

# **MIL / MIL-Lite**

version 7.1

## **Board-Specific Notes**

Manual no. 10515-801-0710

March 1, 2002

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

*Microsoft®, Window®, and Windows NT® are registered trademarks of Microsoft Corporation.*

*PC/104-Plus™ is a trademark of the PC/104 Consortium.*

*CompactPCI™ is a trademark of PCI Industrial Computer Manufacturers' Group.*

*Intel®, Pentium®, and Pentium II® are registered trademarks of Intel Corporation. Intel MMX™ Technology is a trademark of Intel Corporation.*

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

*© Copyright Matrox Electronic Systems Ltd., 2002. All rights reserved.*

*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 may 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.*

*PRINTED IN CANADA*

# Contents

---

## ***Chapter 1: The board-specific notes. . . . . 13***

Board-specific notes . . . . .	14
--------------------------------	----

---

## ***Chapter 2: MIL and the Matrox Corona-II platform. . . 15***

Matrox Corona-II-specific features. . . . .	16
Using Matrox Corona-II with MIL . . . . .	17
Grabbing from multiple cameras with different DCFs . . . . .	18
Using the encoder . . . . .	19
Particularities of existing MIL functions on Matrox Corona-II . . . . .	20
MbufAlloc...() . . . . .	21
MbufCreate...() . . . . .	21
MbufInquire() . . . . .	22
MdigAlloc() . . . . .	22
MdigChannel() . . . . .	23
MdigControl() . . . . .	24
MdigGrab()/MdigGrabContinuous(). . . . .	31
MdigHookFunction() . . . . .	32
MdigInquire() . . . . .	32
MdigLut() . . . . .	36
MdigReference() . . . . .	38
MdispAlloc() . . . . .	39
MdispControl() . . . . .	40
MdispInquire() . . . . .	41

MsysAlloc()	42
MsysControl()	42
MsysInquire()	44

---

## ***Chapter 3: MIL and the Matrox Genesis platform . . . . 45***

Matrox Genesis-specific features	46
Using Matrox Genesis with MIL	48
16-bit buffer simulated display.	48
Grabbing to a Host buffer with Matrox Genesis-LC	49
Optimizing the use of the frame buffers . . .	50
Increasing the default temporary buffer size	50
Grabbing in on-board buffers.	52
Other options	58
Particularities of existing MIL functions on Matrox Genesis	58
MblobFill()	59
MblobInquire()	59
MblobReconstruct()	59
MbufAlloc...()	60
MbufCopyCond(), MbufCopyMask()	60
MbufCreateColor()	61
MbufInquire()	61
MdigAlloc()	61
MdigControl()	62
MdigGrab()	68
MdigGrabWait()	68
MdigHookFunction()	68

MdigInquire()	.68
MdigLut()	.70
MdigReference()	.70
MdispAlloc()	.71
MdispControl()	.72
MdispInquire()	.73
MdispLut()	.73
MgenWarpParameter()	.73
MgraFontScale()	.74
MimConvolve()	.74
MimRank()	.74
MimResize()	.74
MimRotate()	.74
MimWarp()	.74
MpatInquire()	.74
MsysAlloc()	.74
MsysControl()	.75
MsysInquire()	.78
Processing operations performed by Host	.79

---

## ***Chapter 4: MIL and the Matrox Meteor-II platform . . . 85***

Matrox Meteor-II family . . . . .	.86
Types of cameras and supported data format. . . . .	.88
Using Matrox Meteor-II with MIL. . . . .	.89
Transfers to display. . . . .	.90
Grabbing from multiple cameras with different DCFs using Matrox Meteor-II /Multi-Channel. . . . .	.91

Grabbing to a Host buffer with Matrox Meteor-II /Digital or /Camera Link . . . . .	92
Optimizing the use of the frame buffers . . .	93
Compression and decompression with Matrox Meteor-II MJPEG module . . . . .	97
Compression . . . . .	97
Decompression . . . . .	98
IEEE 1394 serial bus-specific features . . . . .	99
Video formats and modes. . . . .	99
Read/write capabilities . . . . .	102
Setting camera features . . . . .	102
An example . . . . .	103
Grabbing from multiple 1394 DCAM-compliant cameras. . . . .	104
Important points to consider. . . . .	106
Particularities of existing MIL functions on Matrox Meteor-II . . . . .	107
MbufAlloc...() . . . . .	108
MbufCreate...(). . . . .	109
MbufInquire(). . . . .	110
MdigAlloc() . . . . .	111
MdigChannel() . . . . .	114
MdigControl(). . . . .	119
MdigGrab/MdigGrabContinuous() . . . . .	137
MdigGrabWait() . . . . .	138
MdigHookFunction() . . . . .	138
MdigInquire() . . . . .	138
MdigLut() . . . . .	147
MdigReference() . . . . .	147

MsysAlloc()	148
MsysControl()	149
MsysInquire()	152

---

## **Chapter 5: MIL and the Matrox Orion platform . . . . .155**

Matrox Orion-specific features.	156
Using Matrox Orion with MIL	158
Grabbing from multiple cameras with different DCFs.	158
Using the encoder	159
Particularities of existing MIL functions on Matrox Orion.	160
MbufAlloc...()	161
MbufCreate...()	161
MbufInquire()	161
MdigAlloc()	162
MdigChannel()	163
MdigControl()	166
MdigGrab()/MdigGrabContinuous()	169
MdigHookFunction()	170
MdigInquire()	170
MdigLut()	172
MdigReference()	172
MdispAlloc()	173
MdispControl()	174
MdispInquire()	175
MsysAlloc()	175
MsysControl()	175
MsysInquire()	176

---

**Chapter 6: MIL and the Matrox 4Sight and  
Matrox 4Sight-II units . . . . . 177**

Specific features of Matrox 4Sight and  
Matrox 4Sight-II units . . . . . 178

    Display capabilities . . . . . 178

    Frame grabbers . . . . . 179

    Input and output data interfaces . . . . . 180

Using MIL with Matrox 4Sight and Matrox  
4Sight-II. . . . . 182

Auxiliary outputs/inputs . . . . . 183

    Auxiliary outputs. . . . . 183

    Auxiliary inputs. . . . . 184

Particularities of existing MIL functions on  
Matrox 4Sight and Matrox 4Sight-II . . . . . 186

    MsysControl() . . . . . 187

    MsysInquire(). . . . . 188

Matrox 4Sight and Matrox 4Sight-II-specific  
MIL functions. . . . . 190

---

**Chapter 7: MIL and the Matrox Cronos platform. . . . 195**

Matrox Cronos-specific features . . . . . 196

Using Matrox Cronos with MIL. . . . . 196

Fast-channel switching . . . . . 197

Grabbing from multiple cameras with  
different DCFs . . . . . 198



Particularities of existing MIL functions on Matrox Cronos . . . . .	199
MbufAlloc...() . . . . .	200
MbufCreate...() . . . . .	200
MdigAlloc() . . . . .	201
MdigChannel() . . . . .	202
MdigControl() . . . . .	203
MdigGrab()/MdigGrabContinuous() . . . . .	208
MdigHookFunction() . . . . .	209
MdigInquire() . . . . .	210
MdigReference() . . . . .	212
MsysAlloc() . . . . .	212
MsysControl() . . . . .	212
MsysHookFunction() . . . . .	214
MsysInquire() . . . . .	215

---

***Chapter 8: MIL and the VGA platform. . . . . 217***

Particularities of existing MIL functions on the VGA system. . . . .	218
Mdig...() . . . . .	218
Additional method of displaying an image in a user-defined window . . . . .	219
VGA board-specific functions . . . . .	220

---

***Appendix A: Board flow diagrams . . . . . 231***

Matrox Corona-II . . . . .	232
Matrox Genesis . . . . .	233
Matrox Meteor-II . . . . .	236
Matrox Orion . . . . .	242
Matrox 4Sight . . . . .	243
Matrox 4Sight motherboard . . . . .	244
Matrox Cronos . . . . .	245

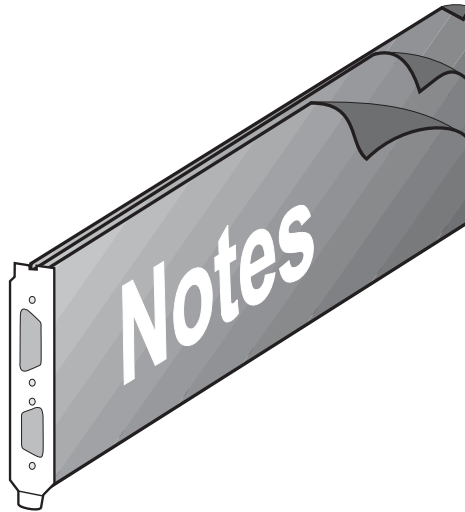
---

***Index***

---

***Product Support***

# *The MIL Board-Specific Notes*



Getting the best of your board with MIL....



---

## ***Chapter 1: The board-specific notes***

---

## Board-specific notes

The board-specific notes are presented as a different chapter for each platform. It is important to review the chapter related to your board before using MIL to understand how your board relates to MIL and its commands.

Each chapter includes general information about the board that might affect its use with MIL. It also lists specific information about any MIL command characteristics that function differently or offer different options with the board. A summary table of commands with board-specific information is provided for a quick overview. This is followed by an alphabetic presentation of individual commands and their board-specific characteristics. Related commands are grouped together because of their nomenclature. For example, all the data allocation and access module commands begin with the letters *Mbuf*.

---

### *Standards and usage*

The standards and usage in this manual are the same as those of the *MIL Command Reference* manual.

---

## ***Chapter 2: MIL and the Matrox Corona-II platform***

*This section discusses features of MIL that are distinct to the Matrox Corona-II platform and ways that optimize the board's performance.*

---

## Matrox Corona-II-specific features

Matrox Corona-II is a PCI frame grabber that is capable of switching between two component RGB, or up to six monochrome standard or non-standard area scan video sources. In addition, when using the optional companion digital-input board, Matrox Corona-II can acquire either RS-422 or LVDS digital video, depending on which companion board is purchased.

When grabbing monochrome analog data, Matrox Corona-II can acquire 8 or 10-bit monochrome data.

Matrox Corona-II has an integrated RGB-to-YUV color space converter that converts 24-bit RGB video data to YUV16.

Matrox Corona-II has a configurable lookup table (LUT). For analog and digital data, the LUT can be configured as triple 256 x 8-bit or a dual 1024 x 16-bit LUT. This allows you to map up to 3 x 8-bit or 2 x 10-bit input through the LUT. For all other inputs (12-, 14-, or 16-bit digital data), the LUT is not available.

---

### *Display features*

With its MGA G400 graphics controller and 32-Mbyte frame buffer, Matrox Corona-II can deliver a true color (32-bit) image display with a 32-bit color overlay, for a completely true-color display at up to 1280x1024 resolution; the maximum refresh rate is 75 Hz, depending on your monitor. Matrox Corona-II can allocate an overlay frame buffer in 8-bit monochrome format or 32-bit RGB format (BGR32 packed), thereby supporting either of these display resolution formats. In addition, Matrox Corona-II features non-destructive graphics overlay on live video, and support for video-in-a-window with arbitrary video scaling (up or down).

Since the display section is controlled by the MGA G400 chip, Matrox Corona-II features DualHead display technology, which provides the ability to simultaneously have an independent analog VGA output and DVI compliant digital VGA, or an analog VGA output and encoder output to devices such as an NTSC/PAL/RGB TV or VCR.

❖ Note that if you use the optional piggyback DVI flat-panel module, the video encoder cannot be used.



- ❖ Note that DirectDraw is not supported under NT 4.0 on an extended desktop. Without DirectDraw, windowed displays will be CPU-assisted.

---

#### UART

Matrox Corona-II also features an on-board UART (Universal Asynchronous Receiver/Transmitter) that provides an RS-232 serial interface. This allows you to remotely control a camera (gain, gamma, control, and operation mode) or a motion control unit, or communicate with a program logic controller (PLC).

---

#### Hardware fingerprint

Matrox Corona-II can be used as a hardware fingerprint for run-time licenses.

See Appendix A for a data flow diagram.

---

## Using Matrox Corona-II with MIL

To use your Matrox Corona-II board with MIL, you must allocate a Corona-II system (`M_SYSTEM_CORONA_II`), using ***MsysAlloc()***. This establishes a MIL system environment in which MIL uses some Host memory, and any available graphics controller for grabbing and displaying images.

This chapter relates Corona-II systems to the following:

*Grabbing from multiple cameras with different DCFs, Using the encoder and Particularities of existing MIL functions on Matrox Corona-II.*

Refer to the *milcor\_II.txt* file in the `\MATROX IMAGING\DRIVERS\DOC` (or user-specified) directory for any additions/modifications to these board specific notes.

## Grabbing from multiple cameras with different DCFs

In some applications, you might want to grab from multiple cameras with different DCFs that are attached to the same digitizer. This procedure normally involves allocating a digitizer, grabbing the required frame, freeing the digitizer, and then allocating the digitizer again with the second DCF; this can increase the time required for operation. On Matrox Corona-II, MIL can circumvent this problem by using a fast DCF-switching technique which is outlined in the steps below:

1. Make as many calls to **MdigAlloc()** as you have cameras, with different formats, from which you want to grab. With each call, the **DigNum** parameter must be set to **M\_DEV0**, since there is only one physical digitizer; each allocation only sets up a virtual instance of the digitizer.
  2. Specify all required digitizer settings using **MdigControl()**, **MdigChannel()**, **MdigReference()**, and **MdigHookFunction()** for each allocated digitizer. Each time a different digitizer is specified, it retains all settings previously specified.
  3. Call **MdigGrab()** with any digitizer identifier. If this call uses a digitizer identifier different from the previous call, a camera switch will occur.
- ❖ If there is a grab in progress on one digitizer, calling any of the following functions with any other digitizer will result in an error: **MdigGrab()**, **MdigGrabContinuous()**, **MdigChannel()**, **MdigControl()**, **MdigInquire()**, **MdigReference()**, **MdigLut()**, **MdigHookFunction()**, **MdigAlloc()**, and **MdigFree()**.

See the example *mdigmultformat.c*.

---

## Using the encoder

Matrox Corona-II's display section features an on-board encoder.

The video encoder can be programmed to output composite (CVBS) and component (Y/C) video in NTSC/PAL formats. It can also output component RGB video with the same resolution and refresh rate as video in NTSC/PAL formats. To output from the encoder, use an auxiliary display with an encoder-supported output format. On Matrox Corona-II, the overlay buffer associated with such an auxiliary display will be driven by the graphics controller.

Note that during a live grab, the display is non-tearing.

## Particularities of existing MIL functions on Matrox Corona-II

Certain commands have special features or functionality on a Corona-II system. This table provides an overview of the affected commands. Details and procedures are described in the individual command descriptions, which follow.

Commands	Corona-II particularities
MbufAlloc...()	Buffer constraints and options.
MbufCreate...()	Required parameter settings.
MbufInquire()	Corona-II-specific inquire options.
MdigAlloc()	Digitizer configuration format (DCF) specifications.
MdigChannel()	Required parameter settings.
MdigControl()	Control type and value additions and restrictions.
MdigGrab/ MdigGrabContinuous()	Destination buffer restriction.
MdigHookFunction()	Hook restrictions.
MdigInquire()	Corona-II-specific inquire options.
MdigLut()	Particularities of look-up tables.
MdigReference()	Corona-II features.
MdispAlloc()	Display format. Overlay setting.
MdispControl()	Required parameter settings.
MdispInquire()	Additional inquiries.
MsysAlloc()	Corona-II-specific allocation options.
MsysControl()	Control type and value additions.
MsysInquire()	Corona-II-specific inquire options.

## MbufAlloc...()

- 8-bit monochrome, 16-bit monochrome, and 3-band 8-bit color buffers can have an M\_GRAB attribute.
- You can add one of the following to the **Attribute** parameter:

Attribute parameter for 3-band 8-bit buffer	Internal storage format for 3-band 8-bit buffer
M_BGR24 + M_PACKED	24-bit BGR packed pixels. (Default for M_DISP buffers.)
M_RGB24 + M_PLANAR	24-bit planar RGB pixels. (Default for non M_DISP buffers.)
M_BGR32 + M_PACKED	32-bit BGR packed pixels.
M_YUV16 + M_PACKED	YUV16 (4:2:2) packed standard.
M_YUV16_YUYV + M_PACKED	YUV16 (4:2:2) packed standard.

Also, note that it might be slower to process buffers that have an M\_PACKED attribute or are in YUV format.

## MbufCreate...()

- Set the **Attribute** parameter to an M\_IMAGE combination (for example, M\_IMAGE + M\_DISP + M\_GRAB + M\_PROC).
- Only the following **ControlFlag** combinations are supported with the M\_GRAB attribute:

M_PHYSICAL_ADDRESS + M_PITCH	<b>ArrayOfDataPtr</b> is an array of physical addresses. The pitch is in pixels (default).
M_PHYSICAL_ADDRESS + M_PITCH_BYTE	<b>ArrayOfDataPtr</b> is an array of physical addresses. The pitch is in bytes.

- The **Attribute** parameter can be set to the same settings as *MbufAlloc...()*.

### MbufInquire()

- The **InquireType** parameter can also be set to:

M_CURRENT_BUF_ID	Identifier of the buffer in which data is currently being grabbed. Valid only for windowed displays. This information is used, for example, to access grabbed data while a continuous grab ( <b>MdigGrabContinuous()</b> ) is being displayed live. Under these circumstances, the destination buffer is not the allocated buffer, but rather an associated buffer, used only for display.
------------------	---

### MdigAlloc()

- The **DataFormat** parameter specifies the DCF of the input device.

The predefined settings for monochrome cameras are:

"M_RS170" "M_RS170_VIA_RGB"	RS-170, 640x480, 8 bits, 12.5MHz, analog.
"M_CCIR" "M_CCIR_VIA_RGB"	CCIR, 768x576, 8 bits, 14.8MHz, analog.
"M_DEFAULT"	Same as M_RS170.

The predefined settings for color cameras are:

"M_NTSC" "M_NTSC_RGB"	RS-170 RGB, 640x480, 3x8 bits, 12.5 MHz
"M_PAL" "M_PAL_RGB"	CCIR RGB, 768x576, 3x8 bits, 14.8 MHz

You can also set the **DataFormat** parameter to the name of a DCF file (\*.*dcf*). This file specifies the input signal format. See the `\MATROX IMAGING\DRIVERS` (or user-specified) directory for the current list of supported formats.

- You can allocate multiple digitizers on one Corona-II system. Make as many calls to **MdigAlloc()** as you have cameras from which you want to grab. With each call, specify the DCF of the camera, and set the **DigNum** parameter to M\_DEV0. Calling **MdigGrab()** with any digitizer identifier will cause

a switch and a grab from the identifier's associated camera. See the section, *Grabbing from multiple cameras with different DCFs* for more information.

## MdigChannel()

- Up to six independent monochrome, two RGB cameras, or two dual-tap cameras can be attached to the Matrox Corona-II board.
- To switch between cameras of a similar type, use:

Camera Type	Channel	Signal/Sync input
Any	M_DEFAULT	Same as M_CH0.
RGB	M_CH0	VID1_IN1, VID1_IN2, VID1_IN3 (data signals)
	M_CH1	VID2_IN1, VID2_IN2, VID2_IN3 (data signals)
	M_RGB	VID1_IN1, VID1_IN2, VID1_IN3 (data signals) and SYNC_IN (sync signal)
Monochrome	M_CH0	VID1_IN1
	M_CH1	VID1_IN2
	M_CH2	VID1_IN3
	M_CH4	VID2_IN1
	M_CH5	VID2_IN2
	M_CH6	VID2_IN3
Monochrome dual-tap	M_CH0	VID1_IN1, VID1_IN2
	M_CH1	VID2_IN1, VID2_IN2

- The default channel of the sync signal can be overridden by adding M\_SYNC to any of the following channel parameters:

Sync channel	Input signal
M_CH0	VID_IN1
M_CH1	VID_IN2

Sync channel	Input signal
M_CH2	VID_IN3
M_CH3	SYNC_IN
M_CH4	VID2_IN1
M_CH5	VID2_IN2
M_CH6	VID2_IN3
M_CH7	SYNC_IN

### MdigControl()

- It is not possible to queue asynchronous grabs on Matrox Corona-II. Therefore, M\_ASYNCHRONOUS\_QUEUED cannot be used as a control value with the M\_GRAB\_MODE control type.
- When using interlaced cameras, the default setting for the M\_GRAB\_START\_MODE control type is M\_FIELD\_START\_ODD.
- For the M\_GRAB\_TRIGGER\_SOURCE and M\_GRAB\_EXPOSURE\_SOURCE control types, note the meaning of the following control values:
  - M\_HARDWARE\_PORT0: Opto-isolated hardware trigger signal using a combination of Pin 34 (OPTOTRIG-) and Pin 35 (OPTOTRIG+) on Video Input connector.
  - M\_HARDWARE\_PORT1: TTL hardware trigger signal using Pin 20 on Video Input connector or Pin 2 (TRIGGER) on Digital Interface connector.
  - M\_HARDWARE\_PORT2: RS-422/LVDS trigger signal using a combination of Pin 73 (TRIG+) and Pin 74 (TRIG-) on the video input connector on the companion digital-input board.
- For the M\_GRAB\_EXPOSURE\_CLOCK\_SOURCE control type, note the following:
  - A clock source must have a frequency greater than or equal to 1 Hz.



- Only a continuous timer can clock another timer. To clock a timer, make the following call:

```
MdigControl(MilDigitizer, M_GRAB_EXPOSURE_CLOCK_SOURCE+M_TIMERn, M_CONTINUOUS)
```

Note that M\_TIMERm and M\_TIMERn can be equal to either M\_TIMER1 or M\_TIMER2. However, M\_TIMERn and M\_TIMERm cannot be equal.

- Circular references with timers are not supported and will generate an error.
- An input filter can be specified for each analog channel using the M\_INPUT\_FILTER control type:

ControlType	Description & ControlValue	
M_INPUT_FILTER	M_LOW_PASS_0	Low-pass filter at 8 Mhz.
	M_LOW_PASS_1	Low-pass filter at 10 Mhz.
	M_LOW_PASS_2	Low-pass filter at 12.5 Mhz.
	M_BYPASS	No input filter used.
	M_DEFAULT	Same as M_LOW_PASS_2+M_ALL_CHANNELS.

- The settings for M\_INPUT\_FILTER can be combined with any of the following to specify the filter on a specific channel. If you do not specify a channel, all channels are assumed.

M_CH0	Specifies the filter for the first A/D converter. Specifying this filter affects the cameras that provide the VID_IN1 and VID_IN4 signals.
M_CH1	Specifies the filter for the second A/D converter. Specifying this filter affects the cameras that provide the VID_IN2 and VID_IN5 signals.
M_CH2	Specifies the filter for the third A/D converter. Specifying this filter affects the cameras that provide the VID_IN3 and VID_IN6 signals.
M_ALL_CHANNELS	Specifies the filter for all A/D converters (default).

- Corona-II supports these additional **ControlType** parameter settings:

ControlType	Description & ControlValue		
M_GRAB_ABORT	Immediately stops a grabs in progress and queued grabs. Useful for cancelling a triggered grab that is waiting for the trigger.		
	M_DEFAULT	Stops the grab.	
M_GRAB_INPUT_GAIN	Set the gain applied to the input signal. Valid input voltages and their corresponding gains are:		
		Input voltage	Gain
	M_GAIN4	2.1 - 2.9Vpp.	1
	M_GAIN0	1.4 - 2.0Vpp.	1.3
	M_GAIN1	1.0 - 1.4Vpp.	2
	M_GAIN2	0.7 - 1.0Vpp.	2.8
	M_GAIN3	0.0 - 0.7Vpp.	4
	M_DEFAULT	Same as M_GAIN2.	Same as M_GAIN2.
M_GRAB_SCALE	Scaling factors: 1, 1/2, 1/3,..., 1/16. When grabbing into a YUV buffer, only the following scaling factors are supported: 1 and 1/2.		
M_GRAB_SCALE_X	Scaling factors for the X-direction: 1, 1/2, 1/3,..., 1/16. When grabbing into a YUV buffer, only the following scaling factors are supported: 1 and 1/2.		
M_GRAB_SCALE_Y	Scaling factors for the Y-direction: 1, 1/2, 1/3,..., 1/16. When grabbing into a YUV buffer, only the following scaling factors are supported: 1 and 1/2.		
M_GRAB_EXPOSURE_BYPASS	Activate the manual or automatic exposure model.		
	M_ENABLE	Manual exposure model M_DISABLE is not supported.	

ControlType	Description & ControlValue	
For the following M_GRAB_EXPOSURE... control types, you can add M_TIMER1 or M_TIMER2 in manual exposure mode, to control the different on-board exposure timers. When omitted, Timer1 is assumed.		
M_GRAB_EXPOSURE	When using a software trigger source, use this control type to activate the specified grab exposure timer. When using a non-software trigger source, enable or disable the specified grab exposure timer. Note, the M_GRAB_EXPOSURE control type has no effect when grabbing using the automatic exposure model.	
	M_ACTIVATE	Activate a software trigger for the specified exposure timer.
	M_ENABLE	Enable exposure timer.
	M_DISABLE	Disable exposure timer.
	M_DEFAULT	same as .dcf. (non-software trigger source)
M_GRAB_EXPOSURE_CLOCK_SOURCE		
	Specifies the clock source that drives the exposure counter.	
	M_SYSCLK	Use a 25 MHz clock source frequency.
	M_PIXCLK	Use the pixel clock frequency (dependent on your camera).
	M_HSYNC	Use the horizontal sync frequency.
	M_TIMER1	Use the frequency of M_TIMER1, if continuous.
	M_TIMER2	Use the frequency of M_TIMER2, if continuous.
	M_DEFAULT	Use the most appropriate clock source (as determined by the driver).
M_GRAB_EXPOSURE_MODE	Sets the exposure signal’s polarity: M_LEVEL_HIGH, M_LEVEL_LOW, or M_DEFAULT (use the DCF’s polarity).	

ControlType	Description & ControlValue											
M_GRAB_EXPOSURE_TIME	<p>Set the time (in nsec) for the active portion of the exposure signal (that is, the exposure time). M_DEFAULT has the same effect as the setting in the digitizer's DCF.</p> <p>When using the automatic exposure model, if a single timer cannot generate the required exposure time, MIL automatically sets up connections with the second timer to generate the requested exposure time length. If <b>ControlValue</b> is set to 0, exposure is disabled and the grab is performed immediately. Note, an error is returned if the specified exposure time cannot be generated.</p>											
M_GRAB_EXPOSURE_TIME_DELAY	<p>Set the delay (in nsec) between the trigger and the start of exposure. If M_DEFAULT, same value as DCF.</p> <p>Note, an error is returned if the specified delay cannot be generated.</p>											
M_GRAB_EXPOSURE_TRIGGER_MODE	<table><tr><td></td><td>Set the trigger activation mode for specified timer.</td></tr><tr><td>M_DEFAULT</td><td>Same as the .dcf file.</td></tr><tr><td>M_EDGE_RISING</td><td>Low to high signal variation.</td></tr><tr><td>M_EDGE_FALLING</td><td>High to low signal variation.</td></tr></table>			Set the trigger activation mode for specified timer.	M_DEFAULT	Same as the .dcf file.	M_EDGE_RISING	Low to high signal variation.	M_EDGE_FALLING	High to low signal variation.		
	Set the trigger activation mode for specified timer.											
M_DEFAULT	Same as the .dcf file.											
M_EDGE_RISING	Low to high signal variation.											
M_EDGE_FALLING	High to low signal variation.											
M_GRAB_WINDOW_RANGE	<p>Limit the range of the grabbed pixel values to between 10 and 245 when grabbing into a 1-band buffer: M_ENABLE or M_DISABLE.</p>											
M_UART_PARITY	<table><tr><td></td><td>Add a data bit (0 or 1) to the character data that is sent or received by the UART as a means of error checking.</td></tr><tr><td>M_DEFAULT</td><td>Same as M_DISABLE.</td></tr><tr><td>M_ODD</td><td>The number of 1's will be odd.</td></tr><tr><td>M_EVEN</td><td>The number of 1's will be even.</td></tr><tr><td>M_DISABLE</td><td>No extra bit is added (no parity).</td></tr></table>			Add a data bit (0 or 1) to the character data that is sent or received by the UART as a means of error checking.	M_DEFAULT	Same as M_DISABLE.	M_ODD	The number of 1's will be odd.	M_EVEN	The number of 1's will be even.	M_DISABLE	No extra bit is added (no parity).
	Add a data bit (0 or 1) to the character data that is sent or received by the UART as a means of error checking.											
M_DEFAULT	Same as M_DISABLE.											
M_ODD	The number of 1's will be odd.											
M_EVEN	The number of 1's will be even.											
M_DISABLE	No extra bit is added (no parity).											
M_UART_STOP_BITS	<table><tr><td></td><td>Add a data bit to signal the end of character data being sent or received.</td></tr><tr><td>1</td><td>1 stop bit will be added.</td></tr><tr><td>2</td><td>2 stop bits will be added</td></tr><tr><td>M_DEFAULT</td><td>1 stop bit will be added.</td></tr></table>			Add a data bit to signal the end of character data being sent or received.	1	1 stop bit will be added.	2	2 stop bits will be added	M_DEFAULT	1 stop bit will be added.		
	Add a data bit to signal the end of character data being sent or received.											
1	1 stop bit will be added.											
2	2 stop bits will be added											
M_DEFAULT	1 stop bit will be added.											

ControlType	Description & ControlValue	
M_UART_DATA_LENGTH	The number of data bits per character that are sent or received by the UART. Valid values are 7 or 8 bits.	
	7	Data length is 7 bits.
	8	Data length is 8 bits.
	M_DEFAULT	Data length is 8 bits.
M_UART_SPEED	Change the baud rate of the UART. Valid values are 230400, 115200, 76800, 57600, 38400, 28800, 19200, 14400, 9600, 7200, 4800, 3600, 2400, 1800, 1200, 600.	
	M_DEFAULT	Default speed is 14400.
M_UART_TIMEOUT	Set the maximum time to wait between each byte when reading incoming data.	
	M_INFINITE	Wait indefinitely.
	M_DEFAULT	Same as M_INFINITE.
	value in msec	Specify time to wait.
M_UART_WRITE_CHAR	Send one character to the UART for transmission. The <b>ControlValue</b> must be a pointer to the character.	
M_UART_READ_CHAR	Read one character from the UART input buffer. The <b>ControlValue</b> must be a pointer in which to save the character. 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.	
M_UART_STRING_DELIMITER	Set the character used to terminate strings of incoming or outgoing data. The delimiter is used but not sent when writing data; it is read for incoming data.	
	M_DEFAULT	The '\0' character.
M_UART_READ_STRING_LENGTH	Set the length of the string (in bytes) to be read by M_UART_READ_STRING.	
	M_DEFAULT	Used to specify the use of M_STRING_DELIMITER to end string.
	Value in bytes	Specify string length.

ControlType		Description & ControlValue	
M_UART_READ_STRING_MAXIMUM_LENGTH		Use this value to specify the size of your read buffer to prevent global protection faults from happening when using the M_UART_READ_STRING control.	
M_UART_READ_STRING		Read a string of incoming data from the UART. The <b>ControlValue</b> must be a pointer to a character array. The size of this array must be set with the M_UART_READ_STRING_MAXIMUM_LENGTH control type. 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.	
M_UART_WRITE_STRING_LENGTH		Sets the length of the string to be sent to the UART for transmission.	
		M_DEFAULT	Used to specify the use of M_STRING_DELIMITER to end string. The delimiter will not be sent through the UART.
		Value in bytes	Specify string length.
M_UART_WRITE_STRING		Send the string of data, specified by the control value, through the UART. Set <b>ControlValue</b> to the character array. The number of characters to send can be specified with M_UART_WRITE_STRING_LENGTH or M_UART_STRING_DELIMITER.	

ControlType	Description & ControlValue	
M_USER_BIT+(3, 4, 5, 6, 9, 10)	Set the state of an output bit of the Video Input connector: M_ON or M_OFF The relationship between the MIL user-bit number and the actual user output bit on the Video Input connector is as follows:	
	User Bit#	Video Input connector signal
	3	USER1OUT signal.
	4	USER2OUT signal.
	5	RS-422 USEROUT0
	6	RS-422 USEROUT1
	9	USER3OUT signal.
	10	USER4OUT signal.
	User-bits 5 and 6 are only supported on the Matrox Corona-II companion digital-input board.	

### MdigGrab()/MdigGrabContinuous()

- It is not possible to grab into a color-band child buffer when the buffer is packed.
- When performing a grab, the width and the horizontal position of the grab destination buffer must be a multiple of 4 bytes. If not, the grab will be clipped accordingly.
- When grabbing from a RGB source into a monochrome buffer, Matrox Corona-II's color space converter first converts the RGB data into YUV, and then stores the Y (luminance) component in the buffer. Data is passed through the input LUTs before the color space converter, therefore the LUTs are applied to the incoming data.
- When performing real-time grabs, Windows 98 and Windows Me do not provide the same performance and consistency as Windows NT/2000. This generally does not cause difficulty except in applications that require very fast response time to a hardware event. For example, an application that must grab sequences of images without missing any frames might not have sufficient time for the

proper response to a hook function. Therefore, if grabbing a sequence, you should try to avoid code that needs an immediate response and favor code that supports a short delay after the event. For example, it is recommended to call the next grab operation (**MdigGrab()**) from the function hooked to the start of the current grab, instead of calling it from a function hooked to the end of the grab. This will queue the next grab operation while the current grab is in progress, and will allow the next grab to start immediately after the current grab. This should prevent the loss of a frame due to delay.

If these restrictions affect a major part of your application(s), we recommend the use of Windows NT/2000.

## MdigHookFunction()

- All settings for the **HookType** parameter are supported.
- A function hooked to an M\_GRAB\_END event is called when all the data of the grab is transferred to the Host. This means that this hooked function can be called after the M\_GRAB\_START or M\_GRAB\_FRAME\_START event of the next frame.
- The **HookType** parameter can also be set to M\_UART\_DATA\_RECEIVED. The function hooked to this event will be called when data is received by the UART.

## MdigInquire()

- The **InquireType** parameter can also be set to the following:

InquireType	Description
M_COLOR_MODE	Digitizer color mode: M_RGB or M_MONO8_VIA_RGB.
M_GRAB_INPUT_GAIN	The gain that is applied to input signal: M_GAIN0, M_GAIN1, M_GAIN2, M_GAIN3, or GAIN4.
M_GRAB_SCALE	Scaling factors: 1, 1/2, 1/3,..., 1/16. When grabbing into a YUV buffer: 1, 1/2.
M_GRAB_SCALE_X	Horizontal scaling factors: 1, 1/2, 1/3,..., 1/16. When grabbing into a YUV buffer: 1, 1/2.



InquireType	Description
M_GRAB_SCALE_Y	Vertical scaling factors: 1, 1/2, 1/3,..., 1/16. When grabbing into a YUV buffer: 1, 1/2.
M_INPUT_FILTER	Input filter on M_CH_0 by default. When combined with M_CH_0, M_CH_1, or M_CH_2 returns the input filter for the specified channel: M_FILTER0, M_FILTER1, M_FILTER2, M_FILTER_BYPASS
M_INPUT_SIGNAL_PRESENT	Video input signal present: M_YES or M_NO.
M_GRAB_EXPOSURE_BYPASS	Exposure model: M_ENABLE (manual). M_DISABLE is not supported.
For the following M_GRAB_EXPOSURE... inquire types, you can add M_TIMER1 or M_TIMER2 in manual exposure mode, to inquire about the different on-board exposure timers. When omitted, Timer1 is assumed.	
M_GRAB_EXPOSURE	Exposure timer state for non-software trigger source: M_ENABLE or M_DISABLE.
M_GRAB_EXPOSURE_CLOCK_FREQUENCY	The frequency in hertz of the clock that is used to generate the exposure signal.
M_GRAB_EXPOSURE_CLOCK_SOURCE	The clock source that drives the exposure counter: M_SYSCLK, M_PIXCLK, M_HSYNC, M_TIMER1, M_TIMER2.
M_GRAB_EXPOSURE_MODE	Exposure signal's polarity: M_LEVEL_HIGH or M_LEVEL_LOW.
M_GRAB_EXPOSURE_TIME	Time for the active portion of the exposure signal (value in nsec). Returned as a double.
M_GRAB_EXPOSURE_TIME_DELAY	Delay (in nsec) between the trigger and the start of exposure. Returned as a double.
M_GRAB_EXPOSURE_TRIGGER_MODE	Trigger activation mode for specified timer: M_EDGE_RISING or M_EDGE_FALLING.
M_FIELD_START_ODD_HANDLER_PTR	The address of the function that is hooked to the M_FIELD_START_ODD event.
M_FIELD_START_ODD_HANDLER_USER_PTR	The address of the user data you want to make available to the function hooked to the M_FIELD_START_ODD event.

InquireType	Description
M_FIELD_START_EVEN_HANDLER_PTR	The address of the function that is hooked to the M_FIELD_START_EVEN event.
M_FIELD_START_EVEN_HANDLER_USER_PTR	The address of the user data you want to make available to the function hooked to the M_FIELD_START_EVEN event.
M_FIELD_START_HANDLER_PTR	The address of the function that is hooked to the M_FIELD_START event.
M_FIELD_START_HANDLER_USER_PTR	The address of the user data you want to make available to the function hooked to the M_FIELD_START event.
M_FRAME_START_HANDLER_PTR	The address of the function that is hooked to the M_FRAME_START_ODD event.
M_FRAME_START_HANDLER_USER_PTR	The address of the user data you want to make available to the function hooked to the M_FRAME_START event.
M_GRAB_IN_PROGRESS	Current grab state: M_YES or M_NO.
M_GRAB_WINDOW_RANGE	State of limiting the range of the grabbed pixel values: M_ENABLE or M_DISABLE.
M_INPUT_SIGNAL_SOURCE	Input-signal source: M_HARDWARE_PORT0 (Video Input connector) or M_HARDWARE_PORT1 (digital port).
M_HOOK_MASTER_THREAD_HANDLE	Returns the handle of the main interrupt thread that dispatches the handling of the grab hooks to their respective threads.
M_HOOK_MASTER_THREAD_ID	Returns the ID of the main interrupt thread that dispatches the handling of the grab hooks to their respective threads.
M_UART_DATA_LENGTH	Current data length in UART configuration: 7 or 8.
M_UART_DATA_PENDING	Indicates the input buffer has some pending data: M_TRUE, M_FALSE.
M_UART_PARITY	Current UART parity setting: M_ODD, M_EVEN, M_DISABLE.

InquireType	Description
M_UART_READ_STRING_LENGTH	Current number of bytes to be read when the M_UART_READ_STRING control is called. Can be M_DEFAULT if the send length is specified by M_UART_STRING_DELIMITER.
M_UART_READ_STRING_MAXIMUM_LENGTH	Current maximum size of the reading buffer
M_UART_SPEED	Current baud rate in UART configuration: 230400, 115200, 76800, 57600, 38400, 28800, 19200, 14400, 9600, 7200, 4800, 3600, 2400, 1800, 1200, 600.
M_UART_STOP_BITS	Current number of stop bits in UART configuration: 1 or 2.
M_UART_STRING_DELIMITER	Current character used to delimit strings if M_UART_READ_STRING_LENGTH or M_UART_WRITE_STRING_LENGTH is set to M_DEFAULT.
M_UART_THREAD_HANDLE	Current UART thread handle.
M_UART_THREAD_ID	Current UART thread ID.
M_UART_TIMEOUT	Current maximum time in milliseconds to wait when reading a byte of incoming data. Can be M_INFINITE.
M_UART_WRITE_STRING_LENGTH	Current number of bytes to be sent when the M_UART_WRITE_STRING control is called. Can be M_DEFAULT if the send length is specified by M_UART_STRING_DELIMITER.

InquireType	Description	
M_USER_BIT+ (1,2,3,4,5,6,7,8,9,10)	Current state of an input/output bit of the Video Input connector: M_ON (1) or M_OFF (0).	
	User Bit #	Video Input connector signal
	1	USER1IN signal.
	2	USER2IN signal.
	3	USER1OUT signal.
	4	USER2OUT signal.
	5	RS-422 USEROUT0
	6	RS-422 USEROUT1
	7	RS-422 USERIN0
	8	RS-422 USERIN1
	9	USER3OUT signal.
	10	USER4OUT signal.
	User-bits 5-8 are only supported on the Matrox Corona-II companion digital-input board.	

MdigLut()

The specified LUT buffer allocated with **MbufAlloc...()** must satisfy the conditions of input data depth, mode, and depth of the destination image buffer. Only the combinations listed below are supported; other combinations are invalid and will produce an error. Note that LUTs are not available for 12-, 14-, or 16-bit digital data.

Input data	Size of the LUT buffer	Image buffer	Notes
8-bit analog or digital  8-bit dual-tap analog or digital	256 x 8-bit	8-bit or YUV16	This is a true 8-bit LUT.
		16-bit	Not supported.
		3 x 8-bit	An 8-bit LUT with the data replicated for each band.
	256 x 16-bit	Not supported	
8-bit analog or digital  8-bit dual-tap analog or digital (cont.)	three 256 x 8-bit	8-bit or YUV16	The single-band data will pass through the 3-band LUT, then through the color-space converter to convert the data to YUV format. When the image buffer is 8-bit, the Y (luminance) component is passed to the monochrome buffer.
		16-bit	Not supported.
		3 x 8-bit	This will achieve a pseudo-color effect.
10-bit analog or digital  10-bit dual-tap analog or digital	1024 x 8-bit	*8-bit	All 1024 entries are mapped to the 8-bit image buffer.
		16-bit	Not supported.
		*3 x 8-bit	Same as the 8-bit case and the data is replicated for each band.
	1024 x 16-bit	*8-bit	The 8 most-significant bits of the maximum range of the 16-bit LUT are used.
		16-bit	True 16-bit LUT.
		*3 x 8-bit	Same as the 8-bit case and the data is replicated for each band.
*When the target buffer is 8-bit, the 8 most-significant bits of the 10-bit data stream are used.			

Input data	Size of the LUT buffer	Image buffer	Notes
3 x 8-bit analog or digital (color)	256 x 8-bit (the three LUTs will contain the same data)  three 256 x 8-bit	8-bit or YUV16	The 3-band data will pass through the 3-band LUT and then through the color-space converter to convert the image to YUV format. When the image buffer is 8-bit, the Y (luminance) component is passed to the monochrome buffer.
		16-bit	Is not supported.
		3 x 8-bit	True 8-bit color LUT.

## MdigReference()

- When grabbing, you can control the black and white reference levels of each input channel separately (M\_BLACK\_REF, M\_WHITE\_REF). To specify the channel, combine M\_CH0\_REF, M\_CH1\_REF, or M\_CH2\_REF with M\_BLACK\_REF or M\_WHITE\_REF.
- For M\_BLACK\_REF, note the meaning of the **ReferenceLevel** parameter settings:
  - The minimum voltage level (M\_MIN\_LEVEL) is 0.6V.
  - The maximum voltage level (M\_MAX\_LEVEL) is 1.6V.
- For M\_WHITE\_REF, note the meaning of the **ReferenceLevel** parameter settings:
  - The minimum voltage level (M\_MIN\_LEVEL) is 1.6V.
  - The maximum voltage level (M\_MAX\_LEVEL) is 2.6V.

