

MIL / MIL-Lite

version 6.1

Board-Specific Notes

Manual no. 10515-801-0610

March 1, 2000

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

DOS/4GW™ is a trademark of Tenberry Software, Inc.

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., 2000. All rights reserved.

Disclaimer: Matrox Electronic Systems Ltd. reserves the right to make changes in specifications at any time and without notice. The information provided in this document is believed to be accurate and reliable. However, no responsibility is assumed by Matrox Electronic Systems Ltd. for its use; nor for any infringements of patents or other rights of third parties resulting from its use. No license is granted under any patents or patent rights 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 platform. . . . 15

Matrox Corona-specific features 16

Using Matrox Corona with MIL 17

Using the encoder 17

Encoder in single-screen mode with NTSC
resolution under Windows 98, or NTSC/PAL
resolution under Windows NT/2000 17

Encoder in single-screen mode with PAL
resolution under Windows 98 19

Encoder in dual-screen mode with NTSC/PAL
resolution under Windows 98 or
Windows NT/2000 20

Particularities of existing MIL functions on
Matrox Corona 21

MbufAlloc...() 22

MbufCreate...() 22

MbufInquire() 23

MdigAlloc() 24

MdigChannel() 25

MdigControl() 26

MdigGrab()/MdigGrabContinuous() 32

MdigHookFunction() 33

MdigInquire() 33

MdigReference() 37

MdispAlloc() 37

MdispControl()	39
MdispInquire()	40
MdispLut()	41
MdispOverlayKey()	43
MdispPan()	44
MdispZoom()	44
MsysAlloc()	45
MsysControl()	45
MsysInquire()	49

Chapter 3: MIL and the Matrox Genesis platform . . . 51

Matrox Genesis-specific features	52
Using Matrox Genesis with MIL	53
16-bit buffer simulated display.	54
Grabbing to a Host buffer with Matrox Genesis-LC	54
Optimizing the use of the frame buffers	55
Increasing the default temporary buffer size	55
Grabbing in on-board buffers.	57
Other options	63
Particularities of existing MIL functions on Matrox Genesis	63
MblobFill()	64
MblobInquire()	64
MblobReconstruct()	64
MbufAlloc...()	64
MbufCopyCond(), MbufCopyMask()	66
MbufCreateColor()	66

MbufInquire()	.66
MdigAlloc()	.66
MdigControl()	.67
MdigGrab()	.73
MdigGrabWait()	.73
MdigHookFunction()	.73
MdigInquire()	.73
MdigLut()	.75
MdigReference()	.75
MdispAlloc()	.75
MdispControl()	.76
MdispInquire()	.77
MdispLut()	.77
MdispPan()	.77
MdispZoom()	.78
MgenWarpParameter()	.79
MgraFontScale()	.79
MimConvolve()	.79
MimResize()	.79
MimRotate()	.79
MimWarp()	.79
MpatInquire()	.80
MsysAlloc()	.80
MsysControl()	.80
MsysInquire()	.85
Processing operations performed by Host	.86

Chapter 4: MIL and the Matrox Meteor-II platform . . . 93

Matrox Meteor-II family	94
Types of cameras and supported data format	96
Using Matrox Meteor-II with MIL	97
Transfers to display	98
Grabbing to a Host buffer with Matrox Meteor-II /Digital	99
Optimizing the use of the frame buffers . . .	99
Compression and decompression with Matrox Meteor-II MJPEG module	103
Compression	103
Decompression	104
IEEE 1394 serial bus-specific features	104
Video formats and modes.	105
Read/write capabilities	106
Setting camera features	106
An example	107
Grabbing from multiple 1394 DCS-based digital cameras	108
Important points to consider.	109
Particularities of existing MIL functions on Matrox Meteor-II	110
MbufAlloc...()	111
MbufCreate...().	112
MbufInquire().	113
MdigAlloc()	114
MdigChannel()	116
MdigControl().	119

MdigGrab/MdigGrabContinuous()	134
MdigGrabWait()	135
MdigHookFunction()	135
MdigInquire()	136
MdigLut()	144
MdigReference()	144
MdispLut()	146
MdispPan()	146
MdispZoom()	146
MsysAlloc()	147
MsysControl()	148
MsysInquire()	153

Chapter 5: MIL and the Matrox Orion platform 157

Matrox Orion-specific features.	158
Using Matrox Orion with MIL	159
Particularities of existing MIL functions on Matrox Orion.	159
MbufAlloc...()	160
MbufCreate...()	160
MbufInquire()	161
MdigAlloc()	161
MdigChannel()	162
MdigControl()	163
MdigGrab()/MdigGrabContinuous()	165
MdigHookFunction()	165
MdigInquire()	166
MdigReference()	168

MdispAlloc()	168
MdispInquire()	168
MdispPan()	168
MdispZoom()	168
MsysAlloc()	169
MsysInquire()	169

Chapter 6: MIL and the Matrox Pulsar platform 171

Matrox Pulsar-specific features	172
Using Matrox Pulsar with MIL	172
16-bit buffer simulated display.	173
Particularities of existing MIL functions on Matrox Pulsar	173
MbufAllocColor()	174
MbufAlloc2d()	174
MbufControl()	174
MbufCreate...()	175
MbufInquire()	176
MdigAlloc()	177
MdigControl()	177
MdigGrab()	180
MdigGrabContinuous()	181
MdigInquire()	181
MdigLut()	184
MdigReference()	185
MdispAlloc()	186
MdispLut()	186
MdispPan()	187

MdispSelect()	187
MdispZoom()	187
MsysAlloc()	188
MsysControl()	189
MsysInquire()	193

Chapter 7: MIL and the Matrox 4Sight platform195

Matrox 4Sight-specific features	196
Using Matrox 4Sight with MIL	197
Auxiliary inputs/outputs	198
Auxiliary outputs	198
Auxiliary inputs	199
Particularities of existing MIL functions on Matrox 4Sight	200
MsysControl()	201
MsysInquire()	202
Matrox 4Sight-specific MIL functions	204

Chapter 8: MIL and the VGA platform209

MIL functions under Windows	210
Mdig...()	210
Additional method of displaying an image in a user-defined window	210
VGA board-specific functions	212

Appendix A: Board flow diagrams 223

Matrox Corona 224

Matrox Genesis 225

Matrox Meteor-II 228

Matrox Orion 233

Matrox Pulsar 234

Matrox 4Sight 235

Matrox 4Sight motherboard 236

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 platform

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

Matrox Corona-specific features

Matrox Corona is a PCI frame grabber that is capable of acquiring images from standard NTSC/PAL, component RGB, and non-standard area scan cameras. Matrox Corona can capture 24-bit RS-422/LVDS digital input in monochrome or color when using the optional companion digital-input board. Matrox Corona also features a VMChannel and grab port for digital input. Images can be simultaneously transferred, in real-time, to Host system memory for processing and to the on-board frame buffers for "video-in-a-window" display.

Matrox Corona features a 2- or 4-Mbyte main (underlay) frame buffer, and a 2- or 4-Mbyte graphics overlay (VGA) frame buffer for non-destructive overlay capabilities. With dual 4-Mbyte frame buffers, Matrox Corona can deliver a 24-bit color image display with a 24-bit color overlay, for a completely true-color display at a 1280x1024 resolution. In the display section, an NTSC/PAL/RGB video encoder provides an additional video output with overlay, from Matrox Corona to external video devices (for example, VCRs). The video encoder can be programmed to output square pixel or CCIR 601 resolutions in either component RGB video or both composite and Y/C video.

Matrox Corona 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).

A low-cost version of Matrox Corona is also available – Matrox Corona-LC. It features only standard acquisition capabilities. It does not support asynchronous reset mode, digital video input, or component RGB acquisition. This chapter does not explicitly refer to Matrox Corona-LC because, in general, any discussion of Matrox Corona also applies to Matrox Corona-LC (except discussions of above mentioned features).

See Appendix A for a data flow diagram.

Using Matrox Corona with MIL

To use your Matrox Corona board with MIL, you must allocate a Corona system (M_SYSTEM_CORONA), using ***MsysAlloc()***. This establishes a MIL system environment in which MIL uses some Host memory, the on-board frame buffers, and the color keying mechanism of the Matrox Corona board to grab and display images (see ***MdispOverlayKey()***).

Matrox Corona can perform simultaneous dual-path image data transfer to Host and on-board frame buffers. See ***MdigAlloc()*** particularities later in this chapter.

This chapter relates Corona systems to the following: *Using the encoder* and *Particularities of existing MIL functions on Matrox Corona*.

Refer to the *milcor.txt* file in the \MIL (user-specified) directory for any additions/modifications to these board specific notes.

Using the encoder

Matrox Corona can be purchased with a factory-installed on-board encoder. When the encoder is enabled, the data that is routed to the VGA monitor (keyed and mapped) is also routed to the encoder for use as an additional output.

Since the video encoder does not have a scaler, the encoder is only enabled when your VGA display mode is 640x480x32-bit (NTSC) or 768x576x32-bit (PAL).

Encoder in single-screen mode with NTSC resolution under Windows 98, or NTSC/PAL resolution under Windows NT/2000

In single-screen mode with NTSC/PAL resolution under Windows NT/2000 or with NTSC resolution under Windows 98, you must adjust the display mode before running your application, as follows:

1. Ensure the display of Matrox Corona is enabled. To enable Corona's VGA, make sure SW1 position 2 is set to OFF.
2. Copy the *mga_enc.mon* file from the *\MGADRV\WINNT400 (WIN2000)* or *\MGADRV\WIN98* directory on the MIL CD to your local MGA monitor directory:
 - For Windows 98, it should be *\MATROX MGA POWERDESK\MON*.
 - For Windows NT/2000, it should be *\MGA NT POWERDESK\MON*.
3. Rename the *mga.mon* file in your local MGA monitor directory to *mga-original.mon*.
4. Then, rename the *mga_enc.mon* file in the local MGA monitor directory (recently copied) to *mga.mon*, replacing the old one.
5. In the same directory, delete the *mga.bin* file.
6. Restart your computer (PC).
7. Once your PC has restarted, launch the MGA Display Properties utility as described in the *Matrox Corona Installation and Hardware Reference* manual. The **Matrox Display Properties** dialog box appears.
8. Select the **Settings** property page. Change the display area setting to **640x480**.
9. Click on the **Apply** button.
10. Now, select the **Monitor** property page. Your PC will now rebuild your MGA monitors database (*mga.bin*) to include an entry for Matrox Corona.
11. While still on the **Monitor** property page, select the **Matrox monitor** option to view the Matrox monitor profiles list box.
12. In the Matrox monitor list box, scroll down and double-click on **Standard NTSC PAL NI** monitor profile.

13. Select the **Matrox Corona** entry for the monitor.
14. Click on the **Apply** button.
15. Return to the **Settings** property page, and choose between 640x480 for NTSC or 768x576 for PAL (Windows NT/2000).
- ❖ Note that it is assumed that the VGA display output is in non-interlaced mode.
16. Now, to use the encoder, add the following function calls to your code:

```
MdispControl(MilDisplay, M_ENCODER, M_ENABLE);
MdispControl(MilDisplay, M_ENCODER_MODE, M_ENABLE, M_COMPOSITE);
```

Encoder in single-screen mode with PAL resolution under Windows 98

To enable the encoder in PAL resolution under Windows 98 in single-screen mode, do the following:

1. Follow steps 1 through 5 above for NTSC resolution.
2. Copy the *palinfo.inf* file, found in the *MGADR\WIN98* directory on the MIL CD, to your root directory.
3. Restart your PC.
4. Once your PC has restarted, launch the MGA Display Properties utility as described in the *Matrox Corona Installation and Hardware Reference* manual. The **Matrox Display Properties** dialog box appears.
5. Select the **Monitor** property page. Then, choose the Windows default monitor.
6. Now, select the **Settings** property sheet. Change the display area setting to **768x576x32-bit**.
7. To use the encoder, add the following function calls to your code:

```
MdispControl(MilDisplay, M_ENCODER, M_ENABLE);
MdispControl(MilDisplay, M_ENCODER_MODE, M_ENABLE, M_COMPOSITE);
```

Encoder in dual-screen mode with NTSC/PAL resolution under Windows 98 or Windows NT/2000

In dual-screen mode, you must adjust the display mode before running your application, as follows:

1. Disable the VGA on Matrox Corona. To do so, set the SW1 position 2 to ON.
2. Open the *Milsetup.h* file found in the local `\MATROX\IMAGING\MIL\INCLUDE` directory.
3. Under the DEFAULT DIGITIZER SPECIFICATIONS section, specify the digitizer configuration format (DCF) file corresponding to the type of video camera signal. That is, if you are using a PAL camera, a "*PAL.DCF*" format must be specified. Similarly, if you are using a NTSC camera, a "*NTSC.DCF*" format must be specified.
4. Under the DEFAULT DISPLAY SPECIFICATIONS section, specify the corresponding video configuration format (VCF) file. That is, if using a "*PAL.DCF*" format, then specify a "*PAL.VCF*" format. Similarly, if using a "*NTSC.DCF*" format, then specify a "*NTSC.VCF*" format.
5. In dual-screen mode, you select the display mode using the **MdispAlloc()** function (specifying the *ntsc.vcf* or *pal.vcf* file).
6. To use the encoder, add the following function calls to your code:

```
MdispControl(MilDisplay, M_ENCODER, M_ENABLE);
MdispControl(MilDisplay, M_ENCODER_MODE, M_ENABLE, M_COMPOSITE);
```

❖ In dual-screen mode, it is advisable to use a Matrox Corona with a Matrox VGA video card. A Matrox video card with G200 or Matrox G400 graphics controller is suggested.

For further information on the encoder, see **MdispControl()** and **MdispInquire()**.

Particularities of existing MIL functions on Matrox Corona

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

Commands	Corona particularities
MbufAlloc...()	Buffer constraints and options.
MbufCreate...()	Required parameter settings.
MbufInquire()	Corona-specific inquire options.
MdigAlloc()	Camera format file specifications.
MdigChannel()	Required parameter settings.
MdigControl()	Control type and value additions and restrictions.
MdigGrab/ MdigGrabContinuous()	Destination buffer restriction.
MdigHookFunction()	Hook restrictions.
MdigInquire()	Corona-specific inquire options.
MdigReference()	Corona features.
MdispAlloc()	Display format. Overlay setting.
MdispControl()	Required parameter settings.
MdispInquire()	Additional inquiries.
MdispLut()	Available for overlay and underlay displays.
MdispOverlayKey()	Particularities and restrictions.
MdispPan()	Panning particularities.
MdispZoom()	Supported zoom factors.
MsysAlloc()	Corona-specific allocation options.
MsysControl()	Control type and value additions.
MsysInquire()	Corona-specific inquire options.

MbufAlloc...()

- Only 8-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_RGB16 + M_PACKED*	16-bit (RGB) packed format.
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.
M_YUV12 + M_PLANAR*	YUV12 planar format.
M_YUV16 + M_PLANAR*	YUV16 planar format.
M_YUV24 + M_PLANAR*	YUV24 planar standard.
* Can only be used when grabbing from the decoder path.	

Also, note that it might be slower to process buffers with these attributes.

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	ArrayOfDataPtr is an array of physical addresses. The pitch is in pixels (default).
M_PHYSICAL_ADDRESS + M_PITCH_BYTE	ArrayOfDataPtr is an array of physical addresses. The pitch is in bytes.

- 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_RGB16 + M_PACKED*	16-bit (RGB) packed format.
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.
M_YUV12 + M_PLANAR*	YUV12 planar format.
M_YUV16 + M_PLANAR*	YUV16 planar format.
M_YUV24 + M_PLANAR*	YUV24 planar standard.
* Can only be used when grabbing from the decoder path.	

Note that it might be slower to process buffers with these attributes.

MbuflInquire()

- 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 in windowed mode. This information is used, for example, to access grabbed data while a continuous grab (MdigGrabContinuous()) 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.
------------------	---

MdigAlloc()

- Matrox Corona features two real-time data paths which transfer grabbed images in real-time to the display and Host memory simultaneously. In order to use this feature, two digitizers must be allocated. One of these digitizers must have the **InitFlag** set to M_DISPLAY_DIGITIZER. The digitizer with this **InitFlag** can only be used to grab in the main (underlay) frame buffer of the display (that is, a buffer with the M_ON_BOARD attribute). See *mdualpat.c* for an example using the dual data path.

Note that the dual data path is only supported with standard NTSC/PAL cameras.

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

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 module.
"M_CCIR_VIA_RGB"	CCIR using RGB module.

