thread was waiting timed out. The returned value for **M_EVENT_SYNCHRONIZE** is **M_UNKNOWN**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.

MthrWaitMultiple

Synopsis

Perform a wait operation on multiple MIL events.

Syntax

```
MIL_INT MthrWaitMultiple(  
    MIL_ID *EventArrayId,  
    MIL_INT EventArraySize,  
    MIL_INT WaitOption,  
    MIL_INT *State  
)
```

Description

This function allows you to synchronize the execution of threads by forcing the current thread to wait for one of the MIL events identified in a user-supplied array to change state. To force the current thread to wait for all events identified in the user-supplied array to change state, add [M_ALL_OBJECTS](#) to [M_EVENT_WAIT](#) or [M_EVENT_SYNCHRONIZE](#) when setting the **WaitOption** parameter.

Parameters

EventArrayId

Specifies the address of a user-supplied array that contains the identifiers of the MIL events for which to wait. All MIL event identifiers must be allocated, using [MthrAlloc\(\)](#), on the same system.

EventArraySize

Specifies the number of events in the user-supplied array.

WaitOption

Specifies the wait option. This parameter should be set to one of the following values.

● For specifying the wait option	
☐ Value	Description
☐ M_EVENT_SYNCHRONIZE +	<p>Allows the current thread to continue executing while forcing its corresponding on-board thread, located on the same board as the events identified in the user-supplied array, to wait for one of the events to be in an M_SINGALED state or for a time out to occur.</p> <p>If the events are allocated on a board that supports asynchronous calls, MIL issues the wait command to that board and allows the current thread to continue executing. Its corresponding thread, on that board, waits for an event identified in the array to change to a signaled state. Any subsequent calls issued on that on-board thread are executed only after an event is signaled. If the events are not allocated on a board that supports asynchronous calls, M_EVENT_SYNCHRONIZE acts exactly like M_EVENT_WAIT.</p> <p>If the event that changes state is of the type that is reset automatically (M_AUTO_RESET), the state of the event is reset to M_NOT_SINGALED, after the wait operation. (summarize)</p>
☐ M_EVENT_WAIT +	<p>Forces the current thread to wait for an event, identified in the user-supplier array, to change to the M_SINGALED state or for the events to timeout. The current thread will not continue until one of the specified events has changed to a signaled state or timed out.</p> <p>If the event that changed state is of the type that is reset automatically (M_AUTO_RESET), after the wait operation, the state of the event is reset to M_NOT_SINGALED. (summarize)</p>

Combination constant for any of the possible values of the [WaitOption](#) parameter

You can add the following value to the above-mentioned values to specify that the current thread must wait for all events to change state or for a timeout to occur.

● For specifying that the thread must wait for all the events to change state	
▢ Value	Description
▢ M_ALL_OBJECTS	Waits for all the events identified in the user-supplied array to be in a M_SIGNALED state or for the events to time out.

Combination constant for any of the possible values of the [WaitOption](#) parameter

You can add the following value to the above-mentioned values to set the time interval after which an event is considered to be timed out.

● For events		
▢ Value	Description	
▢ M_EVENT_TIMEOUT(MIL_INT EventTimeout)	Specifies the time interval after which an event is considered to be timed out. (summarize)	
	Parameters	
	EventTimeout	
	Sets the required timeout interval, in msec. Set this parameter to the following:	
	0	Same as M_INFINITE .
M_INFINITE	Specifies an infinitely long timeout interval.	
Value > 0	Sets the timeout interval to a user-defined value, in msec.	

State

Specifies the address in which to write the returned value. For [M_EVENT_WAIT](#), the returned value is the index of the event that changed to the [M_SIGNALED](#) state, or **M_TIMEOUT** if none of the events in the array have changed state before the time out interval is reached. For [M_EVENT_WAIT](#) + [M_ALL_OBJECTS](#), the returned value is [M_SIGNALED](#) if all the events changed state, or **M_TIMEOUT** if the time out interval is reached before all the events in the array have changed state. For [M_EVENT_SYNCHRONIZE](#), the returned value is **M_UNKNOWN**.

Since the **MthrWaitMultiple()** function also returns the requested information, you can set this parameter to **M_NULL**.

Return value

The returned value for [M_EVENT_WAIT](#) is the index of the event that changed to the [M_SIGNALED](#) state, or **M_TIMEOUT** if none of the events in the array changed state before the time out interval is reached. For [M_EVENT_WAIT](#) + [M_ALL_OBJECTS](#), the returned value is [M_SIGNALED](#) if all the events specified in the array changed state, or **M_TIMEOUT** if the time out interval is reached before all the events in the array have changed state. For [M_EVENT_SYNCHRONIZE](#), the returned value is **M_UNKNOWN**.

Compilation information

Header	Include mil.h.
Library	Use mil.lib.
DLL	Requires mil.dll.