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).

## MdispAlloc()

The current list of Video Configuration Format (VCF) files can be found in the *MATROX IMAGING\DRIVERS\VGA\VCF* directory. The maximum resolution is 1280 x 1024 at 75Hz.

For auxiliary displays, these are the display formats (for encoded video) that you can use to display from the Matrox Corona-II board:

DispFormat	Description
"M_NTSC"*	Encoder enabled in composite format with colorburst.
"M_NTSC_RGB"	Encoder enabled in RGB format with sync on green.
"M_NTSC_YC"*	Encoder enabled in Y/C formats with colorburst.
"M_RS170"	Encoder enabled in composite format without colorburst.
"M_PAL"*	Encoder enabled in composite format with colorburst.
"M_PAL_RGB"	Encoder enabled in RGB format with sync on green.
"M_PAL_YC"*	Encoder enabled in Y/C formats with colorburst.
"M_CCIR"	Encoder enabled in composite format without colorburst.
*Selecting these display formats will produce both composite and Y/C outputs.	

For auxiliary displays, the default ("M\_DEFAULT") display format is 1024 x 768 x 32bpp at 60Hz.

## MdispControl()

When using the encoder, the **ControlType** parameter can be set to one of the following:

ControlType	Description & ControlValue	
M_ENCODER_FILTER	Select a filter for luminance.	
	M_LOW_PASS_0	Low-pass filter type A.
	M_LOW_PASS_1	Low-pass filter type B.
	M_LOW_PASS_2	5.5 MHz low-pass filter.
	M_NOTCH	Subcarrier frequency low-pass filter.
	M_DEFAULT	Same as M_LOW_PASS_0.
M_ENCODER_PEDESTAL	Specify whether a pedestal is to be generated in the composite video signal.	
	M_ENABLE	Generate a pedestal in the composite video signal.
	M_DISABLE	Do not generated a pedestal in the output.
	M_DEFAULT	Same as M_ENABLE
M_SYNC_TYPE	Specifies the output sync signal. This control is only supported when the display format is RGB.	
	M_GREEN	The output sync signal is on green.
	M_SEPARATE	The sync and data signals are separated: the output sync is on pin 12, and the composite data signal is on pin 10 of the HD-44 connector.
	M_DISABLE	The encoder output does not contain a sync.
	M_DEFAULT	Same as M_GREEN.



## MdisplInquire()

- The **InquireType** parameter can also be set to one of the following:

InquireType	Description
M_VGA_PIXEL_FORMAT	The display's pixel format:
	M_MONO8 + M_PLANAR
	M_RGB15 + M_PACKED
	M_RGB16 + M_PACKED
	M_RGB24 + M_PACKED
	M_BGR24 + M_PACKED
	M_RGB32 + M_PACKED
	M_BGR32 + M_PACKED

- When using the encoder, the **InquireType** parameter can be set to one of the following:

InquireType	Description
M_ENCODER_FILTER	Returns the filter selected for luminance. M_LOW_PASS_0, M_LOW_PASS_1, M_LOW_PASS_2, or M_NOTCH.
M_ENCODER_PEDESTAL	Returns whether a pedestal is to be generated in the composite video signal. M_ENABLE or M_DISABLE
M_SYNC_TYPE	Returns which output signal has the sync encoded. M_GREEN, M_SEPARATE, M_DISABLE

**MsysAlloc()**

- To allocate a Corona-II system, you must select M\_SYSTEM\_CORONA\_II as the **SystemType** parameter. This selection opens communication with the board and will automatically allow the system to use some Host memory, and any available graphics controller.

**MsysControl()**

- When displaying a pseudo-live grab operation under Windows, MIL will double-buffer the grab depending on the performance of your Host PC. Double-buffering prevents any frame loss when the grabbed data is copied from the Host to the display. MIL is selective about performing double-buffered pseudo-live grabs because this is more demanding on your Host CPU. If the default behavior is not appropriate for your application, you can force a specific behavior with the following **ControlType**:

ControlType	ControlValue	
M_DISPLAY_DOUBLE_BUFFERING	Can be set to:	
	M_ENABLE	Force double-buffering.
	M_DISABLE	Don't use double-buffering.
	M_DEFAULT	Use double buffering only when MIL considers it appropriate.

- When a live grab operation uses a software trigger and **MdigHalt()** is issued, a software trigger is automatically generated to invoke a last grab (if you did not disable M\_LAST\_GRAB\_IN\_TRUE\_BUFFER). You can disable this automatic trigger; however, you would then have to issue a software trigger call after the **MdigHalt()** call (these calls

must be issued from different threads). To disable the ***MdigHalt()*** automatic trigger, set the following **ControlType** to M\_DISABLE:

ControlType	ControlType	
M_LIVE_GRAB_END_TRIGGER	Can be set to:	
	M_ENABLE	Default for software triggered cameras.
	M_DISABLE	Default for other camera types.

- When a live grab operation uses a hardware trigger, ***MdigHalt()*** will wait indefinitely for a hardware trigger to invoke a last grab (if you did not disable M\_LAST\_GRAB\_IN\_TRUE\_BUFFER). You can override this default so that ***MdigHalt()*** will cause an immediate last grab and will not wait for a hardware trigger. To do so, set the following **ControlType** to M\_DISABLE:

ControlType	ControlValue	
M_LIVE_GRAB_END_TRIGGER	Can be set to:	
	M_ENABLE	Default for hardware triggered cameras.
	M_DISABLE	Default for other camera types.

### **MsysInquire()**

- You can request the following additional information with the **InquireType** parameter:

<b>InquireType</b>	<b>Description</b>
M_BOARD_REVISION	The board revision (long value).
M_BOARD_TYPE	The type of system board: M_CORONA_II or M_CORONA_II_WITH_DIG_MODULE.
M_DIGITAL_MODULE_PRESENT	Returns M_TRUE if the companion digital-input board is present, otherwise it returns M_FALSE.
M_DISPLAY_DOUBLE_BUFFERING	The state of double buffering.
M_LIVE_GRAB_END_TRIGGER	Whether an automatic trigger is generated at end of grab: M_ENABLE or M_DISABLE.
M_PHYSICAL_ADDRESS_VGA	The physical address of the VGA frame buffer.

---

## ***Chapter 3: MIL and the Matrox Genesis platform***

*This section discusses features of MIL that are distinct to the Matrox Genesis platform and ways that optimize the board's performance.*

---

## Matrox Genesis-specific features

The Matrox Genesis family consists of two types of configurable, single-slot, PCI imaging boards: the Matrox Genesis main board (including Matrox Genesis-LC) and the processor board (including Matrox Genesis*Plus*).

The Matrox Genesis main board is a single-slot PCI imaging board which integrates acquisition, high-speed processing, and display capabilities. The processing section (also known as a node) includes a Matrox Video Interface ASIC (VIA), Matrox Neighborhood Operations Accelerator (NOA), a Texas Instruments TMS320C80 (C80) multi-processor, and 64 Mbytes of processing memory (SDRAM) for local processing. The main board has an on-board display section and offers a powerful, integrated grab module. The display section features a 6 Mbyte (color/monochrome) main frame buffer (also referred to as the underlay frame buffer surface). In addition, it has a 2 Mbyte graphics overlay frame buffer (also referred to as the overlay frame buffer surface) for non-destructive overlay capabilities (1600 x 1200 x 8-bit pseudo-color overlay). When the display section is being used to display the Windows desktop, the display format of the on-screen portion of the frame buffers is set using the selected Windows display resolution. The resolution is the same for both the underlay and overlay frame buffers. When the display section is being used for auxiliary display, the display format is set by *MdispAlloc()*.

Matrox Genesis-LC is the cost-sensitive version of the Matrox Genesis main board. This board performs all the acquisition and display functions of the main board, but does not have a processing section (node).

The Matrox Genesis processor board is a board that is dedicated to processing. It is available with either one or two processing nodes. It does not include a grab module or a display section.

Matrox Genesis*Plus* is an upgraded processor board. Each node includes a VIA, a NOA, a Motorola G4 Power PC processor, and 128 or 256 Mbytes of memory (SDRAM).

Several Matrox Genesis boards can be connected to each other by their grab ports and by their VMChannel, allowing high-speed accesses to each other's resources without accessing the primary PCI bus. For example, with multiple processor boards, images can be sent to and processed by all nodes, increasing the speed of all required processing operations.

- ❖ Note that Matrox Genesis cannot be used as a hardware fingerprint for run-time licenses.

This manual generally refers to all Matrox Genesis boards as Matrox Genesis. When it is necessary to distinguish between the Matrox Genesis boards, the manual refers to the Main Board or the Processor Board. In addition, it generally does not explicitly refer to Matrox Genesis-LC, because a discussion of the main board also applies to the Matrox Genesis-LC, excluding any discussion of the processing section.

- ❖ It is important to note that Windows 98 and Windows Me are not supported on Matrox Genesis.

See Appendix A for data flow diagrams.

---

## Using Matrox Genesis with MIL

To use a Matrox Genesis-LC board, or a node on a Main or Processor Board, you must allocate it as a Genesis system (`M_SYSTEM_GENESIS`), using **MsysAlloc()**. This selection opens communication with the board or the specified node, and will allow MIL to use its resources, and any available graphics controller.

❖ For auxiliary displays, the Matrox Genesis will only use its on-board display section.

When using a Processor Board, or when using a node that cannot access Matrox Genesis' display section without accessing the primary PCI bus, MIL will use not only the board/node's resources and Host computer memory, but also the VGA board. Use of the VGA board has been included to allow you to display grabbed images.

In this chapter, we discuss the MIL Genesis system in relation to the following: *16-bit buffer simulated display*, *Grabbing to a Host buffer with Matrox Genesis-LC*, *Particularities of existing MIL functions on Matrox Genesis*, and *Operations performed by Host*.

Refer to the *milgen.txt* file in the `\MATROX IMAGING\DRIVERS\DOC` (or user-specified) directory for any additions/modifications to these board specific notes.

---

## 16-bit buffer simulated display

Although it is possible to grab in monochrome buffers that are deeper than 8-bits on Matrox Genesis, the display adapter permits only 8-bit monochrome depth. As with other boards for windowed displays, it is possible to simulate the display of a 16-bit buffer by using the **MdispControl()** function with the `M_VIEW_MODE` control type. However, on a Matrox Genesis board, these control types are also supported for auxiliary displays.



---

## Grabbing to a Host buffer with Matrox Genesis-LC

When grabbing to a Host buffer, Matrox Genesis-LC uses off-screen frame buffer memory (that is, frame buffer memory that is not configured for display purposes) as a FIFO, to buffer image data while the board waits for access to the PCI bus. Loss of image data could otherwise occur during long bus-access latencies, found in heavily loaded systems.

Matrox Genesis-LC uses hardware/software transfer control logic to double-buffer the grab between two temporary buffers set up in off-screen memory. This logic toggles the grab between one temporary buffer and the other, transferring grabbed data from the buffer in which data is not being grabbed to the Host. The double-buffering is done on a line-block basis rather than on a frame basis.

In general, the default size of the temporary buffers provides a good compromise between maintaining a small latency before the beginning of the transfer and minimizing the transfer's dependency on the PCI bus load. However, when the PCI bus is extremely loaded, the Matrox Genesis-LC board might not be able to maintain the grab and transfer sequence. If the request to transfer is not serviced in time, the grab might overwrite non-transferred data. This is known as a grab over-run.

For example, a typical load on the PCI bus is that of the Matrox Genesis-LC grabbing to a Host buffer while the Host is updating the Matrox Genesis-LC's display. When updating the GUI (for windowed displays) and/or the display of grabbed and processed images, the display updates are frequent. This bi-directional traffic might increase the latency in servicing the request to write grabbed data to Host memory. At some point, the accumulated latencies might cause a grab over-run.

## Optimizing the use of the frame buffers

If you experience grab over-runs, try the following solutions in the specified order:

1. Increase the default temporary buffer size. This should be enough to resolve your problems.
2. If possible, double-buffer the grab on a frame basis rather than a line-block basis. In this case, you have to manage the double-buffering of the grab yourself. This method involves one frame of latency before the beginning of the transfer.
3. Reduce the display resolution or, if possible, reduce the PCI bus load.

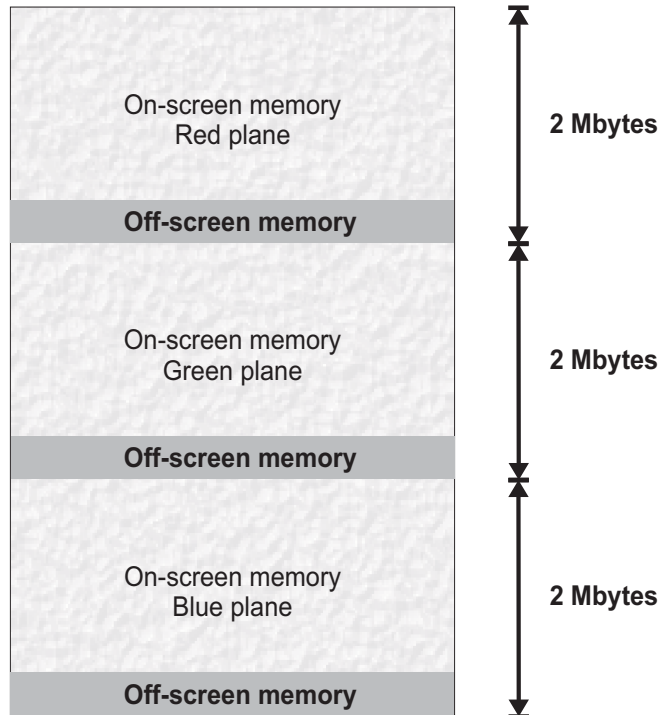
## Increasing the default temporary buffer size

You can increase the default temporary buffer size. Larger temporary buffers make the grab's transfer-to-Host less dependent on the PCI bus load. That is, the grab is less affected by long bus-access latencies, found in heavily loaded systems, because more memory is available to buffer the data. Note, however, that larger temporary buffers also increase the latency before the beginning of the transfer.

The maximum size of each temporary buffer is limited by the amount of off-screen memory divided by three:

$$\text{max temp buffer size} = \text{off-screen memory} / 3$$

The amount of off-screen frame buffer memory is determined by the amount of frame buffer memory that is not configured for display purposes. Decreasing the display resolution increases the amount of off-screen memory.



The amount of off-screen memory per frame buffer plane is determined as follows:

$$\text{off-screen memory/plane} = 2 \text{ Mbytes} - (w * h)$$

where  $w$  and  $h$  are the width and height of the selected display resolution.

The following table lists the amount of off-screen memory available for some typical display resolutions, as well as the maximum temporary buffer size for each resolution.

Display resolution	Bytes of off-screen memory/plane	Maximum temporary buffer size
640 x 480	1789952	596650
800 x 600	1617152	539050
1024 x 768	1310720	436906
1280 x 1024	786432	262144
1600 x 1200	177152	59050

You can increase the default temporary buffer size by adjusting the **SizeOfTmpBufferForHostGrab** field in the *genesis.ini* file.

### Grabbing in on-board buffers

If increasing the size of the temporary buffers does not resolve your over-run problem, you can try double-buffering the grab on a frame basis rather than a line-block basis. The grab is then less dependent on the PCI bus load because the buffers don't have to be transferred as frequently and intermittent bus latencies are averaged over an entire frame. The downside of this process is that there is one frame of latency before the beginning of the transfer.

To implement this model, you have to manage the double-buffering. To do so, allocate two frame-sized buffers on-board and then grab a field/frame into one buffer while copying the other buffer to the Host. Since you control the synchronization between the grab and transfer, you can detect when something goes wrong.

❖ Note that, by default, the copy operation is driven by the Host CPU. To speed up the copy and reduce Host intervention, enable VIA-driven copies by setting the ***MsysControl()*** `M_BUS_MASTER_COPY_TO_HOST` control to `M_ENABLE`.

This method is only possible if sufficient on-board memory is available to allocate the two buffers.

## Grabbing large frames of data

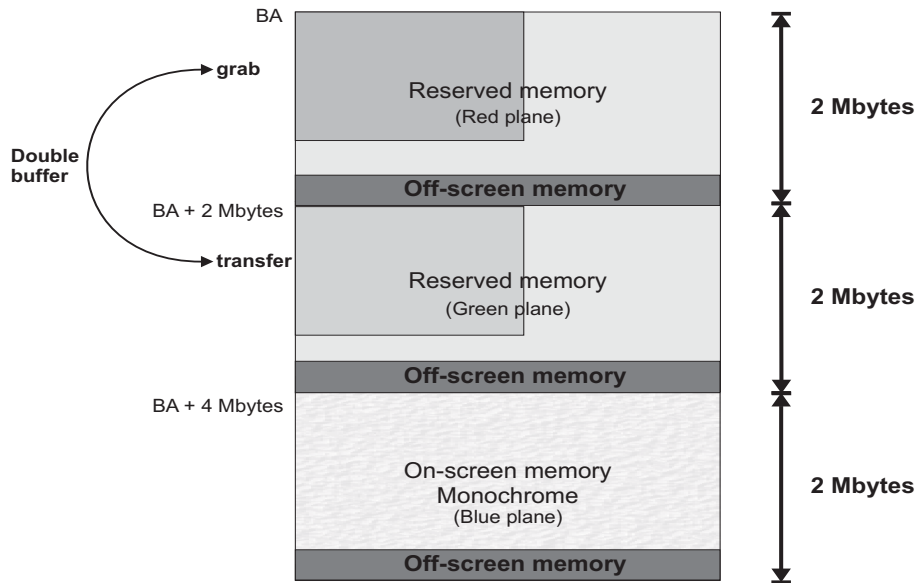
When grabbing large frames of data (for example, 2K X 2K and 4K X 4K), there is no way to store the whole frame in frame buffer memory. In this case, you can only grab to a Host buffer. However, if you are experiencing over-runs using this method, try the suggestions in the subsection titled *Other options*.

## Monochrome display

When the frame buffers are in an 8-bit (monochrome) display resolution, an area of only one frame buffer plane is used to maintain the display. In this case, sufficient memory is available to keep two entire average-sized frames on-board.

Although an area of only one frame buffer plane is actually used to maintain the display, MIL reserves the corresponding area of the two other frame buffer planes so that you can switch between a monochrome and color display without corrupting off-screen memory.

This means that if you try to allocate the buffers with an `M_ON_BOARD` attribute, there might not be sufficient true off-screen memory to allocate the entire buffer. Instead, you might need to allocate the grab buffers in the reserved frame buffer memory. To do so, inquire the address of the underlay frame buffer using **MsysInquire()**, and then, using **MbufCreate2d()**, create your two buffers at the appropriate offset (that is, 0, 2, or 4 Mbytes).



BA = Base address

If you allocate buffers in reserved display memory, you cannot switch to color mode or select a color buffer on the display; otherwise, the grab buffers will be corrupted.

## An example

The following example shows how to allocate buffers in reserved display memory and then grab in these buffers.

```

/* File name: mgenlc.c
 * Synopsis: This program allocates two grab buffers in the
 *           unused bands of the display and then grabs in them.
 *
 *           When the display is in monochrome mode, it
 *           uses only the blue plane and reserves the
 *           red and green planes. This example shows
 *           how to access this unused memory for grabbing
 *           purposes.
 */
/* Headers */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <mil.h>

/* Main function. */
void main(void)
{
    MIL_ID    MilApplication;
    MIL_ID    MilSystem      ;
    MIL_ID    MilDigitizer   ;
    MIL_ID    MilDisplay     ;
    MIL_ID    MilImageOnBoard[2];
    MIL_ID    MilImageDisp;
    void      *UnderlayPhysicalAddress = NULL;

    /* Allocations.*/
    MappAlloc(M_DEFAULT, &MilApplication);
    MsysAlloc(M_SYSTEM_GENESIS, M_DEF_SYSTEM_NUM, M_SETUP, &MilSystem);
    MdigAlloc(MilSystem, M_DEFAULT, M_DEF_DIGITIZER_FORMAT,
              M_DEFAULT, &MilDigitizer);
    MdispAlloc(MilSystem, M_DEFAULT, M_DEF_DISPLAY_FORMAT, M_DEFAULT,
              &MilDisplay);

    /* Force the display in monochrome mode on the blue plane. */
    MdispControl(MilDisplay, M_COLOR_MODE, M_BLUE);

    /* Allocate a display buffer. */
    MbufAlloc2d(MilSystem,
                (long)(MdigInquire(MilDigitizer, M_SIZE_X, M_NULL)),
                (long)(MdigInquire(MilDigitizer, M_SIZE_Y, M_NULL)),
                8L+M_UNSIGNED,
                M_IMAGE+M_PROC+M_DISP+M_NON_PAGED, &MilImageDisp);
    MbufClear(MilImageDisp, 0);

```

(cont...)

```

/* Inquire the base address of the underlay plane. */
MsysInquire(MilSystem, M_PHYSICAL_ADDRESS_UNDERLAY,
            &UnderlayPhysicalAddress);

/* Create a buffer on the red plane (base address). */
MbufCreateColor(MilSystem, 1,
                (long)(MdigInquire(MilDigitizer, M_SIZE_X, M_NULL)),
                (long)(MdigInquire(MilDigitizer, M_SIZE_Y, M_NULL)),
                8L+M_UNSIGNED,
                M_IMAGE+M_GRAB+M_PROC+M_ON_BOARD,
                M_PHYSICAL_ADDRESS+M_PITCH_BYTE,
                M_DEFAULT,
                (void *)&(UnderlayPhysicalAddress),
                &MilImageOnBoard[0]);
/* Move to the green plane (base address+2MEG). */
UnderlayPhysicalAddress = ((unsigned long) UnderlayPhysicalAddress) +
                          0x200000;
MbufCreateColor(MilSystem, 1,
                (long)(MdigInquire(MilDigitizer, M_SIZE_X, M_NULL)),
                (long)(MdigInquire(MilDigitizer, M_SIZE_Y, M_NULL)),
                8L+M_UNSIGNED,
                M_IMAGE+M_GRAB+M_PROC+M_ON_BOARD,
                M_PHYSICAL_ADDRESS+M_PITCH_BYTE,
                M_DEFAULT,
                (void *)&(UnderlayPhysicalAddress),
                &MilImageOnBoard[1]);
/* Enable bus master copies from the Genesis to the Host. */
MsysControl(MilSystem, M_BUS_MASTER_COPY_TO_HOST, M_ENABLE);

/* Display the buffer. */
MdispSelect(MilDisplay, MilImageDisp);
printf("Press enter to grab into the first buffer\n");
getchar();

/* Grab and copy to the Host. */
MdigGrab(MilDigitizer, MilImageOnBoard[0]);
MbufCopy(MilImageOnBoard[0], MilImageDisp);
printf("Press enter to grab into the second buffer\n");
getchar();
/* Grab and copy to the Host. */
MdigGrab(MilDigitizer, MilImageOnBoard[1]);
MbufCopy(MilImageOnBoard[1], MilImageDisp);
getchar();

/* Free allocations. */
MbufFree(MilImageOnBoard[0]);
MbufFree(MilImageOnBoard[1]);
MbufFree(MilImageDisp);

MdispFree(MilDisplay);
MdigFree(MilDigitizer);
MsysFree(MilSystem);
MappFree(MilApplication);
}

```



## Color display

When the frame buffers are in a 24-bit (color) display resolution, the three frame buffer planes are used to maintain the display. In this case, off-screen memory is the only memory available in which to perform double-buffering without affecting the display.

In a 1024 x 768 display resolution, there is 1310720 bytes of off-screen memory/plane available for a double buffering scheme. If, for example, your camera is a 512 x 480 RGB source, only 737280 (512 x 480 x 3) bytes are required to buffer one frame. So, you can allocate the buffers in the off-screen memory of two planes. Since you need to allocate the buffers in off-screen memory, simply allocate the buffers with ***MbufAllocColor()*** and use the `M_ON_BOARD` flag; the buffers will automatically be allocated in the appropriate off-screen memory.

When grabbing larger frames of data or when displaying in a higher resolution, the previous method might not be suitable. For example:

Display	Camera	Bytes missing in off-screen memory
1280 x 1024	1K x 1K mono source	262144
1600 x 1200	1K x 1K mono source	871424
1600 x 1200	512 x 480 mono source	68608

In these cases, the only way to achieve the grab is to use off-screen memory as FIFO and grab to a Host buffer. However, if you are experiencing over-runs using this method, try the suggestions in the following subsection.

### Other options

In the unlikely event that the previous two suggestions do not resolve grab over-runs, you have the following options:

- You can try reducing the display resolution. This reduces the PCI bus load because there is less to update, and it increases the amount of off-screen memory so you can create larger temporary buffers.
- If your application permits, use other methods to reduce the PCI bus load. For example, reduce the number of display updates of the processed grabbed image; this, however, involves manual intervention.

---

### Particularities of existing MIL functions on Matrox Genesis

Certain commands can have special features or functionalities on the Genesis system. This table provides an overview of the affected commands. Details and procedures are described in the individual command descriptions, which follow.

Commands	Matrox Genesis particularities
MblobFill()	Buffer restriction
MblobInquire()	Matrox Genesis-specific inquire options.
MblobReconstruct	Buffer restriction
MbufAlloc...()	Buffer options.
MbufCopyCond() and MbufCopyMask()	Note about copying floating-point buffers.
MbufCreateColor()	Additional control flag option.
MbufInquire()	Matrox Genesis-specific inquire options.
MdigAlloc()	Digitizer configuration format (DCF) specifications.
MdigControl()	Control type and flag additions.
MdigGrab	Destination buffer restriction.
MdigGrabWait()	Unsupported parameter.
MdigHookFunction()	Unsupported hook types.
MdigInquire()	Matrox Genesis-specific inquire options.
MdigReference()	Valid reference types.
MdigLut()	Input LUT buffer size requirements.

Commands	Matrox Genesis particularities
MdispAlloc()	Display format. Overlay setting.
MdispControl()	Matrox Genesis-specific display control options.
MdispInquire()	Matrox Genesis-specific inquire options.
MdispLut()	Available for both windowed and auxiliary displays.
MgenWarpParameter()	Overscan options.
MgraFontScale()	Note about font scale.
MimConvolve()	Note about 16-bit buffers.
MimResize()	Additional interpolation controls.
MpatInquire()	Matrox Genesis-specific inquire options.
MsysAlloc()	Matrox Genesis-specific allocation options.
MsysControl()	Additional controls.
MsysInquire()	Matrox Genesis-specific inquire options.

### MblobFill()

Does not support 1-bit (packed binary) source or destination buffers.

### MblobInquire()

The **InquireType** parameter can also be set to the following:

M_NATIVE_ID	Matrox Genesis library identifier associated with the given MIL blob result or feature list identifier.
-------------	---

### MblobReconstruct()

Does not support 1-bit (packed binary) source or destination buffers.

MbufAlloc...()

- You can allocate up to 4-band buffers with *MbufAllocColor()*.
- Buffers with an M\_PACKED attribute are not supported.
- 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.
- You can add the following to the **Attribute** parameter:

Attribute parameter for 3-band 8-bit buffer	Internal storage format for 3-band 8-bit buffer
M_RGB3+M_PLANAR	3-bit (RGB 1:1:1) planar pixels.
M_RGB24+M_PLANAR	24-bit (RGB) planar pixels.
M_RGB48+M_PLANAR	48-bit (RGB 16:16:16) planar pixels.
M_RGB96+M_PLANAR	96-bit (RGB 32:32:32) planar pixels.

- Only three types of kernel buffers are supported: signed 8-bit, unsigned 8-bit, and signed 16-bit.
- When using a Main Board or Processor Board, all buffers are allocated in processing memory (fast SDRAM) to speed up processing. When using a Matrox Genesis-LC, all buffers are allocated in Host memory.

When a buffer is selected on the display, a display copy is also created in the display memory. When the buffer is processed while selected on the display, the computation is done in processing memory (on Matrox Genesis-LC, done in Host memory) and the display is updated at the completion of processing.

When allocating an M\_DISP or M\_OVR buffer that is for auxiliary displays, it is possible to override the default allocation sequence and allocate the buffer only in display memory. To do so, set the **MbufAlloc() Attribute** parameter to M\_IMAGE+M\_SINGLE+..... Note that processing done on this type of buffer is slower.

MbufCopyCond(), MbufCopyMask()

- MIL uses the Host to copy floating-point buffers conditionally.

## MbufCreateColor()

- An additional **ControlFlag** is available:

M_BUF_ID	This allows you to allocate a multi-band buffer from an array of single-band buffers. <b>ArrayOfDataPtr</b> will be a pointer to an array of MIL buffer identifiers. One identifier per band must be provided. <b>SizeX</b> , <b>SizeY</b> , <b>Type</b> , <b>Attribute</b> and <b>Pitch</b> must be the same for each buffer. <b>Pitch</b> can be set M_DEFAULT.
----------	--

- You can add the following to the **Attribute** parameter:

Attribute parameter for 3-band 8-bit buffer	Internal storage format for 3-band 8-bit buffer
M_RGB3+M_PLANAR	3-bit (RGB 1:1:1) planar pixels.
M_RGB24+M_PLANAR	24-bit (RGB) planar pixels.
M_RGB48+M_PLANAR	48-bit (RGB 16:16:16) planar pixels.
M_RGB96+M_PLANAR	96-bit (RGB 32:32:32) planar pixels.

## MbufInquire()

- The **InquireType** parameter can also be the following:

M_NATIVE_ID	Identifier of the Genesis Native Library buffer that corresponds to the specified MIL buffer.
-------------	---

## MdigAlloc()

- The **DataFormat** parameter specifies the DCF for the input device.

The available settings for monochrome cameras are:

"M_DEFAULT"	RS-170, 640x480, 8 bits, 12.5MHz, analog
"M_RS170"	RS-170, 640x480, 8 bits, 12.5MHz, analog
"M_CCIR"	CCIR, 768x576, 8 bits, 14.8MHz, analog

The available settings for color cameras are:

"M_NTSC_RGB"	RS-170, RGB, 640x480, 3x8 bits, 12.5MHz
"M_PAL_RGB"	PAL I, RGB, 768x576, 3x8 bits, 14.8MHz

You can also set the **DataFormat** parameter to a string that specifies a DCF file. This file specifies the input signal format. See the \GENESIS\DCF directory for the current list of supported formats.

## MdigControl()

- The supported scaling factors for the M\_GRAB\_SCALE\_... control types are: 4, 2, 1, 1/2, 1/3, ..., 1/16.
- For M\_GRAB\_TRIGGER\_SOURCE, note the meaning of the following **ControlValue** settings on a Matrox Genesis:

M_HARDWARE_PORT0	Use opto-isolated hardware trigger signal. Combination of Pin 1 (+) and Pin 3 (-) on video input connector.
M_HARDWARE_PORT1	Use TTL or RS-422 hardware trigger signal. On the companion digital input board, RS-422 trigger is a combination of Pin 47 (TRIGGER, INPUT, +) and Pin 48 (TRIGGER, INPUT, -). On the digital cable adaptor board, RS-422 trigger is a combination of Pin 9 (TRIGGER, INPUT, 422+) and Pin 43 (TRIGGER, INPUT, 422-); whereas TTL trigger on Pin 67 (TRIGGER, INPUT, TTL).

- For windowed displays, you can also set M\_GRAB\_SCALE\_... to M\_FILL\_DESTINATION or M\_FILL\_DISPLAY.
- Matrox Genesis supports these additional **ControlType** parameter settings:

ControlType	Description & ControlValue		
M_GRAB_ABORT	Immediately stops a grab in progress and queued grabs. Useful for cancelling a triggered grab that is waiting for the trigger.		
	M_DEFAULT	Stops the grab.	
M_GRAB_INPUT_GAIN + M_CHX	Select the input-signal gain for channel $x$ , where $x$ is the channel number (0, 1, 2, or 3). More than one channel can be specified. If no channel is specified, the gain is applied to all channels (default). The valid input signal voltages (excluding the sync signal) and their corresponding gains are:		
		Input Voltage	Gain
	M_GAIN0	1.0 - 1.4 Vpp	2
	M_GAIN1	0.8 - 1.0 Vpp	2.7
	M_GAIN2	0.7 - 0.8 Vpp	3.3
	M_GAIN3	0.0 - 0.7 Vpp	4.1
	M_DEFAULT	Same as M_GAIN3	Same as M_GAIN3
M_GRAB_EXPOSURE_BYPASS	Activate the manual or automatic exposure model:		
	M_ENABLE	Manual exposure model.	
	M_DISABLE	Automatic exposure model.	
	M_DEFAULT	Same as M_DISABLE.	

ControlType	Description & ControlValue	
For the following M_GRAB_EXPOSURE... control types, you can add M_TIMER1 or M_TIMER2 in manual exposure mode, to control the different on-board exposure timers. When omitted, Timer1 is assumed.		
M_GRAB_EXPOSURE	When using a software trigger source, use this control type to activate the specified grab exposure timer. When using a non-software trigger source, enable or disable the specified grab exposure timer. Note, the M_GRAB_EXPOSURE control type has no effect when grabbing using the automatic exposure model.	
	M_ACTIVATE	Activate a software trigger for the specified exposure timer.
	M_ENABLE	Enable exposure timer.
	M_DISABLE	Disable exposure timer.
	M_DEFAULT	same as .dcf (non-software trigger source).
M_GRAB_EXPOSURE_MODE	Sets the exposure signal's polarity:	
	M_LEVEL_HIGH	
	M_LEVEL_LOW	
	M_DEFAULT	(same as .dcf).

ControlType	Description & ControlValue	
M_GRAB_EXPOSURE_SOURCE	Select the trigger source for the specified exposure timer: The M_GRAB_EXPOSURE_SOURCE control type has no effect when grabbing using the automatic exposure model.	
	M_NULL	Disable specified exposure timer. This has no effect when grabbing using automatic exposure model.
	M_SOFTWARE	Use software trigger. The exposure signal is generated when <b>MdigControl()</b> with M_GRAB_EXPOSURE + M_TIMERN and M_ACTIVATE is called.
	M_HARDWARE_PORT0	Use opto-isolated hardware trigger signal. Combination of Pin 1 (+) and Pin 3 (-) on Video Input connector.
	M_HARDWARE_PORT1	Use TTL or RS-422 hardware trigger signal. On the companion digital input board, RS-422 trigger is a combination of Pin 47 (TRIGGER, INPUT, +) and Pin 48 (TRIGGER, INPUT, -). On the digital cable adaptor board, RS-422 trigger is a combination of Pin 9 (TRIGGER, INPUT, 422+) and Pin 43 (TRIGGER, INPUT, 422-); whereas TTL trigger on Pin 67 (TRIGGER, INPUT, TTL).
	M_VSYNC	Use vertical sync signal.
	M_HSYNC	Use horizontal sync signal.
	M_TIMER1	Use exposure signal generated by Timer1. Use only if setting trigger source for Timer2.
	M_TIMER2	Use exposure signal generated by Timer2. Use only if setting trigger source for Timer1.
	M_CONTINUOUS	No actual trigger. Run selected exposure timer in periodic mode. Automatically reset timer after each exposure signal is output. Exposure signal loops between delay and active mode



ControlType	Description & ControlValue	
M_GRAB_EXPOSURE_TIME	Set the time (in nsec) for the active portion of the exposure signal (that is, the exposure time). M_DEFAULT has the same effect as setting in the digitizer's *.dcf). When using the automatic exposure model, if a single timer cannot generate the required exposure time, MIL automatically sets up connections with the second timer to generate the requested exposure time length. If <b>ControlValue</b> is set to 0, exposure is disabled and the grab is performed immediately. Note, an error is returned if the specified exposure time cannot be generated.	
M_GRAB_EXPOSURE_TIME_DELAY	Set the delay (in nsec) between the trigger and the start of exposure. If M_DEFAULT, same value as DCF. Note, an error is returned if the specified delay cannot be generated.	
M_GRAB_EXPOSURE_TRIGGER_MODE	Set the trigger activation mode for specified timer.	
	M_EDGE_RISING	Low to high signal variation.
	M_EDGE_FALLING	High to low signal variation.
M_GRAB_LUT_PALETTE	Only the input-LUT palette M_LUT_PALETTE0 is available.	
M_USER_IN_FORMAT	Enable either TTL or RS-422 receivers for digital I/Os:	
	M_TTL	Enable the TTL receivers for trigger and user inputs.
	M_RS422	Enable the RS-422 receivers for trigger and user inputs.
	M_DEFAULT	Same as the *.dcf.
	M_DISABLE	Disable trigger and user inputs.
M_USER_OUT_FORMAT	Enable either TTL or RS-422 drivers for digital I/Os:	
	M_TTL	Enable the TTL drivers for exposure and user outputs.
	M_RS422	Enable the RS-422 drivers for exposure and user outputs.
	M_DEFAULT	Same as the *.dcf.
	M_DISABLE:	Disable exposure and user outputs.

ControlType	Description & ControlValue																				
M_USER_BIT+(0,4)	<p>Set the state of the output bits of the grab module digital port (if applicable): M_ON or M_OFF</p> <p>The relationship between the MIL user-bit number and the actual user output bit on the grab module digital port is as follows.</p> <p>Companion digital input board:</p> <table> <tr> <td>bit 0:</td><td>USER0 OUTPUT (RS-422 only).</td></tr> <tr> <td>bit 1:</td><td>USER1 OUTPUT (RS-422 only).</td></tr> <tr> <td>bit 2:</td><td>CAMERA CONTROL BIT 0 (TTL only).</td></tr> <tr> <td>bit 3:</td><td>CAMERA CONTROL BIT 1 (TTL only).</td></tr> <tr> <td>bit 4:</td><td>CAMERA CONTROL BIT 2 (TTL only).</td></tr> </table> <p>Digital cable adapter:</p> <table> <tr> <td>bit 0:</td><td>USER0 OUTPUT (TTL or RS-422).</td></tr> <tr> <td>bit 1:</td><td>USER1 OUTPUT (TTL or RS-422).</td></tr> <tr> <td>bit 2:</td><td>MODULE LOAD OUTPUT (TTL only).</td></tr> <tr> <td>bit 3:</td><td>MODULE CLOCK OUTPUT (TTL only).</td></tr> <tr> <td>bit 4:</td><td>MODULE DATA OUTPUT (TTL only).</td></tr> </table>	bit 0:	USER0 OUTPUT (RS-422 only).	bit 1:	USER1 OUTPUT (RS-422 only).	bit 2:	CAMERA CONTROL BIT 0 (TTL only).	bit 3:	CAMERA CONTROL BIT 1 (TTL only).	bit 4:	CAMERA CONTROL BIT 2 (TTL only).	bit 0:	USER0 OUTPUT (TTL or RS-422).	bit 1:	USER1 OUTPUT (TTL or RS-422).	bit 2:	MODULE LOAD OUTPUT (TTL only).	bit 3:	MODULE CLOCK OUTPUT (TTL only).	bit 4:	MODULE DATA OUTPUT (TTL only).
bit 0:	USER0 OUTPUT (RS-422 only).																				
bit 1:	USER1 OUTPUT (RS-422 only).																				
bit 2:	CAMERA CONTROL BIT 0 (TTL only).																				
bit 3:	CAMERA CONTROL BIT 1 (TTL only).																				
bit 4:	CAMERA CONTROL BIT 2 (TTL only).																				
bit 0:	USER0 OUTPUT (TTL or RS-422).																				
bit 1:	USER1 OUTPUT (TTL or RS-422).																				
bit 2:	MODULE LOAD OUTPUT (TTL only).																				
bit 3:	MODULE CLOCK OUTPUT (TTL only).																				
bit 4:	MODULE DATA OUTPUT (TTL only).																				
M_GRAB_WAIT	<p>Force the grab of a digitizer to wait for a grab of a second digitizer. This allows you to grab the same frame with two or more MIL digitizers. <b>ControlFlag</b> should be either the MIL identifier of the second digitizer, or M_NULL (default) to not wait for a second digitizer.</p> <p>Note, this mode is only valid when the destination buffers are in different memory banks (for example, on different nodes in a multiple Matrox Genesis board configuration). See the example that follows this table.</p>																				
M_SYNCHRONIZE_ON_STARTED	<p>Synchronize the grab of a digitizer with the grab of a second digitizer. This allows you to grab sequential frames from two or more MIL digitizers. <b>ControlValue</b> should be either the MIL identifier of the second digitizer or M_NULL (default) to not synchronize with a second digitizer. See the M_SYNCHRONIZE_ON_STARTED example that follows this table and the M_GRAB_WAIT example.</p>																				

- **M\_GRAB\_WAIT** can be used in a multiple Matrox Genesis configuration as in the following example:

```

/* Synopsis: This program grabs the same frame through three different
 * digitizers into three buffer in different memory banks.
 */

/* Dig1, Dig2, and Dig3 are allocated on different Matrox Genesis systems and
 * their grab mode is 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 enable the capture. */
MdigGrab(Dig1, Buf1);

```

- **M\_SYNCHRONIZE\_ON\_STARTED** can be used in a multiple Matrox Genesis configuration as in the following example:

```

/* Synopsis: This program grabs three sequential frames through three different
 * digitizers.
 */

/* Dig1, Dig2, and Dig3 are allocated on different Matrox Genesis systems and
 * their grab mode is 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);

```

## MdigGrab()

- While performing a grab, the width of the grab region must be at least 32 bytes after horizontal sub-sampling.

## MdigGrabWait()

- M\_GRAB\_NEXT\_FIELD is not supported.

## MdigHookFunction()

- M\_FRAME\_START and all M\_FIELD... event types are not supported. That is, you can only hook a function to input-signal events that occur while grabbing.
- M\_GRAB\_FRAME\_START is not supported on Matrox Genesis-LC.

## MdigInquire()

- For windowed displays, M\_GRAB\_SCALE... can also return: M\_FILL\_DESTINATION or M\_FILL\_DISPLAY.
- The **InquireType** parameter can also be set to the following:

<b>InquireType</b>	<b>Description</b>
M_COLOR_MODE	Monochrome or color input: M_MONOCHROME or M_RGB.
M_GRAB_IN_PROGRESS	Current grab state: M_YES or M_NO.
M_GRAB_INPUT_GAIN + M_CHx	The input-signal gain for channel <i>x</i> , where <i>x</i> is the channel number. Only one channel can be specified. If no channel is specified, then the gain applied to channel 0 is returned. M_GAIN0, M_GAIN1, M_GAIN2, or M_GAIN3.
M_INPUT_SIGNAL_SOURCE	Input-signal source: M_HARDWARE_PORT0 (analog) or M_HARDWARE_PORT1 (digital).
M_GRAB_EXPOSURE_BYPASS	Exposure model : M_ENABLE (manual) or M_DISABLE (automatic).
For the following M_GRAB_EXPOSURE... inquire types, you can add M_TIMER1 or M_TIMER2 in manual exposure mode, to inquire about the different on-board exposure timers. When omitted, Timer1 is assumed.	
M_GRAB_EXPOSURE	Exposure timer state for non-software trigger source: M_ENABLE or M_DISABLE.
M_GRAB_EXPOSURE_MODE	Exposure signal's polarity: M_LEVEL_HIGH or M_LEVEL_LOW.