The predefined settings for color cameras are:

"M_DEFAULT"	NTSC, 640x480, 3x8 bits, 12.5 MHz, composite
"M_NTSC"	NTSC, 640x480, 3x8 bits, 12.5 MHz, composite
"M_NTSC_RGB"	RS-170 RGB, 640x480, 3x8 bits, 12.5 MHz
"M_NTSC_YC"	RS-170 Y/C(SVHS), 640x480, 3x8 bits, 12.5MHz
"M_PAL"	PAL I, 768x576, 3x8 bits, 14.8 MHz, composite
"M_PAL_RGB"	PAL I RGB, 768x576, 3x8 bits, 14.8 MHz
"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 user-specified directory for the current list of supported formats.

MdigChannel()

- When using the video decoder, 4 independent composite/monochrome or 2 Y/C cameras can be attached to the Matrox Corona board. When using the RGB path, one RGB camera or 3 monochrome cameras can be attached. The DCF selected during digitizer allocation determines the color mode and path used.
- 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_RGB	VID1_IN1, VID1_IN2, VID1_IN3 (data signals) and SYNC_IN (sync signal)
Y/C	M_CH0	(Y camera 1) VID_IN1 and (C camera 1) VID_IN2
	M_CH1	(Y camera 2) VID_IN3 and (C camera 2) VID_IN4
Composite or Monochrome	M_CH0	VID_IN1
	M_CH1	VID_IN2
	M_CH2	VID_IN3
	M_CH3	VID_IN4

MdigControl()

- It is not possible to queue asynchronous grabs on Matrox Corona. Therefore, M_ASYNCHRONOUS_QUEUED cannot be used as a control value with the M_GRAB_MODE control type.
- Corona supports these additional **ControlType** parameter settings:

ControlType	Description & ControlValue		
M_GRAB_AUTOMATIC_INPUT_GAIN	When grabbing using the decoder path, automatically sets the input gain.		
	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.	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.		
M_GRAB_DIRECTION_X	Set the horizontal grab direction:		
	M_REVERSE	Flip the grabbed image horizontally.	
	M_FORWARD	Grab normally in the horizontal direction.	
	M_DEFAULT	Same as M_FORWARD.	
M_GRAB_DIRECTION_Y	Set the vertical grab direction:		
	M_REVERSE	Flip the grabbed image vertically.	
	M_FORWARD	Grab normally in the vertical direction.	
	M_DEFAULT	Same as M_FORWARD.	
M_GRAB_PATH_OVERRIDE	Forces live grabs to make use of the secondary PCI path:		
	M_GRAB_PATH_PCI	Force all grabs to use secondary PCI path.	
	M_DEFAULT	Use the Rainbow Runner whenever possible.	

ControlType	Description & ControlValue	
M_GRAB_SCALE	<p>Scaling factors: 0...1, when grabbing with the decoder. Therefore, Matrox Corona supports arbitrary scaling. However, this feature is only supported when using a board with the hardware rev.B of the video decoder (Samsung KS0127B). If you don't have rev.B of the video decoder, the closest subsampling value will be used.</p> <p>Scaling factors: 1, 1/2, 1/3,..., 1/16, when grabbing from the RGB path.</p>	
M_GRAB_TRIGGER_MODE	Set the hardware trigger activation mode.	
	M_EDGE_RISING	Low to high signal variation.
	M_EDGE_FALLING	High to low signal variation.
	M_LEVEL_LOW	Minimum signal level.
	M_LEVEL_HIGH	Maximum signal level.
M_GRAB_TRIGGER_SOURCE	Set the source of the grab trigger.	
	M_HARDWARE_PORT0	Use the opto-isolated hardware trigger signal. Combination of Pin 34 (OPTOTRIG-) and Pin 35 (OPTOTRIG+) on Video Input connector.
	M_HARDWARE_PORT1	Use the TTL hardware trigger signal. Pin 20 on Video Input connector or Pin 2 (TRIGGER) on Digital Interface connector.
	M_HARDWARE_PORT2	Use the RS-422/LVDS trigger signal. Combination of Pin 73 (TRIG+) and Pin 74 (TRIG-) on the video input connector on the companion digital-input board.
	M_SOFTWARE	Use software trigger.
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, or M_DEFAULT (same as DCF).
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_DEFAULT Same as the .dcf file.
	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 MdigControl() with M_GRAB_EXPOSURE + M_TIMERN and M_ACTIVATE is called.
	M_HARDWARE_PORT0 Use the opto-isolated hardware trigger signal. Combination of Pin 34 (OPTOTRIG-) and Pin 35 (OPTOTRIG+) on Video Input connector.
	M_HARDWARE_PORT1 Use the TTL hardware trigger signal. Pin 20 on Video Input connector or Pin 2 (TRIGGER) on Digital Interface connector.
	M_HARDWARE_PORT2 Use the RS-422/LVDS trigger signal. combination of pin 73 (TRIG+) and pin 74 (TRIG-) on the video input connector on the companion digital-input board.
	M_VSYNC Use vertical sync signal.

ControlType	Description & ControlValue	
M_GRAB_EXPOSURE_SOURCE (cont.)	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.
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 ControlValue 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_DEFAULT	Same as the .dcf file.
	M_EDGE_RISING	Low to high signal variation.
	M_EDGE_FALLING	High to low signal variation.
M_GRAB_TIMEOUT	Set the maximum time to wait for a frame before generating an error.	
	M_DEFAULT	Determined by the frame period.
	M_INFINITE	Wait indefinitely. This is recommended only for triggered cameras.
	Value in msec	Specify time for wait.
M_GRAB_WINDOW_RANGE	Limit the range of the grabbed pixel values to between 10 and 245 for Matrox Corona and between 16 and 235 for Matrox Corona-LC: M_ENABLE or M_DISABLE.	

ControlType	Description & ControlValue	
M_THREAD_PRIORITY	Set the hook function priority under Windows. Valid values are:	
	11 - 15	High priority.
	16 or 22 - 26	Real-time priority.
	If you want to change the hook function priority, use MdigInquire() to determine its current value. Changing these priority settings will affect the overall application priority, due to a Windows restriction.	
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_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_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	The ControlValue is a pointer to a character. Send one character to the output of the UART.	
M_UART_READ_CHAR	The ControlValue is a pointer to a character. 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.	
M_UART_STRING_DELIMITER	Set the character used to terminate strings of incoming or outgoing data. The delimiter is not sent when writing data; it is read for incoming data.	
	M_DEFAULT	The '\0' character.

ControlType	Description & ControlValue											
M_UART_READ_STRING_LENGTH	Set the length of the string (in bytes) to be read by M_UART_READ_STRING. <div>M_DEFAULTUsed to specify the use of M_STRING_DELIMITER to end string.</div> <div>Value in bytesSpecify string length.</div>											
M_UART_READ_STRING_MAXIMUM_LENGTH	Use this value to specify the size of your reading 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 ControlValue 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 output of the UART. <div>M_DEFAULTUsed to specify the use of M_STRING_DELIMITER to end string. The delimiter will not be sent through the UART.</div> <div>Value in bytesSpecify string length.</div>											
M_UART_WRITE_STRING	Send the string of data, specified by the control value, through the UART. Set ControlValue to the character array. The number of characters to send can be specified with M_UART_WRITE_STRING_LENGTH or M_UART_STRING_DELIMITER.											
M_USER_BIT+(3, 4, 5, 6)	<div>Set the state of an output bit of the Video Input connector: M_ON or M_OFF</div> <div>The relationship between the MIL user-bit number and the actual user output bit on the Video Input connector is as follows:</div> <table><thead><tr><th>User Bit#</th><th>Video Input connector signal</th></tr></thead><tbody><tr><td>3</td><td>USER1OUT signal.</td></tr><tr><td>4</td><td>USER2OUT signal.</td></tr><tr><td>5</td><td>RS-422 USEROUT0</td></tr><tr><td>6</td><td>RS-422 USEROUT1</td></tr></tbody></table> <div>User-bits 5 and 6 are only supported on the Matrox Corona companion digital-input board.</div>		User Bit#	Video Input connector signal	3	USER1OUT signal.	4	USER2OUT signal.	5	RS-422 USEROUT0	6	RS-422 USEROUT1
User Bit#	Video Input connector signal											
3	USER1OUT signal.											
4	USER2OUT signal.											
5	RS-422 USEROUT0											
6	RS-422 USEROUT1											

ControlType	Description & ControlValue	
M_VCR_INPUT_TYPE	Set the digitizer input to VCR and improve the image's quality when grabbing from a VCR. This is only valid when using the decoder path:	
	M_DEFAULT	Same as M_DISABLE
	M_DISABLE	Leaves image quality as is.
	M_ENABLE	Improves the image quality when grabbing with a VCR.

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 performing real-time grabs, Windows 98 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.

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 functions to input-signal events that occur while grabbing.
- The **HookType** paramter 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_BRIGHTNESS_REF (when applicable)	Brightness reference level. Refer to <i>MdigReference()</i> for when applicable.
M_COLOR_MODE	Color mode: M_RGB, M_MONO8_VIA_RGB, M_EXTERNAL_CHROMINANCE, M_COMPOSITE, or M_MONOCHROME.
M_CONTRAST_REF (when applicable)	Contrast reference level. Refer to <i>MdigReference()</i> for when applicable.
M_GRAB_AUTOMATIC_INPUT_GAIN	Mode of automatic input gain: M_ENABLE or M_DISABLE.
M_GRAB_DIRECTION_Y	Vertical grab direction: M_REVERSE OR M_FORWARD.
M_GRAB_INPUT_GAIN	The gain that is applied to input signal. When grabbing using the RGB path: M_GAIN0, M_GAIN1, M_GAIN2, or M_GAIN3. When grabbing using the decoder path and M_GRAB_AUTOMATIC_INPUT_GAIN is disabled (M_DISABLE), the value can be any integer from 0 to 255.
M_GRAB_PATH_OVERRIDE	Forced path used for live grabs into the main frame buffer: M_GRAB_PATH_PCI (secondary PCI) or M_DISABLE.

InquireType	Description
M_GRAB_SCALE	Scaling factors: 0,...1, when grabbing from the decoder path. However, this feature is only supported when using a board with the hardware rev.B of the video decoder (Samsung KS0127B). If you don't have rev.B of the video decoder, the closest subsampling value will be used. Scaling factors: 1, 1/2, 1/3,..., 1/16, when grabbing from the RGB path.
M_HUE_REF (when applicable)	Hue reference level. Refer to <i>MdigReference()</i> for when applicable.
M_INPUT_SIGNAL_COLOR_LOCK	Video input signal color lock. Returns M_TRUE when a color lock is achieved, otherwise it returns M_FALSE. Only available when grabbing from the decoder path.
M_INPUT_SIGNAL_HYSNC_LOCK	Video input signal horizontal synchronization. Returns M_TRUE when a line lock is achieved, otherwise it returns M_FALSE. Only available when grabbing from the decoder path.
M_INPUT_SIGNAL_PRESENT	Video input signal present: M_YES or M_NO.
M_SATURATION_REF (when applicable)	Saturation reference level. Refer to <i>MdigReference()</i> for when applicable.
M_GRAB_EXPOSURE_BYPASS	Exposure model: M_ENABLE (manual) or M_DISABLE (auto).
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.
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.

InquireType	Description	
M_GRAB_END_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_END, will run.	
M_GRAB_END_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_END, will run.	
M_GRAB_FIELD_END_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FIELD_END, will run.	
M_GRAB_FIELD_END_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FIELD_END, will run.	
M_GRAB_FIELD_START_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FIELD_START, will run.	
M_GRAB_FIELD_START_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FIELD_START, will run.	
M_GRAB_FRAME_END_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FRAME_END, will run.	
M_GRAB_FRAME_END_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FRAME_END, will run.	
M_GRAB_IN_PROGRESS	Current grab state: M_YES or M_NO.	
M_GRAB_START_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_START, will run.	
M_GRAB_START_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_START, will run.	
M_GRAB_TIMEOUT	Maximum time to wait for the end of grab before generating an error: M_INFINITE or a value in msec.	
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_THREAD_PRIORITY	Hook function priority under Windows:	
	11-15	High priority.
	16 or 22-26	Real-time priority.
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_STRING_RECEIVE 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_STRING_SEND_LENGTH or M_UART_STRING_RECEIVE_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_STRING_SEND control is called. Can be M_DEFAULT if the send length is specified by M_UART_STRING_DELIMITER.	
M_USER_BIT+ (1,2,3,4, 5, 6, 7, 8)	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
	User-bits 5-8 are only supported on the Matrox Corona companion digital-input board.	
	M_VCR_INPUT_TYPE	VCR input type. This is only valid when using the decoder path: M_DISABLE or M_ENABLE

MdigReference()

- When grabbing using the RGB section of Matrox Corona, 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).

- When grabbing monochrome using the decoder section of Matrox Corona, you can control the brightness and contrast levels of the input signal using M_BRIGHTNESS_REF and M_CONTRAST_REF.
- When grabbing color using the decoder section of Matrox Corona, 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.

MdispAlloc()

- When operating under Windows in single-screen mode, set **DispFormat** to the string "M_DEFAULT". This sets the main (underlay) frame buffer's display format to that of the overlay (VGA) frame buffer display.

When operating under Windows in dual-screen mode, set **DispFormat** to a string that specifies the required display resolution for the VGA display. Possible settings are:

Display Resolution	DispFormat				
	8-bit	15-bit	16-bit	24-bit	32-bit
640x480 at 60 Hz	"VM101_60.VCF" or "640x480x8PP"	"VF101_60.VCF" or "640x480x15PP"	"VS101_60.VCF" or "640x480x16PP"	"VV101_60.VCF" or "640x480x24PP"	"VT101_60.VCF" or "640x480x32PP"
640x480 at 72 Hz	"VM101_72.VCF"	"VF101_72.VCF"	"VS101_72.VCF"	"VV101_72.VCF"	"VT101_72.VCF"
800x600 at 60 Hz	"VM103_60.VCF" or "800x600x8PP"	"VF103_60.VCF" or "800x600x15PP"	"VS103_60.VCF" or "800x600x16PP"	"VV103_60.VCF" or "800x600x24PP"	"VT103_60.VCF" or "800x600x32PP"
800x600 at 72 Hz	"VM103_72.VCF"	"VF103_72.VCF"	"VS103_72.VCF"	"VV103_72.VCF"	"VT103_72.VCF"
1024x768 at 60 Hz	"VM105_60.VCF" or "1024x768x8PP"	"VF105_60.VCF" or "1024x768x15PP"	"VS105_60.VCF" or "1024x768x16PP"	"VV105_60.VCF" or "1024x768x24PP"	"VT105_60.VCF" or "1024x768x32PP"
1024x768 at 70 Hz	"VM105_70.VCF"	"VF105_70.VCF"	"VS105_70.VCF"	"VV105_70.VCF"	"VT105_70.VCF"
1152x864 at 60 Hz	"CM001_60.VCF" or "1152x864x8PP"	"CF001_60.VCF" or "1152x864x15PP"	"CS001_60.VCF" or "1152x864x16PP"	"CV001_60.VCF" or "1152x864x24PP"	"CT001_60.VCF" or "1152x864x32PP"
1152x864 at 70 Hz	"CM001_70.VCF"	"CF001_70.VCF"	"CS001_70.VCF"	"CV001_70.VCF"	"CT001_70.VCF"

Display Resolution	DispFormat			
	8-bit	15-bit	16-bit	24-bit
1280x1024 at 60 Hz	"VM107_60.VCF" or "1280x1024x8PP"	"VF107_60.VCF" or "1280x1024x15PP"	"VS107_60.VCF" or "1280x1024x16PP"	"VV107_60.VCF" OR "1280X1024X24PP"
1280x1024 at 75 Hz	"VM107_75.VCF"	"VF107_75.VCF"	"VS107_75.VCF"	"VV107_75.VCF"
1600x1200 at 60 Hz	"VM11C_60.VCF" or "1600x1200x8PP"	"VF11C_60.VCF" or "1600x1200x15PP"	"VS11C_60.VCF" or "1600x1200x16PP"	

See the \MIL (user-specified) directory for the current list of video configuration format (VCF) files.

MdispControl()

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

ControlType	Description & ControlValue	
M_ENCODER	Set the functional state of the encoder. The encoder is automatically enabled when the VGA display mode is compatible with the encoder. See "Using the encoder" at the beginning of this chapter.	
	M_ENABLE (DEFAULT)	
	M_DISABLE	
M_ENCODER_FILTER	Select a filter for luminance.	
	M_LOW_PASS_TYPE_A	Low-pass filter type A.
	M_FILTER_NOTCH	Subcarrier frequency low-pass filter.
	M_FILTER_EXTENDED	5.5 MHz low-pass filter.
	M_FILTER_LOW_PASS_TYPE_B	Low-pass filter type B.
	M_DEFAULT	Same as
	M_FILTER_LOW_PASS_TYPE_A	
M_ENCODER_MODE	Set the encoder output format:	
	M_COMPOSITE or M_YC (DEFAULT)	Selecting either ControlValue will produce both outputs.
	M_RGB	RGB outputs.
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_ENCODER_RGB_SYNC	Set the RGB output with the sync information encoded. This control is only supported when the encoder is set to RGB mode.	
	M_ENABLE	The sync information is encoded in the output.
	M_DISABLE	The encoder output does not contain a sync.
	M_DEFAULT	Same as M_ENABLE.
M_ENCODER_SYNC_SOURCE	Select the encoder synchronization source:	
	M_DEFAULT	Uses the most appropriate synchronization source.