InquireType	Description								
M_GRAB_EXPOSURE_SOURCE	Trigger source for specified timer: M_NULL, M_SOFTWARE, M_HARDWARE_PORT0, M_HARDWARE_PORT1, M_VSYNC, M_HSYNC, M_CONTINUOUS, M_TIMER1, M_TIMER2.								
M_GRAB_EXPOSURE_TIME	Time for the active portion of the exposure signal (value in nsec). Returned as a double.								
M_GRAB_EXPOSURE_TIME_DELAY	Delay (in nsec) between the trigger and the start of exposure. Returned as a double.								
M_GRAB_EXPOSURE_TRIGGER_MODE	Trigger activation mode for specified timer: M_EDGE_RISING or M_EDGE_FALLING.								
M_GRAB_LUT_PALETTE	Input-LUT palette: M_LUT_PALETTE0.								
M_NATIVE_CAMERA_ID	Identifier of the Genesis Native Library camera that is associated with the specified MIL digitizer.								
M_NATIVE_CONTROL_ID	Identifier of the Genesis Native Library control buffer that is associated with the specified MIL digitizer.								
M_NATIVE_ID	Identifier of the Genesis Native Library digitizer that is associated with the specified MIL digitizer.								
M_NATIVE_LAST_GRAB_OSB_ID	Identifier of the Genesis Native Library OSB (operations status block) used in the last grab.								
M_USER_BIT+(0,1)	<p>Current state of the input bit of the external I/O port: M_ON (1) or M_OFF (0).</p> <p>The relationship between the MIL user-bit number and the actual user input on the grab module digital port is as follows.</p> <p><b>Companion digital input board:</b></p> <table> <tr> <td>bit 0:</td><td>USER0 INPUT (RS-422 only)</td></tr> <tr> <td>bit 1:</td><td>USER1 INPUT (RS-422 only)</td></tr> </table> <p><b>Digital cable adapter:</b></p> <table> <tr> <td>bit 0:</td><td>USER0 INPUT (TTL or RS-422)</td></tr> <tr> <td>bit 1:</td><td>USER1 INPUT (TTL or RS-422)</td></tr> </table>	bit 0:	USER0 INPUT (RS-422 only)	bit 1:	USER1 INPUT (RS-422 only)	bit 0:	USER0 INPUT (TTL or RS-422)	bit 1:	USER1 INPUT (TTL or RS-422)
bit 0:	USER0 INPUT (RS-422 only)								
bit 1:	USER1 INPUT (RS-422 only)								
bit 0:	USER0 INPUT (TTL or RS-422)								
bit 1:	USER1 INPUT (TTL or RS-422)								
M_USER_IN_FORMAT	Type of receiver for digital I/Os (M_TTL, M_RS422, or M_DISABLE).								
M_USER_OUT_FORMAT	Type of driver for digital I/Os (M_TTL, M_RS422, or M_DISABLE).								

MdigLut()

- The specified LUT buffer must meet the following requirements when dealing with the indicated types of input mode and restrictions:

Input mode	Restrictions
analog	The LUTs must have 256 entries of 8 bits each.
digital	When grabbing from one or two channels, the LUTs can have up to 8192 (that is, $2^{13}$ ) entries of 8 or 16 bits each. When grabbing from more than two channels, the LUTs must have 256 entries of 8 bits each.

MdigReference()

- The valid **ReferenceType** settings do **not** include:  
M\_BRIGHTNESS\_REF, M\_CONTRAST\_REF, M\_HUE\_REF,  
M\_SATURATION\_REF.
- For M\_BLACK\_REF and M\_WHITE\_REF, note the meaning of the **ReferenceLevel** parameter settings:
  - The minimum voltage level (M\_MIN\_LEVEL) corresponds to 0.0 V.
  - The maximum voltage level (M\_MAX\_LEVEL) corresponds to 2.5 V.

Note that some consecutive **ReferenceLevel** settings might produce the same result due to the fact that there are only 255 distinct adjustments (adjustments of 9.80 mV each). Also note that the white level should be higher than the black level.

## MdispAlloc()

- For auxiliary displays, these are the display formats that you can use to display from the Matrox Genesis board:

Display Format	Display Resolution
"640x480x8PP"	640 x 480 at 60 Hz
"800x600x8PP"	800 x 600 at 60 Hz
"1024x768x8PP"	1024 x 768 at 60 Hz
"1280x1024x8PP"	1280 x 1024 at 60 Hz
"1600x1200x8PP"	1600 x 1200 at 60 Hz
VM101_60.VCF	640 x 480 at 60 Hz
VM101_72.VCF	640 x 480 at 72 Hz
VM103_60.VCF	800 x 600 at 60 Hz
VM103_72.VCF	800 x 600 at 72 Hz
VM105_60.VCF	1024 x 768 at 60 Hz
VM105_72.VCF	1024 x 768 at 72 Hz
CM001_60.VCF	1152 x 882 at 60 Hz
CM001_72.VCF	1152 x 882 at 72 Hz
VM107_60.VCF	1280 x 1024 at 60 Hz
VM107_72.VCF	1280 x 1024 at 72 Hz
VM11C_60.VCF	1600 x 1200 at 60 Hz

See the *GENESIS\VCF* directory for the current list of Video Configuration Format (VCF) files. For auxiliary displays, "M\_DEFAULT" is the VCF specified in the *genesis.ini* file.

- When displaying from the Matrox Genesis board, MIL automatically displays the selected image buffer from the most appropriate frame buffer surface (using hardware keying, when displaying the buffer from the underlay frame buffer surface). You can, however, force a particular surface from which to display. To do so, add one of the following initialization flags to either M\_WINDOWED or M\_AUXILIARY:

M_OVR	Force the display of the selected image buffer to the overlay frame buffer surface. Some CPU intervention might be required if forced to use this surface. An overlay surface is useful because, in general, you can associate a LUT with an auxiliary display.
M_UND	Force the display of the selected image buffer to the underlay frame buffer surface. This surface is typically always used. In this surface, LUTs are available for windowed displays only.

**MdispControl()**

- The **ControlType** and **ControlValue** parameters can be set to the following:

Control Type	ControlValue and Description	
M_COLOR_MODE	M_BLUE	Display Matrox Genesis main frame buffer blue band in monochrome.
	M_COLOR	Display Matrox Genesis main frame buffer in true color.
	M_GREEN	Display Matrox Genesis main frame buffer green band in monochrome.
	M_RED	Display Matrox Genesis main frame buffer red band in monochrome.
M_WINDOW_OVR_WRITE	The display's overlay buffer will always be 1 band.	



## MdisplInquire()

- The **InquireType** parameter can also be set to the following:

InquireType	Description
M_COLOR_MODE	The display color mode ( <b>MdispControl()</b> ).
M_NATIVE_CONTROL_ID	Identifier of the Genesis Native Library control buffer that is associated with the given MIL display.
M_NATIVE_ID	Identifier of the Genesis Native Library display that corresponds to the given MIL display.

## MdispLut()

- A LUT can be associated for both windowed and auxiliary displays. However, for auxiliary displays, the M\_OVR attribute must be specified (**MdispAlloc()**).
- If a display was allocated with the M\_OVR attribute, you can associate it with a monochrome (1 band x 8 bit x 256 entries) or a pseudo-color (3 bands x 8 bit x 256 entries) LUT buffer.

## MgenWarpParameter()

When using LUT buffers for your output, you can add one of the following defines to the **OperationMode** parameter. The default is M\_OVERSCAN\_DISABLE.

M_OVERSCAN_DISABLE	Replace addresses that fall outside of the source buffer of <b>MimWarp()</b> with the address (0, 0). If you use this define, the size of the source buffer is required; specify the x-size with the <b>Val1</b> parameter and the y-size with the <b>Val2</b> parameter. To use the same x- and/or y-size as the LUT buffers, set <b>Val1</b> and/or <b>Val2</b> to M_NULL or M_DEFAULT.
M_OVERSCAN_ENABLE	If addresses fall outside of the source buffer of <b>MimWarp()</b> , use them anyway. Note that, if the source buffer is not a child buffer, these addresses will point to undefined pixel values, leading to unpredictable results.

**MgraFontScale()**

- MIL uses the Host to scale a font by a non-integer factor; integer factors are scaled by the on-board Matrox Genesis processor.

**MimConvolve()**

- All 16-bit buffers are processed as signed buffers.

**MimRank()**

- The predefined mask buffer M\_3X3\_CROSS is not supported.

**MimResize()**

- M\_OVERSCAN\_DISABLE cannot be added to the **InterpolationMode** parameter.

**MimRotate()**

- M\_OVERSCAN\_DISABLE cannot be added to the **InterpolationMode** parameter.

**MimWarp()**

- M\_OVERSCAN\_DISABLE cannot be added to the **InterpolationMode** parameter.

**MpatInquire()**

- For Main Board and Processor Board, the **ParamToInquire** parameter can also be set to the following:

M_NATIVE_ID	Genesis library identifier of the display associated with the given pattern matching model or MIL result buffer.
-------------	--

**MsysAlloc()**

- To allocate a Matrox Genesis-LC board or a node on a Main or Processor Board, you must select M\_SYSTEM\_GENESIS as the **SystemType** parameter. This selection opens communication with the board or the specified node, and will allow MIL to use its resources, and any available graphics controller.
  - ❖ For auxiliary displays, the Matrox Genesis will only use its on-board display section; however, the on-board display section of the Matrox Genesis can be used by other systems.

When using a Matrox Genesis processor board or when using a node that cannot access Matrox Genesis's display section (without accessing the primary PCI bus), MIL will not only use the board/node's resources and Host computer memory, but also the VGA board. The VGA board has been included to allow you to display grabbed images.

## MsysControl()

- You can control the maximum amount of time (in seconds) that the Host will wait for a synchronous function to return before generating a time-out error. The default is 20 seconds. The **ControlType** can be set to the following:

Controltype	ControlValue
M_TIMEOUT	Set the maximum amount of time (in seconds) that the Host will wait for a synchronous function to return before generating a time-out error.
	M_INFINITE or value in seconds
	M_DEFAULT. 20 seconds.

- The NOA is a neighborhood operation accelerator which can be enabled or disabled when applicable (for example, to reduce power consumption when not needed). The **ControlType** can be set to the following:

ControlType	ControlValue
M_USE_NOA	Can be set to: M_ENABLE (default) or M_DISABLE.

- When copying between an M\_ON\_BOARD buffer and a Host buffer, copies can be driven by the digitizer (bus master) or by the Host. When set to M\_ENABLE (the default for the Main Board and the Processor Board), the copy operation is driven by the digitizer, speeding up the copy and reducing Host intervention. When set to M\_DISABLE (the default for the Genesis-LC), the copies are driven by the Host CPU. To override the default settings on any Matrox Genesis, reset the following **ControlType**:

ControlType	ControlValue
M_BUS_MASTER_COPY_FROM_HOST	When copying from a Host buffer to an M_ON_BOARD buffer, <b>ControlValue</b> can be set to: M_ENABLE or M_DISABLE.
M_BUS_MASTER_COPY_TO_HOST	When copying from an M_ON_BOARD buffer to a Host buffer, <b>ControlValue</b> can be set to: M_ENABLE or M_DISABLE.

Note that, if grabbing to a Host buffer, the digitizer cannot drive the copy simultaneously. The copy will have to be driven by the Host (set **ControlType** to M\_DISABLE). Failure to do this might result in unpredictable results.

- For windowed displays (M\_WINDOWED), a snapshot grab is automatically performed in the true grab buffer at the end of a live grab operation. You can override this default; however, in this case, the true buffer will not contain the grabbed data. This default can be overridden by setting the following **ControlType** to M\_DISABLE:

ControlType	ControlValue
M_LAST_GRAB_IN_TRUE_BUFFER	Can be set to:
	M_ENABLE      Grab last frame in true grab buffer (default)
	M_DISABLE      Don't grab last frame in true grab buffer.

- When displaying a pseudo-live grab operation under Windows, MIL will double-buffer the grab depending on the performance of your Host PC. Double-buffering prevents any frame loss when the grabbed data is copied from the Host to the display. MIL is selective about performing double-buffered pseudo-live grabs because this is more

demanding on your Host CPU. If the default behavior is not appropriate for your application, you can force a specific behavior with the following **ControlType**:

ControlType	ControlValue	
M_DISPLAY_DOUBLE_BUFFERING	Can be set to:	
	M_ENABLE	Force double-buffering.
	M_DISABLE	Don't use double-buffering.
	M_DEFAULT	Use double buffering only when MIL considers it appropriate.

- When a live grab operation uses a software trigger and **MdigHalt()** is issued, a software trigger is automatically generated to invoke a last grab (if you did not disable M\_LAST\_GRAB\_IN\_TRUE\_BUFFER). You can disable this automatic trigger; however, you would then have to issue a software trigger call after the **MdigHalt()** call (these calls must be issued from different threads). To disable the **MdigHalt()** automatic trigger, set the following **ControlType** to M\_DISABLE:
- When a live grab operation uses a hardware trigger, **MdigHalt()** will wait indefinitely for a hardware trigger to invoke a last grab (if you did not disable M\_LAST\_GRAB\_IN\_TRUE\_BUFFER). You can override this default so that **MdigHalt()** will cause an immediate last grab and will not wait for a hardware trigger. To do so, set the following **ControlType** to M\_DISABLE:

ControlType	ControlValue	
M_LIVE_GRAB_END_TRIGGER	Can be set to:	
	M_ENABLE	Default for hardware triggered cameras.
	M_DISABLE	Default for other camera types.

## MsysInquire()

- You can request the following information with the **InquireType** parameter:

<b>InquireType</b>	<b>Description</b>
M_BUS_MASTER_COPY_FROM_HOST	Whether copies from a Host buffer to an M_ON_BOARD buffer are driven by the digitizer (M_ENABLE) or the Host (M_DISABLE).
M_BUS_MASTER_COPY_TO_HOST	Whether copies from an M_ON_BOARD buffer to a Host buffer are driven by the digitizer (M_ENABLE) or the Host (M_DISABLE).
M_BOARD_TYPE	The type of system board: M_GENESIS, M_GENESIS_PRO, M_GENESIS_LC, or M_GENESIS_PLUS.
M_DISPLAY_DOUBLE_BUFFERING	The state of double buffering.
M_LAST_GRAB_IN_TRUE_BUFFER	A last grab is done to the true buffer at the end of a continuous grab: M_ENABLE or M_DISABLE.
M_LIVE_GRAB_END_TRIGGER	Generate automatic trigger at end of grab: M_ENABLE or M_DISABLE.
M_LIVE_GRAB_MOVE_UPDATE	Copy performed between windows on update: M_ENABLE or M_DISABLE.
M_NATIVE_ID	Identifier of the Genesis Native Library device that corresponds to the given MIL system.
M_NATIVE_THREAD_ID	Genesis Native Library identifier of the thread currently used by MIL.
M_PHYSICAL_ADDRESS_UNDERLAY	The physical address of the main (underlay) frame buffer.
M_SERIAL_NUMBER	The serial number of the board.
M_TIMEOUT	The time-out period. Value in seconds, or M_INFINITE.
M_USE_NOA	Whether NOA is in use: M_ENABLE or M_DISABLE.

## Processing operations performed by Host

Most of MIL's processing operations are performed by the on-board Matrox Genesis processor. In some cases, however, processing operations are performed by the Host processor rather than the on-board processor. This is done for one of two reasons: in some cases, operations can be performed more efficiently using the Host; in other cases, the on-board processor is not set up to perform the operation. This default feature can be overridden by setting **MappControl()** with **M\_PROCESSING** to **M\_COMPENSATION\_DISABLE**. In this case, each time Matrox Genesis is unable to perform an operation, an error message is produced.

The following is a list of functions and the circumstances under which their operations will be performed by the Host (by default), instead of the on-board processor.

❖ Note also that functions dealing with JPEG 2000 compressed buffers will be performed by the Host (the functions are not included in the following table).

Operation	Conditions under which Host performs the operation
<b>MblobFill()</b> (M_INCLUDED_BLOBS, M_EXCLUDED_BLOBS, OR M_ALL_BLOB) + M_CONTOUR	■ All operations.
<b>MblobCalculate(),</b> <b>MblobGetResult(),</b> <b>MblobGetResultSingle()</b> M_CHAINS, M_CHAIN_X, M_CHAIN_Y, M_CHAIN_INDEX, OR M_NUMBER_OF_CHAINED_PIXELS	■ All operations
Any feature + (M_SORT1_UP, M_SORT1_DOWN, M_SORT2_UP, M_SORT2_DOWN, M_SORT3_UP, M_SORT3_DOWN, OR M_NO_SORT)	■ All operations
<b>MbufBayer()</b>	■ All operations
<b>MdigFocus()</b>	■ Depth of a 1-bit destination buffer.
<b>MgenWarpParameter()</b>	■ 32-bit LUT buffers.

Operation	Conditions under which Host performs the operation
<b>MimArith()</b> (M_ADD, M_SUB, M_ADD_CONST, M_SUB_CONST, M_CONST_SUB, or M_SUB_ABS) + M_SATURATION	<ul style="list-style-type: none"> <li>■ Floating-point buffers.</li> <li>■ Mix of 32-bit buffer types.</li> </ul>
(M_DIV or M_DIV_CONST) + M_FIXED_POINT	<ul style="list-style-type: none"> <li>■ Either source is 32-bit.</li> <li>■ Source is 8-bit and destination is 32-bit.</li> </ul>
M_CONST_DIV + M_FIXED_POINT	<ul style="list-style-type: none"> <li>■ All operations.</li> </ul>
<b>MimArithMultiple()</b>	
M_OFFSET_GAIN or M_WEIGHTED_AVERAGE	<ul style="list-style-type: none"> <li>■ 32-bit buffers.</li> <li>■ One of the buffers is signed.</li> <li>■ Destination buffer's depth is equal to or less than that of the source buffer.</li> <li>■ Source buffer of different types.</li> </ul>
M_MULTIPLY_ACCUMULATE_1	<ul style="list-style-type: none"> <li>■ 32-bit integer buffers.</li> </ul>
M_MULTIPLY_ACCUMULATE_1 + M_SATURATION	<ul style="list-style-type: none"> <li>■ Float buffers.</li> </ul>
M_MULTIPLY_ACCUMULATE_2	<ul style="list-style-type: none"> <li>■ 32-bit integer buffers.</li> <li>■ Source buffers of different signs.</li> </ul>
M_MULTIPLY_ACCUMULATE_2 + M_SATURATION	<ul style="list-style-type: none"> <li>■ Float buffers.</li> </ul>
<b>MimBinarize(), MimClip(), MimCountDifference(), MimLocateEvent()</b>	<ul style="list-style-type: none"> <li>■ Float buffers.</li> </ul>
<b>MimConvert()</b> M_RGB_TO_Y	<ul style="list-style-type: none"> <li>■ All operations.</li> </ul>
<b>MimConnectMap(), MimDilate(), MimDistance(), MimErode(), MimThin(), MimThick(), and MimLabel()</b>	<ul style="list-style-type: none"> <li>■ Buffers of depth greater than 16 bits.</li> </ul>



Operation	Conditions under which Host performs the operation
<b>MimConvolve()</b>	<ul style="list-style-type: none"> <li>■ Source or destination buffer of depth 1 bit or greater than 16 bits.</li> <li>■ Float buffers.</li> <li>■ The neighborhood operation accelerator (NOA) is disabled and the kernel is larger than 15x15. <ul style="list-style-type: none"> <li>□ The maximum kernel size can be as large as 31x31 if all the following conditions are met: <ul style="list-style-type: none"> <li>■ Source buffer is unsigned 8-bit.</li> <li>■ Destination buffer is 8- or 16-bit.</li> <li>■ Kernel is signed 8-bit [-128, +127].</li> </ul> </li> </ul> </li> <li>■ The NOA is enabled and any of the following conditions is met: <ul style="list-style-type: none"> <li>□ The kernel width or height is greater than 33.</li> <li>□ The total number of kernel elements exceeds 1024 and the kernel data is 8-bit, or exceeds 512 and the kernel data is 16-bit.</li> </ul> </li> <li>❖ Here are some exceptions to the above rule: <ul style="list-style-type: none"> <li>■ If both the image and kernel are 8-bit, then the kernel width can go up to 64 (but the kernel height is still limited to 33, and the total number of kernel values has the same limit).</li> <li>■ If the kernel is 1xN (that is, the width = 1), then the maximum kernel height is 64.</li> </ul> </li> </ul>
<b>MimEdgeDetect()</b>	<ul style="list-style-type: none"> <li>■ All modes except M_FAST_EDGE_DETECT, M_FAST_ANGLE, and M_FAST_GRADIENT modes.</li> <li>■ Buffers of depth greater than 16 bits.</li> </ul>
<b>MimFindExtreme()</b>	<ul style="list-style-type: none"> <li>■ Float buffers.</li> </ul>
<b>MimFlip()</b>	<ul style="list-style-type: none"> <li>■ Float buffers.</li> <li>■ Source and destination have different depths.</li> </ul>
<b>MimHistogram()</b>	<ul style="list-style-type: none"> <li>■ Float buffers.</li> <li>■ Source with depth greater than 16 bits.</li> </ul>

Operation	Conditions under which Host performs the operation
<b>MimLutMap()</b>	<ul style="list-style-type: none"> <li>■ Float buffers.</li> <li>■ 32-bit source buffer.</li> <li>■ 32-bit signed destination buffer.</li> </ul>
<b>MimMorphic()</b>	<ul style="list-style-type: none"> <li>■ Buffers with depth greater than 16 bits.</li> </ul>
All cases	
M_ERODE, M_DILATE	■ Structuring elements exceeding 15x15 in size.
M_THIN, M_THICK	<ul style="list-style-type: none"> <li>■ Grayscale structuring elements that are not 3x3 in size.</li> <li>■ Binary structuring elements exceeding 15x15 in size.</li> </ul>
M_HIT_OR_MISS	<ul style="list-style-type: none"> <li>■ Grayscale mode.</li> <li>■ Structuring elements exceeding 15x15 in size.</li> </ul>
M_MATCH	■ Grayscale mode.
<b>MimPolarTransform()</b>	■ All operations.
<b>MimProject()</b>	<ul style="list-style-type: none"> <li>■ Float buffers.</li> <li>■ 32-bit source buffers.</li> </ul>
<b>MimRank()</b>	<ul style="list-style-type: none"> <li>■ Any structuring elements that are not predefined.</li> <li>■ When the rank is not minimum, maximum, or M_MEDIAN.</li> <li>■ Buffers of depth greater than 16 bits.</li> </ul>
<b>MimResize() and MimRotate()</b>	<ul style="list-style-type: none"> <li>■ Float buffers.</li> <li>■ When the interpolation mode is either M_BILINEAR or M_BICUBIC and you are using 32-bit source buffers.</li> <li>■ Source buffer is 32-bit and destination buffer is 1-bit or source buffer is 1-bit and destination is 32-bit.</li> </ul>
M_OVERSCAN_DISABLE	■ All operations
<b>MimShift()</b>	■ Float buffers.
<b>MimTransform()</b>	■ Always, except when performing an M_FFT operation on 32-bit signed buffers and the DC component is not centered (that is, the M_CENTER flag is not used).
<b>MimTranslate()</b>	■ Fractional X or Y displacement when source and/or destination buffer are of depth 1 bit or greater than 16 bits.

Operation	Conditions under which Host performs the operation
<b>MimWarp()</b>	<ul style="list-style-type: none"> <li>■ Float buffers.</li> <li>■ When the interpolation mode is either M_BILINEAR or M_BICUBIC and you are using 32-bit source buffers.</li> <li>■ Source buffer is 32-bit and destination buffer is 1-bit or source buffer is 1-bit and destination is 32-bit.</li> </ul>
M_WARP_LUT	■ Interpolation is M_BICUBIC.
M_OVERSCAN_DISABLE	■ All operations.
<b>MimWatershed()</b>	■ All operations.

Note that functions from the processing modules listed below require that the Host performs part of the operations. The Matrox Genesis on-board processor handles internal calls to basic image processing functions unless the conditions require Host intervention (as listed in the previous table).

- Measurement module
- Calibration module
- Optical character recognition module
- Code module
- Geometric model finder module



---

## ***Chapter 4: MIL and the Matrox Meteor-II platform***

*This section discusses features of MIL that are distinct to the five Matrox Meteor-II platforms and ways that optimize each board's performance.*

## Matrox Meteor-II family

The Matrox Meteor-II family is comprised of five acquisition boards: Matrox Meteor-II /Standard, Matrox Meteor-II /Multi-Channel, Matrox Meteor-II /Digital, Matrox Meteor-II /1394, and Matrox Meteor-II /Camera Link. Each Matrox Meteor-II board is available in one or more of the following form factors: PCI, CompactPCI, and PC/104-Plus.

All Matrox Meteor-II boards share two essential features: no on-board display section, and the ability to transfer images in real-time to Host memory.

A number of other features are not available on every Matrox Meteor-II board. These features include the UART, support for the Matrox Meteor-II MJPEG module, support for software triggers, exposure timers, and asynchronous reset mode (represented in the following table).

The following table outlines the features currently available for each member of the Matrox Meteor-II family:

<b>Matrox Meteor-II</b>	<b>STD</b>	<b>MC</b>	<b>DIG</b>	<b>1394</b>	<b>CL</b>
■ Form factor(s):					
□ PCI	x	x	x	x	x
□ CompactPCI	x				
□ PC-104/Plus	x	x	x		
■ Uses standard VGA for display	x	x	x	x	x
■ On-board UART (PCI form factor)	x	x	x		x
■ VMChannel (PCI form factor)	x	x	x		x
■ Supports mono-tap and dual-tap camera configurations		x	x		x
■ Supports multi-tap camera configurations			x		x
■ Supports hardware triggers	x	x	x	x	x
■ Supports software triggers	x	x	x		x
■ Has 2 exposure timers		x	x		x
■ Supports asynchronous reset mode		x	x		x

Matrox Meteor-II	STD	MC	DIG	1394	CL
■ Can support a Matrox Meteor-II MJPEG compression module (all form factors)	x	x			
■ Supports Windows 98 or Windows Me	x	x		x	
■ Can be used as a fingerprint for run-time licenses				x*	
*If you are not using Windows NT as your operating system, a camera must be plugged-in when calling <b>MappAlloc()</b> ; otherwise, the run-time licence will not work.					

---

### UART

Most Matrox Meteor-II boards for PCI also feature an on-board UART (Universal Asynchronous Receiver/Transmitter) that provides an RS-232 serial interface. This allows you to remotely control a camera or a motion control unit, or communicate with a program logic controller (PLC).

---

### Matrox Meteor-II MJPEG module

In addition, certain Matrox Meteor-II boards support the Matrox Meteor-II MJPEG module, an optional board used for lossy and lossless compression and decompression of data in JPEG format. See the section, *Compression and decompression with Matrox Meteor-II MJPEG module*.

Note, the Matrox Meteor-II MJPEG module is required to compress/decompress images in MIL-Lite. However, the Matrox Meteor-II MJPEG module is not required to compress/decompress images in MIL. See the **MbufAlloc...()** function in the *Matrox Imaging Library Command Reference*.

❖ It is important to note that Windows NT/2000 is supported on all members of the Matrox Meteor-II family.

See Appendix A for a data flow diagram for all Matrox Meteor-II boards.

## Types of cameras and supported data format

The number of composite/monochrome, Y/C, or RGB cameras that can be attached to the different Matrox Meteor-II boards depends on the form factor used and the type of board. The following table lists the maximum number of cameras that can be attached for each available form factor of Matrox Meteor-II boards:

Board	Type of Camera	PCI	CompactPCI	PC/104-Plus
■ Matrox Meteor-II /Standard*	monochrome RS-170/CCIR	12	7	12
	composite NTSC/PAL	12	7	12
	Y/C	6	3	6
■ Matrox Meteor-II /Multi-Channel	standard/ non-standard monochrome	6		6
	RGB	2		2
■ Matrox Meteor-II /Digital	digital monochrome	4		4
	digital RGB	1		1
■ Matrox Meteor-II /1394	digital IEEE 1394 standard	>=1**		
■ Matrox Meteor-II /Camera Link	digital monochrome	2		
	digital RGB	1		

\*Note, Rev. 1 of Matrox Meteor-II /Standard supports only 4 monochrome or 2 Y/C cameras.

\*\*More than one independent IEEE 1394 DCAM-compliant camera can be attached to the Matrox Meteor-II /1394 board, but each 1394 DCAM-compliant camera must be allocated as a separate digitizer. When attaching multiple cameras, they must comply with an IEEE 1394 tree topology.



---

## Using Matrox Meteor-II with MIL

To use a Matrox Meteor-II board with MIL, you must allocate it as a Meteor-II system using

**MsysAlloc**(M\_SYSTEM\_METEOR\_II,...) or

**MsysAlloc**(M\_SYSTEM\_METEOR\_II\_DIG,...) or

**MsysAlloc**(M\_SYSTEM\_METEOR\_II\_1394,...) or

**MsysAlloc**(M\_SYSTEM\_METEOR\_II\_CL,...). This establishes a MIL system environment in which MIL not only uses the Matrox Meteor-II board and Host computer memory, but also any available graphics controller. The graphics controller has been included to allow you to display grabbed images (none of the Matrox Meteor-II boards has an on-board display section). For more information on using your graphics controller, see the Chapter, *MIL and the VGA platform*.

It is important to note that more than one system can be allocated on a Matrox Meteor-II /1394 board. For a discussion on how to use 1394-compliant cameras on a system, see the subsection titled *Grabbing from multiple 1394 DCAM-compliant cameras* in the section, *IEEE 1394 serial bus-specific features*.

This chapter relates Matrox Meteor-II systems to the following: *Transfers to display*, *Grabbing to a Host buffer with Matrox Meteor-II /Digital or /Camera Link*, and *Compression and decompression with Matrox Meteor-II MJPEG module*, and *Particularities of existing MIL functions on Matrox Meteor-II*.

Refer to *milmet2.txt* file in the \MATROX IMAGING\DRIVERS\DOC (or user-specified) directory for any additions/modifications to these board specific notes.

## Transfers to display

The following table indicates when grab and display are performed live or pseudo-live on the MGA G200, G400, and G450 series. For more information on grabbing and displays, see the *MIL User Guide* manual.

Pixel depth of display mode	Grab-and-display type using Matrox display controller	
	Monochrome Camera	Color Camera
8-bit/pixel	live*	pseudo-live
15-bit/pixel	pseudo-live	pseudo-live
16-bit/pixel	pseudo-live	pseudo-live
24-bit/pixel	live	live
32-bit/pixel	live	live

\*Matrox Meteor-II/Digital and /Camera Link can only grab live with a monochrome camera that has a pixel depth of 8 bits.

The following table indicates when grab and display are performed live or pseudo-live on Matrox Meteor-II /1394 for different MGA controllers:

Graphics Controller	Grab-and-display type using Matrox Meteor-II /1394	
	Monochrome Camera	UYVY Camera
Matrox G200	pseudo-live	pseudo-live
Matrox G400	live	live
Matrox G450	live	live
Cyrix companion chip (on Matrox 4Sight)	pseudo-live	live

## Grabbing from multiple cameras with different DCFs using Matrox Meteor-II /Multi-Channel

In some applications, you might want to grab from multiple cameras with different DCFs that are attached to the same digitizer. This procedure normally involves allocating a digitizer, grabbing the required frame, freeing the digitizer, and then allocating the digitizer again with the second DCF; this can increase the time required for operation. On Matrox Meteor-II /Multi-Channel, MIL can circumvent this problem by using a fast DCF-switching technique which is outlined in the steps below:

1. Make as many calls to **MdigAlloc()** as you have cameras, with different formats, from which you want to grab. With each call, the **DigNum** parameter must be set to **M\_DEV0**, since there is only one physical digitizer; each allocation only sets up a virtual instance of the digitizer.
  2. Specify all required digitizer settings using **MdigControl()**, **MdigChannel()**, **MdigReference()**, and **MdigHookFunction()** for each allocated digitizer. Each time a different digitizer is specified, it retains all settings previously specified.
  3. Call **MdigGrab()** with any digitizer identifier. If this call uses a digitizer identifier different from the previous call, a camera switch will occur.
- ❖ If there is a grab in progress on one digitizer, calling any of the following functions with any other digitizer will result in an error: **MdigGrab()**, **MdigGrabContinuous()**, **MdigChannel()**, **MdigControl()**, **MdigInquire()**, **MdigReference()**, **MdigLut()**, **MdigHookFunction()**, **MdigAlloc()**, and **MdigFree()**.

See the example *mdigmultformat.c*.

---

## Grabbing to a Host buffer with Matrox Meteor-II /Digital or /Camera Link

When grabbing to a Host buffer, Matrox Meteor-II /Digital and Matrox Meteor-II /Camera Link use video transfer memory, to temporarily store the grabbed data while the board waits for access to the PCI bus. Loss of image data could otherwise occur during long bus-access latencies, found in heavily loaded systems.

The two boards use hardware/software transfer control logic to double-buffer the grab between two temporary buffers created within the video transfer memory. This logic toggles the grab between one temporary buffer and the other: it transfers to the Host, grabbed data from the buffer into which data is not currently being grabbed. The double-buffering is done on a line-block basis rather than on a frame basis.

In general, the default size of the temporary buffers provides a good compromise between maintaining a small latency before the beginning of the transfer and minimizing the transfer's dependency on the PCI bus load. However, when the PCI bus is extremely loaded, the Matrox Meteor-II /Digital (or Matrox Meteor-II /Camera Link) board might not be able to maintain the grab and transfer sequence. If the request to transfer is not serviced in time, the grab might overwrite non-transferred data. This is known as a *grab over-run*.

For example, a typical load on the PCI bus is that of the Matrox Meteor-II /Digital (or Matrox Meteor-II Camera Link) zooming grabbed data from a high frequency camera before transferring it. A digital camera with a frequency of 40 MHz presents no problem on its own, but zooming by a factor of four would require a bandwidth of 160 Mbytes/sec. The PCI bandwidth has a limit of 133 Mbytes/sec. This situation would result in a grab over-run.

## Optimizing the use of the frame buffers

If you experience grab over-runs, try the following solutions in the specified order:

1. Increase the default temporary buffer size. This should be enough to resolve your problems.
2. If possible, double-buffer the grab on a frame basis rather than a line-block basis. In this case, you have to manage the double-buffering of the grab yourself. This method involves one frame of latency before the beginning of the transfer.
3. Reduce the PCI bus load.

See the subsection *An Example* which outlines the process of allocating buffers in video transfer memory.

## Increasing the default temporary buffer size

You can increase the default temporary buffer size. Larger temporary buffers make the grab's transfer-to-Host less dependent on the PCI bus load. That is, the grab is less affected by long bus-access latencies, found in heavily loaded systems, because more memory is available to buffer the data. Note, however, that larger temporary buffers also increase the latency before the beginning of the transfer.

You can increase the default temporary buffer size by adjusting the **SizeOfTmpBufferForHostGrab** field in the *genesis.ini* file in the `\GENESIS\SHELL` directory.

## Grabbing in on-board buffers

If increasing the size of the temporary buffers does not resolve your over-run problem, you can try double-buffering the grab on a frame basis rather than a line-block basis. The grab is then less dependent on the PCI bus load because the buffers don't have to be transferred as frequently and intermittent bus latencies are averaged over an entire frame. The downside of this process is that there is one frame of latency before the beginning of the transfer.

To implement this model, you have to manage the double-buffering. To do so, allocate two frame-sized buffers on-board and then grab a field/frame into one buffer while copying the other buffer to the Host. Since you control the synchronization between the grab and transfer, you can detect when something goes wrong.

- ❖ Note that, by default, the copy operation is driven by the Host CPU. To speed up the copy and reduce Host intervention, enable board driven (VIA-driven) copies by setting the **MsysControl()** `M_BUS_MASTER_COPY_TO_HOST` control to `M_ENABLE`.

This method is only possible if sufficient on-board memory is available to allocate the two buffers.

### **Grabbing large frames of data on Matrox Meteor-II /Digital**

When grabbing large frames of data (for example, 2K X 2K and 4K X 4K), there is no way to store the whole frame in video transfer memory. In this case, you can only grab to a Host buffer. This restriction does not apply to Matrox Meteor-II /Camera Link, since 32 Mbytes of video transfer memory are available.

## An example

The following example shows how to allocate buffers in video transfer memory and then grab in these buffer.

```

/* File name: mmet2dig.c
* Synopsis: This program allocates 2 grab buffers on the
* SGRAM memory of the Matrox Meteor-II/Digital board and then grabs
* in them.
* The data is then transferred to the HOST using
* the bus-master copy.
*/

/* headers */
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <mil.h>

/* Main function. */
void main(void)
{
    MIL_ID MilApplication;
    MIL_ID MilSystem;
    MIL_ID MilDigitizer;
    MIL_ID MilDisplay;
    MIL_ID MilImageOnBoard[2];
    MIL_ID MilImageDisp;

    /* Allocations. */
    MappAlloc(M_DEFAULT, &MilApplication);
    MsysAlloc(M_SYSTEM_METEOR_II_DIG, M_DEF_SYSTEM_NUM, M_SETUP, &MilSystem);

```

(cont...)

```

MdigAlloc(MilSystem, M_DEFAULT, M_DEF_DIGITIZER_FORMAT,
          M_DEFAULT, &MilDigitizer);
MdispAlloc(MilSystem, M_DEFAULT, M_DEF_DISPLAY_FORMAT, M_DEFAULT,
          &MilDisplay);

/* Allocate a display buffer. */
MbufAlloc2d(MilSystem,
            (long)(MdigInquire(MilDigitizer, M_SIZE_X, M_NULL)),
            (long)(MdigInquire(MilDigitizer, M_SIZE_Y, M_NULL)),
            8L+M_UNSIGNED,
            M_IMAGE+M_PROC+M_DISP+M_NON_PAGED, &MilImageDisp);
MbufClear(MilImageDisp, 0);

/* Allocate 2 grab buffers on the on-board memory of the board. */
MbufAlloc2d(MilSystem,
            (long)(MdigInquire(MilDigitizer, M_SIZE_X, M_NULL)),
            (long)(MdigInquire(MilDigitizer, M_SIZE_Y, M_NULL)),
            8L+M_UNSIGNED,
            M_IMAGE+M_GRAB+M_PROC+M_ON_BOARD, &MilImageOnBoard[0]);
MbufAlloc2d(MilSystem,
            (long)(MdigInquire(MilDigitizer, M_SIZE_X, M_NULL)),
            (long)(MdigInquire(MilDigitizer, M_SIZE_Y, M_NULL)),
            8L+M_UNSIGNED,
            M_IMAGE+M_GRAB+M_PROC+M_ON_BOARD, &MilImageOnBoard[1]);

/* Enable bus master copies from the Matrox Meteor2 to the host. */
MsysControl(MilSystem, M_BUS_MASTER_COPY_TO_HOST, M_ENABLE);

/* Display the buffer. */
MdispSelect(MilDisplay, MilImageDisp);

printf("Press enter to grab into the first buffer\n");
getchar();

/* Grab and copy to the host. */
MdigGrab(MilDigitizer, MilImageOnBoard[0]);
MbufCopy(MilImageOnBoard[0], MilImageDisp);

printf("Press enter to grab into the second buffer\n");
getchar();

/* Grab and copy to the host. */
MdigGrab(MilDigitizer, MilImageOnBoard[1]);
MbufCopy(MilImageOnBoard[1], MilImageDisp);

```

(cont...)



```

getchar();

/* Free allocations. */
MbufFree(MilImageOnBoard[0]);
MbufFree(MilImageOnBoard[1]);
MbufFree(MilImageDisp);
MdispFree(MilDisplay);
MdigFree(MilDigitizer);
MsysFree(MilSystem);
MappFree(MilApplication);
}

```

## Compression and decompression with Matrox Meteor-II MJPEG module

Hardware image compression and decompression are supported only on the Matrox Meteor-II /Standard and Matrox Meteor-II /Multi-Channel boards and only when using a Matrox Meteor-II MJPEG module.

### Compression

The Matrox Meteor-II MJPEG module can compress data from a monochrome source (live or buffer-driven) into the following destination formats:

- 8-bit monochrome (M\_JPEG\_LOSSY, M\_JPEG\_LOSSY\_INTERLACED, M\_JPEG\_LOSSLESS, and M\_JPEG\_LOSSLESS\_INTERLACED).
- M\_YUV16+M\_PACKED (M\_JPEG\_LOSSY only).

The Matrox Meteor-II MJPEG module can compress data from a color source (live or buffer-driven) into the following destination formats:

- 8-bit monochrome (M\_JPEG\_LOSSY, M\_JPEG\_LOSSY\_INTERLACED, M\_JPEG\_LOSSLESS, and M\_JPEG\_LOSSLESS\_INTERLACED).
- M\_YUV16+M\_PACKED (M\_JPEG\_LOSSY and M\_JPEG\_LOSSY\_INTERLACED).

- ❖ If the Matrox Meteor-II MJPEG module is compressing data from a color buffer, it is more efficient if the buffer is in M\_RGB24+M\_PACKED format. If the image data in the buffer is not in the right format, then an internal conversion will be done.

## Decompression

The Matrox Meteor-II MJPEG module can decompress data from a monochrome or color compressed image buffer into the following destination formats:

- 8-bit monochrome.
  - M\_YUV16+M\_PACKED (only from a color compressed image).
  - M\_RGB16+M\_PACKED (only on Matrox Meteor-II /Multi-Channel for PC/104-*Plus*).
  - M\_BGR24+M\_PACKED.
  - M\_BGR32+M\_PACKED.
  - M\_RGB24+M\_PLANAR.
- ❖ The Matrox Meteor-II MJPEG module on a Matrox Meteor-II /Standard board can only decompress a buffer that was compressed in the M\_JPEG\_LOSSY\_INTERLACED or M\_JPEG\_LOSSLESS\_INTERLACED formats. M\_JPEG\_LOSSY and M\_JPEG\_LOSSLESS buffers will be decompressed by the Host.

## IEEE 1394 serial bus-specific features

Although the Matrox Meteor-II /1394 is Open Host Controller Interface (OHCI) compliant, MIL can use Matrox Meteor-II /1394 to acquire serial data only from a IEEE 1394 DCAM-compliant camera. Currently, data transfer rates of up to 400 Mbits per second (Mbps) are possible.

In MIL, to access and acquire data from an IEEE 1394 DCAM-compliant camera, allocate it as a digitizer (**MdigAlloc()**) on a Matrox Meteor-II /1394 system. To access and acquire data from multiple IEEE 1394 DCAM-compliant cameras, refer to the subsection titled *Grabbing from multiple 1394 DCAM-compliant cameras* later in this section.

### Video formats and modes

Matrox Meteor-II /1394 supports monochrome and color IEEE 1394 DCAM-compliant cameras. Most 1394-digital cameras support one or more of four standard formats: format 0, 1, 2, and 7. The format determines the range of maximal image/screen resolutions supported (see the table below):

Format	Resolution Range (Maximum)
0	[160x120, 640x480] (640x480)
1	[800x600, 1024x768] (1024 x 768)
2	[1280x960, 1600x1200] (1600 x1200)
7	Scalable image size format

Each camera format can support a number of modes. Each format/mode combination corresponds to a specific camera setting specifying a resolution, data format, and internal storage size (as per the IEEE 1394 Digital Camera Specification).

For formats 0, 1, and 2, MIL supports all YUV 4:1:1, YUV 4:2:2, and 1-band, 8-bit modes (at all frame rates) on the Matrox Meteor-II /1394.

Format 0 includes two format/mode combinations which approximate the RS-170 (monochrome) and NTSC (color) video format standards, respectively. These video formats are standard on most cameras.

❖ Note that the MIL Matrox Meteor-II /1394 driver does not support native RGB and YUV 4:4:4 camera modes. Use **MdigInquire()** with `M_FORMAT_SUPPORTED` to generate a list of all supported data formats/modes for your 1394 digital camera.

Also, note that on the Matrox Meteor-II/1394, YUV 4:2:2 data is grabbed in UYVY format.

## An example

The following example shows how to inquire and print the supported strings.

```
#include "mil.h"
#include "stdio.h"
#include "string.h"

void main(void)
{
    MIL_ID MilApplication, /* Application identifier. */
    MilSystem,             /* System identifier. */
    MilDigitizer;          /* Digitizer identifier. */

    /* Allocate defaults. */
    MappAlloc(M_DEFAULT, &MilApplication);
    MsysAlloc(M_SYSTEM_METEOR_II_1394, M_DEFAULT, M_SETUP, &MilSystem);
    MdigAlloc(MilSystem, M_DEFAULT, "M_DEFAULT", M_DEFAULT, &MilDigitizer);

    /* Inquire number of strings and length */
    long NbFormats = MdigInquire(MilDigitizer, M_FORMAT_SUPPORTED_NUM,
    M_NULL);
    long MaxLength = MdigInquire(MilDigitizer, M_FORMAT_SUPPORTED_LENGTH,
    M_NULL);

    /* Allocate a buffer to receive the strings */
    char * Strings = new char[NbFormats*MaxLength];
    memset(Strings, 0, NbFormats*MaxLength*sizeof(char));

    /* Inquire the strings */
    MdigInquire(MilDigitizer, M_FORMAT_SUPPORTED, Strings);

    long Offset = 0;
    for(long t = 0; t < NbFormats; t++)
    {
        /* Print each string */
        printf("%s\n", &Strings[Offset]);
        Offset += MaxLength;
    }
    delete [] Strings;

    printf("Press <Enter> to end.\n");
    getchar();

    /* Free MIL objects */
    MdigFree(MilDigitizer);
    MsysFree(MilSystem);
    MappFree(MilApplication);
}
```

## Read/write capabilities

The read and write capabilities of Matrox Meteor-II /1394 are supported through the standard MIL digitizer inquire and control functions.

Use **MdigInquire()** to read the actual, minimum, or maximum value of any status register feature. Use **MdigControl()** and **MdigReference()** to modify the actual value of a status or control register feature.

## Setting camera features

---

### *Setting general camera controls*

There are both general and specific control features for a 1394 DCAM-compliant digital camera. The general camera controls include brightness, hue, saturation and (input) gain. For portability, you should set/control these standard control features in the conventional way using **MdigReference()** with M\_BRIGHTNESS, M\_HUE\_REF, or M\_SATURATION\_REF, and using **MdigControl()** with M\_GRAB\_INPUT\_GAIN.

On a Matrox Meteor-II /1394, the **MdigReference()** function will map the smallest value supported by the camera to M\_MIN\_LEVEL, and map the largest value supported by the camera to M\_MAX\_LEVEL. The number of values supported by your camera can differ such that consecutive reference-level settings might produce the same result. See the **MdigReference()** function in the *Matrox Imaging Library Command Reference* for more details on reference levels.

If you need more flexibility when setting these features, you can use the method described below for specific camera controls.



## Grabbing from multiple 1394 DCAM-compliant cameras

Matrox Meteor-II /1394 supports acquisition from multiple IEEE 1394 DCAM-compliant cameras attached in an IEEE 1394 topology. However, there are issues with respect to both the PCI bus and the IEEE 1394 bus that restricts the actual number of cameras. Refer to the *Hardware reference* chapter in the *Matrox Meteor-II / 1394 Installation and Hardware Reference* manual.

---

### *In general*

To access and grab from multiple cameras, allocate each camera as a digitizer on a Matrox Meteor-II /1394 system (**MdigAlloc()**). With each call, the **DigNum** parameter must be set to M\_DEVn, where n is the camera number, starting from 0 (M\_DEV0). You can then perform simultaneous grabs from each of these cameras, hardware permitting, by making multiple calls to **MdigGrab()** with the appropriate MIL digitizer (camera) identifiers. Note however, it is not possible to perform multiple continuous live grabs to displayed buffers when grabbing from digitizers on the same system.

---

### *Multiple continuous live grabs*

To perform multiple continuous live grabs, allocate a Matrox Meteor-II /1394 system for each camera (**MsysAlloc()**) and then allocate a camera as a digitizer on each respective system. With each call, the **DigNum** parameter must be set to M\_DEVn, where n is the camera number, starting from 0 (M\_DEV0). You can then perform simultaneous continuous grabs from each of these cameras, hardware permitting, by making multiple calls to **MdigGrabContinuous()** with the appropriate MIL digitizer (camera) identifiers.

See the *mdblgrab.c* example that is accessible from the `\MATROX IMAGING\MIL\EXAMPLES\METEOR_II_1394` directory on the MIL CD.

In contrast to other Matrox boards (with the exception of the VGA), multiple Meteor-II /1394 systems are possible on a Matrox Meteor-II /1394 board because the board is only an IEEE 1394-to-PCI adapter.



---

*Establishing a pool of  
IEEE 1394  
DCAM-compliant  
cameras*

All cameras that are attached to a Matrox Meteor-II /1394 board are detected when the first **MsysAlloc()** function call is made. This establishes a "pool" of cameras. Any cameras attached after the "pool" has been established cannot be accessed (allocated). **Warning:** you cannot unplug a camera that has been established within the "pool" before freeing up the system with the **MsysFree()** function.

---

*Assigning a  
camera/digitizer to a  
system*

Although cameras in the pool are detected when the first **MsysAlloc()** function call is made, these cameras are not associated to an allocated system. To be used on a system, a camera from the pool must be assigned to one of the available Meteor-II /1394 systems using **MdigAlloc()**.

A camera in the pool also has no fixed device number. Accordingly, you need to assign an appropriate device number to **MdigAlloc()**. To accomplish this:

1. Inquire the number of digitizers that can be allocated on the system using **MsysInquire()** with M\_DIGITIZER\_NUM. This returns the number of digitizers already allocated on the system, in addition to the number of unassigned cameras (digitizers) in the pool; in other words, it returns the total number of digitizers.
2. Use a device number that is less than this inquired value, and that has not already been used for another digitizer on the system.

### Important points to consider...

- ❖ Once a 1394-compliant camera is allocated as a digitizer on a MIL system, the camera (digitizer) then belongs to the system, until it is freed with **MdigFree()**. When a digitizer is freed, it is returned to the common pool of available digitizers (cameras) that is shared by all Meteor-II /1394 systems.
- ❖ Windows NT 4.0 does not support the IEEE 1394 interface; however by installing the MIL Meteor-II /1394 driver, you can use IEEE 1394 DCAM-compliant cameras through MIL. Once you install this driver, you can only use your unit's IEEE 1394 interface to attach IEEE 1394 DCAM-compliant cameras, and no other IEEE 1394 devices. In addition, you will only be able to use these cameras through MIL.
- ❖ Windows 2000 supports the IEEE 1394 interface, however you must install the MIL Matrox Meteor-II /1394 driver to use IEEE 1394 DCAM-compliant cameras through MIL. Once installed, you can still interface any IEEE 1394 device to the IEEE 1394 interface. However, you will only be able to use your IEEE 1394 DCAM-compliant cameras through MIL.

## Particularities of existing MIL functions on Matrox Meteor-II

Certain commands have special features or functionality on a Meteor-II system. This table provides an overview of the affected commands.

Commands	Meteor-II particularities
MbufAlloc...()	Buffer constraints and options.
MbufCreate...()	Required parameter settings.
MbufInquire()	Matrox Meteor-II-specific inquire options.
MdigAlloc()	Digitizer configuration format (DCF) specifications.
MdigChannel()	Required parameter settings and restrictions.
MdigControl()	Control type and value additions and restrictions.
MdigGrab/ MdigGrabContinuous()	Destination buffer restriction.
MdigGrabWait()	1394-specific restrictions
MdigHookFunction()	Hook restrictions.
MdigInquire()	Matrox Meteor-II-specific inquire options.
MdigLut()	Matrox Meteor-II-specific output formats.
MdigReference()	Matrox Meteor-II features and restrictions.
MsysAlloc()	Matrox Meteor-II-specific allocation options.
MsysControl()	Control type and flag additions.
MsysInquire()	Matrox Meteor-II-specific inquire options.

Details and procedures are described in the individual command descriptions, which follow. Information that is applicable to a particular version of the board is denoted in a table by an 'x'. Any feature or option that is specific to a certain form factor will also be marked by an '\*'. No special denotation will be given when a feature is supported on all available form factors.

## MbufAlloc...()

- On Matrox Meteor-II /Standard and /Multi-Channel, only 8-bit monochrome and 3-band 8-bit color grab buffers can be allocated. On Matrox Meteor-II /Digital and /Camera Link, up to 4-band color buffers can be allocated using **MbufAllocColor()**.
- You can add one of the following to the **Attribute** parameter:

Attribute parameter for 3-band 8-bit buffer	Internal storage format for 3-band 8-bit buffer	STD	MC	DIG/ CL	1394
M_BGR24 + M_PACKED	24-bit (BGR) packed pixels. (Default for M_DISP buffers.)	x	x		x**
M_RGB15 + M_PACKED	15-bit (RGB) packed pixels.				x**
M_RGB16 + M_PACKED	16-bit (RGB) packed pixels.	x	x*		x**
M_RGB4+M_PACKED	24-bit (RGB) packed pixels.	x	x		x**
M_RGB24 + M_PLANAR	24-bit (RGB) planar pixels. (Default for non M_DISP buffers.)	x	x		x**
	24-bit (RGB) planar pixels. (Default in all cases.)			x	x**
M_BGR32 + M_PACKED	32-bit (BGR) packed pixels.	x	x		x**
M_YUV16 + M_PACKED	YUV16 (4:2:2) packed standard.	x			x
M_YUV16_YUYV + M_PACKED	YUV16 (4:2:2) packed standard.	x			x**
M_YUV16_UYVY + M_PACKED	YUV16 (4:2:2) packed standard.				x
M_YUV12 + M_PLANAR	YUV12 planar format.	x*			x**
M_YUV16 + M_PLANAR	YUV16 planar format.	x*			x**
M_YUV24 + M_PLANAR	YUV24 planar standard.	x			x**
*Note, only available on the PC/104-Plus form factor.					
**If you grab into one of these buffers, the grab can only be performed in pseudo-live mode.					

- For detailed information on compression and decompression, refer to the section: *Compression and decompression with Matrox Meteor-II MJPEG module*.
- ❖ Note that it might be slower to process buffers with M\_PACKED attributes.

### MbufCreate...()

- Set the **Attribute** parameter to an M\_IMAGE combination (for example, M\_IMAGE + M\_DISP + M\_GRAB + M\_PROC).
- Note that the following **ControlFlag** combinations are supported on Matrox Meteor-II boards:

ControlFlag	Description	STD	MC	DIG/CL	1394
M_PHYSICAL_ADDRESS+M_PITCH	<b>ArrayOfDataPtr</b> is an array of physical addresses. The pitch is in pixels (default).	x	x	x	x
M_PHYSICAL_ADDRESS+M_PITCH_BYTE	<b>ArrayOfDataPtr</b> is an array of physical addresses. The pitch is in bytes.	x	x	x	x
M_HOST_ADDRESS+M_PITCH	<b>ArrayOfDataPtr</b> is an array of Host addresses. The pitch is in pixels (default).			x	x
M_HOST_ADDRESS+M_PITCH_BYTE	<b>ArrayOfDataPtr</b> is an array of Host addresses. The pitch is in bytes.			x	x
M_BUF_ID	This allows you to allocate a multi-band buffer from an array of single-band buffers. <b>ArrayOfDataPtr</b> will be a pointer to an array of MIL buffer identifiers. One identifier per band must be provided. <b>SizeX</b> , <b>SizeY</b> , <b>Type</b> , <b>Attribute</b> , and <b>Pitch</b> must be the same for each buffer. <b>Pitch</b> can be set to M_DEFAULT.			x	

- For detailed information on compression and decompression, refer to the section: *Compression and decompression with Matrox Meteor-II MJPEG module*.

MbufInquire()

- The **InquireType** parameter can also be set to one of the following on all Matrox Meteor-II boards:

InquireType	Description	STD	MC	DIG /CL	1394
M_NATIVE_ID	Identifier of the <i>Genesis Native Library</i> buffer that corresponds to the specified MIL buffer.			x*	
M_CURRENT_BUF_ID	Identifier of the buffer in which data is currently being grabbed. Valid only for windowed displays. This information is used, for example, to access grabbed data while a continuous grab ( <b>MdigGrabContinuous()</b> ) is being displayed live. Under these circumstances, the destination buffer is not the allocated buffer, but rather an associated buffer, used only for display. Note that this buffer's dimensions are not necessarily identical to those of the true destination buffer, but are related to the portion of the buffer being displayed and the display resolution. Changing the display's window size will change the buffer's size.	x	x	x	x
*Note, Matrox Meteor-II /Digital and /Camera Link can be programmed with the Genesis Native Library.					

## MdigAlloc()

- You can allocate multiple digitizers on one Matrox Meteor-II /Multi-Channel system. Make as many calls to **MdigAlloc()** as you have cameras from which you want to grab. With each call, specify the DCF of the camera, and set the **DigNum** parameter to M\_DEV0. Calling **MdigGrab()** with any digitizer identifier will cause a switch and a grab from the identifier's associated camera. See the section, *Grabbing from multiple cameras with different DCFs using Matrox Meteor-II /Multi-Channel* for more information.
- To grab from multiple cameras using Matrox Meteor-II/1394, see the section *Grabbing from multiple 1394 DCAM-compliant cameras*.
- The **DataFormat** parameter is a string variable that provides the data format of the input device. Each board in the Matrox Meteor-II family has different settings for different camera types.

DataFormat (Camera Type)	Description	Color or Monochrome	STD	MC	DIG /CL	1394
"M_DEFAULT"	Same as "M_RS170"	Mono	x	x		
	The format will be the optimal format available on the camera. This will correspond to: Color data format (if available) largest image dimensions possible, or best frame rate.	Color/Mono				x
"M_RS170"	RS-170, 640x480, 8 bits, 12.5 MHz, analog	Mono	x	x		
	640x480, 8 bits, or the closest available mode.	Mono				x
"M_RS170_VIA_RGB"	RS-170, 640x480, 8 bits, 12.5 MHz, analog	Mono		x		
"M_CCIR"	CCIR, 768x576, 8 bits, 14.8 MHz, analog	Mono	x	x		
"M_CCIR_VIA_RGB"	CCIR, 768x576, 8 bits, 14.8 MHz, analog	Mono		x		

DataFormat (Camera Type)	Description	Color or Monochrome	STD	MC	DIG /CL	1394
"M_NTSC"	NTSC, 640x480, 3x8 bits, 12.5 MHz, composite	Color	x			
	RS-170 RGB, 640x480, 3x8 bits, 12.5 MHz	Color		x		
	640x480, (YUV 4:2:2 or YUV 4:1:1), or the closest available mode.	Color				x
"M_NTSC_YC"	RS-170 Y/C(SVHS), 640x480, 3x8 bits, 12.5 MHz	Color	x			
"M_NTSC_RGB"	RS-170 RGB, 640x480, 3x8 bits, 12.5 MHz	Color		x		
"M_PAL"	PAL I, 768x576, 3x8 bits, 14.8MHz, composite	Color	x			
	PAL RGB, 768x576, 3x8 bits, 14.8MHz	Color		x		
"M_PAL_YC"	PAL Y/C, 768x576, 3x8 bits, 14.8 MHz	Color	x			
"M_PAL_RGB"	PAL RGB, 768x576, 3x8 bits, 14.8MHz	Color		x		
*dcf	A DCF file that specifies the input signal format. See the <i>MATROXIMAGING \DRIVERS \Board's name \DCF</i> directory for the current list of supported formats.		x	x		



DataFormat (Camera Type)	Description	Color or Monochrome	STD	MC	DIG /CL	1394
*dcf	A DCF file that specifies the input signal format. See the \GENESIS\DCF directory for the current list of supported formats on the Matrox Meteor-II /Digital and /Camera Link.				x	
M_XXY_DATA[@Z[FPS]]	<p>The string format, where <i>X</i> is SizeX in pixels, <i>Y</i> is SizeY in pixels, <i>data</i> is the image data format, and <i>z</i> is the frame rate; the information in brackets is optional. For example, M_640X480_YUV411@30FPS, M_320X240_YUV422@7.5, and M_1024X768_Y are accepted strings for one type of camera. Note, if no frame rate is specified, the fastest available frame rate will be utilized. Use the M_FORMAT_SUPPORTED inquire type to generate a list of all supported string formats that correspond to your 1394 digital camera.</p>					x

MdigChannel()

The number of independent composite/monochrome, Y/C or RGB cameras that can be attached to the available channels on different Matrox Meteor-II boards depends on the form factor used and the type of board. The table below lists the maximum number of channels available on each of the Matrox Meteor-II boards.

Board	PCI	CompactPCI	PC/104-Plus
■ Matrox Meteor-II /Standard	12	7	12
■ Matrox Meteor-II /Multi-Channel	6		6
■ Matrox Meteor-II /Digital	4		4
■ Matrox Meteor-II /1394	1*		
■ Matrox Meteor-II /Camera Link	2		
*Note, Matrox Meteor-II /1394 supports the IEEE tree topology for 1394 devices. More than one 1394-compliant camera can be attached to the Matrox Meteor-II /1394 board, and each camera must be allocated as a separate digitizer. However, each digitizer has only 1 channel available.			

- The color mode used, specified in the DCF, determines the channels available for switching. To switch between cameras of the same camera type (for Matrox Meteor-II /Camera Link, see the tables following this one), use:

Camera Type	Channel	Signal/Sync input	STD	MC	DIG	1394
Any	M_DEFAULT	Same as M_CH0.	x	x	x	x
RGB	M_CH0	VID1_IN1, VID1_IN2, VID1_IN3 (data signals)		x		
		DATA, INPUT 0-24 (data signals)			x	
	M_CH1	VID2_IN1, VID2_IN2, VID2_IN3 (data signals)		x		
	M_RGB	VID1_IN1, VID1_IN2, VID1_IN3 (data signals) and SYNC_IN (sync signal)		x		
Y/C	M_CH0	(Y camera 1) VID_IN1 and (C camera 1) VID_IN2	x			x
	M_CH1	(Y camera 2) VID_IN3 and (C camera 2) VID_IN4	x			
	M_CH2	(Y camera 3) VID_IN6 and (C camera 3) VID_IN7	x			
	M_CH3	(Y camera 4) VID_IN8 and (C camera 4) VID_IN5	x			
	M_CH4	(Y camera 5) VID_IN9 and (C camera 5) VID_IN10	x			
	M_CH5	(Y camera 6) VID_IN11 and (C camera 6) VID_IN12	x			

Camera Type	Channel	Signal/Sync input	STD	MC	DIG	1394
Composite* or Monochrome	M_CH0	VID_IN1	x			x**
		VID1_IN1		x		
		DATA, INPUT 0-7			x	
		1394 INPUT				x
	M_CH1	VID_IN2	x			
		VID1_IN2		x		
		DATA, INPUT 8-15			x	
	M_CH2	VID_IN3	x			
		VID1_IN3		x		
		DATA, INPUT 16-23			x	
	M_CH3	VID_IN4	x			
		DATA, INPUT 24-31			x	
	M_CH4	VID_IN5	x			
		VID2_IN1		x		
	M_CH5	VID_IN6	x			
		VID2_IN2		x		
	M_CH6	VID_IN7	x			
		VID2_IN3		x		
	M_CH7 to M_CH11	VID_IN8 to VID_IN12	x***			

\*Composite cameras refer to the Matrox Meteor-II /Standard only.

\*\*Only M\_CH0 is supported for Matrox Meteor-II /1394.

\*\*\*Matrox Meteor-II /Standard for CompactPCI supports up to 7 monochrome cameras only.

- For Matrox Meteor-II /Camera Link, the camera description and the mode, specified in the DCF, determine the channels available for switching, as discussed in the following:
  - In the base mode configuration (one connector per camera), you can switch between two cameras (non simultaneous acquisition):

Camera description	Channel	Signal/Sync Input
8 bit, 1 tap	M_CH0	Connector 1, bit 0-7
	M_CH1	Connector 2, bit 0-7
10-16 bits, 1 tap	M_CH0	Connector 1, bit 0-15
	M_CH1	Connector 2, bit 0-15
8 bits, 2 taps	M_CH0	Connector 1, bit 0-15
	M_CH1	Connector 2, bit 0-15
10-12 bits, 2 taps	M_CH0	Connector 1, bit 0-23*
	M_CH1	Connector 2, bit 0-23*
RGB	M_CH0	Connector 1, bit 0-23
	M_CH1	Connector 2, bit 0-23
*32 bits are used to transfer the 2 taps. Data bits 0-11 (first tap) are transferred to bits 0-15, whereas data bits 12-23 (second tap) are transferred to bits 16-31. Non-data bits are padded with 0.		

- In the medium mode configuration, you can use two connectors per camera. Note that only M\_CH0 is supported in this mode:

Camera description	Channel	Signal/Sync input
8 bits, 2 taps	M_CH0	Connector 1, bit 0-7 and connector 2, bit 0-7
10-16 bits, 2 taps	M_CH0	Connector 1, bit 0-15 and connector 2, bit 0-15 or* Connector 1, bit 0-23 and connector 2, bit 0-7
8 bits, 4 taps	M_CH0	Connector 1, bit 0-15 and connector 2, bit 0-15 or* Connector 1, bit 0-23 and connector 2, bit 0-7
*As specified in the DCF file.		

- In a synchronous mode configuration, you can grab simultaneously from two cameras (the default channel, M\_DEFAULT, is the only channel supported for the data). Note that both cameras must have the same data bit size and the total data must not exceed 32 bits.

When grabbing simultaneously, you should specify one of the two channels for the sync signal.

To override the default channel of the sync signal, add M\_SYNC to the required channel setting (for example, M\_CH1+M\_SYNC).

Camera description		Channel*	Signal/Sync input*
Camera 1	Camera 2		
8 bits, 1 tap	8 bits, 1 tap	M_CH0 M_CH1	Connector 1, bit 0-7 Connector 2, bit 0-7
8 bits, 1 tap	8 bits, 2 taps	M_CH0 M_CH1	Connector 1, bit 0-7 Connector 2, bit 0-15
8 bits, 2 taps	8 bits, 1 tap	M_CH0 M_CH1	Connector 1, bit 0-15 Connector 2, bit 0-7
8 bits, 1 tap	RGB	M_CH0 M_CH1	Connector 1, bit 0-7 Connector 2, bit 0-23
RGB	8 bits, 1 tap	M_CH0 M_CH1	Connector 1, bit 0-23 Connector 2, bit 0-7
10-16 bits, 1 tap	10-16 bits, 1 tap	M_CH0 M_CH1	Connector 1, bit 0-15 Connector 2, bit 0-15
8 bits, 2 taps	8 bits, 2 taps	M_CH0 M_CH1	Connector 1, bit 0-15 Connector 2, bit 0-15

\* Note that the sync signal will be on the specified channel. If no channel is specified, M\_CH0 will be assumed.

- On the Matrox Meteor-II /Multi-Channel, the default channel of the sync signal can be overridden with any of the following:

Sync input channel	Sync input signal
M_CH0	VID1_IN1
M_CH1	VID1_IN2
M_CH2	VID1_IN3
M_CH3	SYNC_IN
M_CH4	VID2_IN1
M_CH5	VID2_IN2
M_CH6	VID2_IN3
M_CH7	SYNC_IN

To override the default channel of the sync signal, add M\_SYNC to the required channel parameter (for example: M\_CH0+M\_SYNC).

### **MdigControl()**

- On Matrox Meteor-II /1394, it is possible to optimize grabs by taking advantage of hardware cropping with a format 7-capable camera. To do so, allocate the camera in format 7. When using the M\_SOURCE\_SIZE... and M\_SOURCE\_OFFSET... control types, the cropping will be done by the camera hardware if it is supported, that is, if the specified offsets and sizes match the boundaries of one of supported subregions.

The **MdigControl()** M\_SOURCE\_COMENSATION control type can be used to disable the software compensation to make sure that hardware cropping is used.

Under any other circumstance, cropping will be done by software.

- For Matrox Meteor-II /Multi-Channel, note the following concerning the M\_GRAB\_EXPOSURE\_CLOCK\_SOURCE control type:
  - A clock source must have a frequency greater than or equal to 1 Hz.
  - A timer cannot clock itself.
  - Only a continuous timer can clock another timer. To clock a timer, make the following call:

```
MdigControl(MilDigitizer, M_GRAB_EXPOSURE_CLOCK_SOURCE+M_TIMERn, M_CONTINUOUS)
```

Note that M\_TIMERm and M\_TIMERn can be equal to either M\_TIMER1 or M\_TIMER2. However, M\_TIMERm and M\_TIMERn cannot be equal.

- Circular references with timers are not supported and will generate an error.
- Existing **ControlType** and additional **ControlType** parameter settings, describing the supported particularities of the Matrox Meteor-II, are shown in the following table:

ControlType	ControlValue & Description		S T D	M D I G	1 3 9 4	C L
M_CAMERALINK_CCX_SOURCE	Route a specific signal to camera control output x, where x is the number of the camera control output (1,2,3,or 4).					x
	M_GRAB_EXPOSURE+ M_TIMER(1,2)	The camera control output comes from an exposure timer. Specify M_TIMER1 or M_TIMER2.				
	M_USER_BIT+(0,1)	The camera control output comes from a user output bit. Specify 0 or 1.				
	M_DEFAULT	Same as the DCF.				
M_GRAB_ABORT	Immediately stops a grab in progress and queued grabs. Useful for cancelling a triggered grab that is waiting for the trigger.		x	x	x	x
	M_DEFAULT	Stops the grab				



ControlType	ControlValue & Description		S T D	M C I	D I G I T A L	1 3 9 4	C L
M_GRAB_AUTOMATIC_INPUT_GAIN	Automatically sets the input gain.		x				
	M_ENABLE	Automatic input gain.					
	M_DISABLE	Set the input gain with the M_GRAB_INPUT_GAIN control type.					
	M_DEFAULT	Same as M_ENABLE.					
M_GRAB_INPUT_GAIN	Set the gain to apply to the input signal. When M_GRAB_AUTOMATIC_INPUT_GAIN is disabled (M_DISABLE), the gain can be set to any integer value from 0 to 255.		x				
	Set the gain to apply to the input signal. The gain can be set to any integer value from 0 to 255. The minimum value supported by the camera will be mapped to 0, the maximum value supported by the camera will be mapped to 255.					x	
	Set the gain to apply to the input signal. Valid input voltages and their corresponding gains are:		x				
		Input Voltage (Gain)					
	M_GAIN4	2.1 - 2.9 Vpp (1)					
	M_GAIN0	1.4 - 2.1 Vpp (1.3)					
	M_GAIN1	1.0 - 1.4 Vpp (2)					
	M_GAIN2	0.7 - 1.0 Vpp (2.8)					
	M_GAIN3	0.0 - 0.7 Vpp (4)					
	M_DEFAULT	Same as M_GAIN2					
M_GRAB_DIRECTION_X	Set the horizontal grab direction:		x	x	x		x
	M_REVERSE	Flip the grabbed image horizontally.					
	M_FORWARD	Grab normally in the horizontal direction.					
	M_DEFAULT	Same as M_FORWARD.					

ControlType	ControlValue & Description	S T D	M C I G 4	D 1 3 9	C L
M_GRAB_DIRECTION_Y	Set the vertical grab direction. On the Matrox Meteor-II /Digital board, this option is only available when a grab buffer is specified with an M_ON_BOARD attribute; there is no effect when grabbing to Host memory.	x	x	x	x
	M_REVERSE				
	M_FORWARD				
	M_DEFAULT				
M_GRAB_SAMPLING_POSITION	This control type fine tunes the pixel clock's sampling position to the analog signal. Control values are between 0 and 255. * Only available on the PC/104-Plus form factor.		x *		
M_GRAB_EXPOSURE_BYPASS	Activate the manual or automatic exposure model. For Matrox Meteor-II /Camera Link, enable M_CAMERALINK_CCX_SOURCE :		x	x	x
	M_ENABLE				
	M_DISABLE				
	M_DEFAULT				
For the following M_GRAB_EXPOSURE... control types, you can add M_TIMER1 or M_TIMER2 in manual exposure mode, to control the different on-board exposure timers. When omitted, Timer1 is assumed.					
M_GRAB_EXPOSURE	When using a software trigger source, use this control type to activate the specified grab exposure timer. When using a non-software trigger source, enable or disable the specified grab exposure timer. Note, the M_GRAB_EXPOSURE control type has no effect when grabbing using the automatic exposure model.		x	x	x
	M_ACTIVATE				
	M_ENABLE				
	M_DISABLE				
	M_DEFAULT				

ControlType	ControlValue & Description		S T D	M D I G	1 3 9 4	C L
M_GRAB_EXPOSURE_CLOCK_SOURCE	Specify the clock source that drives the exposure counter.		x			
	M_SYSCLK	Use a 25 MHz clock source frequency.				
	M_PIXCLK	Use the pixel clock frequency, which depends on the camera used.				
	M_HSYNC	Use the horizontal sync frequency, which depends on the camera used.				
	M_TIMER1	Use the frequency of M_TIMER1, if continuous.				
	M_TIMER2	Use the frequency of M_TIMER2, if continuous.				
	M_DEFAULT	Use the most appropriate clock source (as determined by the driver).				
M_GRAB_EXPOSURE_MODE	Set the exposure signal's polarity:		x	x		x
	M_LEVEL_HIGH					
	M_LEVEL_LOW					
	M_DEFAULT	Same as DCF.				

ControlType	ControlValue & Description		S T D	M C I D	D I G I T A L	1 3 9 4	C L
M_GRAB_EXPOSURE_SOURCE	Select the trigger source for the specified exposure timer. Note that the M_GRAB_EXPOSURE_SOURCE control type has no effect when grabbing using the automatic exposure model:						
	M_DEFAULT	Same as DCF.		x	x		x
	M_HARDWARE_PORT0	Use opto-isolated hardware trigger signal: Both Pin 34 (OPTOTRIG-) and Pin 35 (OPTOTRIG+) on Video Input connector.		x			
		Use opto-isolated hardware trigger signal. Both Pin 7 (OPTOTRIG+) and Pin 2 (OPTOTRIG-) on Trigger Input connector.			x		x
	M_HARDWARE_PORT1	Use TTL hardware trigger signal. Pin 20 (TRIGGER) on the video input connector.		x			
		Use TTL or RS-422/LVDS hardware trigger signal. See M_USER_IN_FORMAT to select the receiver. On the digital interface connector of Matrox Meteor-II /Digital, RS-422/LVDS trigger is a combination of Pin 47 (TRIGGER, INPUT, +) and Pin 48 (TRIGGER, INPUT, -). On the trigger input connector of Matrox Meteor-II /Camera Link, the LVDS trigger is a combination of Pin 3 (TRIGGER, INPUT, +) and Pin 8 (TRIGGER, INPUT, -). For both systems, the TTL trigger is on Pin 1 on the trigger input connector.			x		x
	M_NULL	Disable specified exposure timer. This has no effect when grabbing using automatic exposure model.		x	x		x

ControlType	ControlValue & Description		S T D	M T C I D	D I G I T I Z E R	1 3 9 4	C L
M_GRAB_EXPOSURE_SOURCE (cont.)	M_SOFTWARE	Use software trigger. The exposure signal is generated when <b>MdigControl()</b> with M_GRAB_EXPOSURE + M_TIMERN and M_ACTIVATE is called.		x	x		x
	M_VSYNC	Use vertical sync signal.		x	x		x
	M_HSYNC	Use horizontal sync signal.		x	x		x
	M_TIMER1	Use exposure signal generated by Timer1. Use only if setting trigger source for Timer2.		x	x		x
	M_TIMER2	Use exposure signal generated by Timer2. Use only if setting trigger source for Timer1.		x	x		x
	M_CONTINUOUS	No actual trigger. Run selected exposure timer in periodic mode. Automatically reset timer after each exposure signal is output. Exposure signal loops between delay and active mode.		x	x		x
M_GRAB_EXPOSURE_TIME	Set the time (in nsec) for the active portion of the exposure signal (that is, the exposure time). M_DEFAULT has the same effect as the setting in the digitizer's DCF. When using the automatic exposure model, if a single timer cannot generate the required exposure time, MIL automatically sets up connections with the second timer to generate the requested exposure time length. If <b>ControlValue</b> is set to 0, exposure is disabled and the grab is performed immediately. Note, an error is returned if the specified exposure time cannot be generated.			x	x		x
M_GRAB_EXPOSURE_TIME_DELAY	Set the delay (in nsec) between the trigger and the start of exposure. If M_DEFAULT, then the value is the same value as DCF. Note, an error is returned if the specified delay cannot be generated.			x	x		x

ControlType	ControlValue & Description		STD	MDI	1394	CL
M_GRAB_EXPOSURE_TRIGGER_MODE						
	Set the trigger activation mode for specified timer.			x	x	x
	M_DEFAULT	Same as the DCF.				
	M_EDGE_RISING	Low to high signal variation.				
	M_EDGE_FALLING	High to low signal variation.				
M_GRAB_MODE						
	M_SYNCHRONOUS	Synchronize your application with the end of a grab operation (i.e., wait until a grab has finished before returning from the grab command).	x	x	x	x
	M_ASYNCHRONOUS	Do not synchronize your application with the end of a grab operation, but return immediately after initiating the start of a grab. This allows other operations to be performed while waiting for the next <b>MdigGrab()</b> to be executed. However, only one grab can be queued; a call to another <b>MdigGrab()</b> before the current grab has finished will cause your application to wait until the current grab has finished.	x	x	x	x
	M_ASYNCHRONOUS_QUEUED					
		Do not synchronize your application with the end of a grab operation, but return immediately. This allows other operations to be performed while waiting for the next <b>MdigGrab()</b> to be executed, but in this case more than one <b>MdigGrab()</b> command can be queued.		x	x	x

ControlType	ControlValue & Description		S T D	M T C	D I G	1 3 9 4	C L
M_GRAB_SCALE	M_FILL_DESTINATION (for windowed displays only)		x	x	x	x	x
M_GRAB_SCALE_X	M_FILL_DISPLAY (for windowed displays only)		x	x	x	x	x
M_GRAB_SCALE_Y	Scaling factors: $0 < x \leq 1$ The Matrox Meteor-II /Standard supports arbitrary scaling when grabbing with the hardware revision B (Samsung KS0127B) decoder. If you don't have this revision B, the most appropriate scaling factor close to the specified scaling factor will be used (1/16,...1/4, 1/3, 1/2, 1).		x				
	1/16,...1/4, 1/3, 1/2, 1.			x			
	1/16,...1/4, 1/3, 1/2, 1, 2, and 4.				x		x
						x	
	$0 < x$						
M_GRAB_WINDOW_RANGE	Limit the range of the grabbed pixel values to between 16 and 235.		x				
	Limit the range of the grabbed pixel values to between 10 and 245.			x	x	x	x
M_GRAB_TRIGGER	Set the grab trigger detection state.						
	M_DEFAULT	The trigger state from the DCF file or, if none, M_DISABLE.	x	x	x	x	x
	M_ENABLE	Enables trigger detection.	x	x	x	x	x
	M_DISABLE	Disables trigger detection.	x	x	x	x	x
	M_ACTIVATE	Starts the grab immediately (for software triggers). An asynchronous or continuous grab must be in progress.	x	x	x	x	x

ControlType	ControlValue & Description		S T D	M C I D	D I G I T A L	1 3 9 4	C L
M_GRAB_TRIGGER_MODE	Set the hardware trigger activation mode.						
	M_DEFAULT	The trigger mode in the DCF file or, if none, M_EDGE_RISING. Note that the Matrox Meteor-II /1394 will only accept the default setting.	x	x	x	x	x
	M_EDGE_RISING	Low to high signal variation.	x	x	x		x
	M_EDGE_FALLING	High to low signal variation.	x	x	x		x
	M_LEVEL_LOW	Minimum signal level.	x	x	x		x
	M_LEVEL_HIGH	Maximum signal level.	x	x	x		x
M_GRAB_TRIGGER_SOURCE	Set the source of the grab trigger.						
	M_DEFAULT	The same as the DCF file (if any) or M_NULL.	x	x	x	x	x
	M_NULL	The trigger is inactive.	x	x	x	x	x
	M_SOFTWARE	Uses software trigger.	x	x	x	x	x



ControlType	ControlValue & Description		S T D	M D I G	1 3 9 4	C L
M_GRAB_TRIGGER_SOURCE (cont.)	M_HARDWARE_PORT0	Use opto-isolated hardware trigger signal. Combination of Pin 34 (OPTOTRIG-) and Pin 35 (OPTOTRIG+) on video input connector.	x	x		
		Use opto-isolated hardware trigger signal. Combination of Pin 7 (OPTOTRIG+) and Pin 2 (OPTOTRIG-) on trigger input connector.			x	x
		Description is camera-dependent.				x
	M_HARDWARE_PORT1	Use TTL hardware trigger signal. Pin 20 on video input connector.		x		
		On the digital interface connector of Matrox Meteor-II /Digital, RS-422/LVDS trigger is a combination of Pin 47 (TRIGGER, INPUT, +) and Pin 48 (TRIGGER, INPUT, -). On the trigger input connector of Matrox Meteor-II /Camera Link, RS-422/LVDS trigger is a combination of Pin 3 (TRIGGER, INPUT, +) and Pin 8 (TRIGGER, INPUT, -). For both systems, the TTL trigger is on Pin 1 on the trigger input connector.			x	x
	M_HARDWARE_PORT_CAMERA		x	x	x	x
		Use hardware trigger connected to the same connector as the selected camera (MIL determined).				
	M_TIMER1	Trigger on Timer1 signal.		x	x	x
	M_TIMER2	Trigger on Timer2 signal.		x	x	x

ControlType	ControlValue & Description		S T D	M C I	D I G	1 3 9 4	C L
M_SOURCE_COMPENSATION	Set the source compensation for cropping an input signal capture window.					x	
	M_ENABLE	The capture window will be cropped by software, if not supported by hardware.					
	M_DISABLE	The capture window will be cropped by hardware provided that you are using a format 7-capable camera. In this case, if the capture window specified does not match a subregion, an error will be generated.					
	M_DEFAULT	Same as M_ENABLE.					
For Matrox Meteor-II /Camera Link, you should specify a channel by adding (M_CH0 or M_CH1) to the M_UART control type (for example, M_UART_PARITY+M_CH0). When omitted, M_CH0 is assumed.							
M_UART_DATA_LENGTH	The number of data bits per character that are sent or received by the UART. Valid values are 7 or 8 bits.		x	x	x		x
	7	Data length is 7 bits.					
	8	Data length is 8 bits.					
	M_DEFAULT	Data length is 8 bits.					
M_UART_PARITY	Add a data bit (0 or 1) to the character data that is sent or received by the UART as a means of error checking.		x	x	x		x
	M_DEFAULT	Same as M_DISABLE.					
	M_ODD	The number of 1's will be odd.					
	M_EVEN	The number of 1's will be even.					
	M_DISABLE	No extra bit is added (no parity).					
M_UART_SPEED	Change the baud rate of the UART. Valid values are 230400, 115200, 76800, 57600, 38400, 28800, 19200, 14400, 9600, 7200, 4800, 3600, 2400, 1800, 1200, 600. M_DEFAULT is 14400.		x	x	x		x

ControlType	ControlValue & Description		S T D	M D I G	1 3 9 4	C L
M_UART_STOP_BITS	Add a data bit to signal the end of character data being sent or received.		x	x	x	x
	1	1 stop bit will be added.				
	2	2 stop bits will be added				
	M_DEFAULT	1 stop bit will be added.				
M_UART_TIMEOUT	Set the maximum time to wait between each byte when reading incoming data.		x	x	x	x
	M_INFINITE	Wait indefinitely.				
	M_DEFAULT	Same as M_INFINITE.				
	value in msec	Specify time to wait.				
M_UART_READ_CHAR	Read 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. ControlValue must be a pointer to a char.		x	x	x	x
M_UART_READ_STRING	Read a string of incoming data from the UART. The <b>ControlValue</b> must be a pointer to a character array. The size of this array must be set with the M_UART_READ_STRING_MAXIMUM_LENGTH control type. 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.		x	x	x	x
M_UART_READ_STRING_LENGTH	Set the length of the string (in bytes) to be read by M_UART_READ_STRING.		x	x	x	x
	M_DEFAULT	Used to specify the use of M_STRING_DELIMITER to end string.				
	value in bytes	Specify string length.				
M_UART_READ_STRING_MAXIMUM_LENGTH			x	x	x	x
	Use this value to specify the size of your read buffer to prevent global protection faults from happening when using the M_UART_READ_STRING control.					

ControlType	ControlValue & Description		S T D	M C I D	D I G I T A L	1 3 9 4	C L
M_UART_STRING_DELIMITER	Set the character used to terminate strings of incoming or outgoing data. The delimiter is used but not sent when writing data; it is read for incoming data.		x	x	x		x
	M_DEFAULT	The '\0' character.					
M_UART_WRITE_CHAR	Send one character to the UART. Set <b>ControlValue</b> as a pointer to a character.		x	x	x		x
M_UART_WRITE_STRING	Send the string of data, specified with the control value, through the UART. Set <b>ControlValue</b> to the character array. The number of characters to send can be specified with M_UART_WRITE_STRING_LENGTH or M_UART_STRING_DELIMITER.		x	x	x		x
M_UART_WRITE_STRING_LENGTH	Set the length of the string to be sent to the UART for transmission.		x	x	x		x
	M_DEFAULT	Used to specify the use of M_STRING_DELIMITER to end string. The delimiter will not be sent through the UART.					
	value in bytes	Specify string length.					