ControlType	Description & ControlValue	
M_ENCODER_SYNC_SOURCE (cont.)	M_MASTER	Uses the internal synchronization signal. This control value cannot be selected when grabbing from the decoder path.
	M_SLAVE	Uses the synchronization signal of the camera. The camera must be of the same format as the display (for example an NTSC/PAL camera with an NTSC/PAL display).

In windowed mode, the **ControlType** parameter can also be set to:

ControlType	Description	
M_HARDWARE_PAN	Set whether to use Matrox Corona's hardware pan options, or the software pan options of the display's window.	
	M_ENABLE	Use hardware pan options.
	M_DISABLE	Use software pan options (default).
M_HARDWARE_ZOOM	Set whether to use Matrox Corona's hardware zoom options, or the software zoom options of the display's window.	
	M_ENABLE	Use hardware zoom options.
	M_DISABLE	Use software zoom options (default).

MdisInquire()

- 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	Returns the functional state of the encoder: M_ENABLE or M_DISABLE.
M_ENCODER_FILTER	Returns the filter selected for luminance. M_LOW_PASS_TYPE_A, M_FILTER_NOTCH, M_FILTER_EXTENDED, or M_FILTER_LOW_PASS_TYPE_B
M_ENCODER_MODE	Returns the encoder output format: M_COMPOSITE, M_YC, or M_RGB.
M_ENCODER_PEDESTAL	Returns whether a pedestal is to be generated in the composite video signal. M_ENABLE or M_DISABLE
M_ENCODER_RGB_SYNC	Returns whether the RGB output has the synchronization information encoded. M_ENABLE or M_DISABLE
M_ENCODER_SYNC_SOURCE	Returns the selected encoder synchronization source: M_MASTER or M_SLAVE

- In windowed mode, the **InquireType** parameter can also be set to:

InquireType	Description
M_THREAD_PRIORITY	Priority of the thread used for display (created internally).
M_HARDWARE_PAN	The current hardware pan state.
M_HARDWARE_ZOOM	The current hardware zoom state.

MdispLut()

- If a display has been allocated with an M_WINDOWED or M_DEFAULT initialization flag in single-screen mode:
 - In a 256-color display resolution, you can only associate a LUT buffer with overlay/regular displays (allocated with the M_OVR attribute).
 - In the case of other color resolutions, you can associate a LUT buffer with all overlay/regular displays or one underlay display. You can associate a LUT buffer with all overlay/regular displays because the LUT is handled by

MIL/Windows (not physical hardware). However, you can associate a LUT buffer only with one underlay display at a time because, in this case, the LUT is handled by the physical hardware. That is, the LUT buffer data is loaded directly into the physical LUTs. Therefore, if you associate a LUT buffer with a second underlay display, the first underlay display will lose its association with its LUT. Accordingly, you will lose the effect of the LUT that was associated with the first display.

- If a display has been allocated with an `M_NON_WINDOWED` initialization flag in single-screen mode:
 - In a 256-color display resolution, you can associate a monochrome (1-band x 8-bit x 256 entries) or a pseudo-color (3-band x 8-bit x 256 entries) LUT buffer with the display (overlay or underlay).

- If a display has been allocated with an `M_WINDOWED` initialization flag in dual-screen mode:
 - You can associate a monochrome (1-band x 8-bit x 256 entries) or a pseudo-color (3-band x 8-bit x 256 entries) LUT buffer with the display, regardless if a 1-band or 3-band image buffer has been selected to the display, and regardless if you are displaying from the overlay or underlay surface of the VGA driving your desktop.

The driver does not use the physical LUT to map the data; MIL/Windows handles everything.

- If a display has been allocated with an `M_NON_WINDOWED` or `M_DEFAULT` initialization flag in dual-screen mode:
 - You can associate a monochrome (1-band x 8-bit x 256 entries) or a pseudo-color (3-band x 8-bit x 256 entries) LUT buffer with the display, regardless of the frame buffer surface from which it is being displayed (underlay or overlay) and regardless of the number of bands of the image buffer being displayed.
 - A LUT buffer cannot be associated with an overlay\regular and underlay display at the same time. The restriction occurs because the physical output LUTs

on Matrox Corona can be used to map data from either the overlay or underlay (main) frame buffer surface, but not both.

To associate a LUT buffer with another display in the other surface (overlay or underlay) follow these steps to overwrite the physical output LUT's contents:

- a. Call **MdispLut()** with the old display ID and the M_DEFAULT LUT buffer.
- b. Call **MdispLut()** with the new display ID and the required LUT buffer.

MdispOverlayKey()

- When in a 15-bit display mode, the keying color must always be a multiple of 8 for all components. In addition, all pixel values in the overlay that are not a multiple of 8 will be rounded down to the closest multiple of 8 before being compared to the keying color.

For example, when specifying a keying color of:

```
Red   = 0x70
Green = 0x58
Blue  = 0x60
```

The overlay buffer will be transparent for the range:

```
Red   = [0x70-0x77]
Green = [0x58-0x5F]
Blue  = [0x60-0x68]
```

- When in a 16-bit display mode, the same rules apply except that the green keying color must be a multiple of 4 and the range of transparency on green is equal to 4.

MdispPan()

- Software pan is fully supported in windowed mode.
- The following restrictions apply on the Matrox Corona board, when performing a hardware pan (in non-windowed mode, this is the only type of panning that is supported. In windowed mode, you can enable hardware panning with ***MdispControl()***):
 - The hardware pan (**XOffset**) must be a multiple of 4 (for example, 0, 4, 8, 12, or 16).
 - Any region beyond the image buffer's boundaries cannot be displayed.
 - Although panning is supported in single screen mode on a Matrox Corona with a display section, the display follows that of the VGA; that is, when you pan the Matrox Corona display, the VGA is panned as well, and vice versa.

MdispZoom()

- Zooming is fully supported on a display in windowed mode.
- The following restrictions apply on the Matrox Corona board, when performing a hardware zoom (in non-windowed mode, this is the only type of zoom that is supported. In windowed mode, you can enable hardware zoom with ***MdispControl()***):
 - In the x and y direction, only hardware zoom factors of 1, 2, and 4 are supported.
 - Although hardware zoom is supported in single screen mode, it follows that of the VGA, that is, if you select hardware zoom (using the ***MdispControl()*** function), the VGA shares the same zoom, and vice versa.
 - If you are using single-screen mode under Windows:
 - You cannot use the zoom factor of 4 in the 640 x 480 and 800 x 600 resolutions.
 - Due to hardware limitations, when using the zoom factor of 4 with a 1152 x 882 x 8 resolution, a 32-pixel band of spurious data appears at the right border of the screen.

- In multi-head mode, only the screen where the cursor is positioned appears zoomed. To zoom the other screen, move the cursor to that screen.

MsysAlloc()

- To allocate a Corona system, you must select M_SYSTEM_CORONA as the **SystemTypePtr** parameter. This selection opens communication with the board and will automatically allow the system to use some Host memory, if necessary.

MsysControl()

- To modify the transfer rate and performance of the PCI bus, the Corona system supports an additional combination for the **ControlType** and **ControlValue** parameters:

ControlType	Description and ControlValue
M_PCI_LATENCY	Controls the length of the PCI bus transfer-time slice that is allocated to Matrox Corona. Increasing it can improve transfer time when there is heavy traffic on the PCI bus. However, you should be aware that this can affect the performance of the other bus masters (including the main CPU). Possible values are: 0 - 127. Use MsysInquire() to determine the default value on your system. It is recommended that the chosen value be one near the default value.

- MIL can perform a continuous grab operation live in the overlay (VGA) frame buffer when the display is in windowed mode. When necessary, the grab will switch to pseudo-live (simulating a live grab by grabbing into the Host buffer and updating the display) to prevent the grab from overwriting another window. If there is an instance when automatic live-to-pseudo-live switching does not happen or you want to override the default behavior, you can use the following **ControlType** and **ControlValue** parameters settings.

When grabbing into the underlay, the following control types should be left to their default setting.

ControlType	Description and ControlValue	
M_LIVE_GRAB	Set whether to perform a live grab whenever possible or to force a pseudo-live grab into displayable image buffers:	
	M_ENABLE	Live whenever possible (default).
	M_DISABLE	Force pseudo-live.
M_STOP_LIVE_GRAB_WHEN_MENU	Set whether or not to switch to a pseudo-live grab while an opened menu overlaps the display window:	
	M_ENABLE	Pseudo-live while menu overlaps the display window (default).
	M_DISABLE	Force live.
M_STOP_LIVE_GRAB_WHEN_INACTIVE	Set whether or not to switch to a pseudo-live grab while the display window is inactive (that is, while it does not have the focus):	
	M_ENABLE	Pseudo-live while the display window is inactive (default).
	M_DISABLE	Force live.
M_STOP_LIVE_GRAB_WHEN_DISABLED	Set whether or not to switch to a pseudo-live grab while the display window is disabled (for example, when a pop-up dialog box is opened):	
	M_ENABLE	Pseudo-live while the display window is disabled (default).
	M_DISABLE	Force live.
M_PSEUDO_LIVE_GRAB_WHEN_OVERLAPPED	Set whether to pause the grab or switch to a pseudo-live grab when the live grab is interrupted due to one of the above-mentioned conditions (for example, if the window displaying the grab is overlapped by a menu).	
	M_ENABLE	Pseudo-live (default).
	M_DISABLE	Pause grab.

- When the display is in windowed mode (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.

- In multi-head mode, you can force pseudo-live grabbing when a grab is displayed from a board other than the one performing the grab. To do so, set the following **ControlType** to M_ENABLE:

ControlType	ControlValue	
M_FORCE_PSEUDO_IN_NON_UNDERLAY_DISPLAYS	Can be set to:	
	M_ENABLE	Force pseudo.
	M_DISABLE	Grab live whenever possible (default).

- 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 grab window is displaced, the grab is stopped and restarted at every displacement. When grabbing from a triggered camera, a trigger is probably not issued as often as the window is displaced. To avoid having an empty window, a copy is performed from the old location to the new before restarting the grab. To override this default, set the following **ControlType** to M_DISABLE:

ControlType	ControlValue	
M_LIVE_GRAB_MOVE_UPDATE	Can be set to:	
	M_ENABLE	Perform a copy between the windows. Default for triggered cameras.
	M_DISABLE	Don't perform a copy between the windows. Default for non-triggered cameras.

- 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	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.

- During a live grab operation in windowed mode, the driver keeps track of the grab window. This default can be overridden by setting the following **ControlType** to M_DISABLE:

ControlType	ControlFlag
M_LIVE_GRAB_TRACK	Can be set to:
	M_ENABLE (default) or M_DISABLE.

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_CORONA OR M_CORONA_LC OR M_CORONA_WITH_DIG_MODULE.
M_DIGITAL_MODULE_PRESENT	Returns M_TRUE if a digital grab module is present, otherwise it returns M_FALSE.
M_DISPLAY_DOUBLE_BUFFERING	The state of double buffering.
M_FORCE_PSEUDO_IN_NON_UNDERLAY_DISPLAYS	Pseudo-live grab performed in non-Corona-driven screen: M_ENABLE or M_DISABLE.
M_LAST_GRAB_IN_TRUE_BUFFER	Whether the last grab is done to the true buffer at the end of a continuous grab: M_ENABLE or M_DISABLE.

InquireType	Description
M_LIVE_GRAB	Whether a live grab (not pseudo-live) is performed: M_ENABLE or M_DISABLE.
M_LIVE_GRAB_END_TRIGGER	Whether an automatic trigger is generated at end of grab: M_ENABLE or M_DISABLE.
M_LIVE_GRAB_MOVE_UPDATE	Whether a copy operation is performed between windows on update: M_ENABLE or M_DISABLE.
M_LIVE_GRAB_TRACK	Whether or not MIL keeps track of the live-grab window: M_ENABLE or M_DISABLE.
M_PCI_LATENCY	The length of the PCI bus transfer-time slice.
M_PHYSICAL_ADDRESS_UNDERLAY	The physical address of the underlay frame buffer.
M_PHYSICAL_ADDRESS_VGA	The physical address of the VGA frame buffer.
M_PSEUDO_LIVE_GRAB_WHEN_OVERLAPPED	Whether a switch is made to a pseudo-live grab when the display window is overlapped by another window: M_ENABLE or M_DISABLE.
M_RGB_MODULE_NUM	Number of system RGB modules (0 or 1).
M_STOP_LIVE_GRAB_WHEN_DISABLED	Whether the grab is frozen when the display window is disabled: M_ENABLE or M_DISABLE.
M_STOP_LIVE_GRAB_WHEN_INACTIVE	Whether the grab is frozen when the display window is inactive: M_ENABLE or M_DISABLE.
M_STOP_LIVE_GRAB_WHEN_MENU	Whether the grab is frozen when the display window is overlapped by a menu: M_ENABLE or M_DISABLE.

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.

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 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). In addition, it has a 2 Mbyte graphics overlay (VGA) frame buffer for non-destructive overlay capabilities. Matrox Genesis can use the underlay display architecture.

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.

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.

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 is 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.

When using a Processor Board, or when using a node that cannot access 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 `\MIL` (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 in windowed mode, 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 in a dual-screen configuration.

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 (in single-screen mode) 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.

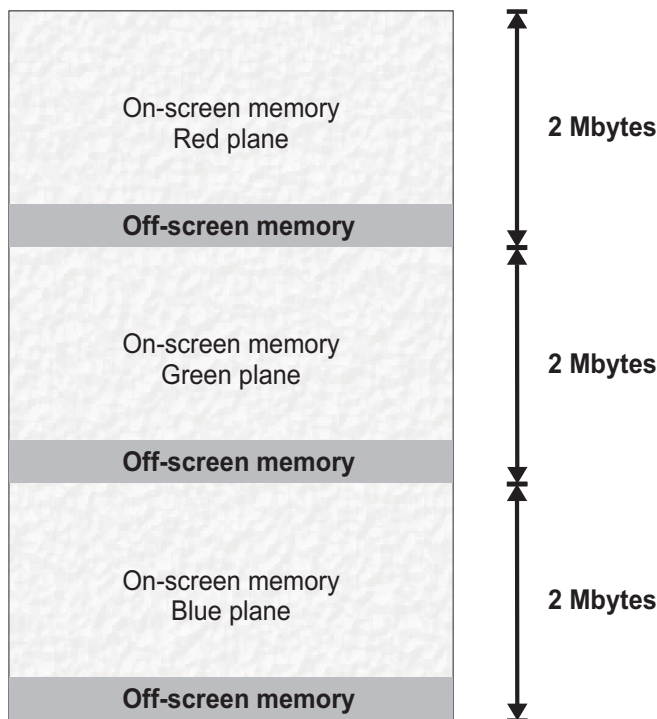
Increasing the default temporary buffer size

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***(0) 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*.

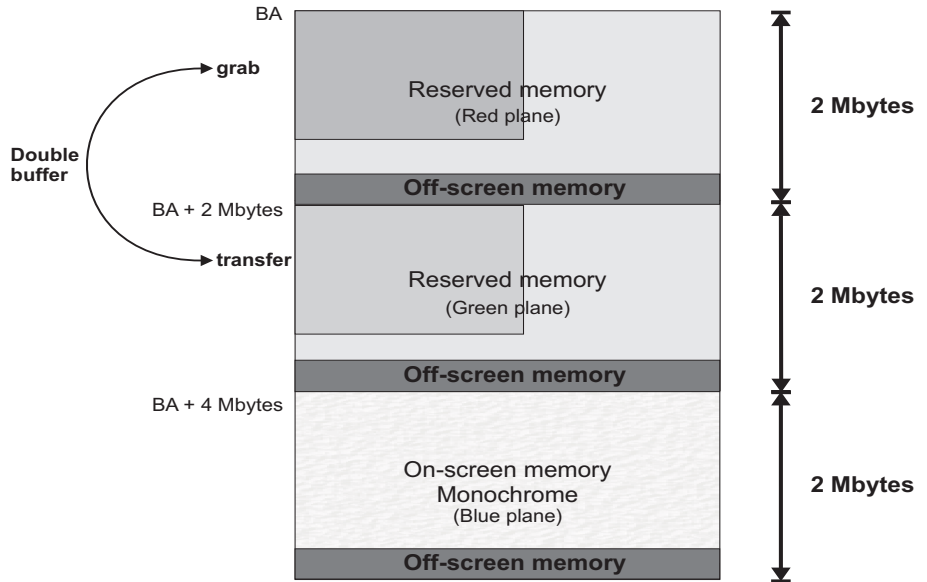
Monochrome display

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 functionality 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()	Camera format file 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.