ControlType	ControlValue & Description	S T D	M D I G	1 3 9 4	C L
M_USER_BIT+(0,1,2,3, or 4) (For example, M_USER_BIT+1)	Set the state of the output bits of the digital interface connector (Matrox Meteor-II /Digital), or of the auxiliary sync and control connector (Matrox Meteor-II /Camera Link): M_ON or M_OFF. The relationship between the MIL user-bit number and the actual user output bit is as follows:				
	<b>MIL User-Bit#</b>				
	<b>Digital interface connector/ Aux. sync and ctrl connector</b>				
	bit 0:		x		
	bit 1:		x		
	bit 2:		x		x
	bit 3:		x		x
	bit 4:		x		x
	Set the state of an output bit of the video input connector: M_ON or M_OFF The relationship between the MIL user-bit number and the actual user output bit on the connectors is as follows:				
	<b>MIL User-Bit#</b>				
	<b>Video Input connector signal</b>				
	bit 3:	x	x		
	bit 4:	x	x		

ControlType	ControlValue & Description		S T D	M C D	D I G	1 3 9 4	C L
M_USER_IN_FORMAT	Enable either TTL or (RS-422/LVDS) receivers for digital I/Os:						
	M_TTL	Enable the TTL receivers for the trigger signal.			x		x
	M_RS422	Enable the RS-422 receivers for trigger and user inputs.			x		
	M_LVDS	Enable the LVDS receivers for trigger and user inputs.			x		x
	M_DEFAULT	Same as the DCF.			x		x
	M_DISABLE	Disable trigger and user inputs.			x		x
M_USER_OUT_FORMAT	Enable either TTL or RS-422/LVDS drivers for digital I/Os. For Matrox Meteor_II /Camera Link, to enable the LVDS drivers on the Camera Link connectors, you must specify to which channel to apply the format. This is done by adding M_CH0, M_CH1, or M_CH0+M_CH1 to the control type (for example, M_USER_OUT_FORMAT+M_CH0 will apply the control to channel 0). If no channel is specified, one of the drivers on the auxiliary I/O connector will be enabled (either LVDS or TTL).						
	M_TTL	Enable the TTL drivers for exposure signals.			x		x
	M_RS422	Enable the RS-422 drivers for exposure and user outputs.			x		
	M_LVDS	Enable the LVDS drivers for exposure and user outputs.			x		x
	M_DEFAULT	Same as the DCF.			x		x
	M_DISABLE:	Disable exposure and user outputs.			x		x

ControlType	ControlValue & Description	S T D	M D I G	1 3 9 4	C L
M_VCR_INPUT_TYPE	Set the digitizer input to VCR and improve the image's stability when grabbing from a VCR.	x			
	M_DEFAULT				
	M_DISABLE				
	M_ENABLE				

### Matrox Meteor-II /1394 DCAM-specific controls

There are 1394-specific **ControlType** settings. The control type values can be set to a value between the minimum and maximum values supported by the camera. Use **MdigInquire()** to determine these values. Similarly, use the M\_DEFAULT control value to set a control to its default value, or to set a camera's control feature to automatic mode (if the control supports an automatic mode).

ControlType	Description
M_AUTO_EXPOSURE	Set to automatic exposure mode, wherein the camera controls the exposure level automatically (and continuously).
M_SHARPNESS	Control the sharpness of the picture.
M_HUE*	Control the color phase of the picture.
M_WHITE_BALANCE_U	Set the U chrominance (color) component of the data.
M_WHITE_BALANCE_V	Set the V chrominance (color) component of the data.
M_SATURATION*	Control the color saturation of the picture.
M_GAMMA	Defines the function between the incoming light level and output picture level.

\*Note, these control types can also be set through the more conventional method. See the *Setting camera features* section that appears at the front of this chapter.

ControlType	Description
M_SHUTTER	Control the integration time of the incoming light.
M_GAIN*	Control the camera circuit gain.
M_IRIS	Control the mechanical lens iris.
M_FOCUS	Control the lens focus capabilities.
M_TARGET_TEMPERATURE	Set the target temperature for the camera.
M_ZOOM	Control the camera lens zoom capabilities.
M_PAN	Control the camera panning capabilities.
M_TILT	Control the camera tilt capabilities.
M_BRIGHTNESS *	Control the black level of the picture.
M_OPTICAL_FILTER	Control the optical filter of the camera lens function.
M_CAPTURE_SIZE	Control the capture size of the image.
M_CAPTURE_QUALITY	Control the capture quality of the image.
M_SOURCE_COMPENSATION	Set the source compensation for cropping an input signal capture window.
*Note, these control types can also be set through the more conventional method. See the <i>Setting camera features</i> section that appears at the front of this chapter.	



## MdigGrab/MdigGrabContinuous()

- When performing a grab on Matrox Meteor-II /Standard, Matrox Meteor-II /Multi-Channel, Matrox Meteor-II /Digital, and Matrox Meteor-II /Camera Link boards, the width and the horizontal position of the grab destination buffer must be a multiple of 4 bytes. If not, the grab will be clipped accordingly. This restriction does not apply to Matrox Meteor-II /1394.
- It is not possible to grab into a color-band child buffer on Matrox Meteor-II /Standard, Matrox Meteor-II /Multi-Channel, Matrox Meteor-II /Digital, or Matrox Meteor-II /Camera Link boards, when the buffer is packed. This restriction does not apply to Matrox Meteor-II /1394.
- When performing real-time grabs, Windows 98 and Windows Me do not provide the same performance and consistency as Windows NT or Windows 2000. This generally does not cause difficulty except in applications that require very fast response time to a hardware event. For example, an application that must grab sequences of images without missing any frames might not have sufficient time for the proper response to a hook function. Therefore, if grabbing a sequence, you should try to avoid code that needs an immediate response and favor code that supports a short delay after the event. For example, it is recommended to call the next grab operation (**MdigGrab()**) from the function hooked to the start of the current grab, instead of calling it from a function hooked to the end of the grab. This will queue the next grab operation while the current grab is in progress, and will allow the next grab to start immediately after the current grab. This should prevent the loss of a frame due to delay.

If these restrictions affect a major part of your application(s), we recommend the use of Windows NT or Windows 2000.

- Matrox Meteor-II can generally transfer all grabbed data of pixel depth 8-bit (monochrome), 24-bit and 32-bit, directly to display memory, when working with a graphics controller that supports fast linear-memory accesses to its frame buffer. For live grabs using Matrox Meteor-II, refer to *Transfers to display* at the beginning of this chapter.

## MdigGrabWait()

- The following digitizer feature is not supported on any of the Matrox Meteor-II boards:
  - M\_GRAB\_NEXT\_FIELD.

## MdigHookFunction()

- A function hooked to an M\_GRAB\_END event is called when all the data of the grab is transferred to the Host. This means that this hooked function can be called after the M\_GRAB\_START or M\_GRAB\_FRAME\_START event of the next frame.
- M\_FRAME\_START and all M\_FIELD... event types are not supported. That is, you can only hook to input-signal events that occur while grabbing.
- The **HookType** parameter can also be set to M\_UART\_DATA\_RECEIVED. The function hooked to this event will be called when the UART receives the data from the camera.
- M\_GRAB\_FRAME\_START is not supported on Matrox Meteor-II /Digital or on Matrox Meteor-II /Camera Link.
- No field-related hooks are supported on Matrox Meteor-II /1394.

## MdigInquire()

- When grabbing from a Matrox Meteor-II board to a windowed display, the M\_GRAB\_SCALE\_... attribute can return: M\_FILL\_DESTINATION or M\_FILL\_DISPLAY. Scaling factors, including arbitrary scaling factors from 0 to 1 can also be returned.
- For Matrox Meteor-II /Digital, Matrox Meteor-II /Camera Link, and Matrox Meteor-II /1394, the M\_GRAB\_MODE attribute can also return M\_ASYNCHRONOUS\_QUEUED.

- Additional **InquireType** parameter settings which are supported by a particular Matrox Meteor-II board are indicated in the following table:

<b>InquireType</b>	<b>Description</b>	<b>S T D</b>	<b>M C</b>	<b>D I G</b>	<b>1 3 9 4</b>	<b>C L</b>
M_BRIGHTNESS_REF	Brightness reference level.	x			x	
M_CAMERALINK_CCX_SOURCE	Signal that is routed to the camera control output (1, 2, 3, 4). Possible signals are: M_GRAB_EXPOSURE+M_TIMER1 M_GRAB_EXPOSURE+M_TIMER2 M_USER_BIT+0 M_USER_BIT+1					x
M_COLOR_MODE	Color mode:					
	M_RGB		x	x		x
	M_MONO8_VIA_RGB		x			
	M_EXTERNAL_CHROMINANCE	x				
	M_COMPOSITE	x				
	M_MONOCHROME	x		x	x	x
M_GRAB_AUTOMATIC_INPUT_GAIN	Mode of automatic input gain: M_ENABLE or M_DISABLE.	x				
M_GRAB_DIRECTION_X	Horizontal grab direction: M_REVERSE or M_FORWARD.	x	x	x		x
M_GRAB_DIRECTION_Y	Vertical grab direction: M_REVERSE or M_FORWARD.	x	x	x		x
M_GRAB_IN_PROGRESS	Current grab state: M_YES or M_NO.	x	x	x	x	x

InquireType	Description	S T D	M C	D I G	1 3 9 4	C L
M_GRAB_INPUT_GAIN	The gain that is applied to the input signal. For the Matrox Meteor-II /Standard, M_GRAB_AUTOMATIC_INPUT_GAIN must be disabled (M_DISABLE) to inquire the gain value. Values are from 0 to 255.	x			x	
	M_GAIN0, M_GAIN1, M_GAIN2, M_GAIN3, or M_GAIN4.		x			
M_GRAB_EXPOSURE_BYPASS	Exposure model: M_ENABLE (manual) or M_DISABLE (automatic).		x	x		x
For the following M_GRAB_EXPOSURE... inquire types, you can add M_TIMER1 or M_TIMER2 in manual exposure mode, to inquire the different on-board exposure timers. When omitted, TIMER1 is assumed.						
M_GRAB_EXPOSURE	Exposure timer state for non-software trigger source: M_ENABLE or M_DISABLE.		x	x		x
M_GRAB_EXPOSURE_CLOCK_FREQUENCY	The frequency in hertz of the clock used to generate the exposure signal.		x			
M_GRAB_EXPOSURE_MODE	Exposure signal's polarity: M_LEVEL_HIGH or M_LEVEL_LOW.		x	x		x
M_GRAB_EXPOSURE_SOURCE	Trigger source for specified timer.		x	x		x
M_GRAB_EXPOSURE_TIME	Time for the active portion of the exposure signal (value in nsec). Returned as a double.		x	x		x
M_GRAB_EXPOSURE_TIME_DELAY	Delay (in nsec) between the trigger and the start of exposure. Returned as a double.		x	x		x
M_GRAB_EXPOSURE_TRIGGER_MODE	Trigger activation mode for specified timer: M_EDGE_RISING or M_EDGE_FALLING.		x	x		x
M_GRAB_SAMPLING_POSITION	The pixel clock's sampling position. Values are between 0 and 255. * Only available for the PC/104-Plus form factor.		x*			

InquireType	Description	S T D	M C	D I G	1 3 9 4	C L
M_GRAB_TRIGGER	Grab trigger state: M_DEFAULT, M_ENABLE, M_DISABLE, M_ACTIVATE.	x	x	x	x	x
M_GRAB_TRIGGER_MODE	Hardware trigger activation mode. See <b>MdigControl()</b> .	x	x	x	x	x
M_GRAB_TRIGGER_SOURCE	Determines the trigger source. See <b>MdigControl()</b> .	x	x	x	x	x
M_GRAB_WINDOW_RANGE	Determines whether to limit the range of the grabbed pixel values: M_ENABLE or M_DISABLE.	x	x	x	x	x
M_HOOK_MASTER_THREAD_HANDLE	Returns the handle of the main interrupt thread that dispatches the handling of the grab hooks to their respective threads.	x	x			
M_HOOK_MASTER_THREAD_ID	Returns the ID of the main interrupt thread that dispatches the handling of the grab hooks to their respective threads.	x	x			
M_INPUT_SIGNAL_PRESENT	Video input signal present: M_YES or M_NO. M_YES is returned whenever the DCF is for a color camera, but only when there is a line lock and color lock. When a monochrome camera with a color DCF is used to grab images, the function will never return "yes". However, this monochrome camera/color DCF combination is still supported on Matrox Meteor-II /Standard, because it is useful when switching between channels.	x			x	
M_INPUT_SIGNAL_HSYNC_LOCK	Video input signal horizontal synchronization. Returns M_TRUE when a line lock is achieved, otherwise it returns M_FALSE.	x				
M_INPUT_SIGNAL_COLOR_LOCK	Video input signal color lock. Returns M_TRUE when a color lock is achieved, otherwise it returns M_FALSE.	x				

InquireType	Description	S T D	M C	D I G	1 3 9 4	C L
M_INPUT_SIGNAL_SOURCE	Input-signal source: M_HARDWARE_PORT0 (analog) or M_HARDWARE_PORT1 (digital port).	x	x	x		x
M_NATIVE_ID	Identifier of the Genesis Native Library* digitizer that is associated with the specified MIL digitizer.			x		x
M_NATIVE_CAMERA_ID	Identifier of the Genesis Native Library* camera that is associated with the specified MIL digitizer.			x		x
M_NATIVE_CONTROL_ID	Identifier of the Genesis Native Library* control buffer that is associated with the specified MIL digitizer.			x		x
M_NATIVE_LAST_GRAB_OSB_ID	Identifier of the Genesis Native Library* OSB (operations status block) used in the last grab.			x		x
*Matrox Meteor-II /Digital and Camera Link can be programmed with the <i>Genesis Native Library</i> .						
M_CONTRAST_REF	Contrast reference level.	x				
M_HUE_REF	Hue reference level.	x			x	
M_SATURATION_REF	Saturation reference level.	x			x	
M_SOURCE_COMPENSATION	Whether the capture window will be cropped by software or by hardware: M_ENABLE or M_DISABLE				x	
For Matrox Meteor-II /Camera Link, you should add a specific channel (M_CH0 or M_CH1) to every M_UART_... inquire type (for example, M_UART_PARITY+M_CH0). When omitted, M_CH0 is assumed.						
M_UART_PARITY	Current UART parity setting: M_ODD, M_EVEN, M_DISABLE.	x	x	x		x
M_UART_STOP_BITS	Current number of stop bits in UART configuration: 1 or 2.	x	x	x		x
M_UART_DATA_LENGTH	Current data length in UART configuration: 7 or 8.	x	x	x		x

InquireType	Description	S T D	M C	D I G	1 3 9 4	C L
M_UART_SPEED	Current baud rate in UART configuration: 230400, 115200, 76800, 57600, 38400, 28800, 19200, 14400, 9600, 7200, 4800, 3600, 2400, 1800, 1200, 600.	x	x	x		x
M_UART_DATA_PENDING	Indicates the input buffer has some pending data: M_TRUE, M_FALSE.	x	x	x		x
M_UART_WRITE_STRING_LENGTH	The length of the string to be sent to the UART for transmission. Can be M_DEFAULT if the string length is specified by M_UART_STRING_DELIMITER.	x	x	x		x
M_UART_READ_STRING_LENGTH	The length of the string, in bytes, to be read by M_UART_READ_STRING. Can be M_DEFAULT if the string length is specified by M_UART_STRING_DELIMITER.	x	x	x		x
M_UART_READ_STRING_MAXIMUM_LENGTH	Current maximum size of the read buffer.	x	x	x		x
M_UART_STRING_DELIMITER	Current character used to delimit strings if M_UART_WRITE_STRING_LENGTH or M_UART_READ_STRING_LENGTH is set to M_DEFAULT.	x	x	x		x
M_UART_TIMEOUT	Current maximum time, in msec, to wait between bytes of incoming data. Can be M_INFINITE or value in msec.	x	x	x		x
M_UART_THREAD_HANDLE	Current UART thread handle.	x	x	x		x
M_UART_THREAD_ID	Current UART thread ID.	x	x	x		x

InquireType	Description	S T D	M C	D I G	1 3 9 4	C L
M_USER_BIT+ (0,1,2,3, or 4) (For example, M_USER_BIT+1)	Current state of an input/output bit of the video input connector: M_ON (1) or M_OFF (0).					
	<b>MIL User-Bit #      Video input connector signal</b>					
	bit 1:              USER1IN SIGNAL.	x	x			
	bit 2:              USER2IN SIGNAL.	x	x			
	bit 3:              USER1OUT SIGNAL.	x	x			
	bit 4:              USER2OUT SIGNAL.	x	x			
	Current state of the input bit of the digital interface connector (Matrox Meteor-II /Digital), or of the auxiliary sync and control connector (Matrox Meteor-II /Camera Link): M_ON (1) or M_OFF (0).					
	<b>MIL User Bit #      Dig. interface connector/Aux. sync and ctrl connector</b>					
	bit 0:              USER0 INPUT (RS-422/LVDS)			x		x
	bit 1:              USER1 INPUT (RS-422/LVDS)			x		x
M_USER_IN_FORMAT	Type of receiver for digital I/Os (M_TTL, M_RS422, M_LVDS, or M_DISABLE).			x		x



InquireType	Description	S T D	M C	D I G	1 3 9 4	C L
M_USER_OUT_FORMAT	Type of driver for digital I/Os (M_TTL, M_RS422, M_LVDS, or M_DISABLE). For Matrox Meteor-II /Camera Link, you must specify the channel of the driver about which to inquire. This is done by adding M_CH0 or M_CH1 to the inquire type (for example, M_USER_OUT_FORMAT+CH_0 will inquire channel 0). If no channel is specified, the auxiliary I/O connector will be the one inquired.			X		X
M_VCR_INPUT_TYPE	Digitizer input set to VCR. M_DISABLE or M_ENABLE.	X				

### Matrox Meteor-II /1394 DCAM-specific inquires

In addition to the above general digitizer inquiries, 1394 camera-specific features can also be inquired.

You can inquire whether the camera has a required camera feature by using the **MdigInquire()** function with the feature's corresponding inquire type. An error message will be generated if the requested feature (inquire type) is not supported.

You can also inquire the minimum and maximum values supported by the camera. Add M\_MIN\_VALUE, M\_MAX\_VALUE to the **InquireType** parameter to read the smallest value supported and largest value supported, respectively. If nothing is specified, the current value for a particular camera feature is returned. Note that if the camera is in automatic mode, the default value is returned.

InquireType	Description
M_AUTO_EXPOSURE	Read the camera exposure level.
M_SHARPNESS	Read the sharpness of the picture.
M_HUE	Read the color phase of the picture.
M_WHITE_BALANCE_U M_WHITE_BALANCE_V	Inquire one of the two chrominance (color) components of the data.
M_SATURATION	Read the color saturation of the picture.

InquireType	Description
M_GAMMA	Read the function between the incoming light level and output picture level.
M_SHUTTER	Read the integration time of the incoming light.
M_GAIN	Read the camera circuit gain control.
M_IRIS	Read the mechanical lens iris control.
M_FOCUS	Read the lens focus control.
M_TEMPERATURE	Read the temperature inside the camera.
M_TARGET_TEMPERATURE	Read the target temperature for the camera.
M_ZOOM	Read the camera lens zoom capabilities.
M_PAN	Read the camera panning capabilities.
M_TILT	Read the camera tilt capabilities.
M_BRIGHTNESS	Read the black level of the picture.
M_OPTICAL_FILTER	Read the optical filter of the camera lens function.
M_CAPTURE_SIZE	Read the capture size of the image.
M_CAPTURE_QUALITY	Read the capture quality of the image.
M_FORMAT_SUPPORTED	List the names of all supported formats that can be accepted by <b>MdigAlloc()</b> for your 1394 digital camera. This list includes "M_RS170", "M_NTSC" (if available), and "M_DEFAULT".
M_FORMAT_SUPPORTED_LENGTH	Inquire the maximum number of characters of the names of the supported formats.
M_FORMAT_SUPPORTED_NUM	Inquire the number of supported strings.
M_SOURCE_COMPENSATION	Inquire whether the capture window will be cropped by software or by hardware.
M_SERIAL_NUMBER_0	Inquire the low part of the serial number of the camera.
M_SERIAL_NUMBER_1	Inquire the high part of the serial number of the camera.

The following inquire types are not supported, since they are not currently supported by the hardware.

- M\_BLACK\_REF.
- M\_WHITE\_REF.
- M\_CONTRAST\_REF.

## MdigLut()

- The Meteor-II /Multi-Channel has three 256 8-bit LUTs.
- There are no LUTs on Matrox Meteor-II /Standard and Matrox Meteor-II /1394.
- Matrox Meteor-II /Digital and Matrox Meteor-II /Camera Link have four 8-bit x 256 programmable LUTs, which can be associated with a digitizer that was allocated using **MdigAlloc()**. These LUTs can be operated as four 8-bit LUTs (256 entries), two 10-bit LUTs (1024 entries), or two 12-bit LUTs (4096 entries). For Matrox Meteor-II /Digital, when grabbing from more than two channels, the LUTs must be 8-bit.

## MdigReference()

- The reference type parameter specifies the reference level type to adjust for the frame grabber. This parameter can be set to one of the following:.

ReferenceType	Description and Reference Level	STD	MC	DIG	1394	CL
M_BLACK_REF*	Digitizer black reference level.** For M_BLACK_REF, the minimum voltage level (M_MIN_LEVEL) corresponds to 0.6 V. The maximum voltage level (M_MAX_LEVEL) corresponds to 1.6 V.		x			
M_WHITE_REF*	Digitizer white reference level.** For M_WHITE_REF, the minimum voltage level (M_MIN_LEVEL) corresponds to 1.6 V. The maximum voltage level (M_MAX_LEVEL) corresponds to 2.6 V		x			
M_HUE_REF	Digitizer hue reference level.	x			x	
M_SATURATION_REF	Digitizer saturation reference level.	x			x	

ReferenceType	Description and Reference Level	STD	MC	DIG	1394	CL
M_BRIGHTNESS_REF	Digitizer brightness reference level.	x			x	
M_CONTRAST_REF	Digitizer contrast reference.	x				
*Note: To specify the channel with the Matrox Meteor-II /Multi-Channel, combine M_CH0_REF, M_CH1_REF, or M_CH2_REF with M_BLACK_REF or M_WHITE_REF. **Note that some consecutive <b>ReferenceLevel</b> settings might produce the same result due to the fact that there are only 98 distinct adjustments on Meteor-II /Multi-Channel (adjustments of 10.23 mV each).						

**MsysAlloc()**

- To allocate a Meteor-II system, you must set the **SystemType** parameter to one of the following:

Board	System
Matrox Meteor-II /Standard	M_SYSTEM_METEOR_II
Matrox Meteor-II /Multi-Channel	M_SYSTEM_METEOR_II
Matrox Meteor-II /Digital	M_SYSTEM_METEOR_II_DIG
Matrox Meteor-II /1394	M_SYSTEM_METEOR_II_1394
Matrox Meteor-II /Camera Link	M_SYSTEM_METEOR_II_CL

This selection opens communication with the board and will automatically allow the system to use some Host memory and any available graphics controller, if necessary.

To perform multiple continuous grabs, MIL allows you to allocate multiple M\_SYSTEM\_METEOR\_II\_1394 systems on a Matrox Meteor-II /1394 board. Then, use **MdigAlloc()** to specify which camera (digitizer) belongs to a particular system. Refer to section *Grabbing from multiple 1394 DCAM-compliant cameras* for more details.

- ❖ The number of 1394-compliant cameras available is detected when **MsysAlloc()** function is called the first time. **Warning:** you cannot unplug a camera that has been detected by **MsysAlloc()** before freeing up the system with the **MsysFree()** function.



ControlType	Description and ControlValue		STD	MC	DIG	1394	CL
M_DISPLAY_DOUBLE_BUFFERING	When displaying a pseudo-live grab operation under Windows, MIL will double-buffer the grab depending on the performance of your Host PC. Double-buffering prevents any frame loss when the grabbed data is copied from the Host to the display. MIL is selective about performing double-buffered pseudo-live grabs because this is more demanding on your Host CPU. If the default behavior is not appropriate for your application, you can force a specific behavior with the following <b>ControlType</b> :		x	x	x	x	x
	M_ENABLE	Force double-buffering.					
	M_DISABLE	Disable double-buffering					
M_HARDWARE_COMPRESSION	Specifies whether the JPEG compression is done by the hardware or the software.		x	x			
	M_ENABLE	Compression is done by the hardware.					
	M_DISABLE	Compression is done by the software.					
	M_DEFAULT	Compression is done by the hardware if a hardware module is present.					

ControlType	Description and ControlValue		S T D	M C	D I G	1 3 9 4	C L
M_HARDWARE_DECOMPRESSION	Specifies whether the JPEG decompression is done by the hardware or the software.		x	x			
	M_ENABLE	Decompression is done by the hardware.					
	M_DISABLE	Decompression is done by the software.					
	M_DEFAULT	Decompression is done by the hardware if a hardware module is present.					
M_LIVE_GRAB_END_TRIGGER	When a live grab operation uses a software trigger and <b>MdigHalt()</b> is issued, a software trigger is automatically generated to invoke a last grab (if you did not disable M_LAST_GRAB_IN_TRUE_BUFFER). You can disable this automatic trigger; however, you would then have to issue a software trigger call after the <b>MdigHalt()</b> call (these calls must be issued from different threads). To disable the <b>MdigHalt()</b> automatic trigger, set the <b>ControlType</b> to M_DISABLE:		x	x	x		x
	M_ENABLE	Default for hardware triggered cameras.					
	M_DISABLE	Default for other camera types.					

## MsysInquire()

- You can request the following information with the **InquireType** parameter:

InquireType	Description	S T D	M C	D I G	1 3 9 4	C L
M_BUS_MASTER_COPY_TO_HOST	Whether copies from an M_ON_BOARD buffer to a Host buffer are driven by the digitizer (M_ENABLE) or the Host (M_DISABLE).			x		x
M_BUS_MASTER_COPY_FROM_HOST	Whether copies from a Host buffer to an M_ON_BOARD buffer are driven by the digitizer (M_ENABLE) or the Host (M_DISABLE).			x		x
M_BOARD_TYPE	The type of system board: M_METEOR_II_STD M_METEOR_II_MC M_METEOR_II_WITH_COMPRESSION_MODULE M_METEOR_II_MC_WITH_COMPRESSION_MODULE M_METEOR_II_DIG M_METEOR_II_1394 M_METEOR_II_CL	x	x	x	x	x
M_COMPRESSION_MODULE_PRESENT	Returns M_TRUE if a compression board is present, otherwise it returns M_FALSE.	x	x			
M_DIGITIZER_NUM	The number of digitizers that are available to be allocated (including those already allocated) on a particular system. This corresponds to the number of digitizers allocated on the system, plus the number of digitizers still in the pool.	x	x	x	x	x
M_HARDWARE_COMPRESSION	Specifies whether the JPEG compression is done by the hardware or the software.		x	x		
	M_ENABLE	Compression is done by the hardware.				
	M_DISABLE	Compression is done by the software.				
	M_DEFAULT	Compression is done by the hardware if a hardware module is present.				



InquireType	Description	S T D	M C	D I G	1 3 9 4	C L
M_HARDWARE_DECOMPRESSION	Specifies whether the JPEG decompression is done by the hardware or the software.	x	x			
	M_ENABLE					
	M_DISABLE					
	M_DEFAULT					
M_BOARD_REVISION	The board revision (long value).	x	x			
M_DISPLAY_DOUBLE_BUFFERING	The state of double-buffering: M_ENABLE or M_DISABLE.	x	x	x	x	x
M_LIVE_GRAB_END_TRIGGER	An automatic trigger is generated at the end of a grab: M_ENABLE or M_DISABLE.	x	x	x		x



---

## ***Chapter 5: MIL and the Matrox Orion platform***

*This section discusses features of MIL that are distinct to the Matrox Orion platform and ways that optimize the board's performance.*

## Matrox Orion-specific features

Matrox Orion is a frame grabber capable of acquiring standard monochrome video in RS-170/CCIR format, composite (CVBS) and component (Y/C) color video in NTSC/PAL format, and RGB video. Matrox Orion is available in both PCI and AGP configurations, and as grab modules on Matrox 4Sight-II.

The number of channels with which you can grab depends on the version of Matrox Orion you are using. The number of channels supported by each version of Matrox Orion, and the type of input supported is listed below:

Board	Type of Camera	Number of channels
Matrox Orion (for AGP/PCI)	monochrome RS-170/CCIR	6 (using the RGB path) 8 (using the decoder path)
	composite NTSC/PAL	8
	Y/C	4
	RGB	2
Matrox Orion /RGB for 4Sight-II	monochrome RS-170/CCIR	6 (using the RGB path) 12 (using the decoder path)
	composite NTSC/PAL	12
	Y/C	6
	RGB	2
Matrox Orion /Standard for 4Sight-II	monochrome RS-170/CCIR	12
	composite NTSC/PAL	12
	Y/C	6

Matrox Orion also has three 256 x 8-bit input lookup tables (LUTs) that support any input source and buffer format. Note that there are no input LUTs on Matrox Orion /Standard for 4Sight-II.

Matrox Orion supports a grab feature that allows you to simultaneously grab live into the display, and a Host buffer for processing. For details, see the section, *MdigAlloc()*.

When acquiring data on Matrox Orion (for AGP), the values 0 and 255 are reserved. Therefore, the range of possible input values spans from 1 to 254.

---

### *Display features*

With its MGA-400 controller and 32-Mbyte frame buffer, Matrox Orion can deliver a true color (32-bit) image display with a 32-bit color overlay, for a completely true-color display at up to 1280x1024 resolution; the maximum refresh rate is 75 Hz, depending on your monitor. Matrox Orion can allocate an overlay frame buffer in 8-bit monochrome format or 32-bit RGB format (BGR32 packed), thereby supporting either of these Windows display resolutions. In addition, Matrox Orion features non-destructive graphics overlay on live video, and support for video-in-a-window with arbitrary video scaling (up or down).

Since the display section is controlled by the MGA G400 chip, Matrox Orion features DualHead display technology, which provides the ability to simultaneously have an independent analog VGA output and TV output.

❖ Note that DirectDraw is not supported under NT 4.0 on an extended desktop. Without DirectDraw, windowed displays will be CPU-assisted.

---

### *Miscellaneous information*

Matrox Orion can be used as a hardware fingerprint for run-time licenses.

See Appendix A for a data flow diagram.

❖ Matrox Orion supports Windows NT/2000 and Windows Me.

---

## Using Matrox Orion with MIL

To use your Matrox Orion board with MIL, you must allocate an Orion system (M\_SYSTEM\_ORION), using **MsysAlloc()**. This establishes a MIL system environment in which MIL uses some Host memory, and any available graphics controller for grabbing and displaying images.

This chapter relates Orion systems to the following: *Grabbing from multiple cameras with different DCFs*, *Using the encoder*, and *Particularities of existing MIL functions on Matrox Orion*.

Refer to the *milorion.txt* file in the  
\\MATROX IMAGING\\DRIVERS\\DOC (or user-specified)  
directory for any additions/modifications to these board specific notes.

---

## Grabbing from multiple cameras with different DCFs

In some applications, you might want to grab from multiple cameras with different DCFs that are attached to the same digitizer. This procedure normally involves allocating a digitizer, grabbing the required frame, freeing the digitizer, and then allocating the digitizer again with the second DCF; this can increase the time required for operation. On Matrox Orion, MIL can circumvent this problem by using a fast DCF-switching technique which is outlined in the steps below:

1. Make as many calls to **MdigAlloc()** as you have cameras, with different formats, from which you want to grab. With each call, the **DigNum** parameter must be set to M\_DEV0, since there is only one physical digitizer; each allocation only sets up a virtual instance of the digitizer.
2. Specify all required digitizer settings using **MdigControl()**, **MdigChannel()**, **MdigReference()**, and **MdigHookFunction()** for each allocated digitizer. Each time a different digitizer is specified, it retains all settings previously specified.

3. Call **MdigGrab()** with any digitizer identifier. If this call uses a digitizer identifier different from the previous call, a camera switch will occur.
- ❖ If there is a grab in progress on one digitizer, calling any of the following functions with any other digitizer will result in an error: **MdigGrab()**, **MdigGrabContinuous()**, **MdigChannel()**, **MdigControl()**, **MdigInquire()**, **MdigReference()**, **MdigLut()**, **MdigHookFunction()**, **MdigAlloc()**, and **MdigFree()**.

See the example *mdigmultformat.c*.

---

## Using the encoder

Matrox Orion's display section features an on-board encoder.

The video encoder can be programmed to output composite (CVBS) and component (Y/C) video in NTSC/PAL formats. It can also output component RGB video with the same resolution and refresh rate as video in NTSC/PAL formats. To output from the encoder, use an auxiliary display with an encoder-supported output format. On Matrox Orion, the overlay buffer associated with such an auxiliary display will be driven by the graphics controller.

Note that during a live grab, the display is non-tearing.

## Particularities of existing MIL functions on Matrox Orion

Certain commands have special features or functionality on an Orion system. This table provides an overview of the affected commands. Details and procedures are described in the individual command descriptions, which follow.

Commands	Matrox Orion particularities
MbufAlloc...()	Buffer constraints and options.
MbufCreate...()	Required parameter settings.
MbufInquire()	Orion-specific inquire options.
MdigAlloc()	Digitizer configuration format (DCF) specifications.
MdigChannel()	Required parameter settings.
MdigControl()	Control type and value additions and restrictions.
MdigGrab/MdigGrabContinuous()	Destination buffer restriction.
MdigHookFunction()	Hook restrictions.
MdigInquire()	Orion-specific inquire options.
MdigReference()	Orion features.
MdispAlloc()	Display format specifications.
MdispInquire()	Additional inquiries.
MsysAlloc()	Orion-specific allocation options.
MsysControl()	Control type and value additions.
MsysInquire()	Orion-specific inquire options.



## MbufAlloc...()

- Only 8-bit monochrome and 3-band 8-bit color buffers in the following formats can have an M\_GRAB attribute:

Attribute parameter for 3-band 8-bit buffer	Internal storage format for 3-band 8-bit buffer
M_BGR32 + M_PACKED	32-bit (BGR) packed pixels (default).
M_YUV16_YUYV + M_PACKED	YUV16 (4:2:2) packed standard.

## MbufCreate...()

- Set the **Attribute** parameter to an M\_IMAGE combination (for example, M\_IMAGE + M\_DISP + M\_GRAB + M\_PROC).
- Only the following **ControlFlag** combinations are supported:

M_PHYSICAL_ADDRESS + M_PITCH	<b>ArrayOfDataPtr</b> is an array of physical addresses. The pitch is in pixels (default).
M_PHYSICAL_ADDRESS + M_PITCH_BYTE	<b>ArrayOfDataPtr</b> is an array of physical addresses. The pitch is in bytes.

- The **Attribute** parameter can be set to the same settings as *MbufAlloc...()*.

## MbufInquire()

- The **InquireType** parameter can also be set to:

M_CURRENT_BUF_ID	Identifier of the buffer in which data is currently being grabbed. Valid only for windowed displays. This information is used, for example, to access grabbed data while a continuous grab ( <b>MdigGrabContinuous()</b> ) is being displayed live. Under these circumstances, the destination buffer is not the allocated buffer, but rather an associated buffer, used only for display.
------------------	---

## MdigAlloc()

- The **DataFormat** parameter specifies the DCF for the input device. Formats denoted with an asterisk (\*) are not supported on Matrox Orion /Standard for 4Sight-II.

The predefined settings for monochrome cameras are:

"M_RS170"	RS-170, 640x480, 8 bits, 12.5MHz, analog.
"M_CCIR"	CCIR, 768x576, 8 bits, 14.8MHz, analog.
"M_RS170_VIA_RGB"*	RS-170 using RGB input.
"M_CCIR_VIA_RGB"*	CCIR using RGB input.
"M_DEFAULT"	Same as RS-170.

The predefined settings for color cameras are:

"M_NTSC"	NTSC, 640x480, 3x8 bits, 12.5MHz, composite
"M_NTSC_RGB"*	RS-170 RGB, 640x480, 3x8 bits, 12.5MHz
"M_NTSC_YC"	RS-170 Y/C(SVHS), 640x480, 3x8 bits, 12.5MHz
"M_PAL"	PAL I, 768x576, 3x8 bits, 14.8MHz, composite
"M_PAL_RGB"*	PAL RGB, 768x576, 3x8 bits, 14.8MHz
"M_PAL_YC"	PAL Y/C, 768x576, 3x8 bits, 14.8 MHz

You can also set the **DataFormat** parameter to the name of a DCF file (\*.dcf). This file specifies the input signal format. See the \MATOX IMAGING\DRIVERS\ (or user-specified) directory for the current list of supported formats.

- You can allocate multiple digitizers on one Orion system. Make as many calls to **MdigAlloc()** as you have cameras from which you want to grab. With each call, specify the DCF for the camera, and set the **DigNum** parameter to M\_DEV0. Calling **MdigGrab()** with any digitizer identifier will cause a switch and a grab from the identifier's associated camera. See the section, *Grabbing from multiple cameras with different DCFs* for more information.

---

*Multi-destination support*

- Matrox Orion has a special feature that permits real-time transfers of grabbed images to the display and Host memory simultaneously. For example, an application can have a color live grab in the display and at the same time, grab in a monochrome Host buffer for processing.

To use this feature, two digitizers must be allocated. Both digitizers must have the same the device number (`M_DEV0`) and the same DCF because physically, the same digitizer is used for both calls to **MdigAlloc()**.

After the digitizers are allocated, you can then call **MdigGrabContinuous()** into a buffer selected on the display, using one of the digitizer identifiers, and call **MdigGrab()** to grab into a Host buffer using the other digitizer identifier. This results in the same grabbed image being transferred to each buffer in their respective destination formats.

Both the buffer selected to the display and the Host buffer need not be in the same format or the same size. In addition the `M_GRAB_MODE` and `M_GRAB_SCALE` control types can be independently set for each digitizer; all other controls will be used for both digitizers. Changing any other control for one digitizer, will affect the other.

See the example, `mdigmultdestination.c`

## MdigChannel()

The channels available for acquisition depend on the color mode and data path used; this is specified by the DCF during digitizer allocation.

- When using the video decoder on Matrox Orion, you can attach and switch between 8 monochrome cameras (CCIR/RS-170 format), 8 composite color cameras (NTSC/PAL format), or 4 Y/C cameras (NTSC/PAL format). When using the RGB path, 6 monochrome or 2 RGB cameras can be attached.

When using the video decoder on Matrox Orion /RGB for 4Sight-II, you can attach and switch between 12 monochrome cameras (CCIR/RS-170 format) when using the decoder path, or 6 cameras when using the RGB path. You can also attach 12 composite color cameras (NTSC/PAL format), or 6 Y/C cameras (NTSC/PAL format), or 2 RGB cameras.

When using video decoder on Matrox Orion /Standard for 4Sight-II, you can attach and switch between 12 monochrome cameras (CCIR/RS-170 format), 12 composite color cameras (NTSC/PAL format), or 6 Y/C cameras (NTSC/PAL format).

- When grabbing from the RGB path, two calls **MdigChannel**(...,M\_CHx + M\_SIGNAL) and **MdigChannel**(...,M\_CHx + M\_SYNC) can be made to control the channels of the data and synchronization (sync) signals, respectively.
- To switch between cameras of a similar type, use:

Camera Type	Channel	Signal	Orion	Orion/Std for 4Sight-II	Orion /RGB for 4Sight-II
Any	M_DEFAULT	Same as M_CH0.	x	x	x
Monochrome (RGB path)	M_CH0	VID_IN1	x		x
	M_CH1	VID_IN2	x		x
	M_CH2	VID_IN3	x		x
	M_CH4	VID_IN5	x		x
	M_CH5	VID_IN6	x		x
	M_CH6	VID_IN7	x		x
Color RGB (RGB path)	M_CH0	VID_IN1, VID_IN2, VID_IN3 (data signals, and sync can be set on any input signal).	x		x
	M_CH1	VID_IN5, VID_IN6, VID_IN7 (data signals, and sync can be set on any input signal).	x		x
	M_RGB	VID_IN1, VID_IN2, VID_IN3 (data signals) and sync signal is forced on VID_IN4.	x		x

Camera Type	Channel	Signal	Orion	Orion/Std for 4Sight-II	Orion /RGB for 4Sight-II
Composite-sync monochrome/ composite color (decoder path)	M_CH0	VID_IN1	x	x	x
	M_CH1	VID_IN2	x	x	x
	M_CH2	VID_IN3	x	x	x
	M_CH3	VID_IN4	x	x	x
	M_CH4	VID_IN5	x	x	x
	M_CH5	VID_IN6	x	x	x
	M_CH6	VID_IN7	x	x	x
	M_CH7	VID_IN8	x	x	x
	M_CH8	VID_IN9		x	x
	M_CH9	VID_IN10		x	x
	M_CH10	VID_IN11		x	x
	M_CH11	VID_IN12		x	x
Y/C (decoder path)	M_CH0	(Y camera 1) VID_IN1 and (C camera 1) VID_IN2	x	x	x
	M_CH1	(Y camera 2) VID_IN3 and (C camera 2) VID_IN4	x	x	x
	M_CH2	(Y camera 3) VID_IN5 and (C camera 3) VID_IN6	x	x	x
	M_CH3	(Y camera 4) VID_IN7 and (C camera 4) VID_IN8	x	x	x
	M_CH4	(Y camera 5) VID_IN9 and (C camera 5) VID_IN10		x	x
	M_CH5	(Y camera 6) VID_IN11 and (C camera 6) VID_IN12		x	x

When using the RGB path on Matrox Orion or Matrox Orion /RGB for 4Sight-II, the default channel of the sync signal can be overridden with any of the following by adding M\_SYNC to the required channel:

CameraType	Data channel	Sync channel	Signal
Monochrome	When the data channel is M_CH0, M_CH1, or M_CH2, the sync can be on any of these channels.	M_CH0	VID_IN1
		M_CH1	VID_IN2
		M_CH2	VID_IN3
		M_CH3	VID_IN4
	When the data channel is M_CH4, M_CH5, OR M_CH6, the sync can be on any of these channels.	M_CH4	VID_IN5
		M_CH5	VID_IN6
		M_CH6	VID_IN7
		M_CH7	VID_IN8
Color RGB	When the data channel is M_CH0 + M_SIGNAL, the sync can be on any of these channels.	M_CH0	VID_IN1
		M_CH1	VID_IN2
		M_CH2	VID_IN3
		M_CH3	VID_IN4
	When the data channel is M_CH1 + M_SIGNAL, the sync can be on any of these channels.	M_CH4	VID_IN5
		M_CH5	VID_IN6
		M_CH6	VID_IN7
		M_CH7	VID_IN8

## MdigControl()

- M\_GRAB\_SCALE, M\_GRAB\_SCALE\_X, and M\_GRAB\_SCALE\_Y can be set to any value between 0 and 255 since Matrox Orion supports arbitrary scaling. Passing a value between 0 and 1 will reduce the image size, while passing a value greater than 1 will zoom the image. A bilinear filter is always applied when the scaling value is not 1 or 0.5.

The values M\_FILL\_DESTINATION and M\_FILL\_DISPLAY are also supported for this control type.