Commands	Matrox Genesis particularities
MdigLut()	Input LUT buffer size requirements.
MdispAlloc()	Display format. Overlay setting.
MdispControl()	Matrox Genesis-specific display control options.
MdispInquire()	Matrox Genesis-specific inquire options.
MdispLut()	Available only in dual-screen mode.
MdispPan()	Panning particularities.
MdispZoom()	Supported zoom factors.
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.
- Buffers with the **Attribute** parameter set to M_GRAB 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_RGB24+M_PLANAR	24-bit planar (RGB) 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 to be displayed in non-windowed mode, 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.

- Although not commonly done, you can display a buffer in a display that has been allocated only in the overlay frame buffer. MIL allocates these displayable buffers on the Host, and then copies them to the overlay (VGA) frame buffer when selected (**MdispSelect()**) on displays allocated with M_OVR **InitFlag** (**MdispAlloc()**).

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. ArrayOfDataPtr will be a pointer to an array of MIL buffer identifiers. One identifier per band must be provided. SizeX , SizeY , Type , Attribute and Pitch must be the same for each buffer. Pitch 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_RGB24+M_PLANAR	24-bit planar (RGB) 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 of 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).

- In windowed mode, 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_DIRECTION_X	Set the horizontal grab direction:	
	M_REVERSE	Flip the grabbed image horizontally.
	M_FORWARD	Grab normally in the horizontal direction.
	M_DEFAULT	Same as M_FORWARD.
M_GRAB_DIRECTION_Y	Set the vertical grab direction. 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.	
	M_REVERSE	Flip the grabbed image vertically
	M_FORWARD	Grab normally in the vertical direction.
	M_DEFAULT	Same as M_FORWARD.

ControlType	Description & ControlValue		
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 voltages 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.	
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).	
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 MdigControl() with M_GRAB_EXPOSURE + M_TIMER N and M_ACTIVATE is called.	

ControlType	Description & ControlValue	
M_GRAB_EXPOSURE_SOURCE (cont.)	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
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 ControlValue 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.	

ControlType	Description & ControlValue	
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_THREAD_PRIORITY	Set the hook function priority under Windows. Valid values are:	
	11 - 15	High priority.
	16 or 22 - 26	Real-time priority.
	If you want to change the hook function priority, use MdigInquire() to determine its current value. Changing these priority settings will affect the overall application priority, due to a Windows restriction.	
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 <i>.dcf</i> .
	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 <i>*.dcf</i> .
	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. ControlFlag 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. ControlValue 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()

- In windowed mode, M_GRAB_SCALE... can also return: M_FILL_DESTINATION or M_FILL_DISPLAY.
- The **InquireType** parameter can also be set to the following:

InquireType	Description
M_GRAB_DIRECTION_X	Horizontal grab direction: M_REVERSE OR M_FORWARD.
M_GRAB_DIRECTION_Y	Vertical grab direction: M_REVERSE or M_FORWARD.
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).

InquireType	Description
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.
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_GRAB_THREAD_ID	Thread identifier for all grab interrupt handlers.
M_GRAB_THREAD_HANDLE	Thread handle for all grab interrupt handlers.
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_THREAD_PRIORITY	Hook function priority under Windows:
	11 - 15 High priority.
	16 or 22 - 26 Real-time priority.
M_USER_BIT+(0,1)	Current state of the input bit of the external I/O port: M_ON (1) or M_OFF (0).
	The relationship between the MIL user-bit number and the actual user input on the grab module digital port is as follows.
	Companion digital input board:
	bit 0: USER0 INPUT (RS-422 only)
	bit 1: USER1 INPUT (RS-422 only)
	Digital cable adapter:
	bit 0: USER0 INPUT (TTL or RS-422)
M_USER_IN_FORMAT	Type of receiver for digital I/Os (M_TTL, M_RS422, or M_DISABLE).

InquireType	Description	
M_USER_OUT_FORMAT	Type of driver for digital I/Os (M_TTL, M_RS422, OR M_DISABLE).	
	bit 1:	USER1 INPUT (TTL or RS-422)

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()

- When operating under Windows in single-screen mode, set **DispFormat** to "M_DEFAULT". This sets the main (underlay) frame buffer's display format to that of the on-board overlay (VGA) frame buffer display.

When operating under Windows in dual-screen mode, set **DispFormat** to the required display resolution for the VGA display. Possible settings are:

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.

MdispControl()

- When 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_HARDWARE_PAN	M_ENABLE	Use your system's hardware pan options.
	M_DISABLE	Use the software pan options of the display's window. (default).

Control Type	Control Value and Description	
M_HARDWARE_ZOOM	M_ENABLE	Use your system's hardware zoom options.
	M_DISABLE	Use the software zoom options of the display's window. (default).

Mdisplnquire()

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

InquireType	Description
M_COLOR_MODE	The display color mode (<i>MdispControl()</i>).
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.
M_HARDWARE_PAN	Whether your system's hardware pan options are enabled or disabled (default).
M_HARDWARE_ZOOM	Whether your system's hardware zoom options are enabled or disabled (default).

MdispLut()

- A LUT can only be associated with a display that has the M_OVR attribute.
- If a display was allocated with the M_OVR attribute, using *MdispAlloc()*, 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.

MdispPan()

- Software panning is fully supported in windowed mode.
- The following restrictions apply on the Matrox Genesis board, when performing a hardware pan (in non-windowed mode, this is the only type of panning that is supported. In windowed mode, you can enable hardware panning with *MdispControl()*):
 - The hardware pan (**XOffset**) must be a multiple of 4 (for example, 0, 4, 8, 12, or 16).
 - Any region beyond the image buffer's boundaries cannot be displayed.

- Although panning is supported in single screen mode on a Matrox Genesis with a display section, the display follows that of the VGA; that is, when you pan the Matrox Genesis display, the VGA is panned as well, and vice versa.

MdispZoom()

- Software zooming is fully supported on a display in windowed mode.
- The following restrictions apply on the Matrox Genesis board, when performing a hardware zoom (in non-windowed mode, this is the only type of zoom that is supported. In windowed mode, you can enable hardware zoom with ***MdispControl()***):
 - In the x and y direction, only hardware zoom factors of 1, 2, and 4 are supported.
 - Although hardware zoom is supported in single screen mode, it follows that of the VGA, that is, if you select hardware zoom (using the ***MdispControl()*** function), the VGA shares the same zoom, and vice versa.
 - If you are using single-screen mode under Windows:
 - You cannot use the zoom factor of 4 in the 640 x 480 and 800 x 600 resolutions.
 - Due to hardware limitations, when using the zoom factor of 4 with a 1152 x 882 x 8 resolution, a 32-pixel band of spurious data appears at the right border of the screen.
 - In multi-head mode, only the screen where the cursor is positioned appears zoomed. To zoom the other screen, move the cursor to that screen.

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 <i>MimWarp()</i> with the address (0, 0). If you use this define, the size of the source buffer is required; specify the x-size with the Val1 parameter and the y-size with the Val2 parameter. To use the same x- and/or y-size as the LUT buffers, set Val1 and/or Val2 to M_NULL or M_DEFAULT.
M_OVERSCAN_ENABLE	If addresses fall outside of the source buffer of <i>MimWarp()</i> , 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.

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 **SystemTypePtr** parameter. This selection opens communication with the board or the specified node, and will allow MIL to use its resources.

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	
	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, ControlValue 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, ControlValue 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.

- MIL can perform a continuous grab operation live in the overlay (VGA) frame buffer when the display is in windowed mode. When necessary, the grab will switch to pseudo-live (simulating a live grab by grabbing into the Host buffer and updating the display) to prevent the grab from overwriting another window. If there is an instance when automatic live-to-pseudo-live switching does not happen or you want to override the default behavior, you can use the following

ControlType and **ControlValue** parameter settings. When grabbing into the underlay, the following control types should be left to their default setting.

ControlType	ControlValue & Description	
M_LIVE_GRAB	Set whether to perform a live grab whenever possible or to force a pseudo-live grab into displayable image buffers:	
M_PSEUDO_LIVE_GRAB_WHEN_OVERLAPPED	Set whether to pause the grab or switch to a pseudo-live grab when the live grab is interrupted due to one of the above-mentioned conditions (for example, if the window displaying the grab is overlapped by a menu).	
	M_ENABLE	Pseudo-live (default).
	M_DISABLE	Pause grab.
	M_ENABLE	Live whenever possible (default).
	M_DISABLE	Force pseudo-live
M_STOP_LIVE_GRAB_WHEN_DISABLED	Set whether or not to switch to a pseudo-live grab while the display window is disabled (for example, when a pop-up dialog box is opened):	
	M_ENABLE	Pseudo-live while the display window is disabled (default).
	M_DISABLE	Force live.
M_STOP_LIVE_GRAB_WHEN_INACTIVE	Set whether or not to switch to a pseudo-live grab while the display window is inactive (that is, while it does not have the focus):	
	M_ENABLE	Pseudo-live while the display window is inactive (default).
	M_DISABLE	Force live.
M_STOP_LIVE_GRAB_WHEN_MENU	Set whether or not to switch to a pseudo-live grab while an opened menu overlaps the display window:	
	M_ENABLE	Pseudo-live while menu overlaps the display window (default).
	M_DISABLE	Force live.

- When the display is in windowed mode (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.

- In multi-head mode, you can force pseudo-live grabbing when a grab is displayed from a board other than the one performing the grab. To do so, set the following **ControlType** to M_ENABLE:

ControlType	ControlValue	
M_FORCE_PSEUDO_IN_NON_UNDERLAY_DISPLAYS	Can be set to:	
	M_ENABLE	Force pseudo
	M_DISABLE	Grab live whenever possible (default).

- 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 grab window is displaced, the grab is stopped and restarted at every displacement. When grabbing from a triggered camera, a trigger is probably not issued as often as the window is displaced. To avoid having an empty window,

a copy is performed from the old location to the new before restarting the grab. To override this default, set the following **ControlType** to M_DISABLE:

ControlType	ControlValue	
M_LIVE_GRAB_MOVE_UPDATE	Can be set to:	
	M_ENABLE	Perform a copy between the windows. Default for triggered cameras.
	M_DISABLE	Don't perform a copy between the windows. Default for non-triggered cameras.

- 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.

- During a live grab operation in windowed mode, the MIL driver keeps track of the grab window. This default can be overridden by setting the following **ControlType** to **M_DISABLE**:

ControlType	ControlFlag
M_LIVE_GRAB_TRACK	Can be set to: M_ENABLE (default) or M_DISABLE .

MsysInquire()

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

InquireType	Description
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 , M_GENESIS_WITH_DAC_MODULE , or M_GENESIS_LC_WITH_DAC_MODULE .
M_DISPLAY_DOUBLE_BUFFERING	The state of double buffering.
M_FORCE_PSEUDO_IN_NON_UNDERLAY_DISPLAYS	Pseudo-live grab performed in non-Genesis-driven screen: M_ENABLE or M_DISABLE .
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	A live grab (not pseudo-live) is performed: 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_LIVE_GRAB_TRACK	Whether or not MIL keeps track of the live-grab window: M_ENABLE or M_DISABLE .

InquireType	Description
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_PSEUDO_LIVE_GRAB_WHEN_OVERLAPPED	A switch is made to a pseudo-live grab when the display window is overlapped by another window: M_ENABLE or M_DISABLE.
M_SERIAL_NUMBER	The serial number of the board.
M_STOP_LIVE_GRAB_WHEN_DISABLED	Grabbing is frozen when the display window is disabled: M_ENABLE or M_DISABLE.
M_STOP_LIVE_GRAB_WHEN_INACTIVE	Grabbing is frozen when the display window is inactive: M_ENABLE or M_DISABLE.
M_STOP_LIVE_GRAB_WHEN_MENU	Grabbing is frozen when the display window is overlapped by a menu: M_ENABLE or M_DISABLE.
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 operation will be performed by the Host (by default), instead of the on-board processor:

Operation	Conditions under which Host performs the operation
<i>MblobFill()</i> (M_INCLUDED_BLOB, M_EXCLUDED_BLOB, OR M_ALL_BLOB) + M_CONTOUR	<ul style="list-style-type: none"> ■ All operations.
<i>MblobCalculate(), MblobGetResult(), MblobGetResultSingle()</i> M_CHAINS, M_CHAIN_X, M_CHAIN_Y, M_CHAIN_INDEX, OR M_NUMBER_OF_CHAINED_PIXELS Any feature + (M_SORT1_UP, M_SORT1_DOWN, M_SORT2_UP, M_SORT2_DOWN, M_SORT3_UP, M_SORT3_DOWN, OR M_NO_SORT)	<ul style="list-style-type: none"> ■ All operations ■ All operations
<i>MdigFocus()</i>	<ul style="list-style-type: none"> ■ Depth of a 1-bit destination buffer.
<i>MgenWarpParameter()</i>	<ul style="list-style-type: none"> ■ 32-bit LUT buffers.
<i>MimArith()</i> (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"> ■ Floating-point buffers. ■ Mix of 32-bit buffer types.
(M_DIV or M_DIV_CONST) + M_FIXED_POINT	<ul style="list-style-type: none"> ■ Either source is 32-bit. ■ Source is 8-bit and destination is 32-bit.
M_CONST_DIV + M_FIXED_POINT	<ul style="list-style-type: none"> ■ All operations.

Operation	Conditions under which Host performs the operation
<i>MimArithMultiple()</i> M_OFFSET_GAIN or M_WEIGHTED_AVERAGE	<ul style="list-style-type: none"> ■ 32-bit buffers. ■ One of the buffers is signed. ■ Destination buffer's depth is equal to or less than that of the source buffer. ■ Source buffer of different types.
M_MULTIPLY_ACCUMULATE_1	■ 32-bit integer buffers.
M_MULTIPLY_ACCUMULATE_1 + M_SATURATION	■ Float buffers.
M_MULTIPLY_ACCUMULATE_2	<ul style="list-style-type: none"> ■ 32-bit integer buffers. ■ Source buffers of different signs.
M_MULTIPLY_ACCUMULATE_2 + M_SATURATION	■ Float buffers.
<i>MimBinarize(), MimClip(), MimCountDifference(), MimLocateEvent()</i>	■ Float buffers.
<i>MimConvert()</i> M_RGB_TO_Y	■ All operations.
<i>MimConnectMap(), MimDilate(), MimDistance(), MimErode(), MimThin(), MimThick(), and MimLabel()</i>	■ Buffers of depth greater than 16 bits.

Operation	Conditions under which Host performs the operation
<i>MimConvolve()</i>	<ul style="list-style-type: none"> ■ Source or destination buffer of depth 1 bit or greater than 16 bits. ■ Float buffers. ■ The neighborhood operation accelerator (NOA) is disabled and the kernel is larger than 15x15. <ul style="list-style-type: none"> □ The maximum kernel size can be as large as 31x31 if all the following conditions are met: <ul style="list-style-type: none"> ■ Source buffer is unsigned 8-bit. ■ Destination buffer is 8- or 16-bit. ■ Kernel is signed 8-bit [-128, +127]. ■ The NOA is enabled and any of the following conditions is met: <ul style="list-style-type: none"> □ The kernel width or height is greater than 33. □ 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.
<i>MimConvolve()</i> (cont.)	<ul style="list-style-type: none"> ❖ Here are some exceptions to the above rule: <ul style="list-style-type: none"> ■ 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). ■ If the kernel is 1xN (that is, the width = 1), then the maximum kernel height is 64.
<i>MimEdgeDetect()</i>	<ul style="list-style-type: none"> ■ All modes except M_FAST_EDGE_DETECT, M_FAST_ANGLE AND M_FAST_GRADIENT modes. ■ Buffers of depth greater than 16 bits.

Operation	Conditions under which Host performs the operation
<i>MimFindExtreme()</i>	<ul style="list-style-type: none"> ■ Float buffers.
<i>MimFlip()</i>	<ul style="list-style-type: none"> ■ Float buffers. ■ Source and destination have different depths.
<i>MimHistogram()</i>	<ul style="list-style-type: none"> ■ Float buffers. ■ Source with depth greater than 16 bits.
<i>MimLutMap()</i>	<ul style="list-style-type: none"> ■ Float buffers. ■ 32-bit source buffer. ■ 32-bit signed destination buffer.
<i>MimMorphic()</i> All cases	<ul style="list-style-type: none"> ■ Buffers with depth greater than 16 bits.
M_ERODE, M_DILATE	<ul style="list-style-type: none"> ■ Structuring elements exceeding 15x15 in size.
M_THIN, M_THICK	<ul style="list-style-type: none"> ■ Grayscale structuring elements that are not 3x3 in size. ■ Binary structuring elements exceeding 15x15 in size.
M_HIT_OR_MISS	<ul style="list-style-type: none"> ■ Grayscale mode. ■ Structuring elements exceeding 15x15 in size.
M_MATCH	<ul style="list-style-type: none"> ■ Grayscale mode.
<i>MimPolarTransform()</i>	<ul style="list-style-type: none"> ■ All operations.
<i>MimProject()</i>	<ul style="list-style-type: none"> ■ Float buffers. ■ 32-bit source buffers.
<i>MimRank()</i>	<ul style="list-style-type: none"> ■ Any structuring elements that are not predefined. ■ When the rank is not minimum, maximum, or M_MEDIAN. ■ Buffers of depth greater than 16 bits.

Operation	Conditions under which Host performs the operation
<i>MimResize()</i> and <i>MimRotate()</i>	<ul style="list-style-type: none"> ■ Float buffers. ■ When the interpolation mode is either M_BILINEAR or M_BICUBIC and you are using 32-bit source buffers. ■ Source buffer is 32-bit and destination buffer is 1-bit or source buffer is 1-bit and destination is 32-bit.
M_OVERSCAN_DISABLE	<ul style="list-style-type: none"> ■ All operations
<i>MimShift()</i>	<ul style="list-style-type: none"> ■ Float buffers.
<i>MimTransform()</i>	<ul style="list-style-type: none"> ■ 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).
<i>MimTranslate()</i>	<ul style="list-style-type: none"> ■ Fractional X or Y displacement when source and/or destination buffer are of depth 1 bit or greater than 16 bits.
<i>MimWarp()</i>	<ul style="list-style-type: none"> ■ Float buffers. ■ When the interpolation mode is either M_BILINEAR or M_BICUBIC and you are using 32-bit source buffers. ■ Source buffer is 32-bit and destination buffer is 1-bit or source buffer is 1-bit and destination is 32-bit.
M_WARP_LUT	<ul style="list-style-type: none"> ■ Interpolation is M_BICUBIC.
M_OVERSCAN_DISABLE	<ul style="list-style-type: none"> ■ All operations.
<i>MimWatershed()</i>	<ul style="list-style-type: none"> ■ All operations.

Chapter 4: MIL and the Matrox Meteor-II platform

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

Matrox Meteor-II family

The Matrox Meteor-II family is comprised of four acquisition boards: Matrox Meteor-II /Standard, Matrox Meteor-II /Multi-Channel, Matrox Meteor-II /Digital, and Matrox Meteor-II /1394. 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, software triggers, exposure timers, and asynchronous reset mode (discussed below table).

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

Matrox Meteor-II	STD	MC	DIG	1394
■ Form factor(s):				
□ PCI	x	x	x	x
□ CompactPCI	x			
□ PC-104/Plus	x	x		
■ Uses standard VGA for display	x	x	x	x
■ On-board UART (PCI form factor)	x	x	x	
■ VMChannel (PCI form factor)	x	x	x	
■ Supports mono-tap and dual-tap camera configuration		x	x	
■ Supports multi-tap camera configuration			x	
■ Supports hardware triggers	x	x	x	x
■ Supports software triggers	x	x	x	
■ Has 2 exposure timers		x	x	
■ Supports asynchronous reset mode		x	x	
■ Can support a Matrox Meteor-II MJPEG compression module (all form factors)	x	x		
■ Supports Windows 98	x	x		
■ Supports square pixels and CCIR resolutions	x			

UART

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. See the *Compression/Decompression with Matrox Meteor-II MJPEG* section.

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. However, under Windows 2000 the Matrox Meteor-II /1394 driver does not support "plug and play".

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
■ Meteor-II /Standard	monochrome RS-170/CCIR	12	7	12
	composite NTSC/PAL	12	3	12
	Y/C	6	3	6
Note, Rev. 1 of Matrox Meteor-II /Standard supports only 4 monochrome or 2 Y/C cameras.				
■ Meteor-II /Multi-Channel	standard/ non-standard monochrome	6		12
	RGB	2		6
■ Meteor-II /Digital	digital monochrome	4		
	digital RGB	1		
■ Meteor-II /1394	digital IEEE 1394 standard	>=1*		
*Note, more than one independent IEEE 1394 Digital Camera Specification (DCS)-compliant camera can be attached to the Matrox Meteor-II /1394 board, but each 1394 DSC-based 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,...). This establishes a MIL system environment in which MIL not only uses the Matrox Meteor-II board and Host computer memory, but also the VGA board. The VGA board has been included to allow you to display grabbed images (none of the Matrox Meteor-II boards has an on-board display section).

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 DCS-based cameras".

This chapter relates Meteor-II systems to the following:

Transfers to display, Grabbing to a Host buffer, 1394-specific features, and Particularities of existing MIL functions on Matrox Meteor-II.

Refer to *milmet2.txt* file in the \MIL (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 and G400 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

*The Meteor-II /Digital 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
Cyrix companion chip (on Matrox 4Sight)	pseudo-live	live

Grabbing to a Host buffer with Matrox Meteor-II /Digital

When grabbing to a Host buffer, Matrox Meteor-II /Digital uses SGRAM 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.

Matrox Meteor-II /Digital uses hardware/software transfer control logic to double-buffer the grab between two temporary buffers created within SGRAM. 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 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 zooming grabbed data from a high frequency camera before transferring it. A digital camera with a frequency of 40MHz 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 SGRAM 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

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 SGRAM. In this case, you can only grab to a Host buffer.

An example

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

```
/* File name: mmet2dig.c
* Synopsis: This program allocates 2 grab buffers on the
* SGRAM memory of the meteor2-dig 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(Mi1ImageOnBoard[0]);
MbufFree(Mi1ImageOnBoard[1]);
MbufFree(Mi1ImageDisp);
MdispFree(Mi1Display);
MdigFree(Mi1Digitizer);
MsysFree(Mi1System);
MappFree(Mi1Application);
}
```

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 only when using a MatroxMeteor-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 and M_JPEG_LOSSY_INTERLACED).

The Matrox MJPEG module can compress data from a color source into the following destination formats:

- 8-bit monochrome (in M_JPEG_LOSSY format only).
- 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 Meteor-II /Multi-Channel/PC/104-*Plus*[™]).
 - M_BGR24+M_PACKED.
 - M_BGR24+M_PLANAR.
 - M_BGR32+M_PACKED.
 - M_BGR32+M_PLANAR.
- ❖ The MJPEG module on a Matrox Meteor-II /Standard board can only decompress a buffer that was compressed in the M_JPEG_LOSSY_INTERLACED format. M_JPEG_LOSSY and M_JPEG_LOSSLESS buffer 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 digital camera that complies with the IEEE 1394 Digital Camera Specification (DCS) standard. Currently, data transfer rates of up to 400 Mb/s are possible.

In MIL, to access and acquire data from an IEEE 1394 DCS-based camera, allocate it as a digitizer (**MdigAlloc()**) on a Matrox Meteor-II /1394 system. To access and acquire data

from multiple IEEE 1394 DCS-based cameras, refer to the subsection titled "Grabbing from multiple 1394 DCS-based cameras"

Video formats and modes

Matrox Meteor-II /1394 supports monochrome and color IEEE 1394 DCS-based 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 YUV4:1:1, YUV4:2:2, 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.

For format 7, only mode 2 or mode 3 (depending on the camera type) is supported on Meteor-II /1394. Format 7 (mode 2 and 3; 640 x 480, 1-band, 8-bit) approximates the RS-170 monochrome video data format.

❖ 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.

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 DCS-based 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.

Setting specific camera controls

The specific camera controls include auto-exposure, sharpness, chrominance U/V, gamma, shutter, iris, focus, target temperature, zoom, pan, tilt, optical filter, capture size, and capture quality. For these specific camera controls, there is a non-standard approach which allows flexible management of any camera.

1. Inquire whether the camera has a required camera feature using the **MdigInquire()** function. An error message will be generated if the requested feature (inquire type) is not supported.

If the required feature is available, inquire the minimum and maximum possible values of a required control feature using the **MdigInquire()** function.

To do so, add M_MIN_VALUE or M_MAX_VALUE to the specified **InquireType**, respectively. For example, to inquire the minimum sharpness value, use M_SHARPNESS + M_MIN_VALUE.

2. Once you have determined the range of possible values, you can set or adjust the camera-control option using the **MdigControl()** function. See the subsection titled "1394 DCS-specific controls" of *MdigControl() on the Matrox Meteor-II* for restrictions.

An example

The following code snippet demonstrates how to use the specific method to set the sharpness level for a compliant digital camera. In this example, the sharpness level is set to the mean (average) sharpness level for a particular camera:

```
/* Inquire the minimum and maximum values, respectively, for the image sharpness.
 * Calculate the mean sharpness, and then set the sharpness value to the mean.
 */

long MinVal, MaxVal, NewVal;
MdigInquire(DigID, M_SHARPNESS+M_MIN_VALUE, &MinVal); /* Inquire minimum value */
MdigInquire(DigID, M_SHARPNESS+M_MAX_VALUE, &MaxVal); /* Inquire maximum value */
NewVal = (MinVal + MaxVal) / 2;                        /* Calculate average */
MdigControl(DigID, M_SHARPNESS, NewVal);               /* Set current value to average */
```

Grabbing from multiple 1394 DCS-based digital cameras

Matrox Meteor-II /1394 supports acquisition from multiple IEEE 1394 DCS-based 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 *Using multiple Matrox Meteor-II boards* chapter in the *Matrox Meteor-II 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()**). 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 Meteor-II /1394 system for each camera (**MsysAlloc()**) and then allocate a camera as a digitizer on each respective system. 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 `\MIL\EXAMPLES\METEOR-2\` 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 IEEE1394-DCS based 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, 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.
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.
- ❖ Once you install the MIL Matrox Meteor-II /1394 driver, you can only use cameras that are compliant with the IEEE 1394 Digital Camera Specifications, and you can use these cameras only 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()	Meteor-II-specific inquire options.
MdigAlloc()	Camera format file 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()	Meteor-II-specific inquire options.
MdigLut()	Meteor-II-specific output formats.
MdigReference()	Meteor-II features and restrictions.
MdispAlloc()	Display format. Overlay setting.
MdispLut()	Meteor-II /Digital-specific feature.
MdispPan()	Note on panning.
MdispZoom()	Note on zooming.
MsysAlloc()	Meteor-II-specific allocation options.
MsysControl()	Control type and flag additions.
MsysInquire()	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, 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	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	x*		x**
M_RGB16 + M_PACKED	16-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			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.					

❖ 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 the following **ControlFlag** combinations are supported on Matrox Meteor-II boards:

ControlFlag	Description	STD	MC	DIG	1394
M_PHYSICAL_ADDRESS+M_PITCH	ArrayOfDataPtr is an array of physical addresses. The pitch is in pixels (default).	x	x	x	x
M_PHYSICAL_ADDRESS+M_PITCH_BYTE	ArrayOfDataPtr is an array of physical addresses. The pitch is in bytes.	x	x	x	x
M_HOST_ADDRESS+M_PITCH	ArrayOfDataPtr is an array of Host addresses. The pitch is in pixels (default).			x	x
M_HOST_ADDRESS+M_PITCH_BYTE	ArrayOfDataPtr 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. ArrayOfDataPtr will be a pointer to an array of MIL buffer identifiers. One identifier per band must be provided. SizeX , SizeY , Type , Attribute , and Pitch must be the same for each buffer. Pitch can be set to M_DEFAULT.			x	

MbuflInquire()

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

InquireType	Description	STD	MC	DIG	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 in windowed mode. This information is used, for example, to access grabbed data while a continuous grab (MdigGrabContinuous()) 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 can be programmed with the Genesis Native Library.					

MdigAlloc()

- Note that each camera that is connected requires a separate **MdigAlloc()** call.
- 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	1394
"M_DEFAULT"	NTSC, 640x480, 3x8 bits, 12.5 MHz, composite	Color	x			x
	RS-170 RGB, 640x480, 3x8 bits, 12.5 MHz	Color		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	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 I 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 I RGB, 768x576, 3x8 bits, 14.8MHz	Color		x		
*.dcf	A DCF file that specifies the input signal format. See the \MIL (user-specified) directory for the current list of supported formats.		x	x		
*.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.				x	

DataFormat (Camera Type)	Description	Color or Monochrome	STD	MC	DIG	1394
M_XXY_DATA[@z[FPS]]	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. 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.					x

MdigChannel()

The number of independent composite/monochrome, Y/C or RGB cameras than 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
■ Meteor-II Standard	12	7	12
■ Meteor-II Multi-Channel	6		12
■ Meteor-II Digital	4		
■ Meteor-II /1394	1*		
*Note, Matrox Meteor-II /1394 supports the IEEE tree topology for 1394 devices. More than one 1394-compliant camera can be attached to the Meteor-II /1394 board, and each camera must be allocated as a separate digitizer. However, each digitizer has only 1 channel available.			

- The DCF selected during digitizer allocation, determines the color mode used. To switch between cameras of the same camera type, 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			
Composite** or Monochrome	M_CH0	VID_IN1	x			x***
		VID1_IN1		x		
		DATA, INPUT 0-7			x	
		1394 INPUT				x

Camera Type	Channel	Signal/Sync input	STD	MC	DIG	1394
Composite** or Monochrome (cont.)	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.						

- 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

Sync input channel	Sync input signal
M_CH4	VID2_IN1
M_CH5	VID2_IN2
M_CH6	VID2_IN3
M_CH7	SYNC_IN

MdigControl()

- It is possible to optimize grabs by taking advantage of hardware cropping with a format 7-capable camera on Matrox Meteor-II /1394. However, all format 7-capable cameras need an external trigger to operate in format 7 (mode 2 and 3). Therefore, when using the M_SOURCE_SIZE... and M_SOURCE_OFFSET... control types with a format 7-capable camera, MIL will automatically switch to format 7 (mode 2 or 3) to optimize grabs, if:
 - The specified offsets and sizes match the boundaries of one of supported subregions.
 - The trigger is enabled (use M_ENABLE control value with the M_GRAB_TRIGGER control type)

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

- ❖ Note that changing the M_SOURCE_SIZE... and M_SOURCE_OFFSET... control types will not affect the trigger mode.