- The default setting for the M\_GRAB\_START\_MODE control type is M\_FIELD\_START\_ODD.
- For the M\_GRAB\_TRIGGER\_SOURCE control type, note the meaning of the following control values:
  - M\_HARDWARE\_PORT0: Opto-isolated hardware trigger signal using a combination of Pin 34 (OPTOTRIG-) and Pin 35 (OPTOTRIG+) on Video Input connector.
  - M\_HARDWARE\_PORT1: TTL hardware trigger signal using Pin 20 on Video Input connector or Pin 2 (TRIGGER) on Digital Interface connector.
- The M\_GRAB\_MODE control type can also be set to M\_ASYNCHRONOUS\_QUEUED.
- The M\_GRAB\_EXPOSURE\_XX control types are not supported on Matrox Orion.
- Matrox Orion supports these additional **ControlType** parameter settings:

ControlType	Description & ControlValue		
M_GRAB_ABORT	Immediately stops a grabs in progress and queued grabs. Useful for cancelling a triggered grab that is waiting for the trigger.		
	M_DEFAULT	Stops the grab.	
M_GRAB_AUTOMATIC_INPUT_GAIN	Sets the input gain automatically when grabbing from the decoder path.		
	M_ENABLE	Automatic input gain.	
	M_DISABLE	Set the input gain with the M_GRAB_INPUT_GAIN control type.	
	M_DEFAULT	Same as M_ENABLE.	
M_GRAB_INPUT_GAIN	Set the gain applied to the input signal. When grabbing using the RGB path, valid input voltages and their corresponding gains are:		
		Input voltage	Gain
	M_GAIN0	1.4 - 2.0Vpp.	1.3
	M_GAIN1	1.0 - 1.4Vpp.	2
	M_GAIN2	0.7 - 1.0Vpp.	2.8
	M_GAIN3	0.0 - 0.7Vpp.	4
	M_DEFAULT	Same as M_GAIN2.	
	When grabbing using the decoder path and M_GRAB_AUTOMATIC_INPUT_GAIN is M_DISABLE, the gain can be set to any integer value from 0 to 255.		

ControlType	Description & ControlValue	
M_GRAB_SCALE_QUALITY	M_LOW	Each field is scaled separately before being composed into one scaled frame. This setting can cause some vertical scaling artifacts, but grab latency is minimal.
	M_HIGH	An entire frame is grabbed in on-board memory before scaling. This setting yields a high quality scaling, but grab latency is doubled.
	M_DEFAULT	Same as M_LOW.
M_GRAB_WINDOW_RANGE	Limit the range of the pixels grabbed in a monochrome buffer with the decoder path to between 16 and 235.	
	Limit the range of the pixels grabbed in a monochrome buffer with the RGB path to between 10 and 245.	
M_INPUT_FILTER	Specifies the filter of incoming data for all channels.	
	M_LOW_PASS_1	Low-pass filter at 10 Mhz.
	M_BYPASS	No input filter used.
	M_DEFAULT	Same as M_LOW_PASS_1.
M_INTERPOLATION_MODE	Specifies the type of interpolation used during a grab. This control type can also be used when the scaling factor is set to 1. Interpolation can be performed on each field separately, or on a frame, depending on the control value of M_GRAB_SCALE_QUALITY.	
	M_NEAREST_NEIGHBOR	Nearest neighbor interpolation.
	M_AVERAGE	Averaging interpolation.
	M_BILINEAR	Bilinear interpolation.
	M_DEFAULT	When the scaling factor is 1, same as M_NEAREST_NEIGHBOR, otherwise, M_BILINEAR.
M_USER_BIT+(3, 4)	Set the state of an output bit of the expanded video I/O connector: M_ON or M_OFF. The relationship between the MIL user-bit number and the actual user output bit on the Video Input connector is as follows:	
	User Bit#	<b>Expanded video I/O connector signal</b>
	3	USER1OUT signal.
	4	USER2OUT signal.



ControlType	Description & ControlValue	
M_VCR_INPUT_TYPE	Set the digitizer input to VCR. Improves the stability of the image when grabbing from a VCR. This is only valid when using the decoder path:	
	M_DEFAULT	Same as M_DISABLE
	M_DISABLE	Does not adjust the stability of the image.
	M_ENABLE	Improves the stability of the image when grabbing with a VCR.

## MdigGrab()/MdigGrabContinuous()

- It is not possible to grab into a color-band child buffer.
- When performing a grab, the width and the horizontal position of the grab destination buffer must be a multiple of 4 bytes. If not, the grab will be clipped accordingly.
- When acquiring data on Matrox Orion, the values 0 and 255 are reserved. Therefore, the range of possible input values spans from 1 to 254.
- When grabbing from a color source into a monochrome buffer, the hardware color space converter first converts the RGB data into YUV, and then stores the Y (luminance) component in the buffer. Data is passed through the input LUTs before the color space converter, therefore the LUTs are applied to the incoming data.
- When performing real-time grabs, Windows 98 and Windows Me does not provide the same performance and consistency as Windows NT/2000. This generally does not cause difficulty except in applications that require very fast response time to a hardware event. For example, an application that must grab sequences of images without missing any frames might not have sufficient time for the proper response to a hook function. Therefore, if grabbing a sequence, you should try to avoid code that needs an immediate response and favor code that supports a short delay after the event. For example, it is recommended to call the next grab operation (**MdigGrab()**) from the function hooked to the start of the current grab, instead of calling it from a function hooked to the end of the grab. This will queue

the next grab operation while the current grab is in progress, and will allow the next grab to start immediately after the current grab. This should prevent the loss of a frame due to delay.

If these restrictions affect a major part of your application(s), we recommend the use of Windows NT/2000 in addition to the above recommendations.

## MdigHookFunction()

- A function hooked to an M\_GRAB\_END event is called when all the data of the grab is transferred to the Host. This means that this hooked function can be called after the M\_GRAB\_START or M\_GRAB\_FRAME\_START event of the next frame.
- The **HookType** parameter M\_UART\_DATA\_RECEIVED is not supported.

## MdigInquire()

- The **InquireType** parameter can also be set to the following:

InquireType	Description
M_BRIGHTNESS_REF*	Brightness reference level.
M_COLOR_MODE	Color mode: M_RGB, M_MONO8_VIA_RGB, M_EXTERNAL_CHROMINANCE, M_COMPOSITE, or M_MONOCHROME.
M_CONTRAST_REF*	Contrast reference level.
M_HUE_REF*	Hue reference level.
M_SATURATION_REF*	Saturation reference level.
*Note that these inquire types are only supported on composite and Y/C cameras (decoder path). However, for monochrome cameras (decoder path), the M_HUE_REF and M_SATURATION_REF inquire types are not supported.	
M_GRAB_AUTOMATIC_INPUT_GAIN	Mode of automatic input gain: M_ENABLE or M_DISABLE.
M_GRAB_IN_PROGRESS	Current grab state: M_YES or M_NO.
M_GRAB_INPUT_GAIN	When grabbing using the RGB path, the gain applied to the input signal: M_GAIN0, M_GAIN1, M_GAIN2, or M_GAIN3. When grabbing using the decoder path and M_GRAB_AUTOMATIC_INPUT_GAIN is M_DISABLE, any integer value from 0 to 255.

InquireType	Description
M_GRAB_MODE	Whether the grab is synchronous: M_SYNCHRONOUS, M_ASYNCHRONOUS, and M_ASYNCHRONOUS_QUEUED.
M_GRAB_SCALE_QUALITY	Whether scaling is performed on a frame or field basis. M_LOW (field) or M_HIGH (frame).
M_GRAB_SCALE	Scaling factor.
M_GRAB_SCALE_X	Horizontal scaling factor.
M_GRAB_SCALE_Y	Vertical scaling factor.
M_GRAB_WINDOW_RANGE	State of limiting the range of the grabbed pixel values: A value between 16 and 235 when using the decoder path. A value between 10 and 245 when using the RGB path.
M_HOOK_MASTER_THREAD_HANDLE	Returns the handle of the main interrupt thread that dispatches the handling of the grab hooks to their respective threads.
M_HOOK_MASTER_THREAD_ID	Returns the ID of the main interrupt thread that dispatches the handling of the grab hooks to their respective threads.
M_INPUT_FILTER	Input filter of incoming data for all channels: M_LOW_PASS_1 or M_BYPASS.
M_INPUT_SIGNAL_SOURCE	Input-signal source: M_HARDWARE_PORT0 (Video Input connector).
M_INPUT_SIGNAL_PRESENT	Video input signal present: M_YES or M_NO. M_YES: whenever the DCF is for a color camera, but only when there is a line lock and color lock. When a monochrome camera with a color DCF is used to grab images, the function will never return M_YES. However, this monochrome camera/color DCF combination is still supported on Matrox Orion, because it is useful when switching between channels.
M_INPUT_SIGNAL_COLOR_LOCK	Video input signal color lock. Returns M_TRUE when a color lock is achieved, otherwise it returns M_FALSE. Not supported when grabbing with the RGB path.
M_INTERPOLATION_MODE	The type of interpolation used during a grab: M_NEAREST_NEIGHBOR, M_AVERAGE or M_BILINEAR.

InquireType	Description
M_USER_BIT+ (1,2,3,4)	Current state of an input/output bit of the expanded video I/O connector: M_ON (1) or M_OFF (0).
	<b>User Bit #</b>
	<b>Expanded video I/O connector signal</b>
	1 USER1IN signal.
	2 USER2IN signal.
	3 USER1OUT signal.
	4 USER2OUT signal.
M_VCR_INPUT_TYPE	Whether image's stability is improved when grabbing from a VCR: M_DISABLE or M_ENABLE.

## MdigLut()

Matrox Orion has three 256 x 8-bit input lookup tables (LUTs) that can be used when grabbing from the decoder path or the RGB path. When grabbing into a YUV buffer, data is passed through the input LUTs before the color space converter, therefore the LUTs are applied to the incoming data. Note that there are no input LUTs on Matrox Orion /Standard for 4Sight-II.

## MdigReference()

- When grabbing using the RGB section of Matrox Orion, you can control the black and white reference levels of each input channel separately (M\_BLACK\_REF, M\_WHITE\_REF). To specify the channel, combine M\_CH0\_REF, M\_CH1\_REF, or M\_CH2\_REF with M\_BLACK\_REF or M\_WHITE\_REF.
- For M\_BLACK\_REF, note the meaning of the **ReferenceLevel** parameter settings:
  - The minimum voltage level (M\_MIN\_LEVEL) is 0.6 V.
  - The maximum voltage level (M\_MAX\_LEVEL) is 1.6 V.
- For M\_WHITE\_REF, note the meaning of the **ReferenceLevel** parameter settings:
  - The minimum voltage level (M\_MIN\_LEVEL) is 1.6 V.
  - The maximum voltage level (M\_MAX\_LEVEL) is 2.6 V.

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).

- When grabbing monochrome data using the decoder section of Matrox Orion, you can control the brightness and contrast levels of the input signal using M\_BRIGHTNESS\_REF and M\_CONTRAST\_REF. Valid values are between M\_MIN\_LEVEL to M\_MAX\_LEVEL.
- When grabbing color using the decoder section of Matrox Orion, you can control the brightness, contrast, hue, and saturation levels of the composite input signal, using M\_BRIGHTNESS\_REF, M\_CONTRAST\_REF, M\_HUE\_REF, and M\_SATURATION\_REF. Valid values are between M\_MIN\_LEVEL to M\_MAX\_LEVEL.

## MdispAlloc()

For auxiliary displays, these are the display formats (for encoded video) that you can use to display from the Matrox Orion board:

DispFormat	Description
"M_NTSC"*	Encoder enabled in composite format with colorburst (default).
"M_NTSC_RGB"	Encoder enabled in RGB format with sync on green.
"M_NTSC_YC"*	Encoder enabled in Y/C formats with colorburst.
"M_RS170"	Encoder enabled in composite format without colorburst.
"M_PAL"*	Encoder enabled in composite format with colorburst.
"M_PAL_RGB"	Encoder enabled in RGB format with sync on green.
"M_PAL_YC"*	Encoder enabled in Y/C formats with colorburst.
"M_CCIR"	Encoder enabled in composite format without colorburst.
*Selecting these display formats will produce both composite and Y/C outputs.	

## MdispControl()

When using the encoder, the **ControlType** parameter can be set to one of the following:

ControlType	Description & ControlValue	
M_ENCODER_FILTER	Select a filter for luminance.	
	M_LOW_PASS_0	Low-pass filter type A.
	M_LOW_PASS_1	Low-pass filter type B.
	M_LOW_PASS_2	5.5 MHz low-pass filter.
	M_NOTCH	Subcarrier frequency low-pass filter.
	M_DEFAULT	Same as M_LOW_PASS_0.
M_ENCODER_PEDESTAL	Specify whether a pedestal is to be generated in the composite video signal.	
	M_ENABLE	Generate a pedestal in the composite video signal.
	M_DISABLE	Do not generated a pedestal in the output.
	M_DEFAULT	Same as M_ENABLE
M_SYNC_TYPE	Specifies the output sync signal. This control is only supported when the display format is RGB.	
	M_GREEN	The output sync signal is on green.
	M_SEPARATE	The sync and data signals are separated: the output sync is on pin 12, and the composite data signal is on pin 10 of the HD-44 connector.
	M_DISABLE	The encoder output does not contain a sync.
	M_DEFAULT	Same as M_GREEN.

## MdisplInquire()

- For windowed displays, the **InquireType** parameter can also be set to:

InquireType	Description
M_VGA_PIXEL_FORMAT	The display's pixel format:
	M_MONO8 + M_PLANAR
	M_RGB32 + M_PACKED
	M_BGR32 + M_PACKED

- When using the encoder, the **InquireType** parameter can be set to one of the following:

InquireType	Description
M_ENCODER_FILTER	Returns the filter selected for luminance. M_LOW_PASS_0, M_LOW_PASS_1, M_LOW_PASS_2, or M_NOTCH.
M_ENCODER_PEDESTAL	Returns whether a pedestal is to be generated in the composite video signal. M_ENABLE or M_DISABLE
M_SYNC_TYPE	Returns which output signal has the sync encoded. M_GREEN, M_SEPARATE, M_DISABLE

## MsysAlloc()

- To allocate an Orion system, you must select M\_SYSTEM\_ORION as the **SystemType** parameter. This selection opens communication with the board and will automatically allow the system to use some Host memory, and any available graphics controller, if necessary.
- To allocate an Orion system, your display resolution must be in an 8-bit or true color (32-bit) display mode (display depth).

## MsysControl()

- When a live grab operation uses a software trigger and **MdigHalt()** is issued, a software trigger is automatically generated to invoke a last grab (if you did not disable M\_LAST\_GRAB\_IN\_TRUE\_BUFFER). You can disable this automatic trigger; however, you would then have to issue a software trigger call after the **MdigHalt()** call (these calls

must be issued from different threads). To disable the **MdigHalt()** automatic trigger, set the following **ControlType** to M\_DISABLE:

ControlType	ControlValue	
M_LIVE_GRAB_END_TRIGGER	M_ENABLE	Default for software triggered cameras.
	M_DISABLE	Default for other camera types.

- When a live grab operation uses a hardware trigger, **MdigHalt()** will wait indefinitely for a hardware trigger to invoke a last grab (if you did not disable M\_LAST\_GRAB\_IN\_TRUE\_BUFFER). You can override this default so that **MdigHalt()** will cause an immediate last grab and will not wait for a hardware trigger. To do so, set the following **ControlType** to M\_DISABLE:

ControlType	ControlValue	
M_LIVE_GRAB_END_TRIGGER	M_ENABLE	Default for hardware triggered cameras.
	M_DISABLE	Default for other camera types.

## MsysInquire()

- You can request the following information with the **InquireType** parameter:

InquireType	Description
M_BOARD_REVISION	The board revision (long value).
M_BOARD_TYPE	The type of system board: M_ORION, M_ORION_4SIGHT_II_STD, M_ORION_4SIGHT_II_RGB
M_PHYSICAL_ADDRESS_VGA	The physical address of the VGA frame buffer surface.
M_LIVE_GRAB_END_TRIGGER	Generate automatic trigger at end of grab: M_ENABLE or M_DISABLE.



---

## ***Chapter 6: MIL and the Matrox 4Sight and Matrox 4Sight-II units***

*This section discusses features of MIL that are specific to the Matrox 4Sight and Matrox 4Sight-II units, and ways that optimize their performance.*

---

## Specific features of Matrox 4Sight and Matrox 4Sight-II units

Matrox 4Sight and Matrox 4Sight-II are self-contained units that integrate image capture, storage, processing, and display, along with general purpose input/output (I/O), standard 10/100BaseT Ethernet interfacing capabilities, and IEEE 1394 tree topology capabilities. In addition, Matrox 4Sight and Matrox 4Sight-II support add-on boards compliant with the PC-104/*Plus*<sup>™</sup> form factor.

Matrox 4Sight and Matrox 4Sight-II currently support the Microsoft Windows NT Embedded, Microsoft Windows NT Workstation and Microsoft Windows 2000 operating systems. Microsoft Windows CE will be supported in a future release.

This chapter outlines basic features of both platforms, and relates them to specific functions in MIL. However, for more detailed information on the capabilities of Matrox 4Sight and Matrox 4Sight-II, you should refer to their respective user guides.

For block diagrams and flow diagrams of the Matrox 4Sight and Matrox 4Sight-II motherboards, you can also refer to *Appendix A: Board flow diagrams*.

### Display capabilities

Matrox 4Sight includes an integrated graphics controller that can output to a standard VGA monitor or flat panel, for display resolutions of 8 bits per pixel (pseudo-color) or 16 bits per pixel (hi-color). You can also configure the interface for output to an NTSC/PAL/RGB device, such as a TV or VCR.

Matrox 4Sight-II integrates the Matrox G450 graphics controller for all display functions. It features a 2X, 32-bit, master AGP Host interface, a primary and secondary RAMDAC, a TV/video encoder, a TMDS transmitter, and 16 Mbytes DDR SDRAM. The controller allows for independent display on one or two display devices, which include a VGA monitor, an analog flat-panel, a digital flat-panel, or an NTSC/PAL/RGB device, such as a TV or VCR.

---

### Non-destructive graphic annotations

To drive the display section, Matrox 4Sight uses its processor's companion chip, while Matrox 4Sight-II uses Matrox's G450. Both support non-destructive graphic annotations displayed on top of live video.

---

### NTSC/PAL/RGB video output

The video encoder on Matrox 4Sight and Matrox 4Sight-II allows for either composite, component (Y/C), or component RGB output to a TV or VCR. On Matrox 4Sight, the video encoder is controlled by the Matrox 4Sight BIOS. Refer to the *Matrox 4Sight User Guide* for details on controlling the encoder output from the BIOS.

On Matrox 4Sight-II, such capabilities are controlled directly by MIL. Please refer to the *Matrox Imaging Library User guide* for information on displaying images on a NTSC/PAL/RGB device.

## Frame grabbers

Matrox 4Sight supports the following Matrox frame grabbers:

- Matrox Meteor-II /Standard.
- Matrox Meteor-II /Multi-Channel.

Both frame grabbers support the optional Matrox Meteor-II MJPEG module for PC/104-Plus. Please refer to *Chapter 4: MIL and the Matrox Meteor-II platform*, for details on the capabilities of Matrox Meteor-II frame grabbers for PC/104-Plus.

Matrox 4Sight-II supports all of the above frame grabbers. It supports the following Matrox frame grabbers as well:

- Matrox Meteor-II /Digital.
- Matrox Orion /Standard for 4Sight-II.
- Matrox Orion /RGB for 4Sight-II.

Refer to *Chapter 4: MIL and the Matrox Meteor-II platform*, for details on the particularities of existing MIL functions on Matrox Meteor-II.

Refer to *Chapter 5: MIL and the Matrox Orion platform*, for information on the particularities of existing MIL functions on the Matrox Orion modules for 4Sight-II<sup>1</sup>.

## Input and output data interfaces

Matrox 4Sight and Matrox 4Sight-II both integrate an IEEE 1394 interface and an auxiliary I/O interface for data exchange between Matrox 4Sight/4Sight-II and specialized devices.

---

### *The IEEE 1394 interface*

The IEEE 1394 interface is a very fast external bus, which supports isochronous (real-time) digital data transfer between IEEE 1394 devices (such as, IEEE 1394 DCAM-compliant cameras) and Matrox 4Sight/4Sight-II at rates of up to 400 Mbit/sec.

Despite being an integrated component on the motherboard, this IEEE 1394 serial bus controller provides identical functionality as the Matrox Meteor-II /1394 frame grabber. Please refer to *Chapter 4: MIL and the Matrox Meteor-II platform*, for information on the particularities of MIL functions using a Meteor-II /1394 system.

### **Δ Important**

Windows NT 4.0 does not support the IEEE 1394 interface; however by installing the MIL Meteor-II /1394 driver, you can use IEEE 1394 DCAM-compliant cameras through MIL. Once you install this driver, you can only use your unit's IEEE 1394 interface to attach IEEE 1394 DCAM-compliant cameras, and no other IEEE 1394 devices. In addition, you will only be able to use these cameras through MIL.

Windows 2000 supports the IEEE 1394 interface, however you must install the MIL Matrox Meteor-II /1394 driver to use IEEE 1394 DCAM-compliant cameras through MIL. Once installed, you can still interface any IEEE 1394 device to the IEEE 1394 interface. However, you will only be able to use your IEEE 1394 DCAM-compliant cameras through MIL.

- 
1. For specific details regarding the hardware components of the Matrox Orion modules for 4Sight-II, please refer to the *Matrox 4Sight-II Installation and Hardware reference manual*.

---

### The auxiliary I/O interface

The auxiliary I/O interface is a discrete, digital interface, used for operating external devices, such as a PLC or motion control unit.

Matrox 4Sight can send and receive up to twenty low-voltage TTL (LVTTTL) signals to and from external LVTTTL devices through the auxiliary I/O interface. Matrox 4Sight-II can send and receive up to sixteen LVTTTL signals.

You can also connect non-LVTTTL devices to Matrox 4Sight-II's auxiliary I/O interface, using the Matrox *opto-coupling module*. This device electrically isolates the external device from your unit, and acts as the device's electrical switch. Please refer to the *Matrox 4Sight-II Installation and Hardware reference* manual for details.

Auxiliary input signals have interrupt-generation capabilities. On Matrox 4Sight, you can assign either IRQ 3, 4, 5, 7, 9, 10, or 11 to transmit the interrupt. On Matrox 4Sight-II, you can assign IRQ 10 to transmit the interrupt. The IRQ is assigned by entering the *BIOS Setup* program, and configuring the appropriate menu item.

Your application can also poll any of the auxiliary I/O bits to determine their state, if you do not want to assign an IRQ.

Please refer to the BIOS reference appendix, in the Matrox 4Sight and Matrox 4Sight-II documentation, for details on configuring the auxiliary I/O interface interrupt. You can then refer to the *Auxiliary inputs/outputs* section, later in this chapter, for more information on programming and using the auxiliary I/O bits in your applications.

---

## Using MIL with Matrox 4Sight and Matrox 4Sight-II

MIL treats Matrox 4Sight and Matrox 4Sight-II as a normal computer. Accordingly, you don't allocate Matrox 4Sight or Matrox 4Sight-II as a system. However, to use a Matrox Orion module for 4Sight-II, or a Matrox Meteor-II frame grabber, you must first allocate the board as an Orion or Meteor-II system.

To use IEEE 1394 DCAM-compliant cameras, you must allocate Meteor-II /1394 systems. Then, assign a camera from the pool of IEEE 1394 DCAM-compliant cameras, to a particular system, by allocating it as a digitizer on that system. Please refer to *Chapter 4: MIL and the Matrox Meteor-II platform* for details on allocating a Meteor-II /1394 system, allocating a Meteor-II /1394 digitizer, and managing multiple IEEE 1394 DCAM-compliant cameras.

This chapter relates the Matrox 4Sight and Matrox 4Sight-II unit to the following:

- Auxiliary input and output signals.
- Particularities of existing MIL functions on Matrox 4Sight and Matrox 4Sight-II.
- Matrox 4Sight/4Sight-II-specific MIL functions.

Please refer to *milmet2.txt* file in the `\MATROX IMAGING\DRIVERS\DOC` (user-specified) directory for any additions/modifications to these board-specific notes.

Note that when installing a redistribution (run-time) version of MIL on Matrox 4Sight and Matrox 4Sight-II, the license number will be based on a hardware fingerprint on your unit. This means that you will not be able to copy the licensed version of MIL to another computer.

## Auxiliary outputs/inputs

The **MsysControl**(M\_DEFAULT\_HOST, M\_USER\_BIT...) command permits control and access to the auxiliary I/O pins (user bits) on the Matrox 4Sight and Matrox 4Sight-II platforms. For example, you define whether a user bit is in input or output mode using **MsysControl()**.

The **MsysInquire**(M\_DEFAULT\_HOST, M\_USER\_BIT...) command permits reading of the user bits on the Matrox 4Sight and Matrox 4Sight-II systems. You can manipulate individual user bits, or groups of user bits, using a mask.

### Auxiliary outputs

To set a user bit to output mode and enable it, perform the following steps:

1. In MIL, set the required I/O pin or group of user bits to output mode using the **MsysControl()** function. Use the M\_USER\_BIT\_MODE control type and M\_OUTPUT control value combination.
2. Set the functional state of the required user bit(s) using the same **MsysControl()** function. In this case, use the M\_USER\_BIT\_VALUE control type and M\_ON control value combination.

The following snippet of code shows how to set the mode and functional state of a single user bit, as well as how to set the mode and functional state of two user bits using a bit-encoded mask.

```
/* Set bit 1 to output mode */
MsysControl(M_DEFAULT_HOST,M_USER_BIT_MODE+1, M_OUTPUT);

/* Set the functional state of bit 1 to on.*/
MsysControl(M_DEFAULT_HOST,M_USER_BIT_VALUE+1, M_ON);

/* Set bits 2 and 3 to output mode. */
MsysControl(M_DEFAULT_HOST,M_USER_BIT_MODE+M_BIT_MASK(0xC), 0xC);

/* Set bit 2 to on, but bit 3 to off. */
MsysControl(M_DEFAULT_HOST,M_USER_BIT_VALUE+M_BIT_MASK(0xC), 0x4);
```

## Auxiliary inputs

To handle input user bits, you can either poll their state or hook a function to their change of state.

To hook a function to the change in a user bit state, follow the steps below:

1. In the BIOS, assign an interrupt line to the entire group of user bits. Refer to the Matrox 4Sight and Matrox 4Sight-II documentation for more information on assigning an interrupt line to the auxiliary I/O user bits.
2. In MIL, set the required user bit or group of user bits to input mode using the **MsysControl()** function. Use the `M_USER_BIT_MODE` control type and `M_INPUT` control value combination.
3. In MIL, specify whether an input user bit or group of input user bits should generate an interrupt upon a rising edge or falling edge (interrupt mode). To set the mode, use the `M_USER_BIT_INTERRUPT_MODE` control type and either the `M_EDGE_RISING` or `M_EDGE_FALLING` control value combination.
4. Hook a function to a user bit's specific change of functional state, using the **MsysHookFunction()**. The change in functional state is determined according to the control value specified for the `M_USER_BIT_INTERRUPT_MODE` control type (see step 3 above).

Within the scope of the hook function itself, inquire which user bit(s) generated the interrupt, using the **MsysGetHookInfo()** function. Then, execute the next step according to the result of the inquiry.



5. Enable the interrupt state for the required user bit(s) using the same **MsysControl()** function. In this case, use the **M\_USER\_BIT\_INTERRUPT\_STATE** control type and **M\_ENABLE** control value combination.

The following snippet of code demonstrates how to hook a function and its related hook data to a particular change of state of a user bit (rising edge, in this case). When the interrupt mode is set to rising edge, a user bit's change of state from 0 (OFF) to 1 (ON) is required to generate the interrupt. When no longer required, the specified function is then unhooked from this event.

```
/* Set user bit 1 to input mode. */
MsysControl(M_DEFAULT_HOST,M_USER_BIT_MODE+1, M_INPUT);

/* Set the interrupt mode to be generated on a rising edge for bit 1. */
MsysControl(M_DEFAULT_HOST,M_USER_BIT_INTERRUPT_MODE+1, M_EDGE_RISING);

/* Hook your function (YourHookFunction) and any related hook data (YourHookData)
 * to a change of functional state of the user bit.
 */
MsysHookFunction(M_DEFAULT_HOST,M_USER_BIT_CHANGE,YourHookFunction,
    (void*)&YourHookData);

/* Enable the interrupt state of bit 1. */
MsysControl(M_DEFAULT_HOST,M_USER_BIT_INTERRUPT_STATE+1, M_ENABLE);

/* Perform processing operations...*/

/* Unhook the function */
MsysHookFunction(M_DEFAULT_HOST,M_USER_BIT_CHANGE+M_UNHOOK,YourHookFunction,
    (void*)&YourHookData);
```

## Particularities of existing MIL functions on Matrox 4Sight and Matrox 4Sight-II

Certain functions have special features or functionality on Matrox 4Sight and Matrox 4Sight-II. This table provides an overview of the affected commands. Details and procedures are described in the individual command descriptions, which follow.

Commands	Matrox 4Sight/4Sight-II particularities
MsysControl()	Matrox 4Sight/4Sight-II-specific control type additions.
MsysInquire()	Matrox 4Sight/4Sight-II-specific inquire options.

- ❖ Refer to *Chapter 4: MIL and the Matrox Meteor-II platform*, for information on the integrated 1394 serial bus controller and ports, as well as details on the particularities of the other Matrox Meteor-II boards for PC/104-Plus, and the Matrox Meteor-II MJPEG module.
- ❖ Refer to *Chapter 5: MIL and the Matrox Orion platform*, for information on the particularities of existing MIL functions on the Matrox Orion modules for 4Sight-II.

## MsysControl()

To control a specific user bit or a group of user bits on Matrox 4Sight or Matrox 4Sight-II, set the **ControlType** and **ControlValue** combination to the following. To affect a single user bit, add its user bit number to the control type. To affect a group of user bits, add M\_BIT\_MASK to the control type and identify which bits to affect in a bit-encoded mask value. Enabled bits in the mask value identify which user bit to affect. For example, M\_USER\_BIT\_VALUE + M\_BIT\_MASK(0x4f) affects user bits 0,1,2,3, and 6. If a group of user bits is specified, pass a bit-encoded control value that specifies the setting for each of these user bits.

Note that for these control types, only M\_DEFAULT\_HOST or a Host system can be used as a target system (**SystemId**).

ControlType	ControlValue	Description
M_USER_BIT_VALUE + (0-15) (for Matrox 4Sight-II) or + (0-19) (for Matrox 4Sight) or + M_BIT_MASK (bit-encoded mask value)	Set the functional state of the system's specified user bit(s). The specified bit(s) must be in output mode.	
	M_ON or <b>ControlValue</b> bit set to 1.	Set the user bit to 1.
	M_OFF or <b>ControlValue</b> bit set to 0.	Set the user bit to 0. (default)
M_USER_BIT_MODE + (0-15) (for Matrox 4Sight-II) or + (0-19) (for Matrox 4Sight) or + M_BIT_MASK (bit-encoded mask value)	Set the system's specified user bit(s) input/output mode.	
	M_INPUT or <b>ControlValue</b> bit set to 0.	Set the user bit to input. (default)
	M_OUTPUT or <b>ControlValue</b> bit set to 1.	Set the user bit to output.
M_USER_BIT_INTERRUPT_STATE + (0-15) (for Matrox 4Sight-II) or + (0-19) (for Matrox 4Sight) or + M_BIT_MASK (bit-encoded mask value)	Set the system's specified user bit(s) interrupt state. The specified bit(s) must be in input mode.	
	M_ENABLE or <b>ControlValue</b> bit set to 1.	Enable interrupt.
	M_DISABLE or <b>ControlValue</b> bit set to 0.	Disable interrupt. (default)
M_USER_BIT_INTERRUPT_MODE + (0-15) (for Matrox 4Sight-II) or + (0-19) (for Matrox 4Sight) or + M_BIT_MASK (bit-encoded mask value)	Set the system's specified user bit(s) interrupt mode. This control type specifies whether the interrupt will be generated because of an off-to-on (rising edge) or on-to-off (falling edge) change in functional state of the user bit(s).	
	M_EDGE_RISING or <b>ControlValue</b> bit set to 0.	Set interrupt mode to rising edge mode. (default)
	M_EDGE_FALLING or <b>ControlValue</b> bit set to 1.	Set interrupt mode to falling edge mode.

MsysInquire()

The following additional inquire types are supported on Matrox 4Sight and Matrox 4Sight-II, using M\_DEFAULT\_HOST, or a Host system, as the target system (**SystemId**).

InquireType	Description	
For the following inquire types, it is required that your unit's BIOS version be equal to or greater than ft1.01.000 (in the case of Matrox 4Sight-II), or fs1.02.022 (in the case of Matrox 4Sight).		
M_USER_BIT_COUNT	Determine the number of user bits available to the user on the auxiliary I/O interface connector. Either: 16 (in the case of Matrox 4Sight-II), or 20 (in the case of Matrox 4Sight) will be returned.	
M_USER_BIT_OPTOMODULE	Determine if the Matrox opto-module is present or not. It will return: M_TRUE if present, or M_FALSE if absent. Note that the return value will always indicate M_FALSE on Matrox 4Sight, as the Matrox opto-coupling module is not supported by this unit.	
For the following inquire types, you should consider the following: ■ To inquire about a single user bit, add the user bit number to the inquire type. ■ To inquire about a group of user bits, add M_BIT_MASK to the inquire type and identify which bits to inquire about in a bit-encoded mask value. In this case, a bit-encoded inquire value is returned to the <b>UserVarPtr</b> , which must be a pointer to a long.		
M_USER_BIT_VALUE + (0-15) (for Matrox 4Sight-II) or + (0-19) (for Matrox 4Sight) or + M_BIT_MASK (bit-encoded mask value)	Read the functional state of the system's specified user bit(s).	
	M_ON or bit value is 1.	The user bit is set to 1.
	M_OFF or bit value is 0.	The user bit is set to 0.

InquireType	Description	
M_USER_BIT_MODE + (0-15) (for Matrox 4Sight-II) or + (0-19) (for Matrox 4Sight) or + M_BIT_MASK (bit-encoded mask value)	Read the system's specified user bit(s) mode.	
	M_INPUT or bit value is 0.	The user bit is in input mode.
	M_OUTPUT or bit value is 1.	The user bit is in output mode.
M_USER_BIT_INTERRUPT_STATE + (0-15) (for Matrox 4Sight-II) or + (0-19) (for Matrox 4Sight) or + M_BIT_MASK (bit-encoded mask value)	Read the system's specified user bit(s) interrupt state.	
	M_ENABLE or bit value is 1.	Interrupt is enabled.
	M_DISABLE or bit value is 0.	Interrupt is disabled.
M_USER_BIT_INTERRUPT_MODE + (0-15) (for Matrox 4Sight-II) or + (0-19) (for Matrox 4Sight) or + M_BIT_MASK (bit-encoded mask value)	Read the system's specified user bit(s) interrupt mode.	
	M_EDGE_RISING or bit value is 0.	Interrupt mode is in rising edge mode.
	M_EDGE_FALLING or bit value is 1.	Interrupt mode is in falling edge mode.

---

## Matrox 4Sight and Matrox 4Sight-II-specific MIL functions

There are Matrox 4Sight and Matrox 4Sight-II-specific functions that inquire about or deal with hooking a function to a user bit’s specific change of functional state. This table provides an overview of the affected commands. Details and procedures are described in the individual command descriptions, which follow.

Commands	Matrox 4Sight/4Sight-II particularities
MsysGetHookInfo()	Matrox 4Sight/4Sight-II-specific hook event inquire.
MsysHookFunction()	Matrox 4Sight/4Sight-II-specific system hook function.

## MsysGetHookInfo

**Synopsis** Get information about a hook event.

**Format** `long MsysGetHookInfo(SystemId, EventId, InquireType, UserVarPtr)`

MIL_ID SystemId;	System identifier
MIL_ID EventId;	Identifier received by the hook-handler function
long InquireType;	Type of information which is inquired
void MPTYPE *UserVarPtr;	Storage location for requested information.

**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.

The **SystemId** parameter specifies the identifier of the system. This parameter must be set to M\_DEFAULT\_HOST.

The **EventId** parameter is the system event identifier received by the hook-handler function (see **MsysHookFunction()**).

The **InquireType** parameter specifies the type of information about which to inquire. If the hook-handler function was called with a **HookType** parameter equal to M\_USER\_BIT\_CHANGE, the supported value for **Type** is:

InquireType	Description
M_USER_BIT	User bit that triggered the hook. Possible values are 0 to 15 (for Matrox 4Sight-II), or 0 to 19 (for Matrox 4Sight). Each value corresponds to one of the 16 or 20 user bits (auxiliary I/O pins) available. If more than one user bit triggers an interrupt at the same time, then the hook-handler function (or chain of hook-handler functions) is called for each input user bit that generated an interrupt. As a result, any and all user specified function(s) hooked to M_USER_BIT_CHANGE ( <b>MsysHookFunction()</b> ) will be executed for each interrupt.

The **UserVarPtr** parameter specifies the address of the variable 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.

**See also** **MsysHookFunction()**



# MsysHookFunction

**Synopsis** Hook a function to a system event.

**Format** void MsysHookFunction(SystemId, HookType, HookHandlerPtr, UserDataPtr)

MIL_ID SystemId;	System identifier
long HookType;	Type of event to hook
MSYSHOOKFCTPTR HookHandlerPtr;	Pointer to the function to call when the specified system event occurs
void MPTYPE *UserDataPtr;	User data pointer

**Description** This function allows the user 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, 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 **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.

The **SystemId** parameter specifies the identifier of the system. You can hook a function to an event only on an M\_DEFAULT\_HOST system.

The **HookType** parameter specifies the system event to which to hook the function. This parameter can be set to one of the following values. Note that a hooked function must be unhooked by combining the **HookType** parameter with M\_UNHOOK.

HookType	Description
M_USER_BIT_CHANGE	Calls the function only when an input user bit changes in accordance with its specified interrupt mode, M_EDGE_RISING or M_EDGE_FALLING (see <b>MsysControl()</b> ). You can hook a function to this event only on an M_DEFAULT_HOST system. It is prudent to verify that the appropriate user bit triggered the hook-handler function. To determine which bit caused the event, call <b>MsysGetHookInfo()</b> from within your hook-handler function.
M_UNHOOK + M_USER_BIT_CHANGE	Unhooks the specified function if hooked to an M_USER_BIT_CHANGE event. You can unhook a function to this event only on an M_DEFAULT_HOST system.

The **HookHandlerPtr** parameter specifies the address of the function that should be called when the specified event occurs. The hook-handler function must be declared as follows:

long MFTYPE HookHandler(HookType, EventId, UserDataPtr);	
long HookType;	Type of system event that generated the call
MIL_ID EventId;	Event identifier to pass to <b>MsysGetHookInfo()</b> when inquiring about the hooked event
void MPTYPE *UserDataPtr;	User data pointer that was passed (as UserDataPtr) by MsysHookFunction()

Upon successful completion, the hook-handler function should return M\_NULL. Note, MSYSHOOKFCTPTR, MFTYPE, and MPTYPE are reserved MIL predefined types for functions and data pointers.

The **UserDataPtr** parameter 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.

**See also**    **MsysGetHookInfo()**

---

## ***Chapter 7: MIL and the Matrox Cronos platform***

*This section discusses features of MIL that are distinct to the Matrox Cronos platform and ways that optimize the board's performance.*

---

## Matrox Cronos-specific features

Matrox Cronos is a frame grabber capable of acquiring standard monochrome video in RS-170/CCIR format, composite (CVBS) and component (Y/C) color video in NTSC/PAL format. The Matrox Cronos base board can switch between a maximum of four monochrome/composite cameras. Matrox Cronos can switch and grab from up to 12 additional monochrome/composite cameras with the Composite Video Input modules, and up to four Y/C cameras with the Y/C Video Input module. Finally, connecting the Matrox Cronos I/O module allows you to control user bits.

---

Miscellaneous  
information

Matrox Cronos can be used as a hardware fingerprint for run-time licenses.

See Appendix A for a data flow diagram.

❖ Matrox Cronos supports Windows Me/2000/XP.

---

## Using Matrox Cronos with MIL

To use your Matrox Cronos board with MIL, you must allocate a Cronos system (M\_SYSTEM\_CRONOS), using **MsysAlloc()**. This establishes a MIL system environment in which MIL uses some Host memory, and any available graphics controller for grabbing and displaying images.

This chapter relates Cronos systems to the following:  
*Fast-channel switching, Grabbing from multiple cameras with different DCFs and Particularities of existing MIL functions on Matrox Cronos.*

Refer to the *milcronos.txt* file in the  
`\MATROX IMAGING\DRIVERS\DOC` (or user-specified)  
 directory for any additions/modifications to these board specific notes.

---

## Fast-channel switching

Channel switching is always performed with **MdigChannel()**. If you intend to switch channels after grabbing a single field or frame, use the fast-channel switching mode to minimize time between grabs; set the `M_CAMERA_LOCK` control type to `M_ENABLE+M_FAST`. Although this mode is optimized for applications that use several cameras of the same type, the `M_ENABLE+M_FAST` control value does impose certain restrictions:

- Some digitizer hook events might only be generated a few frames after the channel switch: `M_FRAME_START`, `M_FIELD_START`, `M_FIELD_START_ODD`, and `M_FIELD_START_EVEN`.
- When using the Matrox Cronos I/O module, only user bits 7, 8, 9, and 10 are available. User bits 0 to 6 are used internally, and connecting external devices to these user bits risks a hardware failure.
- Both trigger inputs and triggered I/O are unavailable, since they are used internally. Connecting external devices to these inputs risks a hardware failure.
- In frame mode, the following hook events are unreliable: `M_GRAB_FIELD_END`, `M_GRAB_FIELD_END_ODD`, and `M_GRAB_FIELD_END_EVEN`. Note these hook events are reliable in field mode.

---

## Grabbing from multiple cameras with different DCFs

In some applications, you might want to switch between cameras with different DCFs that are attached to the same digitizer. This procedure normally involves allocating a digitizer, grabbing the required frame, freeing the digitizer, and then allocating the digitizer again with the second DCF; this can increase the time required for the operation. On Matrox Cronos, MIL can circumvent this problem by using a fast DCF-switching technique which is outlined in the steps below:

1. Make as many calls to **MdigAlloc()** as you have cameras, with different formats, from which you want to grab. With each call, the **DigNum** parameter must be set to **M\_DEV0**, since there is only one physical digitizer; each allocation only sets up a virtual instance of the digitizer.
  2. Specify all required digitizer settings using **MdigControl()**, **MdigChannel()**, **MdigReference()**, and **MdigHookFunction()** for each allocated digitizer. Each time a different digitizer is specified, it retains all settings previously specified.
  3. Call **MdigGrab()** with any digitizer identifier. If this call uses a digitizer identifier different from the previous call, a DCF switch will occur.
- ❖ If there is a grab in progress on one digitizer, calling any of the following functions with any other digitizer will result in an error: **MdigGrab()**, **MdigGrabContinuous()**, **MdigInquire()**, **MdigAlloc()**, and **MdigFree()**.

See the example *mdigmultformat.c*.

## Particularities of existing MIL functions on Matrox Cronos

Certain commands have special features or functionality on a Cronos system. This table provides an overview of the affected commands. Details and procedures are described in the individual command descriptions, which follow.