- 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 C	D I G	1 3 9 4
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.				
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 interger 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

ControlType	ControlValue & Description		STD	M/C	DIG	1394
M_GRAB_INPUT_GAIN (cont.)	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	
	M_REVERSE	Flip the grabbed image horizontally.				
	M_FORWARD	Grab normally in the horizontal direction.				
	M_DEFAULT	Same as M_FORWARD.				
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	
	M_REVERSE	Flip the grabbed image vertically.				
	M_FORWARD	Grab normally in the vertical direction.				
	M_DEFAULT	Same as M_FORWARD.				
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*		

ControlType	ControlValue & Description	S T D	M C	D I G	1 3 9 4
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 MdigGrab() to be executed. However, only one grab can be queued; a call to another MdigGrab() 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 after initiating the start of the grab. Queue the grab on-board if another grab is issued before the first one has finished. This allows other operations to be performed while waiting for the next MdigGrab() to be executed, but in this case more than one MdigGrab() command can be queued. Note that the grab command is queued on the Host (not on-board). Therefore, the number of queued grabs is limited only by the amount of Host memory (RAM).			x	x
M_GRAB_SCALE	M_FILL_DESTINATION (windowed mode only)	x	x	x	x
	M_FILL_DISPLAY (windowed mode only)	x	x	x	x
	Scaling factors: $0 < x \leq 1$. The 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 will be used (1/16,...1/4, 1/3, 1/2, 1).	x			

ControlType	ControlValue & Description		S T D	M C	D I G	1 3 9 4
M_GRAB_SCALE (cont.)	1/16,...1/4, 1/3, 1/2, 1.			x		
	1/16,...1/4, 1/3, 1/2, 1, 2, and 4.				x	
	$0 < x$.					x
M_GRAB_TIMEOUT	Set the maximum time to wait for a frame before generating an error.		x	x		x
	M_DEFAULT	Determined by the frame period.				
	M_INFINITE	Wait indefinitely. This is recommended only for triggered cameras.				
	Value in msec	Specify time to wait.				
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
M_THREAD_PRIORITY	Set the hook function priority under Windows. Valid values are:		x	x	x	x
	11 - 15	High priority.				
	16 or 22 - 26	Real-time priority.				
	If you want to change the hook function priority, use MdigInquire() to determine its current value. Changing these priority settings will affect the overall application priority, due to a Windows restriction.					
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
	M_ENABLE	Enables trigger detection.	x	x	x	x

ControlType	ControlValue & Description		S T D	M C	D I G	1 3 9 4
M_GRAB_TRIGGER (cont.)	M_DISABLE	Disables trigger detection.	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	
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
	M_NULL	The trigger is inactive.	x	x	x	x
	M_SOFTWARE	Uses software trigger.	x	x	x	
	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	
		Description is camera-dependent.				x
	M_HARDWARE_PORT1	Use TTL hardware trigger signal. Pin 20 on Video Input connector.		x		
		TTL trigger is on Pin 1 on the Trigger Input connector. RS-422/LVDS trigger is a combination of Pin 47 (TRIGGER, INPUT, +) and Pin 48 (TRIGGER, INPUT, -) on Digital Interface connector.			x	
	M_HARDWARE_PORT2	Use the RS-422/LVDS trigger signal. Pin 17 (422+) and Pin 18 (422-) on RS-422/LVDS input connector.		x		

ControlType	ControlValue & Description		S T D	M C	D I G	1 3 9 4
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	
	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:			x	x	
	M_LEVEL_HIGH					
	M_LEVEL_LOW					
	M_DEFAULT	Same as DCF.				
M_GRAB_EXPOSURE_SOURCE	Select the trigger source for the specified exposure timer:					
	M_DEFAULT	Same as the *.dcf				
	M_HARDWARE_PORT0	Use opto-isolated hardware trigger signal: Both Pin 34 (OPTOTRIG-) and Pin 35 (OPTOTRIG+) on Video Input connector.				
		Use opto-isolated hardware trigger signal. Both Pin 7 (OPTOTRIG+) and Pin 2 (OPTOTRIG-) on Trigger Input connector.				
	M_HARDWARE_PORT1	Use TTL hardware trigger signal. Pin 20 (TRIGGER) on Video input connector.				

Note that the M_GRAB_EXPOSURE_SOURCE control type has no effect when grabbing using the automatic exposure model.

ControlType	ControlValue & Description		S T D	M C	D I G	1 3 9 4
M_GRAB_EXPOSURE_SOURCE (cont.)	M_HARDWARE_PORT1 (cont.)	Use TTL or RS-422/LVDS hardware trigger signal. See MdigControl0 (M_USER_IN_FORMAT) to select. On the digital interface connector, RS-422/LVDS trigger is a combination of Pin 47 (TRIGGER, INPUT, +) and Pin 48 (TRIGGER, INPUT, -). The TTL trigger is on Pin 1 on the trigger input connector.			x	
	M_HARDWARE_PORT2	Use the RS-422/LVDS trigger signal. Pin 17 (422+) and Pin 18 (422-) on RS-422/LVDS input connector.		x		
	M_NULL	Disable specified exposure timer. This has no effect when grabbing using automatic exposure model.		x	x	
	M_SOFTWARE	Use software trigger. The exposure signal is generated when MdigControl0 with M_GRAB_EXPOSURE + M_TIMER/N and M_ACTIVATE is called.		x	x	
	M_VSYNC	Use vertical sync signal.		x	x	
	M_HSYNC	Use horizontal sync signal.		x	x	
	M_TIMER1	Use exposure signal generated by Timer1. Use only if setting trigger source for Timer2.		x	x	
	M_TIMER2	Use exposure signal generated by Timer2. Use only if setting trigger source for Timer1.		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	

ControlType	ControlValue & Description	S T D	M C	D I G	1 3 9 4										
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 ControlValue is set to 0, exposure is disabled and the grab is performed immediately.</p> <p>Note, an error is returned if the specified exposure time cannot be generated.</p>		x	x											
M_GRAB_EXPOSURE_TIME_DELAY	<p>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.</p> <p>Note, an error is returned if the specified delay cannot be generated.</p>		x	x											
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</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	M_EDGE_RISING	Low to high signal variation.	M_EDGE_FALLING	High to low signal variation.		x	x			
	Set the trigger activation mode for specified timer.														
M_DEFAULT	Same as the *.dcf														
M_EDGE_RISING	Low to high signal variation.														
M_EDGE_FALLING	High to low signal variation.														
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).	x	x	x	
	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).														

ControlType	ControlValue & Description		S T D	M C	D I G	1 3 9 4
M_UART_STOP_BITS	Add a data bit to signal the end of character data being sent or received.		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_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	
	7	Data length is 7 bits.				
	8	Data length is 8 bits.				
	M_DEFAULT	Data length is 8 bits.	x	x		
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	
M_UART_TIMEOUT	Set the maximum time to wait between each byte when reading incoming data.		x	x	x	
	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 output of the UART. Set ControlValue as a pointer to a character.		x	x	x	
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	
M_UART_STRING_DELIMITER	Set the character used to terminate strings of incoming or outgoing data. The delimiter is not sent when writing data; it is read for incoming data.		x	x	x	
	M_DEFAULT	The '\0' character.				

ControlType	ControlValue & Description		S T D	M C	D I G	1 3 9 4
M_UART_READ_STRING_LENGTH	Set the length of the string (in bytes) to be read by M_UART_READ_STRING.		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	Use this value to specify the size of your reading buffer to prevent global protection faults from happening when using the M_UART_READ_STRING control.		x	x	x	
M_UART_READ_STRING	Read a string of incoming data from the UART. The ControlValue 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	
M_UART_WRITE_STRING_LENGTH	Sets the length of the string to be sent to the output of the UART.		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.				
M_UART_WRITE_STRING	Send the string of data, specified with the control value, through the UART. Set ControlValue 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	

ControlType	ControlValue & Description	S T D	M C	D I G	1 3 9 4
M_USER_BIT+(0,1,2,3,4,7,8)	Set the state of the output bits of the digital interface connector: M_ON or M_OFF. The relationship between the MIL user-bit number and the actual user output bit is as follows:			x	
	MIL User-Bit#				
	bit 0:				
	bit 1:				
	bit 2:			x	
	bit 3:				
	bit 4:				
	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:				
	MIL User-Bit#	x	x		
	bit 3:				
	bit 4:				
	MIL User-Bit#		x		
	bit 7:				
	bit 8:				

ControlType	ControlValue & Description		S T D	M C	D I G	1 3 9 4
M_USER_IN_FORMAT	Enable either TTL or (RS-422/LVDS) receivers for digital I/Os:				x	
	M_TTL	Enable the TTL receivers for trigger and user inputs.				
	M_RS422	Enable the RS-422 receivers for trigger and user inputs.				
	M_LVDS	Enable the LVDS receivers for trigger and user inputs.				
	M_DEFAULT	Same as the <i>.dcf</i> .				
	M_DISABLE	Disable trigger and user inputs.				
M_USER_OUT_FORMAT	Enable either TTL or RS-422/LVDS drivers for digital I/Os:				x	
	M_TTL	Enable the TTL drivers for exposure and user outputs.				
	M_RS422	Enable the RS-422 drivers for exposure and user outputs.				
	M_LVDS	Enable the LVDS receivers for trigger and user inputs.				
	M_DEFAULT	Same as the <i>*.dcf</i> .				
	M_DISABLE:	Disable exposure and user outputs.				
M_VCR_INPUT_TYPE	Set the digitizer input to VCR and improve the image's quality when grabbing from a VCR. This is only valid when using the decoder path.		x			
	M_DEFAULT	Same as M_DISABLE				
	M_DISABLE	Leaves image quality as is.				
	M_ENABLE	Improves the image quality when grabbing with a VCR.				

Matrox Meteor-II /1394 DCS-specific controls

There are 1394-specific **ControlType** settings. The control type values can be set 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**	
M_WHITE_BALANCE_V**	
M_SATURATION***	Control the color saturation of the picture.
M_GAMMA	Defines the function between the incoming light level and output picture level.
M_SHUTTER	Control the integration time of the incoming light.
M_GAIN***	Control the camera circuit gain.
M_IRIS	Control the mechanical lens iris.
This control type sets one of the two chrominance (color) components of the data. *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.	

ControlType	Description
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.
<p>**This control type sets one of the two chrominance (color) components of the data.</p> <p>***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.</p>	

MdigGrab/MdigGrabContinuous()

- When performing a grab on Matrox Meteor-II /Standard, Matrox Meteor-II /Multi-Channel, and Matrox Meteor-II /Digital 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, or Matrox Meteor-II /Digital boards, when the buffer is packed. This restriction does not apply to Matrox Meteor-II /1394.

- When performing real-time grabs, Windows 98 does not provide the same performance and consistency as Windows NT. 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.

- 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 VGA that supports fast linear-memory accesses to its frame buffer. For live grabs using Matrox Meteor-II, refer to *Displaying using a Meteor-II system* at the beginning of this chapter.

MdigGrabWait()

The following digitizer feature is not supported on Matrox Meteor-II/ 1394:

- 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.
- M_GRAB_FRAME_START is not supported on Matrox Meteor-II /Digital.
- No field-related hooks are supported on Matrox Meteor-II /1394.

MdigInquire()

- While in windowed mode and using a Matrox Meteor-II board, 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 and Matrox Meteor-II /1394, the M_GRAB_MODE attribute can also return M_ASYNCHRONOUS_QUEUED.
- For Meteor-II /1394, M_GRAB_TRIGGER_MODE is a camera-specific control, since the trigger is connected directly to the camera (not the board). As a result, this setting cannot be inquired.
- Additional **InquireType** parameter settings which are supported by a particular Matrox Meteor-II board are indicated in the table:

InquireType	Description	S T D	M C	D I G	1 3 9 4
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	

InquireType	Description	S T D	M C	D I G	1 3 9 4
M_GRAB_DIRECTION_Y	Vertical grab direction: M_REVERSE or M_FORWARD.	x	x	x	
M_GRAB_IN_PROGRESS	Current grab state: M_YES or M_NO.	x	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		
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_TIMEOUT	Maximum time to wait for the end of grab before generating an error: M_INFINITE or a value in msec.	x	x		
M_GRAB_WINDOW_RANGE	State of limiting the range of the grabbed pixel values: M_ENABLE or M_DISABLE.	x	x	x	x
M_THREAD_PRIORITY	Hook function priority under Windows:		x	x	x
	11 - 15	High priority.			
	16 or 22 - 26	Real-time priority.			
M_INPUT_SIGNAL_SOURCE	Input-signal source: M_HARDWARE_PORT0 (analog) or M_HARDWARE_PORT1 (digital port).	x	x	x	
M_NATIVE_ID	Identifier of the Genesis Native Library* digitizer that is associated with the specified MIL digitizer.			x	
M_NATIVE_CAMERA_ID	Identifier of the Genesis Native Library* camera that is associated with the specified MIL digitizer.			x	

InquireType	Description	S T D	M C	D I G	1 3 9 4
M_NATIVE_CONTROL_ID	Identifier of the Genesis Native Library* control buffer that is associated with the specified MIL digitizer.			X	
M_NATIVE_LAST_GRAB_OSB_ID	Identifier of the Genesis Native Library* OSB (operations status block) used in the last grab.			X	
*Matrox Meteor-II /Digital can be programmed with the <i>Genesis Native Library</i> .					
M_BRIGHTNESS_REF	Brightness reference level.	X			X
M_COLOR_MODE	Color mode:				
	M_RGB		X	X	
	M_MONO8_VIA_RGB		X		
	M_EXTERNAL_CHROMINANCE	X			
	M_COMPOSITE	X			
	M_MONOCHROME	X		X	X
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 "yes". However, this monochrome camera/color DCF combination is still supported on Meteor-II /Standard, because it is useful when switching between channels.	X			X
M_INPUT_SIGNAL_HYSNC_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			
M_CONTRAST_REF	Contrast reference level.	X			

InquireType	Description	S T D	M C	D I G	1 3 9 4
M_HUE_REF	Hue reference level.	X			X
M_SATURATION_REF	Saturation reference level.	X			X
M_GRAB_TRIGGER	Grab trigger state: M_DEFAULT, M_ENABLE, M_DISABLE, M_ACTIVATE.	X	X	X	X
M_GRAB_TRIGGER_SOURCE	Determines the trigger source. See <i>MdigControl()</i> .	X	X	X	X
M_GRAB_TRIGGER_MODE	Hardware trigger activation mode. See <i>MdigControl()</i> .	X	X	X	
M_GRAB_EXPOSURE_BYPASS	Exposure model: M_ENABLE (manual) or M_DISABLE (automatic).		X	X	
For the following M_GRAB_EXPOSURE... inquire 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	Exposure timer state for non-software trigger source: M_ENABLE or M_DISABLE.		X	X	
M_GRAB_EXPOSURE_MODE	Exposure signal's polarity: M_LEVEL_HIGH or M_LEVEL_LOW.		X	X	
M_GRAB_EXPOSURE_SOURCE	Trigger source for specified timer: See <i>MdigControl()</i> .		X	X	
M_GRAB_EXPOSURE_TIME	Time for the active portion of the exposure signal (value in nsec). Returned as a double.		X	X	
M_GRAB_EXPOSURE_TIME_DELAY	Delay (in nsec) between the trigger and the start of exposure. Returned as a double.		X	X	
M_GRAB_EXPOSURE_TRIGGER_MODE	Trigger activation mode for specified timer: M_EDGE_RISING or M_EDGE_FALLING.		X	X	
M_GRAB_END_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_END, will run.	X	X		X
M_GRAB_END_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_END, will run.	X	X		X

InquireType	Description	S T D	M C	D I G	1 3 9 4
M_GRAB_FIELD_END_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FIELD_END, will run.	x	x		
M_GRAB_FIELD_END_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FIELD_END, will run.	x	x		
M_GRAB_FIELD_START_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FIELD_START, will run.	x	x		
M_GRAB_FIELD_START_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FIELD_START, will run.	x	x		
M_GRAB_FRAME_END_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FRAME_END, will run.	x	x		x
M_GRAB_FRAME_END_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FRAME_END, will run.	x	x		x
M_GRAB_START_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_START, will run.	x	x		x
M_GRAB_START_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_START, will run.	x	x		x
M_GRAB_THREAD_HANDLE	Thread handle for all grab interrupt handlers.			x	
M_GRAB_THREAD_ID	Thread identifier for all grab interrupt handlers.			x	
M_UART_PARITY	Current UART parity setting: M_ODD, M_EVEN, M_DISABLE.	x	x	x	
M_UART_STOP_BITS	Current number of stop bits in UART configuration: 1 or 2.	x	x	x	
M_UART_DATA_LENGTH	Current data length in UART configuration: 7 or 8.	x	x	x	

InquireType	Description	S T D	M C	D I G	1 3 9 4
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	
M_UART_DATA_PENDING	Indicates the input buffer has some pending data: M_TRUE, M_FALSE.	x	x	x	
M_UART_WRITE_STRING_LENGTH	The length of the string to be sent to the output of the UART. Can be M_DEFAULT if the send length is specified by M_UART_STRING_DELIMITER.	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 send length is specified by M_UART_STRING_DELIMITER.	x	x	x	
M_UART_READ_STRING_MAXIMUM_LENGTH	Current maximum size of the reading buffer.	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	
M_UART_TIMEOUT	Current maximum time in milliseconds to wait between bytes of incoming data. Can be M_INFINITE or value in msec.	x	x	x	
M_UART_THREAD_HANDLE	Current UART thread handle.	x	x	x	
M_UART_THREAD_ID	Current UART thread ID.	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			
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			

InquireType	Description	S T D	M C	D I G	1 3 9 4
M_USER_BIT+ (0,1,2,3,4,5,6,7,8)	Current state of an input/output bit of the Video Input connector: M_ON (1) or M_OFF (0).				
	MIL User-Bit # Video Input connector signal	X	X		
	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 an input/output bit of RS-422/LVDS connector: M_ON (1) or M_OFF (0).				
	MIL User Bit # RS-422/LVDS connector signal		X		
	bit 5: USER1IN_422		X		
	bit 6: USER2IN_422		X		
	bit 7: USER1OUT_422		X		
	bit 8: USER2OUT_422		X		
	Current state of the input bit of the Digital interface connector: M_ON (1) or M_OFF (0).			X	
	MIL User Bit # RS-422/LVDS connector signal				
	bit 0: USER0 INPUT (TTL or RS-422/LVDS)				
	bit 1: USER1 INPUT (TTL or RS-422/LVDS)				
M_USER_IN_FORMAT	Type of receiver for digital I/Os (M_TTL, M_RS422, M_LVDS, or M_DISABLE).			X	
M_USER_OUT_FORMAT	Type of driver for digital I/Os (M_TTL, M_RS422, M_LVDS, or M_DISABLE).			X	

InquireType	Description	S T D	M C	D I G	1 3 9 4
M_VCR_INPUT_TYPE	Digitizer input set to VCR. This is only valid when using the decoder path. M_DISABLE or M_ENABLE	X			

Matrox Meteor-II /1394 DCS-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	This inquire type inquires one of the two chrominance (color) components of the data.
M_SATURATION	Read the color saturation of the picture.
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	Reads the lens focus control.
M_TEMPERATURE	Read the temperature inside the camera.
M_TARGET_TEMPERATURE	Read the target temperature for the camera.

InquireType	Description
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	Reads the capture size of the image.
M_CAPTURE_QUALITY	Read the capture size of the image.
M_FORMAT_SUPPORTED	Generates a list of all supported string formats that will be accepted through MdigAlloc() for your 1394 digital camera. This list includes "M_RS170", "M_NTSC" (if available), and "M_DEFAULT".

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 input LUTs on Matrox Meteor-II can output data in 8:8:8, 5:6:5, or 5:5:5 format. The output data format is determined by the format of the destination buffer used during the grab operation. For example, if a 32-bit grab buffer is being used, the LUTs will output data in 8:8:8 format. Note that the LUTs will output the least significant bits of the LUT data values.

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 and Reference Level	Description	STD	MC	DIG	1394
M_BLACK_REF*	Digitizer black reference level.** For M_BLACK_REF, the minimum voltage level (M_MIN_LEVEL) corresponds to 0.6V. The maximum voltage level (M_MAX_LEVEL) corresponds to 1.6V.		X		
M_WHITE_REF*	Digitizer white reference level.** For M_WHITE_REF, the minimum voltage level (M_MIN_LEVEL) corresponds to 1.6V. The maximum voltage level (M_MAX_LEVEL) corresponds to 2.6V		X		
M_HUE_REF	Digitizer hue reference level for composite input signals.	X			X
M_SATURATION_REF	Digitizer saturation reference level for composite input signals.	X			X
M_BRIGHTNESS_REF	Digitizer brightness reference level for composite input signals.	X			X
M_CONTRAST_REF	Digitizer contrast reference for composite input signals.	X			
<p>*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.</p> <p>**Note that some consecutive ReferenceLevel 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)</p>					

MdispLut()

- The Matrox Meteor-II /Digital board has four 8-bit x 256 programmable LUTs, which can be associated with a display that was allocated using ***MdispAlloc()***. 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). When grabbing from more than two channels, the LUTs must be 8-bit.

MdispPan()

- Software pan is fully supported in windowed mode.
- Hardware pan can be enabled (***MdispControl()***) only if you are using a Matrox MGA display card and are under Windows. The following restrictions apply:
 - The hardware pan (***XOffset***) must be a multiple of 4 (for example, 0, 4, 8, 12, or 16).
 - Hardware panning of any region beyond the image buffer's boundaries cannot be displayed.

MdispZoom()

- Software zoom is fully supported in windowed mode.
- Hardware zoom can be enabled (***MdispControl()***) only if you are using a Matrox MGA display card. The following restrictions apply:
 - In the X and Y direction, only factors of 1, 2, and 4 are supported for hardware zooming.
 - You cannot use the zoom factor of 4 in the 640 x 480 and 800 x 600 resolutions.
 - Due to hardware limitations, when using the zoom factor of 4 with the 1152 x 882 x 8 resolution, a 32-pixel-wide band of spurious data appears at the right border of the screen.
 - In multi-head mode, only the screen where the cursor is positioned appears zoomed. To zoom the other screen, move the cursor to that screen.

MsysAlloc()

- To allocate a Meteor-II system, you must select the **SystemTypePtr** parameter as 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

This selection opens communication with the board and will automatically allow the system to use some Host memory and your VGA board, if necessary.

You can 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.

- ❖ 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.

MsysControl()

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

ControlType	Description and ControlValue	S T D	M C	D I G	1 3 9 4
M_BUS_MASTER_COPY_TO_HOST	<div>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_DISABLE (the default for Matrox Meteor-II /Digital), the copy operation is driven by the Host CPU. When set to M_ENABLE, the copies are driven by the digitizer, speeding up the copy and reducing Host intervention.**</div> <div>M_ENABLE or M_DISABLE.</div>			x	
M_BUS_MASTER_COPY_FROM_HOST	<div>When copying between a Host buffer and an M_ON_BOARD buffer, copies can be driven by the digitizer (bus master) or by the Host. When set to M_DISABLE (the default for Matrox Meteor-II /Digital), the copy operation is driven by the Host CPU. When set to M_ENABLE, the copies are driven by the digitizer, speeding up the copy and reducing Host intervention.**</div> <div>M_ENABLE or M_DISABLE.</div>			x	
**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 so might result in unpredictable results.					

ControlType	Description and ControlValue	S T D	M C	D I G	1 3 9 4
M_PCI_LATENCY	<p>To modify the transfer rate and performance of the PCI bus, Meteor-II systems support an additional combination for the ControlType and ControlValue parameters.</p> <p>Controls the length of the PCI bus transfer-time slice that is allocated to Matrox Meteor-II. Increasing it can improve transfer time when there is heavy traffic on the PCI bus. However, you should be aware that this can affect the performance of the other bus masters (including the main CPU). Possible values are: 0 - 127. Use MsysInquire() to determine the default value on your system. It is recommended that the chosen value be one near the default value.</p>	X	X		
<p>MIL can perform a continuous grab operation live in the VGA frame buffer when the display is in windowed mode.</p> <p>When necessary, the grab will switch to pseudo-live (simulating a live grab by grabbing into the Host buffer and updating the display) to prevent the grab from overwriting another window. If there is an instance when automatic live-to-pseudo-live switching does not happen or you want to override the default behavior, you can use the following ControlType and ControlValue parameter settings:</p>					
M_LIVE_GRAB	Set whether to perform a live grab whenever possible or to force a pseudo-live grab into displayable image buffers:	X	X	X	X
	M_ENABLE (default) Live whenever possible.				
	M_DISABLE Force pseudo-live				
M_STOP_LIVE_GRAB_WHEN_MENU	Set whether or not to switch to a pseudo-live grab while an opened menu overlaps the display window:	X	X	X	X
	M_ENABLE Pseudo-live while menu overlaps the display window (default).				
	M_DISABLE Force live.				

ControlType	Description and ControlValue		S T D	M C	D I G	1 3 9 4
M_STOP_LIVE_GRAB_WHEN_INACTIVE	Set whether or not to switch to a pseudo-live grab while the display window is inactive (i.e., while it does not have the focus):		x	x	x	x
	M_ENABLE	Pseudo-live while the display window is inactive (default).				
	M_DISABLE	Force live.				
M_STOP_LIVE_GRAB_WHEN_DISABLED	Set whether or not to switch to a pseudo-live grab while the display window is disabled (e.g. when a pop-up dialog box is opened):		x	x	x	x
	M_ENABLE	Pseudo-live while the display window is disabled (default).				
	M_DISABLE	Force live.				
M_PSEUDO_LIVE_GRAB_WHEN_OVERLAPPED	Set whether to pause the grab or switch to a pseudo-live grab when the live grab is interrupted due to one of the above-mentioned conditions (e.g., if the window displaying the grab is overlapped by a menu).		x	x	x	x
	M_ENABLE	Pseudo-live (default).				
	M_DISABLE	Pause grab.				
M_LAST_GRAB_IN_TRUE_BUFFER	When the display is in windowed mode (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 ControlType to M_DISABLE:		x	x	x	x
	M_ENABLE	Grab last frame in true grab buffer (default).				
	M_DISABLE	Don't grab last frame in true grab buffer.				

ControlType	Description and ControlValue		S T D	M C	D I G	1 3 9 4
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 ControlType :		x	x	x	x
	M_ENABLE	Force double-buffering.				
	M_DISABLE	Disable double-buffering				
M_LIVE_GRAB_MOVE_UPDATE	When a grab window is displaced, the grab is stopped and restarted at every displacement. When grabbing from a triggered camera, a trigger is probably not issued as often as the window is displaced. To avoid having an empty window, a copy is performed from the old location to the new before restarting the grab. To override this default, set the ControlType to M_DISABLE:		x	x	x	x
	M_ENABLE	Default for hardware triggered cameras.				
	M_DISABLE	Default for other camera types.				

ControlType	Description and ControlValue		S T D	M C	D I G	1 3 9 4
M_LIVE_GRAB_END_TRIGGER	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 ControlType to M_DISABLE:		x	x	x	x
	M_ENABLE	Default for hardware triggered cameras.				
	M_DISABLED	Default for other camera types.				
M_LIVE_GRAB_TRACK	During a live grab operation in windowed mode, the MIL driver keeps track of the grab window. This default (M_ENABLE) can be overridden by setting the ControlType to M_DISABLE.		x	x	x	x
M_HARDWARE_COMPRESSION	Specifies whether the 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
M_HARDWARE_DECOMPRESSION	Specifies whether the decompression is done by the hardware or the software (0 to 255).	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.				

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
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	
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	
M_BOARD_TYPE	The type of system board: M_METEOR_II_STD M_METEOR_II_MC M_METEOR_II_DIG M_METEOR_II_WITH_COMPRESSION_MODULE M_METEOR_II_MC_WITH_COMPRESSION_MODULE M_METEOR_II_1394	x	x	x	x
M_COMPRESSION_MODULE_PRESENT	Returns M_TRUE if a compression board is present, otherwise it returns M_FALSE.	x	x		

InquireType	Description	S T D	M C	D I G	1 3 9 4
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
M_HARDWARE_COMPRESSION	Specifies whether the 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.			
M_HARDWARE_DECOMPRESSION	Whether the 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_PCI_LATENCY	The length of the PCI bus transfer time slice.	x	x		
M_BOARD_REVISION	The board revision (long value).	x	x		
M_RGB_MODULE_NUM	Number of system RGB modules (0 or 1).	x	x		
M_DISPLAY_DOUBLE_BUFFERING	The state of double-buffering.	x	x	x	x

InquireType	Description	S T D	M C	D I G	1 3 9 4
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.	x	x	x	x
M_LIVE_GRAB	A live grab (not pseudo-live) is performed: M_ENABLE or M_DISABLE.	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
M_LIVE_GRAB_MOVE_UPDATE	Copy performed between windows on update: M_ENABLE or M_DISABLE.	x	x	x	x
M_LIVE_GRAB_TRACK	Whether or not MIL keeps track of the live-grab window: M_ENABLE or M_DISABLE.	x	x	x	x
M_PSEUDO_LIVE_GRAB_WHEN_OVERLAPPED	A switch is made to a pseudo-live grab when the display window is overlapped by another window: M_ENABLE or M_DISABLE.	x	x	x	x
M_STOP_LIVE_GRAB_WHEN_DISABLED	Grabbing is frozen when the display window is disabled: M_ENABLE or M_DISABLE.	x	x	x	x
M_STOP_LIVE_GRAB_WHEN_INACTIVE	Grabbing is frozen when the display window is inactive: M_ENABLE or M_DISABLE.	x	x	x	x
M_STOP_LIVE_GRAB_WHEN_MENU	Grabbing is frozen when the display window is overlapped by a menu: 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, and composite (CVBS) and component (Y/C) color video in NTSC/PAL format. There are two versions of the board: a PCI version and an AGP version.

Eight software selectable channels are available to switch between up to 8 composite or 4 Y/C cameras.

With its MGA-G400 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 1600x1200 resolution. Matrox Orion can allocate an overlay frame buffer surface in 8-bit monochrome format or 32-bit RGB format (BGR32 packed), thereby supporting either of these display modes. In addition, Matrox Orion features non-destructive, live graphics and video overlay display capabilities. Because of these capabilities, Matrox Orion supports the Direct Draw underlay-surface display architecture, and can dynamically allocate an underlay frame buffer surface that is in 8-bit monochrome format or YUV16 format (YUYV).

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.

See Appendix A for a data flow diagram.

❖ Note that Matrox Orion also features an encoder, component RGB path, programmable display LUTs, simultaneous update of Host buffer and display, dual-screen capabilities, and triggers. However, these features are not currently supported in software and, therefore, are not mentioned in this document. These features will be available in a future release.

Also, note that Matrox Orion supports Windows NT/2000.

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, the on-board frame buffer, and the color keying mechanism of Matrox Orion to grab and display images.

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

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()	Camera format file 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. Overlay setting.
MdispInquire()	Additional inquiries.
MdispPan()	Panning particularities.
MdispZoom()	Supported zoom factors.

Commands	Matrox Orion particularities
MsysAlloc()	Orion-specific allocation options.
MsysInquire()	Orion-specific inquire options.

MbufAlloc...()

- Only 8-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_BGR32 + M_PACKED	32-bit (BGR) packed pixels (default for M_DISP buffer).
M_YUV16 + M_PACKED	YUV16 (4:2:2) packed standard.
M_YUV16_YUYV + M_PACKED	YUV16 (4:2:2) packed standard.

Note that it might be slower to process buffers with these attributes.

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	ArrayOfDataPtr is an array of physical addresses. The pitch is in pixels (default).
M_PHYSICAL_ADDRESS + M_PITCH_BYTE	ArrayOfDataPtr is an array of physical addresses. The pitch is in bytes.

- The **Attribute** parameter can be changed to one of the following:

Attribute parameter for 3-band 8-bit buffer	Internal storage format for 3-band 8-bit buffer
M_BGR32 + M_PACKED	32-bit packed (BGR) pixels (default for M_DISP buffers).
M_YUV16 + M_PACKED	YUV16 (4:2:2) packed standard.
M_YUV16_YUYV + M_PACKED	YUV16 (4:2:2) packed standard.

Note that it might be slower to process buffers with these attributes.

MbuInquire()

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

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

MdigAlloc()

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

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.

The predefined settings for color cameras are:

"M_DEFAULT"	NTSC, 640x480, 3x8 bits, 12.5MHz, composite
"M_NTSC"	NTSC, 640x480, 3x8 bits, 12.5MHz, composite
"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_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 \MIL (user-specified) directory for the current list of supported formats.

MdigChannel()

- When using the video decoder, 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).
- A single **MdigChannel**(...,M_CHx) call can be used to manage simultaneously the data signal and sync channel for all types of supported cameras. This method will assign the data signal input and sync input to the same channel.
- To switch between cameras of a similar type, use:

Camera Type	Channel	Signal/Sync input
Any	M_DEFAULT	Same as M_CH0.
Composite-sync monochrome/ composite color	M_CH0	VID_IN1
	M_CH1	VID_IN2
	M_CH2	VID_IN3
	M_CH3	VID_IN4
	M_CH4	VID_IN5
	M_CH5	VID_IN6
	M_CH6	VID_IN7
	M_CH7	VID_IN8
Y/C	M_CH0	(Y camera 1) VID_IN1 and (C camera 1) VID_IN2
	M_CH1	(Y camera 2) VID_IN3 and (C camera 2) VID_IN4
	M_CH2	(Y camera 3) VID_IN5 and (C camera 3) VID_IN6
	M_CH3	(Y camera 4) VID_IN7 and (C camera 4) VID_IN8

MdigControl()

- It is not possible to queue asynchronous grabs on Matrox Orion. Therefore, M_ASYNCHRONOUS_QUEUED cannot be used as a control value with the M_GRAB_MODE control type.
- The supported scaling factors for M_GRAB_SCALE are as follows: $0 < x \leq 256$.

Note that you cannot scale grabbed data if the destination buffer is in a YUV format.

- M_GRAB_FIELD_NUM can only be set to 2.
- Matrox Orion supports these additional **ControlType** parameter settings:

ControlType	Description & ControlValue	
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 decoder path and M_GRAB_AUTOMATIC_INPUT_GAIN is M_DISABLE, the gain can be set to any integer value from 0 to 255.	
M_GRAB_TIMEOUT	Set the maximum time to wait for a frame before generating an error.	
	M_DEFAULT	Determined by the frame period.
	M_INFINITE	Wait indefinitely. This is recommended only for triggered cameras.
	value in msec	Specify time for wait.

ControlType	Description & ControlValue	
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#	Expanded video I/O connector signal
	3	USER1OUT signal.
	4	USER2OUT signal.
M_VCR_INPUT_TYPE	Set the digitizer input to VCR. Improves the image's quality when grabbing from a VCR. This is only valid when using the decoder path:	
	M_DEFAULT	Same as M_DISABLE
	M_DISABLE	Leaves image quality as is.
	M_ENABLE	Improves the image quality when grabbing with a VCR.
M_THREAD_PRIORITY	Set the hook function priority under Windows. Valid values are:	
	11 - 15	High priority.
	16 or 22 - 26	Real-time priority.
	If you want to change the hook function priority, use MdigInquire() to determine its current value. Changing these priority settings might affect the overall application priority, due to a Windows restriction.	