Commands	Matrox Cronos particularities
MbufAlloc...()	Buffer constraints and options.
MbufCreate...()	Required parameter settings.
MbufInquire()	Cronos-specific inquire options.
MdigAlloc()	Digitizer configuration format (DCF) specifications.
MdigChannel()	Required parameter settings.
MdigControl()	Control type and value additions and restrictions.
MdigGrab/MdigGrabContinuous()	Destination buffer restriction.
MdigHookFunction()	Hook restrictions.
MdigInquire()	Cronos-specific inquire options.
MdigReference()	Cronos features.
MsysAlloc()	Cronos-specific allocation options.
MsysControl()	Control type and value additions.
MsysHookFunction()	Cronos-specific system hook function.
MsysInquire()	Cronos-specific inquire options.

MbufAlloc...()

- Matrox Cronos supports scatter-gather DMA transfers to paged memory. By default, grab buffers are grabbed into non-paged memory, but this default can be overridden by adding M\_PAGED to the buffer attribute.
- Only 8-bit monochrome buffers, and 3-band 8-bit color buffers in the following formats, can have an M\_GRAB attribute:

Attribute parameter for 3-band 8-bit buffer	Internal storage format for 3-band 8-bit buffer
M_BGR32 + M_PACKED	32-bit (BGR) packed pixels (default).
M_YUV16_YUYV + M_PACKED	YUV16 (4:2:2) packed standard.
M_BGR24 + M_PACKED	24-bit (BGR) packed pixels.
M_RGB16 + M_PACKED	16-bit (RGB) packed pixels.
M_RGB15 + M_PACKED	16-bit (RGB) packed pixels.

MbufCreate...()

- Set the **Attribute** parameter to an M\_IMAGE combination (for example, M\_IMAGE + M\_DISP + M\_GRAB + M\_PROC).
- Only the following **ControlFlag** combinations are supported:

M_PHYSICAL_ADDRESS + M_PITCH	<b>ArrayOfDataPtr</b> is an array of physical addresses. The pitch is in pixels (default).
M_PHYSICAL_ADDRESS + M_PITCH_BYTE	<b>ArrayOfDataPtr</b> is an array of physical addresses. The pitch is in bytes.
M_HOST_ADDRESS + M_PITCH	<b>ArrayOfDataPtr</b> is an array of host addresses. The pitch is in pixels (default).
M_HOST_ADDRESS + M_PITCH_BYTE	<b>ArrayOfDataPtr</b> is an array of host addresses. The pitch is in bytes.

- The **Attribute** parameter can be set to the same settings as *MbufAlloc...()*.



## MdigAlloc()

- The **DataFormat** parameter specifies the DCF for the input device.

The predefined settings for monochrome cameras are:

"M_RS170"	RS-170, 640x480, 8 bits, 12.27MHz, analog.
"M_CCIR"	CCIR, 768x576, 8 bits, 14.75MHz, analog.
"M_DEFAULT"	Same as M_RS170.

The predefined settings for color cameras are:

"M_NTSC"	NTSC, 640x480, 3x8 bits, 12.27MHz, composite
"M_NTSC_YC"	RS-170 Y/C(SVHS), 640x480, 3x8 bits, 12.27MHz
"M_PAL"	PAL, 768x576, 3x8 bits, 14.75MHz, composite
"M_PAL_YC"	PAL Y/C, 768x576, 3x8 bits, 14.75 MHz

You can also set the **DataFormat** parameter to the name of a DCF file (\*.dcf). This file specifies the input signal format. See the `\MATOX IMAGING\DRIVERS\CRONOS` (or user-specified) directory for the current list of supported formats.

- You can allocate multiple digitizers on one Cronos system. Make as many calls to **MdigAlloc()** as you have different cameras from which you want to grab. With each call, specify the DCF for the camera, and set the **DigNum** parameter to M\_DEV0. Calling **MdigGrab()** with any digitizer identifier will cause a switch and a grab from the identifier's associated camera. See the section, *Grabbing from multiple cameras with different DCFs* for more information.

## MdigChannel()

The channels available for acquisition depend on whether you have input modules attached to your Matrox Cronos base board.

- When using the Matrox Cronos base board, you can attach and switch between 4 monochrome cameras (CCIR/RS-170 format), or 4 composite color cameras (NTSC/PAL format).
- Additional input modules give you more channels for grabbing, but those channels are determined by the connector to which the input module is attached.
- To switch between cameras of a similar type, use:

Input type	Channel	Board	Connector
Monochrome or composite	M_CH0	Base board	BNC 0
	M_CH1		BNC 1
	M_CH2		BNC 2
	M_CH3		BNC 3
	M_CH4	Composite Video Input module on connector J5	BNC 0
	M_CH5		BNC 1
	M_CH6		BNC 2
	M_CH7		BNC 3
	M_CH8	Composite Video Input module on connector J6	BNC 0
	M_CH9		BNC 1
	M_CH10		BNC 2
	M_CH11		BNC 3
	M_CH12	Composite Video Input module on connector J7	BNC 0
	M_CH13		BNC 1
	M_CH14		BNC 2
	M_CH15		BNC 3

Input type	Channel	Board	Connector
Y/C	M_CH0	Y/C Video Input module on connector J7	DIN 0
	M_CH1		DIN 1
	M_CH2		DIN 2
	M_CH3		DIN 3
Refer to the <i>Cronos Installation and Hardware Reference</i> manual for more information about these connectors and channels they make available.			

- ❖ To perform fast channel-switching use the **MdigControl()** control type M\_CAMERA\_LOCK.

### MdigControl()

- M\_GRAB\_SCALE, M\_GRAB\_SCALE\_X, and M\_GRAB\_SCALE\_Y can be set to any value between 1 and 1/8 in both frame and field modes since Matrox Cronos supports arbitrary scaling.
  - When M\_GRAB\_SCALE\_Y is set to a value with an integer in the denominator (that is, 1, 1/2, 1/3, 1/4, ...1/8), there is no interpolation. For any other value, bilinear interpolation is used.
  - The values M\_FILL\_DESTINATION and M\_FILL\_DISPLAY are also supported for aforementioned control types.
- The default setting for the M\_GRAB\_START\_MODE control type is M\_FIELD\_START\_ODD.
- The M\_GRAB\_DIRECTION\_X control type is not supported. The M\_GRAB\_DIRECTION\_Y control type supports the following control values.

Control value	Description
M_REVERSE	Flip the grabbed image vertically.
M_FORWARD	Grab normally in the vertical direction.
M_DEFAULT	Same as M_FORWARD.

- The M\_GRAB\_EXPOSURE\_XX control types are not supported on Matrox Cronos.
- Matrox Cronos supports these additional **ControlType** parameter settings:

ControlType	Description & ControlValue	
M_GRAB_ABORT	Immediately stops the grab in progress and cancels all queued grabs. Useful for cancelling a triggered grab that is waiting for the trigger.	
	M_DEFAULT	Stops the grab.
M_GRAB_AUTOMATIC_INPUT_GAIN	Sets the input gain automatically when grabbing.	
	M_ENABLE	Enable automatic input gain.
	M_DISABLE	Disable automatic input gain.
	M_DEFAULT	Same as M_ENABLE.
M_GRAB_FAIL_CHECK	Sets whether and when to log a grab-fail error.	
	M_ENABLE	Log all grab failures.
	M_DISABLE	Log no grab failures.
	M_FINAL_GRAB	Log monoshot grab failures or that of the last frame of a continuous grab.
	M_DEFAULT	Same as M_FINAL_GRAB.
M_GRAB_FAIL_RETRY_NUMBER	Sets the number of retries when a field or frame grab fails. This control type can be set to the required number (default is 1).	

ControlType	Description & ControlValue	
M_CAMERA_LOCK	Controls the camera lock mechanism, which is useful during channel-switching when the digitizer might not be synchronized with the source.	
	M_ENABLE	Causes <b>MdigGrab()</b> to wait until the digitizer is locked with the video source before starting the grab.
	M_DISABLE	Causes <b>MdigGrab()</b> to grab without verifying whether the digitizer is locked with the video source. Corrupt images or a synchronization-lost error can occur if the digitizer is not locked.
	M_ENABLE+M_FAST	Enables fast channel-switching. Note that this control value is only supported on Matrox Cronos boards that are Rev. 1 or higher. See the section, <i>Fast-channel switching</i> , for the restrictions of this control.
	M_DEFAULT	Same as M_ENABLE.
M_CAMERA_LOCK_SENSITIVITY	Controls the line-lock sensitivity when M_CAMERA_LOCK is enabled. Valid values are from 0 to 255, where 0 is the least reliable but fastest lock speed. M_DEFAULT is 16.	
M_CAMERA_PRESENT_SENSITIVITY		
	Controls the number of vsync signals the digitizer will wait, after a channel switch, before generating a "Signal not present" error. Valid values are between 0 and 255. M_DEFAULT is 10, and a value of 0 means the digitizer will not check for a camera before grabbing. This control type is only available when the control type M_CAMERA_LOCK is set to M_ENABLE+M_FAST.	
M_GRAB_TRIGGER_SOURCE	Sets the source of the grab trigger.	
	M_HARDWARE_PORT0	
		BNC3 on the base board with jumper J12 set to pins 2-3, and Pin 22 on the I/O module connector.
	M_DEFAULT	The same as the DCF file.
	M_SOFTWARE	Uses software trigger.
	M_HARDWARE_PORT_CAMERA	
		Same as M_HARDWARE_PORT0.

ControlType	Description & ControlValue	
M_GRAB_WINDOW_RANGE	Limit the range of the pixels grabbed in a monochrome buffer to between 16 and 253.	
The following control types are only available when the I/O module is attached to the Matrox Cronos base board.		
M_UART_DATA_LENGTH	The number of data bits per character that are sent or received by the UART. Valid values are 7 or 8 bits.	
	7	Data length is 7 bits.
	8	Data length is 8 bits.
	M_DEFAULT	Data length is 8 bits.
M_UART_OUTPUT	Specify the UART output datalink.	
	M_RS232	Use the RS-232 serial data communication port.
	M_RS485	Use the RS-485 serial data communication port.
	M_DEFAULT	Same as M_RS232.
M_UART_PARITY	Add a data bit (0 or 1) to the character data that is sent or received by the UART as a means of error checking.	
	M_DEFAULT	Same as M_DISABLE.
	M_ODD	The number of 1's will be odd.
	M_EVEN	The number of 1's will be even.
	M_DISABLE	No extra bit is added (no parity).
M_UART_READ_CHAR	Read 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. ControlValue must be a pointer to a char.	
M_UART_READ_STRING	Read a string of incoming data from the UART. The <b>ControlValue</b> must be a pointer to a character array. The size of this array must be set with the M_UART_READ_STRING_MAXIMUM_LENGTH control type. 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.	

ControlType	Description & ControlValue	
M_UART_READ_STRING_LENGTH	Sets the length of the string (in bytes) to be read by M_UART_READ_STRING.	
	M_DEFAULT	Used to specify the use of M_UART_STRING_DELIMITER to end string.
	value in bytes	Specify string length.
M_UART_READ_STRING_MAXIMUM_LENGTH	Use this value to specify the size of your read buffer to prevent global protection faults from happening when using the M_UART_READ_STRING control.	
M_UART_SPEED	Change the baud rate of the UART. Valid values are 230400, 115200, 76800, 57600, 38400, 28800, 19200, 14400, 9600, 7200, 4800, 3600, 2400, 1800, 1200, 600. M_DEFAULT is 14400.	
M_UART_STOP_BITS	Add a data bit to signal the end of character data being sent or received.	
	1	1 stop bit will be added.
	2	2 stop bits will be added.
	M_DEFAULT	1 stop bit will be added.
M_UART_STRING_DELIMITER	Sets the character used to terminate strings of incoming or outgoing data. The delimiter is used but not sent when writing data; it is read for incoming data.	
	M_DEFAULT	The '\0' character.
M_UART_TIMEOUT	Sets the maximum time to wait between each byte when reading incoming data.	
	M_INFINITE	Wait indefinitely.
	M_DEFAULT	Same as M_INFINITE.
	value in msec	Specify time to wait.
M_UART_WRITE_CHAR	Send one character to the UART. Set <b>ControlValue</b> as a pointer to a character.	
M_UART_WRITE_STRING	Send the string of data, specified with the control value, through the UART. Set <b>ControlValue</b> to the character array. The number of characters to send can be specified with M_UART_WRITE_STRING_LENGTH or M_UART_STRING_DELIMITER.	

ControlType	Description & ControlValue	
M_UART_WRITE_STRING_LENGTH	Sets the length of the string to be sent to the UART for transmission.	
	M_DEFAULT	Used to specify the use of M_UART_STRING_DELIMITER to end string. The delimiter will not be sent through the UART.
	value in bytes	Specify string length.

### MdigGrab()/MdigGrabContinuous()

- It is not possible to grab into a color-band child buffer.
- When acquiring data on Matrox Cronos, the range of input values depends on the buffer into which you are grabbing:
  - When grabbing into a monochrome or RGB buffer, the range of possible input values spans from 0 to 255.
  - When grabbing into a YUV buffer, the range of possible input values spans from 0 to 255 for the Y component, and from 2 to 253 for both the U and V components.
- Matrox Cronos uses an internal FIFO to buffer image data while waiting to access the PCI bus; therefore image data could be lost due to long bus-access latencies found in heavily loaded systems. When the PCI bus is extremely loaded, Matrox Cronos might not be able to empty the FIFO when required. This problem causes an acquisition error known as a "Data transfer error (FIFO overflow)".

A way to reduce the occurrence of this error is to grab into a M\_YUV16 packed buffer instead of a M\_BGR24 or M\_BGR32 buffer. In addition using an AGP graphics controller for display instead of a PCI graphics controller will also reduce the PCI load. Another method is to increase the value of the M\_GRAB\_FAIL\_RETRY\_NUMBER control type; when an error occurs, the driver will restart the grab as many times as specified by this control type.



- When performing real-time grabs, Windows Me does not provide the same performance and consistency as Windows 2000/XP. This generally does not cause difficulty except in applications that require very fast response time to a hardware event. For example, an application that must grab sequences of images without missing any frames might not have sufficient time for the proper response to a hook function. Therefore, if grabbing a sequence, you should try to avoid code that needs an immediate response and favor code that supports a short delay after the event. For example, it is recommended to call the next grab operation (**MdigGrab()**) from a function hooked to the start of the current grab, instead of calling it from a function hooked to the end of the grab. This will queue the next grab operation while the current grab is in progress, and will allow the next grab to start immediately after the current grab. This should prevent the loss of a frame due to delay.

If these restrictions affect a major part of your application(s), we recommend the use of Windows 2000/XP in addition to the above recommendations.

- Due to a limitation on Matrox Cronos, grabbing into a color grab buffer, with a color camera and a monochrome DCF will result in a color image, not monochrome.

### **MdigHookFunction()**

- A function hooked to an M\_GRAB\_END event is called when all the data of the grab is transferred to the Host. This means that this hooked function can be called after the M\_GRAB\_START or M\_GRAB\_FRAME\_START event of the next frame.

MdiglInquire()

- The **InquireType** parameter can also be set to the following:

InquireType	Description
M_BRIGHTNESS_REF	Brightness reference level.
M_CONTRAST_REF	Contrast reference level.
M_HUE_REF*	Hue reference level.
M_SATURATION_REF*	Saturation reference level.
*Note that the inquire types M_HUE_REF and M_SATURATION_REF inquire types are not supported for monochrome cameras.	
M_CAMERA_LOCK	Inquires the state of the camera lock mechanism: M_ENABLE, M_DISABLE, or M_ENABLE+M_FAST.
M_CAMERA_LOCKED	Inquires whether a horizontal lock is achieved: M_YES or M_NO.
M_CAMERA_PRESENT	Inquires whether a camera is present: M_YES or M_NO.
M_COLOR_MODE	Color mode: M_EXTERNAL_CHROMINANCE, M_COMPOSITE, or M_MONOCHROME.
M_GRAB_AUTOMATIC_INPUT_GAIN	Mode of automatic input gain: M_ENABLE or M_DISABLE.
M_GRAB_FAIL_CHECK	The state of the grab-fail check control: M_FINAL_GRAB, M_ENABLE, or M_DISABLE.
M_GRAB_FAIL_RETRY_NUMBER	The number of retries to attempt upon a grab failure.
M_GRAB_FAIL_STATUS	The status of the fail check on the last grab: M_YES if failed, or M_NO.
M_GRAB_IN_PROGRESS	Current grab state: M_YES or M_NO.
M_GRAB_SCALE_X	Horizontal scaling factor.
M_GRAB_SCALE_Y	Vertical scaling factor.

InquireType	Description
M_GRAB_WINDOW_RANGE	Inquires whether the range of the pixels grabbed is limited: M_ENABLE or M_DISABLE.
M_HOOK_MASTER_THREAD_HANDLE	Returns the handle of the main interrupt thread that dispatches the handling of the grab hooks to their respective threads.
M_HOOK_MASTER_THREAD_ID	Returns the ID of the main interrupt thread that dispatches the handling of the grab hooks to their respective threads.
M_UART_DATA_LENGTH	Returns the number of data bits per character: 7 or 8 (default).
M_UART_DATA_PENDING	Indicates the input buffer has some pending data: M_TRUE, M_FALSE.
M_UART_OUTPUT	Returns the data communication port: M_RS232 or M_RS485.
M_UART_PARITY	Current UART parity setting: M_ODD, M_EVEN, M_DISABLE.
M_UART_READ_STRING_LENGTH	The length of the string, in bytes, to be read by M_UART_READ_STRING. Can be M_DEFAULT if the string length is specified by M_UART_STRING_DELIMITER.
M_UART_READ_STRING_MAXIMUM_LENGTH	Current maximum size of the read buffer.
M_UART_SPEED	Current baud rate in UART configuration: 230400, 115200, 76800, 57600, 38400, 28800, 19200, 14400, 9600, 7200, 4800, 3600, 2400, 1800, 1200, 600.
M_UART_STOP_BITS	Current number of stop bits in UART configuration: 1 or 2.
M_UART_STRING_DELIMITER	Current character used to delimit strings if M_UART_WRITE_STRING_LENGTH or M_UART_READ_STRING_LENGTH is set to M_DEFAULT.
M_UART_THREAD_HANDLE	Current UART thread handle.
M_UART_THREAD_ID	Current UART thread ID.

InquireType	Description
M_UART_TIMEOUT	Current maximum time, in msec, to wait between bytes of incoming data. Can be M_INFINITE or value in msec.
M_UART_WRITE_STRING_LENGTH	The length of the string to be sent to the UART for transmission. Can be M_DEFAULT if the string length is specified by M_UART_STRING_DELIMITER.

## MdigReference()

- When grabbing monochrome data, you can control the brightness and contrast levels of the input signal using M\_BRIGHTNESS\_REF and M\_CONTRAST\_REF. Valid values are between M\_MIN\_LEVEL to M\_MAX\_LEVEL.
- When grabbing color, you can control the brightness, contrast, hue, and saturation levels of the composite input signal, using M\_BRIGHTNESS\_REF, M\_CONTRAST\_REF, M\_HUE\_REF, and M\_SATURATION\_REF. Valid values are between M\_MIN\_LEVEL to M\_MAX\_LEVEL.

## MsysAlloc()

- To allocate a Cronos system, you must select M\_SYSTEM\_CRONOS as the **SystemType** parameter. This selection opens communication with the board and will automatically allow the system to use some Host memory, and any available graphics controller, if necessary.

## MsysControl()

To control a specific user bit or a group of user bits on Matrox Cronos, set the **ControlType** and **ControlValue** combination according to the following table. To affect a single user bit, add its user bit number to the control type. To affect a group of user bits, add M\_BIT\_MASK to the control type and identify which bits to affect in a bit-encoded mask value. Enabled bits in the mask value identify which user bit to affect. For example, M\_USER\_BIT\_VALUE + M\_BIT\_MASK(0x4f) affects

user bits 0,1,2,3, and 6. If a group of user bits is specified, pass a bit-encoded control value that specifies the setting for each of these user bits.

ControlType	ControlValue	Description
M_USER_BIT_VALUE + (0-10) or + M_BIT_MASK (bit-encoded mask value)	Sets the functional state of the system's specified user bit(s). The specified bit(s) must be in output mode.	
	M_OFF or <b>ControlValue</b> bit set to 0.	Sets the user bit to 0. (default)
	M_ON or <b>ControlValue</b> bit set to 1.	Sets the user bit to 1.
M_USER_BIT_MODE + (0-10) or + M_BIT_MASK (bit-encoded mask value)	Sets the system's specified user bit(s) input/output mode.	
	M_INPUT or <b>ControlValue</b> bit set to 0.	Sets the user bit to input. (default)
	M_OUTPUT or <b>ControlValue</b> bit set to 1.	Sets the user bit to output.
M_USER_BIT_INTERRUPT_STATE	Sets the system's user bit interrupt state. The user bit interrupt uses the same input pin as the trigger. The trigger can use either BNC 3 on the base board if jumper J12 is set to pins 2 and 3, or the twelfth I/O line on the I/O module. Note these two inputs are wired together; see the <i>Matrox Cronos Installation and Hardware Reference</i> manual for details.	
	M_ENABLE.	Enable interrupt.
	M_DISABLE.	Disable interrupt. (default)
M_USER_BIT_INTERRUPT_MODE	Sets the system's user bit interrupt mode. This control type specifies whether the interrupt will be generated due to an off-to-on (rising edge) or on-to-off (falling edge) change in functional state of the user bit. The user bit interrupt uses the same interrupt pin as the grab trigger, and must be compatible with the <b>MdigControl()</b> M_GRAB_TRIGGER_MODE control type. For example, when M_USER_BIT_INTERRUPT_MODE is set to M_EDGE_RISING, M_GRAB_TRIGGER_MODE can be set to M_EDGE_RISING or M_LEVEL_HIGH.	
	M_EDGE_RISING.	Sets interrupt mode to rising edge mode (default).
	M_EDGE_FALLING.	Sets interrupt mode to falling edge mode.

- When a live grab operation uses a software trigger and **MdigHalt()** is issued, a software trigger is automatically generated to invoke a last grab (if you did not disable `M_LAST_GRAB_IN_TRUE_BUFFER`). You can disable this automatic trigger; however, you would then have to issue a software trigger call after the **MdigHalt()** call (these calls must be issued from different threads). To enable the trigger for the last grab, set the **ControlType** `M_LIVE_GRAB_END_TRIGGER` to `M_ENABLE`.

Similarly, when a live grab operation uses a hardware trigger, **MdigHalt()** will wait indefinitely for a hardware trigger to invoke a last grab (if you did not disable `M_LAST_GRAB_IN_TRUE_BUFFER`). You can override this default so that **MdigHalt()** will cause an immediate last grab and will not wait for a hardware trigger. To do so, set the following **ControlType** below to `M_DISABLE`.

ControlType	ControlValue	
M_LIVE_GRAB_END_TRIGGER	M_ENABLE	Default for hardware triggered cameras.
	M_DISABLE	Default for software-triggered cameras.

**MsysHookFunction()**

- The **HookType** parameter can be set to one of the values below. For more information, see *MsysHookFunction* for Matrox 4Sight-II.

HookType	Description
M_USER_BIT_CHANGE	Calls the function when an input user bit changes in accordance with its specified interrupt mode, <code>M_EDGE_RISING</code> or <code>M_EDGE_FALLING</code> .
M_UNHOOK + M_USER_BIT_CHANGE	Unhooks the specified function if hooked to an <code>M_USER_BIT_CHANGE</code> event.

## MsysInquire()

- You can request the following information with the **InquireType** parameter:

InquireType	Description	
M_BOARD_REVISION	The board revision (long value).	
M_BOARD_TYPE	The type of system board: M_CRONOS.	
M_LIVE_GRAB_END_TRIGGER	Generate automatic trigger at end of grab: M_ENABLE or M_DISABLE.	
M_USER_BIT_COUNT	The number of user bits available on the digital I/O and serial port connector: 11.	
For the following inquire types, you should consider the following: ■ To inquire about a single user bit, add the user bit number to the inquire type. ■ To inquire about a group of user bits, add M_BIT_MASK to the inquire type and identify which bits to inquire about in a bit-encoded mask value. In this case, a bit-encoded inquire value is returned to the <b>UserVarPtr</b> , which must be a pointer to a long.		
M_USER_BIT_VALUE + (0-10) or + M_BIT_MASK (bit-encoded mask value)	Read the functional state of the system’s user bit.	
	M_OFF or bit value is 0.	The user bit is set to 0.
	M_ON or bit value is 1.	The user bit is set to 1.
M_USER_BIT_MODE + (0-10) or + M_BIT_MASK (bit-encoded mask value)	Read the system’s specified user bit(s) mode.	
	M_INPUT or bit value is 0.	The user bit is in input mode.
	M_OUTPUT or bit value is 1.	The user bit is in output mode.
M_USER_BIT_INTERRUPT_STATE	Read the interrupt state of the system’s user bit.	
	M_ENABLE.	Interrupt is enabled.
	M_DISABLE.	Interrupt is disabled.

InquireType	Description	
M_USER_BIT_INTERRUPT_MODE	Read the interrupt mode for the system's user bit.	
	M_EDGE_RISING.	Interrupt mode is in rising edge mode.
	M_EDGE_FALLING.	Interrupt mode is in falling edge mode.



---

## ***Chapter 8: MIL and the VGA platform***

*This chapter describes the distinctive MIL functionality under a VGA platform.*

## Particularities of existing MIL functions on the VGA system

Certain commands can have special features or functionality on the VGA system. This table provides an overview of the affected commands. Details and procedures are described in the individual command descriptions, which follow.

Commands	VGA particularities
Mdig...()	Not available.
MdispAlloc()	Auxiliary display.
MdispInquire()	M_WINDOW_BUF_ID

### Mdig...()

These functions are not available since there is no digitizer on a VGA system.

**MdispAlloc()** The display format parameter of **MdispAlloc()** specifies the video output format for auxiliary displays (M\_AUXILIARY). If you are using a Matrox display card, the supported display formats can be found in the *MATROX IMAGING\DRIVERS\VGA\VCF* directory. For auxiliary displays, the default ("M\_DEFAULT") display format is 1024 x 768 x 32bpp at 60Hz.

If you are using the second CRT output provided with the Matrox Millennium G400, G450, or G550 for your auxiliary screen, your display driver's DualHead mode must be disabled; otherwise, both the display driver and the MIL driver will attempt to access the second CRT output. In addition, the G400's second CRT output does not support encoded video formats, but the G450 and G550 do.

**MdispInquire()** On graphics controllers that do not have non-destructive overlay capabilities, the M\_WINDOW\_BUF\_ID inquire type returns 0. In this case, M\_WINDOW\_OVR\_BUF\_ID should be used instead.

---

## Additional method of displaying an image in a user-defined window

Under Windows, an additional method exists for image display in a customized (user-defined) window. Custom windows can be created and images can be displayed in them using the Windows API functions (for example, **DrawState()**).

---

### *Using the Windows API functions*

Using the Windows API functions for displaying images in a user-defined window is for advanced users that need very fine control over the display behavior. This method gives you the flexibility of choosing where and how to display an image (for example, in a custom window, a child window, or as an icon), and when to update the display. For example, the Windows' function, **UpdateWindow()**, updates the client area of a window.

The Windows API functions need to access the DIB (Device Independent Bitmap) of an image to display it. To access the DIB of an image, use **MdisInquire()**. Alternatively, allocate a special information structure, **M\_DIB\_INFO**, and associate it with the DIB using **MvgaDispAllocDIBInfo()**. You can later use **MvgaDispFreeDIBInfo()** to free it.

The **M\_DIB\_INFO** structure contains DIB information, such as:

- The pointer to the DIB header, which describes the data format.
- The color table usage of the DIB header.
- The pointer to the DIB data itself.
- The pointer to the display palette to visualize the DIB (optional usage).

The actual fields of this structure are described later in this chapter.

---

### *Window is not redrawn*

When using the Windows API functions, an image is not redrawn in its window automatically. You are responsible for updating the display window when your application receives a Windows refresh message (**WM\_PAINT**) or when your image data is modified.

You can create your own function (using the Windows API functions) to handle the redrawing of an image in its window. This function can be hooked to DIB modifications with the **MvgaHookModified()** function. Once the function is hooked, it is called each time the image is modified by a MIL function.

See the *mwindib.c* example provided in the *MATROX IMAGING\MIL\EXAMPLES* directory.

---

## VGA board-specific functions

The following describes the MIL VGA system-specific function. This function is an extension to the regular MIL function set. You must include the *windows.h* file before using this function.

This function has not been replaced by regular functions in the **Mdisp** module. It is currently maintained for compatibility purposes only and might not be included in a future release.

MIL VGA Commands	Command Parameters	Description
MvgaHookModified()	MilVgaDisplayId, HookType, HookHandlerPtr, UserDataPtr	Hook a user-specified function to a MIL VGA display event.

## MvgaHookModified

**Synopsis** Hook a user-specified function to a MIL VGA display event.

**Format** **MVGAHOOKFCTPTR MvgaHookModified(MilVgaDisplayId, HookType, HookHandlerPtr, UserDataPtr)**

long MilVgaDisplayId;	MIL VGA display identifier
long HookType;	Type of event to hook
MVGAHOOKFCTPTR *HookHandlerPtr;	Address of function to call when a change occurs
void *UserDataPtr;	User-defined data pointer

**Description** This function allows you to attach or detach a user-defined function to a specified MIL VGA display event. Once a hook handler function is defined and hooked to an event, the function 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.

The type of events that can be hooked to a user-defined function are modifications to:

- The specified display (for example, using **MdispZoom()**).
- The image buffer associated with the specified display (for example, using **MbufPut()**).
- The window associated with the specified display. Such modifications include any size or position change of the window, button selection from the system menu, scroll bar usage, or zoom button usage.

The first two types of events are considered DIB events since they affect the DIB of an image buffer.

The **MilVgaDisplayId** parameter specifies the MIL VGA display identifier.

The **HookType** parameter specifies the event type. This parameter can be one of the following:

M_HOOK_MODIFIED_DIB	Call the hook handler function when the image buffer's DIB changes.
M_HOOK_MODIFIED_WINDOW + M_HOOK_BEFORE	Call the hook handler function just before the window changes.
M_HOOK_MODIFIED_WINDOW + M_HOOK_AFTER	Call the hook handler function just after the window changes.
M_UNHOOK + M_HOOK_MODIFIED_DIB	Detach the hook handler function being called when the image buffer's DIB changes.
M_UNHOOK + M_HOOK_MODIFIED_WINDOW + M_HOOK_BEFORE	Detach the hook handler function being called just before the window changes.
M_UNHOOK + M_HOOK_MODIFIED_WINDOW + M_HOOK_AFTER	Detach the hook handler function being called just after the window changes.

The **HookHandlerPtr** parameter specifies the address of the function that should be called when an event occurs (that is, the hook handler function). The following pages describe how to declare the function and what information is passed to it. Declare its parameters, using the appropriate type for the event being hooked (DIB or window modification). Upon successful completion, the hook-handler function should return M\_NULL.

When hooking to a **window modification**, the hook-handler function, pointed to by **HookHandlerPtr**, must be declared as follows:

long MFTYPE HookHandlerPtr(ModifiedDispWindowPtr, UserDataPtr)	
M_WINDOW_INFO MPTYPE *ModifiedDispWindowPtr;	Address of the information structure associated with the modified display window.
void MPTYPE *UserDataPtr;	User data pointer

When a window-modification event occurs, the M\_WINDOW\_INFO structure is passed to your hook handler function. The information structure has the following format:

typedef struct			
{			
HWND	WindowHandle;	/* Display's window handle	*/
LPBITMAPINFO	DIBHeaderPtr;	/* Pointer to the DIB header	*/
UINT	DIBColorUse;	/* Color usage of the DIB header	*/
LPSTR	DIBDataPtr;	/* Pointer to the DIB data	*/
LPLOGPALETTE	DIBPalettePtr;	/* Pointer to the DIB palette	*/
HPALETTE	DIBPaletteHdl;	/* Handle of the DIB palette	*/
HBITMAP	DIBBitmapHdl;	/* Handle of the DIB bitmap	*/
LPBITMAPINFO	DIBDisplayHeaderPtr;	/* Pointer to the display DIB header	*/
UINT	DIBDisplayColorUse;	/* Color usage of the display DIB header	*/
LPSTR	DIBDisplayDataPtr;	/* Pointer to the display DIB data	*/
LPLOGPALETTE	DIBDisplayPalettePtr;	/* Pointer to the display DIB palette	*/
HPALETTE	DIBDisplayPaletteHdl;	/* Handle of the display DIB palette	*/
HBITMAP	DIBDisplayBitmapHdl;	/* Handle of the display DIB bitmap	*/
MIL_ID	MilVgaBufferId;	/* MIL Id of the MIL VGA buffer	*/
MIL_ID	MilVgaDisplayId;	/* MIL Id of the MIL VGA display	*/
long	VgaDriverBufferId;	/* Reserved	*/
long	VgaDriverDisplayId;	/* Reserved	*/
long	VgaDriverSystemId;	/* Reserved	*/
long	ModificationHookType;	/* Type of hook on modification	*/
long	ModificationType;	/* Type of modification to window	*/
long	ModificationOffsetX;	/* Offset X of the window modification	*/
long	ModificationOffsetY;	/* Offset Y of the window modification	*/
long	ModificationSizeX;	/* Width of the window modification	*/
long	ModificationSizeY;	/* Height of the window modification	*/
long	ModificationZoomX;	/* X zoom factor of window modification	*/
long	ModificationZoomY;	/* Y zoom factor of window modification	*/
long	ModificationPanX;	/* Horizontal scroll of window	*/
		/* modification	*/
long	ModificationPanY;	/* Vertical scroll of window modification	*/

long	ModificationMenu;	/* Menu being used	*/
long	ModificationValue;	/* Optional value for modification	*/
long	VisibleOffsetX;	/* X offset of the window's visible display	*/
		/* area	*/
long	VisibleOffsetY;	/* Y offset of the window's visible display	*/
		/* area	*/
long	VisibleSizeX;	/* Width of the window's visible display	*/
		/* area	*/
long	VisibleSizeY;	/* Height of the window's visible display	*/
		/* area	*/
LPVOID	ExpansionPtr;	/* Reserved	*/
long	ExpansionFlag;	/* Reserved	*/
long	ReservedValue1;	/* Reserved	*/
long	ReservedValue2;	/* Reserved	*/
long	ReservedValue3;	/* Reserved	*/
long	ReservedValue4;	/* Reserved	*/
} M_WINDOW_INFO;		/* Reserved	*/

Your hook-handler function can read its modification fields to identify the type of change that has occurred.

- The M\_WINDOW\_INFO **ModificationHookType** field indicates the type of event that is hooked. This field could have been passed with one of the following values:

M_HOOK_MODIFIED_WINDOW + M_HOOK_BEFORE
M_HOOK_MODIFIED_WINDOW + M_HOOK_AFTER

- The M\_WINDOW\_INFO **ModificationType** field indicates the type of window modification. This field could have been passed with one of the following values:

M_MODIFIED_WINDOW_CREATION	Window has been created.
M_MODIFIED_WINDOW_DESTRUCTION	Window has been destroyed.
M_MODIFIED_WINDOW_LOCATION	Window has been moved or sized.
M_MODIFIED_WINDOW_ICONIZED	Window has been iconized (minimized).
M_MODIFIED_WINDOW_ZOOM	Window has been zoomed using the menu buttons.



M_MODIFIED_WINDOW_PAN	Window has been panned using the scroll bars.
M_MODIFIED_WINDOW_MENU	Window menu has been selected.
M_MODIFIED_WINDOW_PAINT	Window has been repainted with the buffer image.
M_MODIFIED_WINDOW_ACTIVE	Window has been activated or deactivated.
M_MODIFIED_WINDOW_ENABLE	The window enable state has been changed.

Note, when menu or title bars are changed, the window location is changed, but the hook function is not called.

- If an M\_MODIFIED\_WINDOW\_ACTIVE modification type has occurred, the M\_WINDOW\_INFO **ModificationValue** field indicates the activation state of the window. This field could have been passed with the following values:

M_MODIFIED_ON	The window is activated.
M_MODIFIED_OFF	The window is deactivated.
M_MODIFIED_STATE_FROM_WINDOW	Activation or deactivation of the window is controlled from the display window itself.
M_MODIFIED_STATE_FROM_PARENT	Activation or deactivation of the window is controlled from the application window that is a parent of the display window. (used in MDI applications)

- If an M\_MODIFIED\_WINDOW\_ENABLE modification type has occurred, the M\_WINDOW\_INFO **ModificationValue** field indicates the enable state of the window. These values can be combined:

M_MODIFIED_ON	The window is enabled.
M_MODIFIED_OFF	The window is disabled.
M_MODIFIED_STATE_FROM_WINDOW	The enablement or disablement of the window is controlled from the display window itself.
M_MODIFIED_STATE_FROM_PARENT	The enablement or disablement of the window is controlled from the application window that is a parent of the display window. (used in MDI applications)

- If an M\_MODIFIED\_WINDOW\_MENU modification type has occurred, the M\_WINDOW\_INFO **ModificationMenu** field indicates the menu that has been used. This field could have been passed with one of the following values:

M_MODIFIED_SYS_MENU	The window's system menu has been selected.
M_MODIFIED_APP_MENU	The window's application menu has been selected.

The **ModificationMenu** field can also indicate the selected menu item. When an M\_HOOK\_AFTER event is hooked, the M\_MODIFIED\_SYS\_MENU comes combined with one of the following (for example, M\_MODIFIED\_SYS\_MENU + M\_MODIFIED\_MOVE\_MENUITEM):

M_MODIFIED_RESTORE_MENUITEM	Restore menu item.
M_MODIFIED_MOVE_MENUITEM	Move menu item.
M_MODIFIED_SIZE_MENUITEM	Size menu item.
M_MODIFIED_MINIMIZE_MENUITEM	Minimize menu item.
M_MODIFIED_MAXIMIZE_MENUITEM	Maximize menu item.
M_MODIFIED_CLOSE_MENUITEM	Close menu item.
M_MODIFIED_TASKLIST_MENUITEM	Switch to menu item.
M_MODIFIED_MENUBAR_MENUITEM	Menu bar menu item.
M_MODIFIED_TITLEOFF_MENUITEM	Title off menu item.

Likewise, when a M\_HOOK\_BEFORE or M\_HOOK\_AFTER event is hooked, M\_MODIFIED\_APP\_MENU comes combined with one of the following:

M_MODIFIED_ZOOMIN_MENUITEM	Zoom in menu item.
M_MODIFIED_ZOOMOUT_MENUITEM	Zoom out menu item.
M_MODIFIED_NOZOOM_MENUITEM	Zoom 1 menu item.

- In general, DIBXXXXPtr and DIBDisplayXXXXPtr are the same. However, when 24-bit RGB image buffers are displayed with an 8-bit VGA Windows driver, DIBDisplayXXXXPtr might point to 8-bit precalculated 3:3:2 versions of DIBXXXXPtr 24-bit buffers. These 8-bit versions are created for a faster display of color images.

When hooking to a **DIB modification**, the hook-handler function, pointed to by **HookHandlerPtr**, must be declared as follows:

```
long MFTYPE HookHandlerPtr(ModifiedVgaBufferPtr, UserDataPtr)

M_DIB_INFO MPTYPE *ModifiedVgaBufferPtr;    Address of the DIB structure
                                              associated with the modified
                                              display image buffer.

void MPTYPE *UserDataPtr;                    User data pointer
```

When a DIB modification event occurs, the M\_DIB\_INFO structure is passed to your hook-handler function. The information structure has the following format:

```
typedef struct
{
LPBITMAPINFO    DIBHeaderPtr;           /* Pointer to the DIB header.      */
UINT            DIBColorUse;            /* Color usage of DIB header.      */
LPSTR           DIBDataPtr;            /* Pointer to the DIB data.        */
LPLOGPALETTE    DIBPalettePtr;         /* Pointer to the DIB palette.*    */
HPALETTE        DIBPaletteHdl;         /* Handle of the DIB palette.*    */
HBITMAP         DIBBitmapHdl;          /* Handle of the DIB bitmap.*     */
LPBITMAPINFO    DIBDisplayHeaderPtr;    /* Pointer to the display DIB header. */
UINT            DIBDisplayColorUse;     /* Color usage of display DIB header. */
LPSTR           DIBDisplayDataPtr;      /* Pointer to the display DIB data. */
LPLOGPALETTE    DIBDisplayPalettePtr;   /* Pointer to the display DIB palette. */
HPALETTE        DIBDisplayPaletteHdl;   /* Handle of the display DIB palette.* */
HBITMAP         DIBDisplayBitmapHdl;    /* Handle of the display DIB bitmap.* */
MIL_ID          MILVgaBufferId;         /* MIL ID of the MIL VGA buffer.    */
MIL_ID          MILVgaDisplayId;        /* MIL ID of the MIL VGA display.   */
long            VgaDriverBufferId;      /* Reserved.                        */
long            VgaDriverDisplayId;     /* Reserved.                        */
}
```

long	VgaDriverSystemId;	/* Reserved.	*/
long	ModificationHookType;	/*Type of hook on DIB modifications.	*/
long	ModificationType;	/* Type of modification to buffer.	*/
long	ModificationOffsetX;	/* Offset x of modification in buffer.	*/
long	ModificationOffsetY;	/* Offset y of modification in buffer.	*/
long	ModificationSizeX;	/* Size x of modification in buffer.	*/
long	ModificationSizeY;	/* Size y of modification in buffer.	*/
long	ModificationZoomX;	/* Zoom x of modification in buffer.	*/
long	ModificationZoomY;	/* Zoom y of modification in buffer.	*/
long	ModificationPanX;	/* Pan x of modification in buffer.	*/
long	ModificationPanY;	/* Pan y of modification in buffer.	*/
LPVOID	ExpansionPtr;	/* Reserved.	*/
} M_DIB_INFO;			

Your hook-handler function can read the structure's modification fields to identify the type of change that has occurred.

- The M\_DIB\_INFO **ModificationHookType** field indicates the type of event that is hooked. It should be set to M\_HOOK\_MODIFIED\_DIB.
- The M\_DIB\_INFO **ModificationType** field indicates the type of DIB modification. This field could have been passed with one of the following values:

M_MODIFIED_DIB_CREATION	An image has been associated with the display (using <b>MdispSelect()</b> or <b>MdispSelectWindow()</b> ).
M_MODIFIED_DIB_DESTRUCTION	An image has been disassociated from the display (using <b>MdispDeselect()</b> ).
M_MODIFIED_DIB	The content of the image has been modified.
M_MODIFIED_PAN	The pan of the display has been modified.
M_MODIFIED_ZOOM	The zoom of the display has been modified.
M_MODIFIED_LUT	The LUT associated with the display has been modified.

- In general, DIBXXXPtr and DIBDisplayXXXPtr are the same. However, when 24-bit RGB image buffers are displayed with an 8-bit VGA Windows driver, DIBDisplayXXXPtr might point to 8-bit precalculated 3:3:2 versions of DIBXXXPtr 24-bit buffers. These 8-bit versions are created for a faster display of color images.

The **UserDataPtr** parameter 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.

**Return value** The returned value is the address of the hook-handler function (if any) that was previously hooked to the specified type of event. This allows you to chain hooked functions, or to restore the old hook function when unhooking. The prototype is as follows:

```
MVGAHOOKFCTPTR OldHookFctPtr;
```

**See also** **MdispSelect()**, **MdispSelectWindow()**

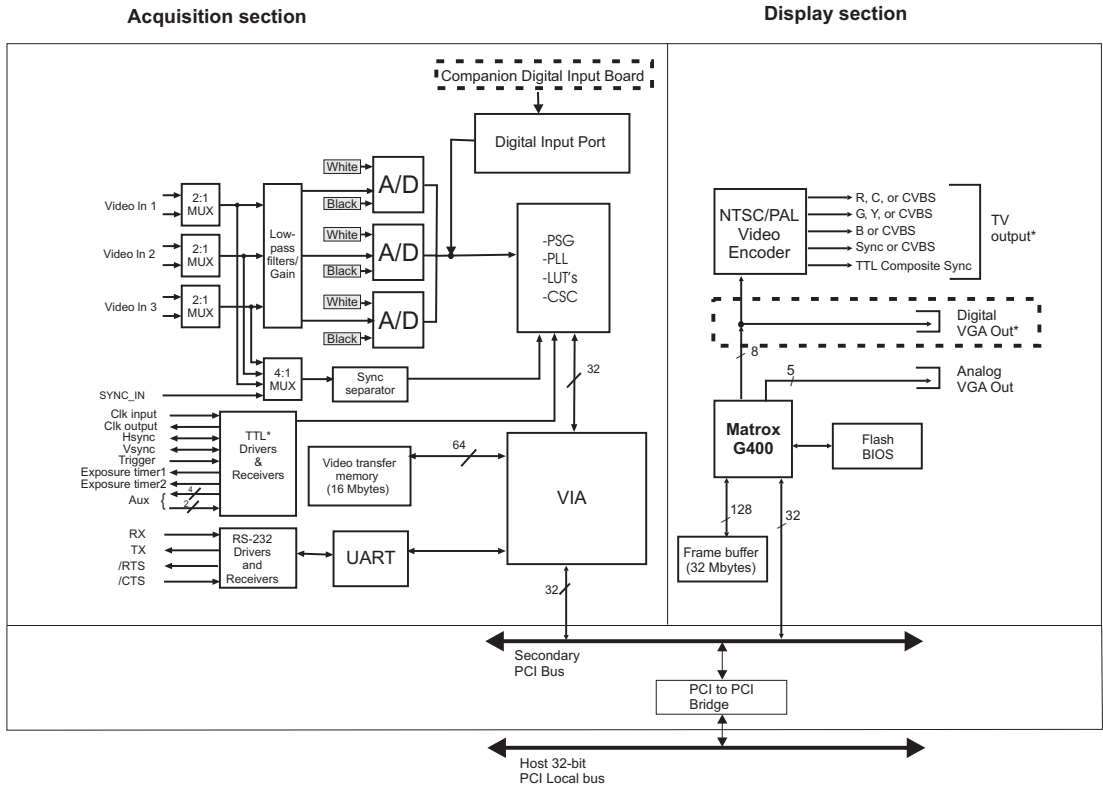


---

## ***Appendix A: Board flow diagrams***

*This section contains data flow diagrams for all Matrox frame grabbers.*

# Matrox Corona-II



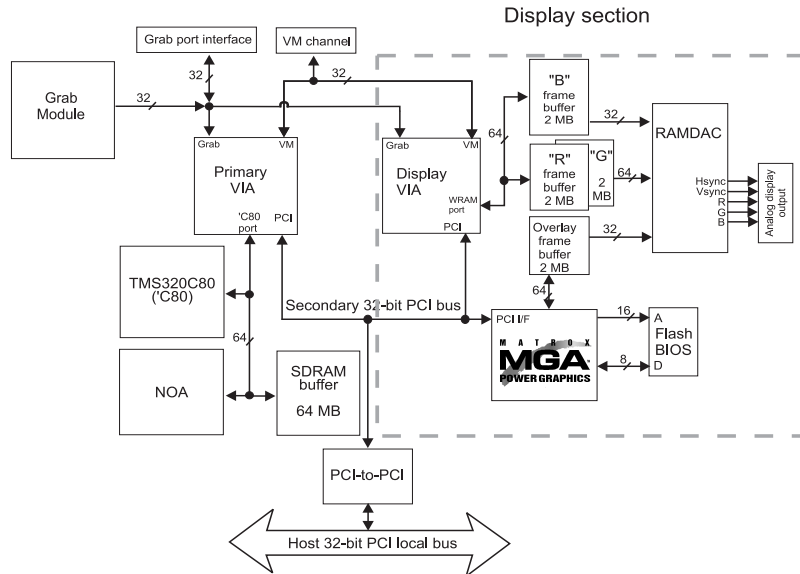
Optional

\* Note that the TV output and Digital VGA Outputs cannot be used at the same time

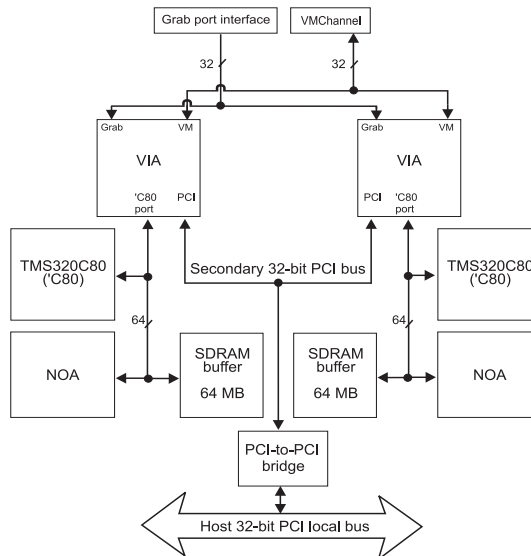


# Matrox Genesis

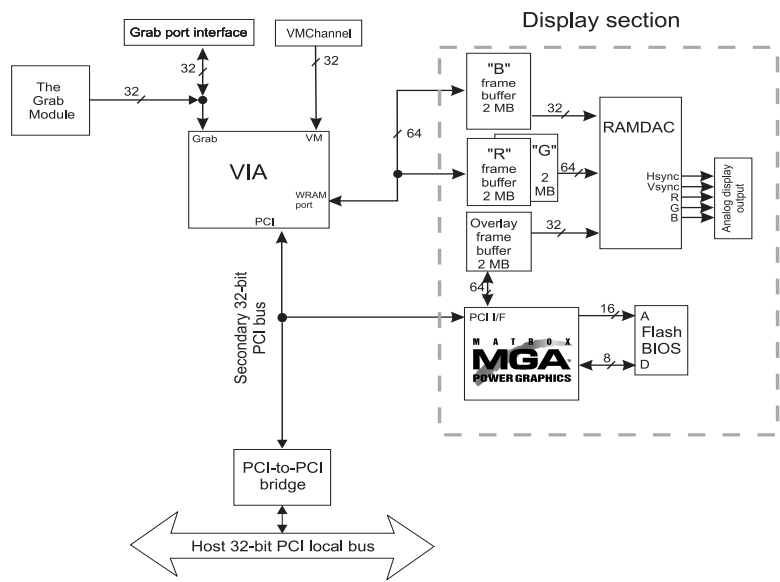
## Genesis Main Board



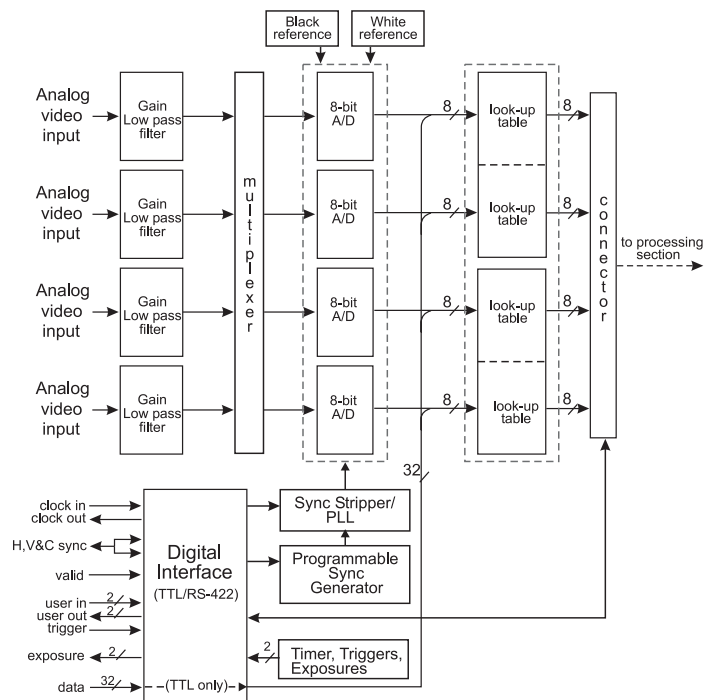
## Genesis Processor Board



# Genesis-LC

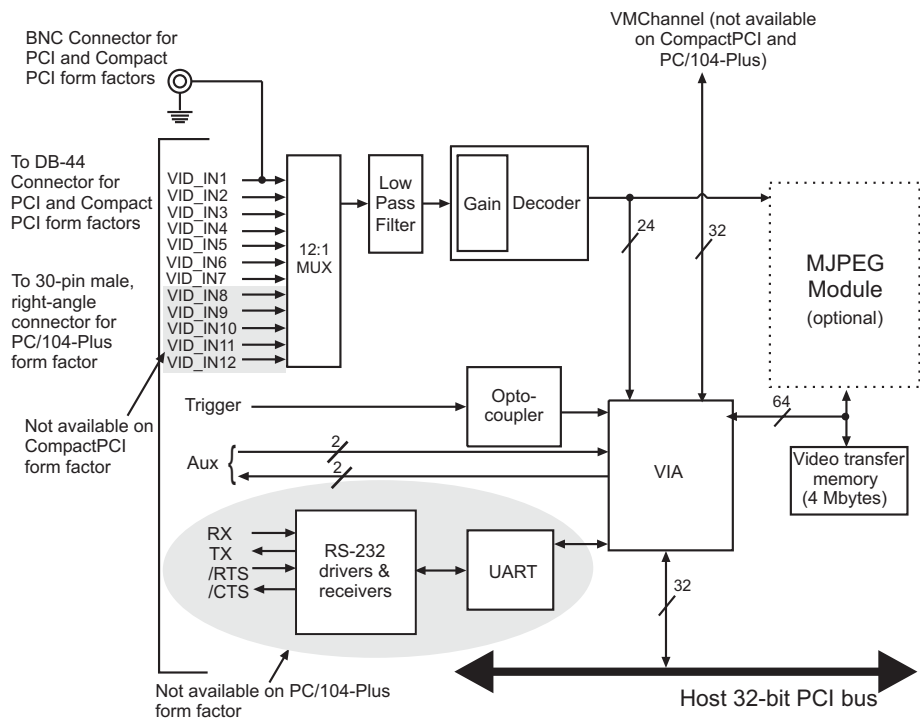


## Grab module

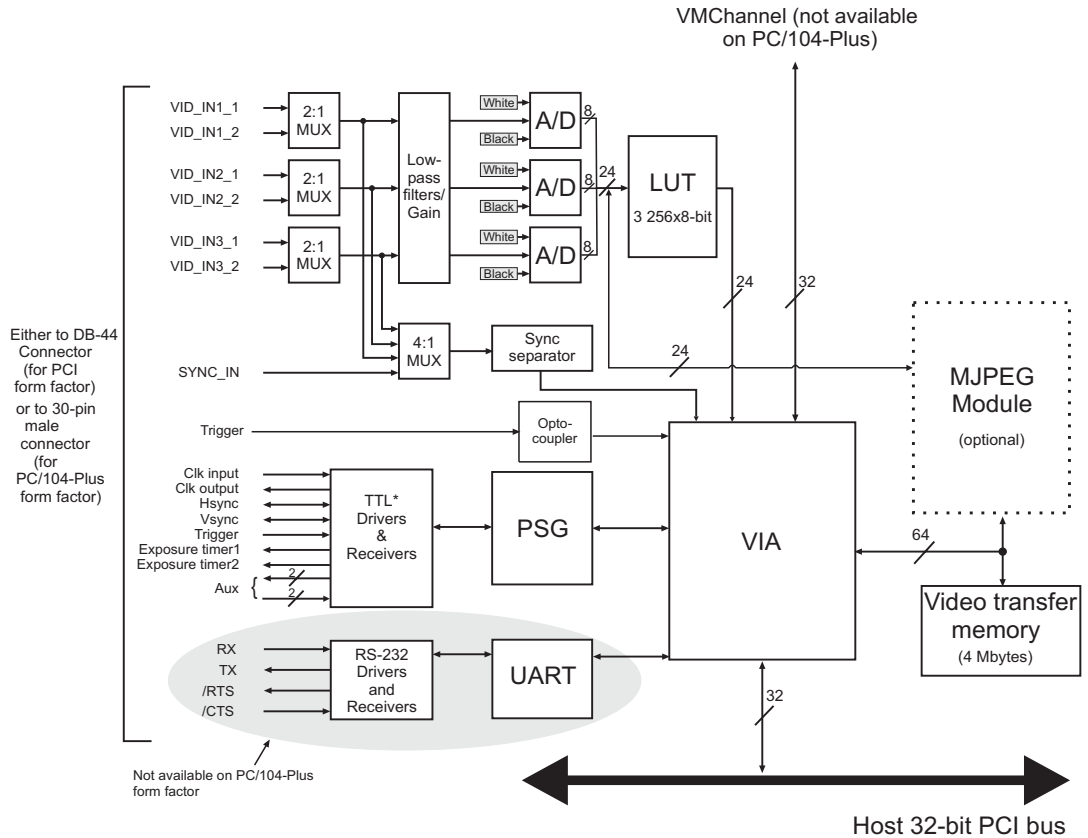


# Matrox Meteor-II

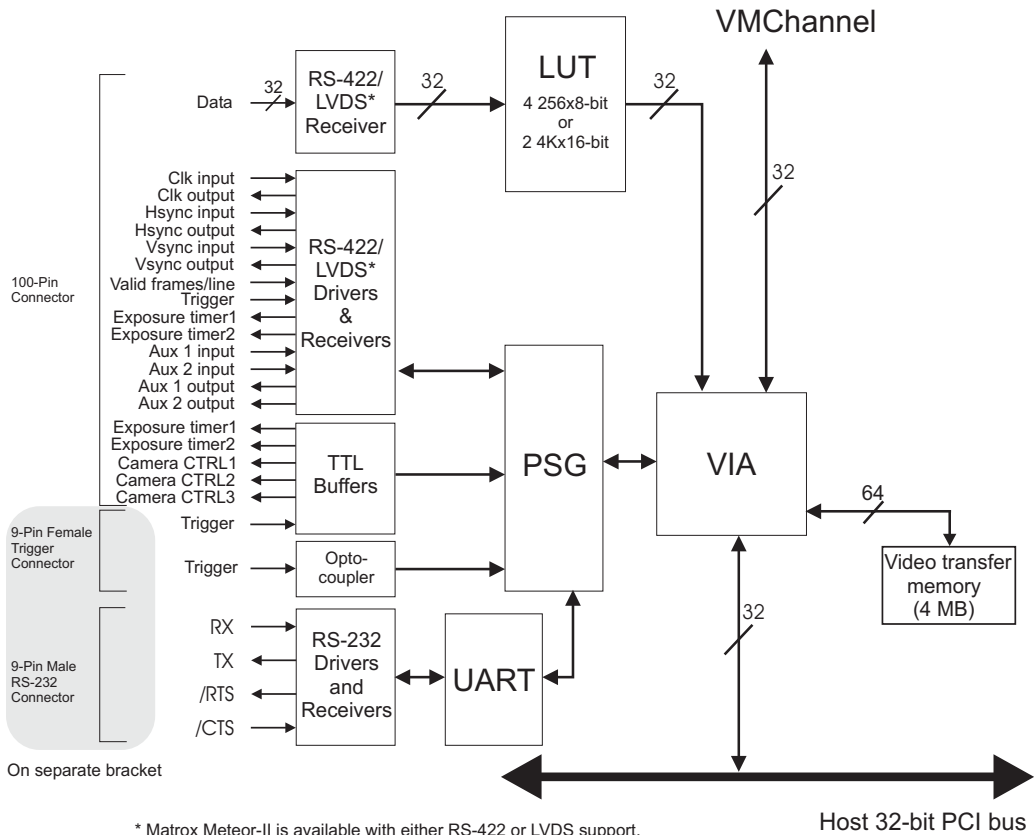
## Matrox Meteor-II /Standard



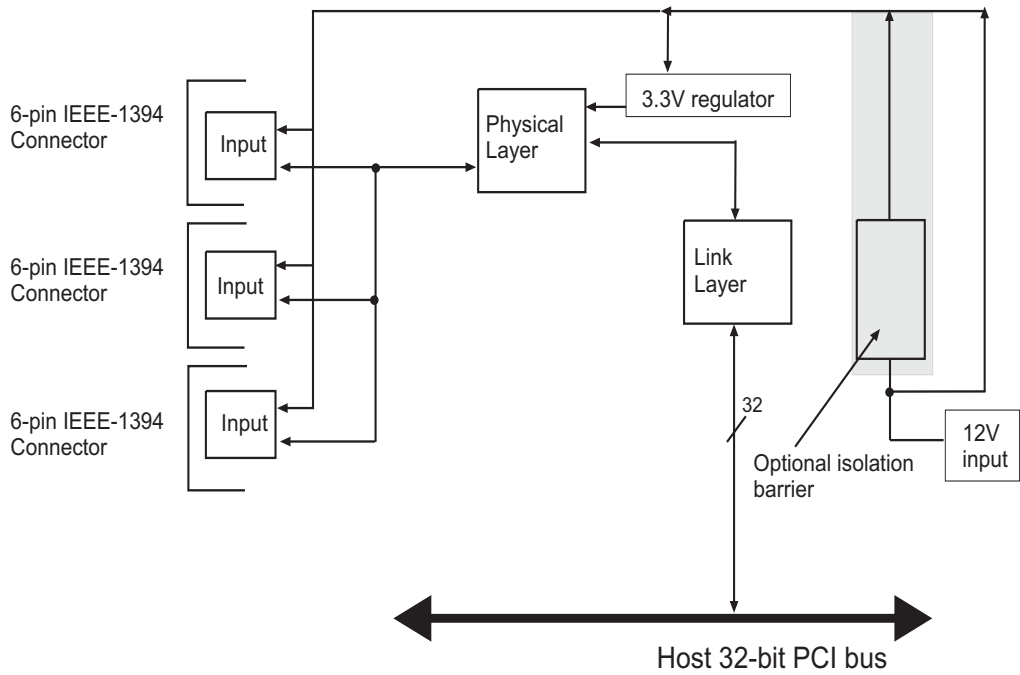
## Matrox Meteor-II /Multi-Channel



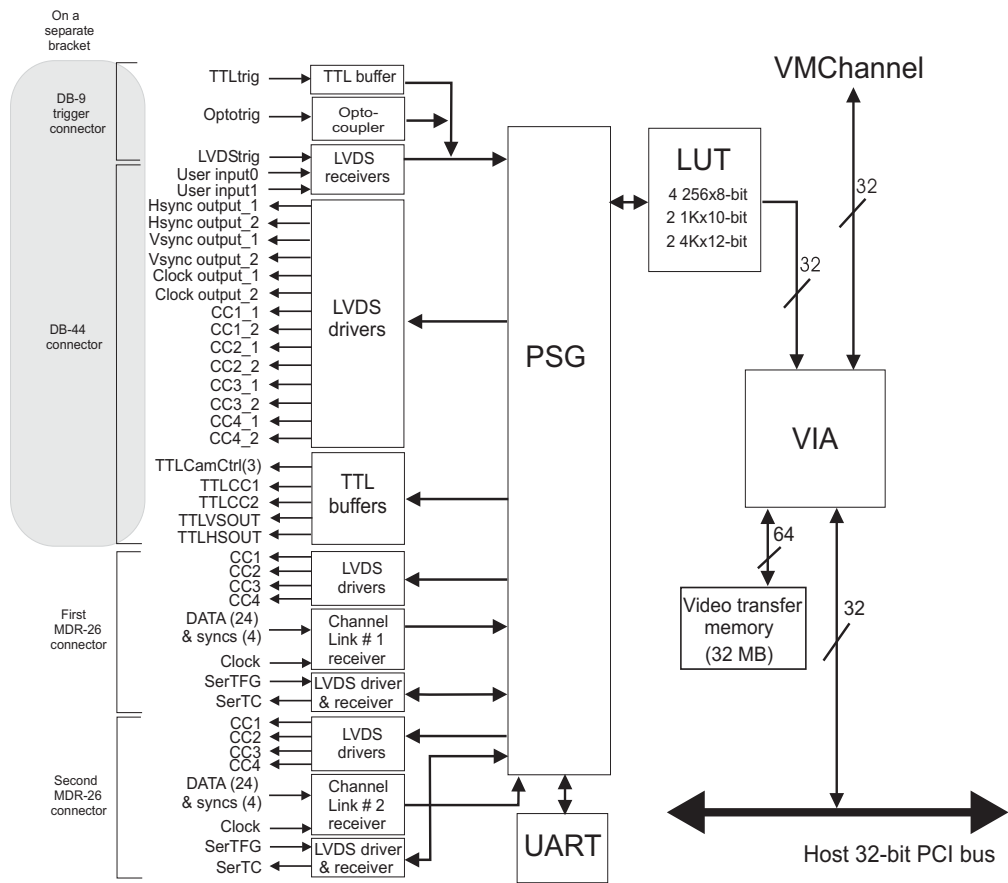
**Matrox Meteor-II /Digital**



## Matrox Meteor-II /1394

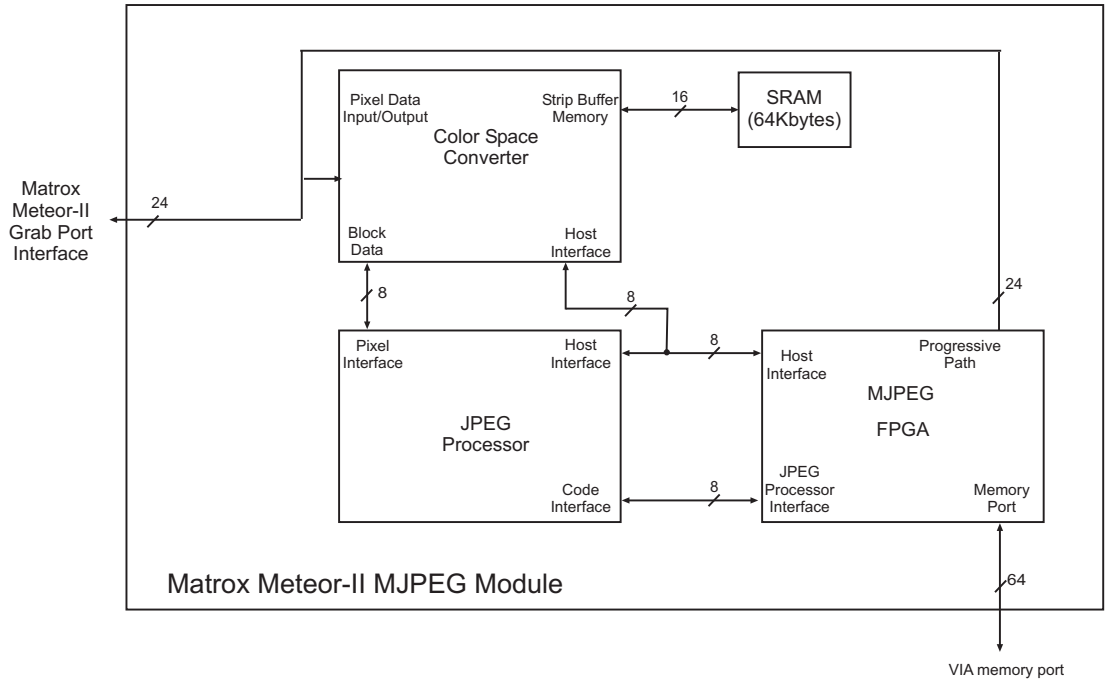


**Matrox Meteor-II /Camera Link**

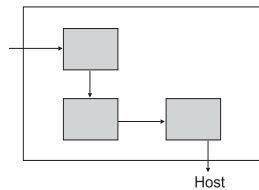




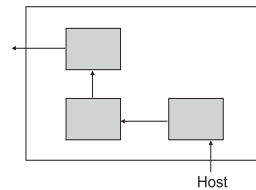
## Matrox Meteor-II /MJPEG Module



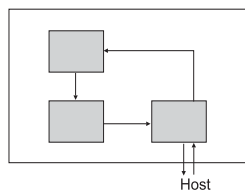
### MJPEG compression



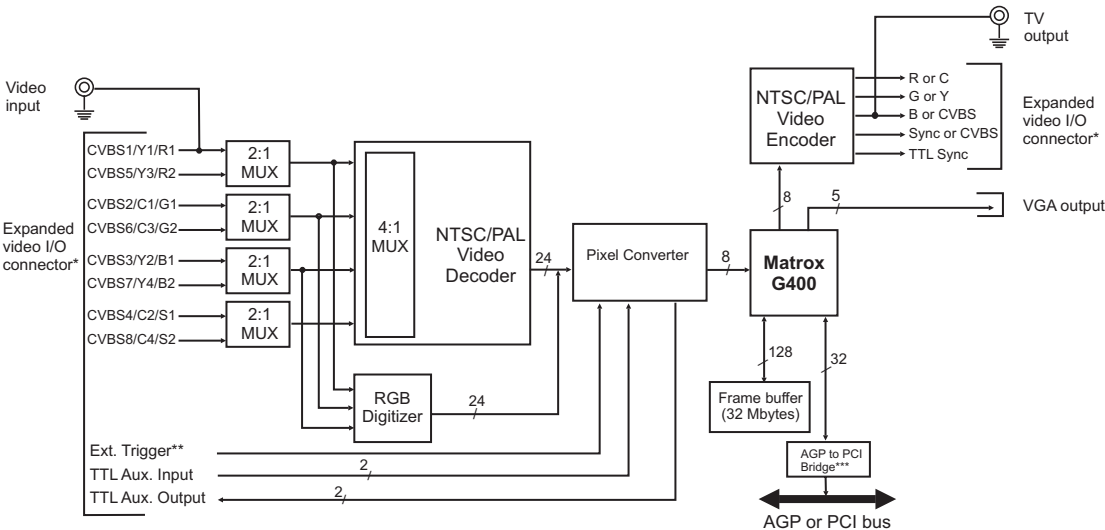
### Decompression



### JPEG compression



# Matrox Orion



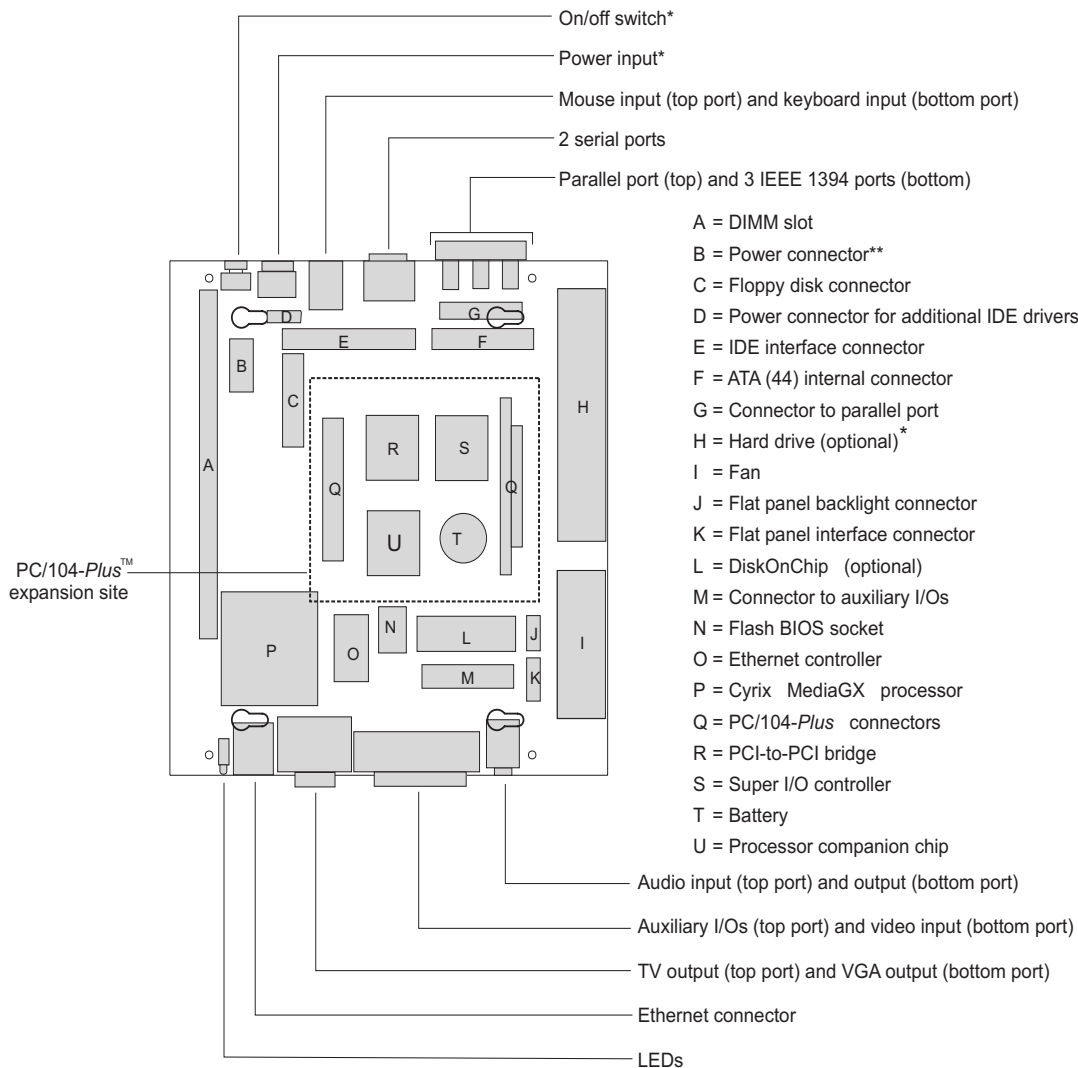
\* The expanded video I/O connector is used for both inputs and outputs, and is located on a separate bracket.

\*\* The external trigger can be either TTL or opto-isolated.

\*\*\* Only present on the PCI version.

# Matrox 4Sight

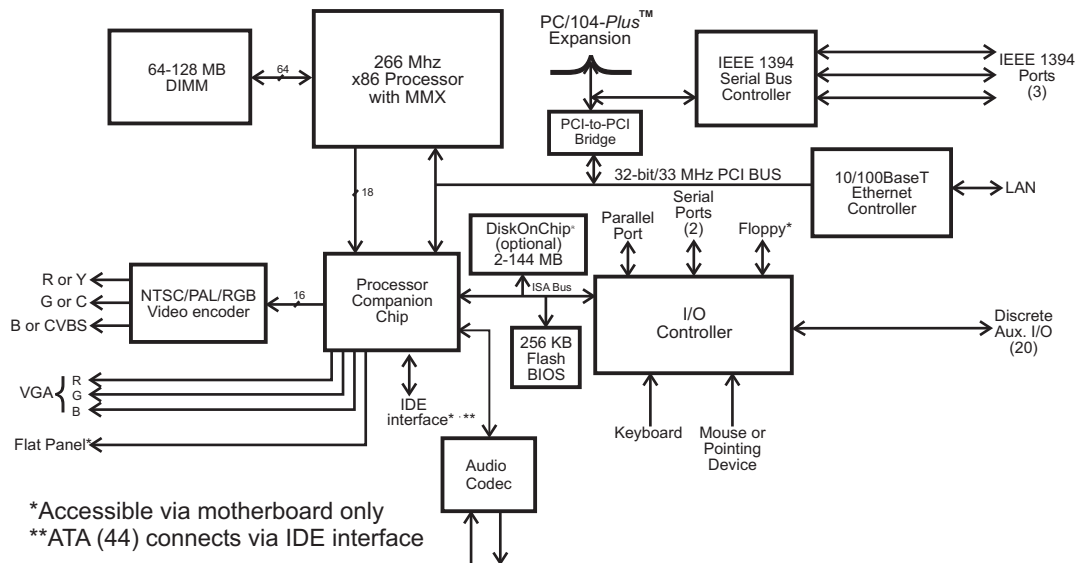
## Matrox 4Sight components and connectors



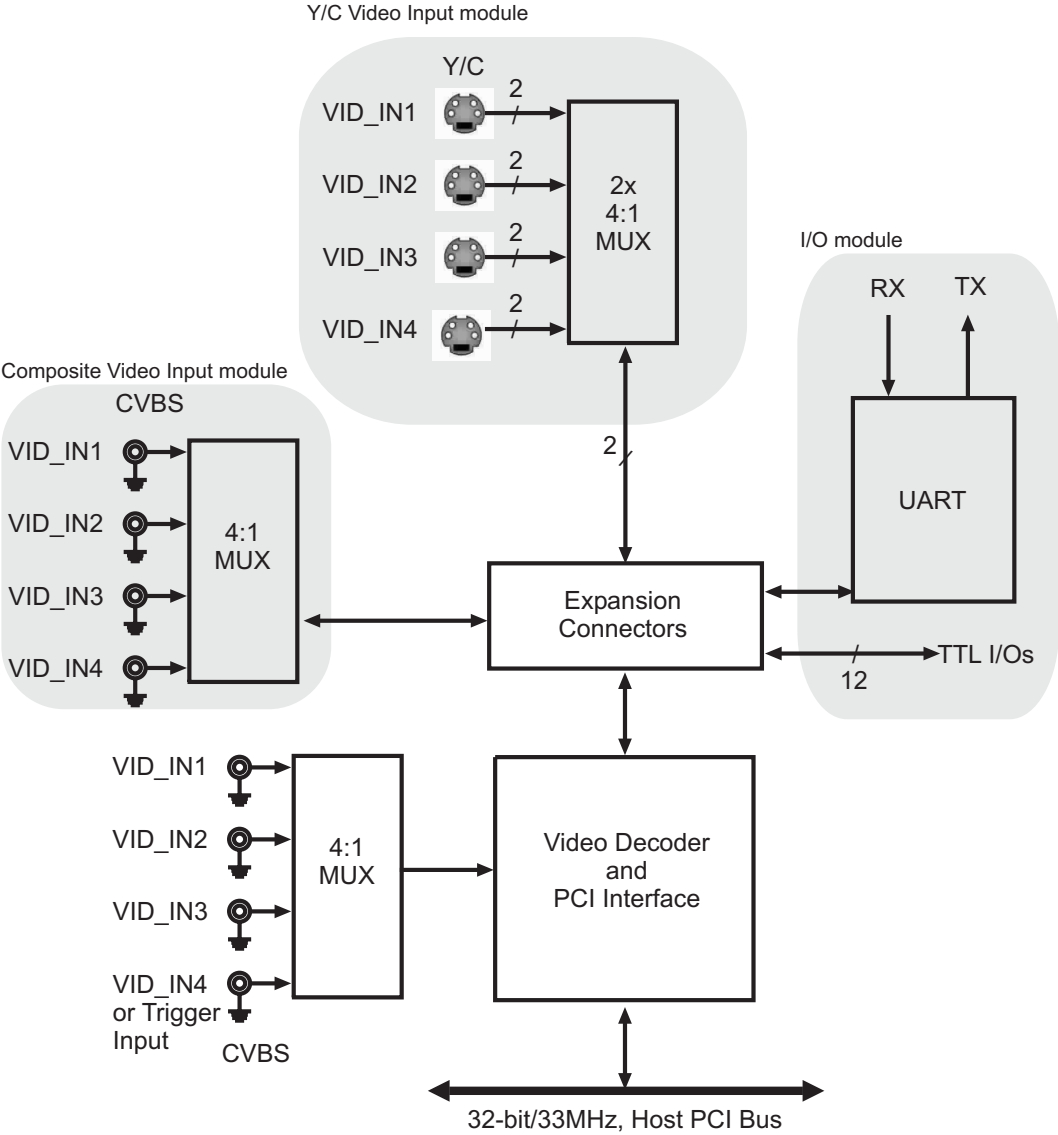
\* Available if you purchase the Matrox 4Sight integrated unit

\*\* Available on the stand-alone Matrox 4Sight motherboard

## Matrox 4Sight motherboard



# Matrox Cronos





# Index

## A

---

architecture, display 72  
Auxiliary inputs/outputs  
    4Sight 183–184

## B

---

buffers, Mbuf...()  
    Corona-II 21–22  
    Cronos 200  
    Genesis 60–61  
    Meteor-II 108  
    Orion 161

## C

---

command reference  
    Corona-II 20  
    Cronos 199  
    Genesis 58  
    Meteor-II 107  
    Orion 160  
    VGA, under Windows 218  
commands  
    VGA-specific 220  
Corona-II  
    encoder 19  
custom  
    window, VGA 219

## D

---

DCF files  
    Corona-II 22  
    Cronos 201  
    Genesis 61  
    Meteor-II 114  
    Orion 162  
DIB info structure, VGA 219  
digitizer 162, 201

digitizer, Mdig...()  
    Corona-II 22–24, 31–32, 38  
    Cronos 208–210, 212  
    Genesis 61–62, 68  
    Meteor-II 114, 137, 147  
    Orion 169–170, 172  
display  
    16-bit buffer simulated, Genesis 48  
    VGA system 219  
    Windows, VGA 219  
display architecture 72  
display, Mdisp...()  
    Corona-II 39–41, 174  
    Genesis 72  
    Orion 173, 175

## E

---

encoder 19, 159  
error reporting  
    hook, VGA 221

## G

---

Genesis main board  
    general features 46  
Genesis processor board  
    features 46  
Genesis-LC 47  
    grabbing to Host 49  
grab  
    double-buffering  
        Meteor-II /Camera Link 92  
        Meteor-II /Digital 92  
    grabbing to Host  
        Genesis-LC 49  
grab over-run 49–50, 92–93  
    Genesis-LC, solutions  
        buffer size 50  
        grabbing in on-board buffers 52  
        large frames of data 53  
    Meteor-II /Digital, solutions  
        buffer size 93  
        grabbing in on-board buffers 93  
        large frames of data 94

## H

---

### hook

- error, VGA 221
- trace, VGA 221
- user-defined function, VGA 219

## I

---

### IEEE 1394 cameras

- 4Sight 182

### inquire

- system 191, 193

## M

---

### M\_DIB\_INFO, VGA 219

### MappHookFunction()

- VGA 221

### Matrox 4Sight

- specific features 178

### MblobFill()

- Genesis 59

### MblobInquire()

- Genesis 59

### MblobReconstruct()

- Genesis 59

### MbufAlloc...()

- Corona-II 21
- Genesis 60
- Meteor-II 108
- Orion 161

### MbufAlloc2d()

- Genesis 60

### MbufCopyCond()

- Genesis 60

### MbufCopyMask()

- Genesis 60

### MbufCreate...()

- Corona-II 21
- Cronos 200
- Meteor-II 109
- Orion 161

### MbufCreateColor()

- Genesis 61

### MbufInquire()

- Corona-II 22
- Genesis 61
- Meteor-II 110
- Orion 161

### Mdig...()

- Cronos 201
- Orion 162
- VGA 218

### MdigAlloc()

- Corona-II 22
- Cronos 201
- Genesis 61
- Meteor-II 111
- Orion 162

### MdigChannel()

- Corona-II 23
- Cronos 202
- Meteor-II 114
- Orion 163

### MdigControl()

- Corona-II 24
- Genesis 62
- Meteor-II 119
- Orion 166

### MdigGrab 68

### MdigGrab()

- Corona-II 31
- Cronos 208
- Genesis 68
- Meteor-II 137
- Orion 169

### MdigGrab()/MdigGrabContinuous()

- Cronos 208
- Orion 169

### MdigGrabContinuous()

- Corona-II 31
- Cronos 208
- Meteor-II 137
- Orion 169

### MdigGrabWait()

- Genesis 68
- Meteor-II 138



MdigHookFunction()	Meteor-II /Digital
Corona-II 32	double-buffering 92
Cronos 209	MgenWarpParameter()
Genesis 68	Genesis 73
Meteor-II 138	MgraFontScale()
Orion 170	Genesis 74
MdigInquire()	milcor.txt
Corona-II 32	Cronos 196
Cronos 210	Orion 158
Genesis 68	milcor_II.txt
Meteor-II 138	Corona-II 17
Orion 170	milgen.txt
MdigLut()	Genesis 48
Corona-II 36	milmet2.txt 182
Genesis 70	Meteor-II 89
Meteor-II 147	MimConvolve()
Orion 172	Genesis 74
MdigReference()	MimRank()
Corona-II 38	Genesis 74
Cronos 212	MimResize()
Genesis 70	Genesis 74
Meteor-II 147	MimRotate()
Orion 172	Genesis 74
MdispAlloc()	MimWarp()
Corona-II 39	Genesis 74
Genesis 71	MJPEG
Orion 173	Meteor-II 87
MdispControl()	MpatInquire()
Corona-II 40, 174	Genesis 74
Genesis 72	MsysAlloc()
Orion 174	Corona-II 42
MdispDeselect()	Cronos 212
VGA 228	Genesis 74
MdispInquire()	Meteor-II 148
Corona-II 41	Orion 175
Genesis 72–73	MsysControl()
Orion 175	4Sight/4Sight-II 187
MdispLut()	Corona-II 42
Genesis 73	Cronos 212
MdispPan()	Genesis 75
Genesis 73	Meteor-II 149
Meteor-II	Orion 175
data compression/decompression	MsysGetHookInfor()
Meteor-II, lossless 87	4Sight 191
Meteor-II, lossy 87	
Meteor-II /Camera Link	
double-buffering 92	

MsysInquire() 191, 193  
4Sight/4Sight-II 188  
Corona-II 44  
Cronos 215  
Genesis 78  
Meteor-II 152  
Orion 176

MvgaAllocDIBInfo()  
VGA 228

MvgaDispFreeDIBInfo()  
VGA 219

MvgaDispSelectClientArea() 228

MvgaHookModifiedDIB() 219

## O

---

Orion  
encoder 159

## R

---

redraw image, VGA 219  
resolutions supported  
4Sight 178

## S

---

system  
inquire 191, 193  
system, Msys...()  
Corona-II 42, 44  
Cronos 212, 215  
Genesis 74, 78  
Meteor-II 152  
Orion 175–176

## T

---

trace  
hook, VGA 221

## U

---

UART  
Corona-II 17  
Cronos 196, 206  
Meteor-II 87

## V

---

VGA, board-specific functions 220  
Video Configuration Format  
Genesis 39, 71  
video formats supported  
4Sight 179

## W

---

Windows  
custom window, VGA 219  
redraw image, VGA 219

# Product Support

## Product Assistance Request Form

[illegible]