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 performing real-time grabs, Windows 98 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.

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_GRAB_AUTOMATIC_INPUT_GAIN	Mode of automatic input gain: M_ENABLE or M_DISABLE.
M_GRAB_INPUT_GAIN	When grabbing using the decoder path and M_GRAB_AUTOMATIC_INPUT_GAIN is M_DISABLE, any integer value from 0 to 255.
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 . However, for monochrome cameras , the M_HUE_REF and M_SATURATION_REF inquire types are not supported.	
M_GRAB_END_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_END, will run.
M_GRAB_END_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_END, will run.
M_GRAB_FIELD_END_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FIELD_END, will run.
M_GRAB_FIELD_END_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FIELD_END, will run.
M_GRAB_FIELD_START_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FIELD_START, will run.
M_GRAB_FIELD_START_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FIELD_START, will run.
M_GRAB_FRAME_END_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FRAME_END, will run.
M_GRAB_FRAME_END_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FRAME_END, will run.
M_GRAB_FRAME_START_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_FRAME_START, will run.
M_GRAB_FRAME_START_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_FRAME_START, will run.
M_GRAB_IN_PROGRESS	Current grab state: M_YES or M_NO.
M_GRAB_START_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_GRAB_START, will run.

InquireType	Description	
M_GRAB_START_THREAD_ID	The identifier of the thread in which the function, hooked to M_GRAB_START, will run.	
M_GRAB_TIMEOUT	Maximum time to wait for the end of grab before generating an error: M_INFINITE or a value in msec.	
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).	
M_INPUT_SIGNAL_PRESENT	Video input signal present: M_YES or M_NO.	
M_INPUT_SIGNAL_HYSNC_LOCK	Video input signal horizontal synchronization. Returns M_TRUE when a line lock is achieved, otherwise it returns M_FALSE.	
M_INPUT_SIGNAL_COLOR_LOCK	Video input signal color lock. Returns M_TRUE when a color lock is achieved, otherwise it returns M_FALSE.	
M_START_THREAD_HANDLE	The handle of the thread in which the function, hooked to M_START, will run.	
M_START_THREAD_ID	The identifier of the thread in which the function, hooked to M_START, will run.	
M_THREAD_PRIORITY	Hook function priority under Windows:	
	11-15	High priority.
	16 or 22-26	Real-time priority.
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).	
	User Bit #	Expanded video I/O connector signal
	1	USER1IN signal.
	2	USER2IN signal.
	3	USER1OUT signal.
	4	USER2OUT signal.
M_VCR_INPUT_TYPE	VCR input type: M_DISABLE or M_ENABLE.	

MdigReference()

- When grabbing monochrome, you can control the brightness and contrast levels of the input signal using `M_BRIGHTNESS_REF` and `M_CONTRAST_REF`.
- 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`.

MdispAlloc()

- When operating under Windows in single-screen mode, set **DispFormat** to the string "M_DEFAULT". This sets the display format to that of the VGA frame buffer display.

MdispInquire()

- In windowed mode, 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

MdispPan()

- Software pan is fully supported in windowed mode.

MdispZoom()

- Software zooming is fully supported in windowed mode.

MsysAlloc()

- To allocate an Orion system, you must select M_SYSTEM_ORION as the **SystemTypePtr** parameter. This selection opens communication with the board and will automatically allow the system to use some Host memory, if necessary.
- To allocate an Orion system, your display mode must be in an 8-bit or true color (32-bit) display mode (display depth).

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_PHYSICAL_ADDRESS_VGA	The physical address of the VGA frame buffer surface.

Chapter 6: MIL and the Matrox Pulsar platform

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

Matrox Pulsar-specific features

The Matrox Pulsar board is a monochrome PCI frame grabber that features on-board acquisition, display capabilities, and real-time transfers to Host memory. It features a 2-Mbyte main (underlay) frame buffer, and a 2-Mbyte graphics overlay (VGA) frame buffer for non-destructive overlay capabilities. It is important to note that the display adapter can only be configured in 256 colors.

See Appendix A for a data flow diagram.

❖ Note that Matrox Pulsar supports Windows 98 and Windows NT/2000.

Using Matrox Pulsar with MIL

To use your Matrox Pulsar board with MIL, you must allocate a Pulsar system (M_SYSTEM_PULSAR), using ***MsysAlloc()***. This establishes a MIL system environment in which MIL uses some Host memory, the on-board frame buffers and the VGA color keying mechanism of the Matrox Pulsar board to grab and display images (see ***MdispOverlayKey()***).

In this chapter, we discuss the MIL Pulsar system in relation to the following: *16-bit buffer simulated display and Particularities of existing MIL functions on Matrox Pulsar.*

Refer to the *milpul.txt* file in the \MIL (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 Pulsar, the display adapter permits only 8-bit monochrome depth. As with other boards in windowed mode, it is possible to simulate the display of a 16-bit buffer by using the **MdispControl()** function with the `M_VIEW_MODE` control type.

Particularities of existing MIL functions on Matrox Pulsar

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

Commands	Matrox Pulsar particularities
MbufAllocColor()	Limited support for multiple bands.
MbufAlloc2d()	Buffer constraints and options.
MbufControl()	Buffer access constraints.
MbufCreate...()	Required parameter settings.
MbufInquire()	Matrox Pulsar-specific inquire options.
MdigAlloc()	Camera format file specifications.
MdigChannel()	Matrox Pulsar-specific restriction.
MdigControl()	Control type and flag additions.
MdigGrab()	Limitations on buffer size, grab-restart.
MdigGrabContinuous()	Limitations on buffer size.
MdigInquire()	Matrox Pulsar-specific inquire options.
MdigLut()	Input LUT buffer size requirements.
MdigReference()	Valid reference types.
MdispAlloc()	Display format. Overlay setting.
MdispLut()	Available only in dual-screen mode.
MdispPan()	Panning particularities.

Commands	Matrox Pulsar particularities
MdispSelect()	Display particularities.
MdispZoom()	Supported zoom factors.
MsysAlloc()	Matrox Pulsar-specific allocation options.
MsysControl()	Control type and flag additions.
MsysInquire()	Matrox Pulsar-specific inquire options.

MbufAllocColor()

- Although it is possible to allocate a multi-band buffer, its use is limited since Matrox Pulsar is a monochrome frame grabber and can display only in 8-bit resolution.
- Only 8- and 16-bit monochrome buffers can have an M_GRAB attribute. See **MbufAlloc2d()** for other Matrox Pulsar restrictions.

MbufAlloc2d()

- To maximize the use of the memory space available on Matrox Pulsar and to avoid memory organization conflicts, you must allocate your display buffer first, and then the largest buffer(s).
- Only 8- and 16-bit buffers can be allocated with an M_GRAB attribute. Buffers of 16-bit depth can only be displayed in windowed mode.
- Under dual-screen, when setting the **Attribute** parameter to M_IMAGE+M_DISP, the value assigned to the **SizeX** parameter must not be greater than the actual display resolution of the Matrox Pulsar display format (VCF).

MbufControl()

- Buffers allocated with an M_ON_BOARD attribute are mapped to the Host. Due to PCI latency, it is faster to access these buffers as a whole rather than pixel by pixel. This means that processing operations are slower, but copy

operations are faster. This default can be overridden, to not map on the Host, by setting the following **ControlType** to M_DISABLE:

ControlType	ControlFlag
M_MAP_BUFFER_TO_HOST	Can be set to: M_ENABLE (default) or M_DISABLE.

MbufCreate...()

- Set the **Attribute** parameter to an M_IMAGE combination (for example, M_IMAGE + M_DISP + M_GRAB + M_PROC).
- Matrox Pulsar cannot grab into a buffer that crosses a 4-Mbyte memory boundary.
- 8-bit monochrome buffers can have an M_GRAB attribute.
- To grab in Host system memory via the PCI bus, buffers must be allocated in a physically linear memory array (non-paged memory).
- Only the following **ControlFlag** combinations are supported:

ControlFlag	Description
M_PHYSICAL_ADDRESS + M_PITCH	ArrayOfDataPtr is an array of physical addresses. The pitch is in pixels (default).
M_PHYSICAL_ADDRESS + M_PITCH_BYTE	ArrayOfDataPtr is an array of physical addresses. The pitch is in bytes.

MbufInquire()

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

InquireType	Description
M_CURRENT_BUF_ID	Identifier of the buffer in which data is currently being grabbed. Valid only in windowed mode. This information is used, for example, to access grabbed data while a continuous grab (MdigGrabContinuous()) 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.
M_MAP_BUFFER_TO_HOST	Whether on-board buffers are mapped to the Host: (M_ENABLE or M_DISABLE)
M_VALID_GRAB_BUFFER	Whether the buffer is valid for grab or not. Note that a buffer on a 4Mbyte boundary is not valid for the grab. This inquire type returns M_YES if the buffer is valid for the grab operation or M_NO if it is not. If an invalid buffer is used in a grab operation then the grab operation will output an error.

MdigAlloc()

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

The predefined 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, 768x572, 8 bits, 14.8MHz, analog

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

- **MdigChannel()** Matrox Pulsar must have its data signal and synchronization set to the same channel; therefore, you cannot use the M_SIGNAL or M_SYNC flag. Otherwise, an error message will be generated. Instead, use the M_CHX flag to specify the appropriate data signal and synchronization channel.

MdigControl()

- It is not possible to queue asynchronous grabs on Matrox Pulsar. Therefore, M_ASYNCHRONOUS_QUEUED cannot be used as a control value with the M_GRAB_MODE control type.
- The supported scaling factor combinations for the M_GRAB_SCALE... control types are as follows:

X	Y
1.0	1.0
1.0	0.5
0.5	1.0
0.5	0.5
0.25	0.25

- For M_GRAB_TRIGGER_SOURCE, note the meaning of the following **ControlValue** settings on a Matrox Pulsar:

M_HARDWARE_PORT0	Use TTL hardware trigger signal. Pin 1 on Video Input Connector.
M_HARDWARE_PORT1	When using the 37-pin connector on the ribbon cable: if the DCF detects a TTL signal, pin 8 is used. If it is an RS-422 signal, it uses a combination of pin 17 and pin 35. When using the 68-pin I/O connector of the RS-422 digital interface board, a combination of pin 27 and pin 61 is used.

- Matrox Pulsar supports these additional **ControlType** parameter settings:

ControlType	Description & ControlValue	
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_BYPASS	Activate the manual or automatic exposure model:	
	M_ENABLE	Manual exposure model.
	M_DISABLE	Automatic exposure model.
	M_DEFAULT	Same as M_DISABLE.
M_GRAB_EXPOSURE_MODE	Select the exposure signal polarity: M_LEVEL_HIGH, M_LEVEL_LOW, or M_DEFAULT (same as .dcf).	
M_GRAB_EXPOSURE_SOURCE	Select the exposure source: M_SOFTWARE, M_HARDWARE_PORT0, M_HARDWARE_PORT1, M_VSYNC, M_HSYNC, M_CONTINUOUS, or M_TIMER1.	
M_GRAB_EXPOSURE_TIME	Set the grab exposure time (in nsec) after exposure start and before active exposure (if 0, exposure is disabled), or M_DEFAULT (same as .dcf). Note, an error might occur if unable to generate the specified exposure time.	

ControlType	Description & ControlValue		
M_GRAB_EXPOSURE_TIME_DELAY	Set the delay (in nsec) between the trigger and the start of exposure or M_DEFAULT (same as .dcf). Note, an error might occur if unable to generate the specified delay.		
M_GRAB_FAIL_CHECK	Set whether and when to log a grab-fail error. Can be set to:		
	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 (default).	
M_GRAB_FAIL_RETRY_NUMBER	Set the number of retries when a field or frame grab fails. Can be set to the required number (default is 1).		
M_GRAB_INPUT_GAIN	Select the input-signal gain. Valid input voltages and their corresponding gains are:		
		Input Voltage	Gain
	M_GAIN0	2.14 - 2.86 Vpp	1.25
	M_GAIN1	1.43 - 2.14 Vpp	1.66
	M_GAIN2	1.0 - 1.43 Vpp	2.5
	M_GAIN3 (default)	0.71 - 1.0 Vpp	3.5
	M_GAIN4	0.0 - 0.71 Vpp	5.0
M_GRAB_LUT_PALETTE	Select the input-LUT palette: M_LUT_PALETTE0, M_LUT_PALETTE1, M_LUT_PALETTE2, M_LUT_PALETTE3, or M_DEFAULT. No palette is available when grabbing in digital mode.		

ControlType	Description & ControlValue	
M_GRAB_TIMEOUT	Set the maximum time to wait for a frame before generating an error.	
	M_DEFAULT	Determined by the frame period.
	M_INFINITE	Wait indefinitely. This is recommended only for triggered cameras.
	value in msec	Specify time for wait.
M_THREAD_PRIORITY	Set the hook function priority under Windows. Valid values are:	
	11 - 15	High priority.
	16 or 22 - 26	Real-time priority.
	If you want to change the hook function priority, use MdigInquire() to determine its current value. Changing these priority settings will affect the overall application priority, due to a Windows restriction.	
M_USER_BIT+(1...5)	Set the state of the specified output bit on the digital connector or the RS-422 module (if applicable): M_ON or M_OFF	

The relationship between the MIL user-bit number and the actual user output bit on the TTL/RS422 digital connector is shown in the table below:

User Bit	Matrox Pulsar TTL/RS-422 digital connector signal output
1	CTRL_USR1 (TTL, or RS-422 if digital module present).
2	CTRL_USR2 (TTL, or RS-422 if digital module present).
3	TTL_USR0 on RS-422 module connector.
4	TLL_USR1 on RS-422 module connector.
5	TLL_USR2 on RS-422 module connector.

MdigGrab()

- When performing a grab, the horizontal position and size of the grab destination buffer must be set to a multiple of 4 divided by the scaling factor. For example, for a scaling factor of 0.5, set the horizontal position and size to a multiple of 8; for a scaling factor of 0.25, to a multiple of 16. If this is not done, the image will be internally clipped within the specified grab region.
- When a field or frame drop (grab failure) is detected, MIL will retry grabbing for the number of times specified with **MdigControl()**.

- When performing real-time grabs, Windows 98 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 the 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.

MdigGrabContinuous()

- The limitations on **MdigGrab()** apply to this function as well.

MdigInquire()

- Matrox Pulsar does not support 2, 3, or 4 input color bands. Accordingly, there are restrictions on the possible values returned when using the M_SIZE_BAND inquire type.
- The **InquireType** parameter can also be set to the following:

InquireType	Description
M_FIELD_START_THREAD_HANDLE	Handle of the thread in which the function(s) hooked to M_GRAB_START, M_GRAB_FRAME_START, M_FRAME_START, M_FIELD_START_ODD, M_FIELD_START_EVEN, and M_FIELD_START will run.
M_FIELD_START_THREAD_ID	Identifier of the thread in which the function(s) hooked to M_GRAB_START, M_GRAB_FRAME_START, M_FRAME_START, M_FIELD_START_ODD, M_FIELD_START_EVEN, and M_FIELD_START will run.

InquireType	Description
M_GRAB_EXPOSURE	Exposure timer state for non-software trigger source: M_ENABLE or M_DISABLE.
M_GRAB_EXPOSURE_BYPASS	Exposure bypass mode: M_ENABLE or M_DISABLE.
M_GRAB_EXPOSURE_MODE	Exposure signal polarity: M_LEVEL_HIGH or M_LEVEL_LOW.
M_GRAB_EXPOSURE_SOURCE	Exposure trigger source: M_SOFTWARE, M_HARDWARE_PORT0, M_HARDWARE_PORT1, M_VSYNC, M_HSYNC, M_CONTINUOUS, M_TIMER1.
M_GRAB_EXPOSURE_TIME	Grab exposure time (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_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, else M_NO.
M_GRAB_FIELD_END_EVEN_THREAD_HANDLE	Handle of the thread in which the function(s) hooked to M_GRAB_FIELD_END_EVEN will run. If M_GRAB_START_MODE is set to M_FIELD_START_ODD, this is also the thread in which the function(s) hooked to M_GRAB_END, M_GRAB_FIELD_END, and M_GRAB_FRAME_END will run.
M_GRAB_FIELD_END_ODD_THREAD_HANDLE	Handle of the thread in which the function(s) hooked to M_GRAB_FIELD_END_ODD will run. If M_GRAB_START_MODE is set to M_FIELD_START_EVEN, this is also the identifier for the thread in which the function(s) hooked to M_GRAB_END, M_GRAB_FIELD_END, and M_GRAB_FRAME_END will run.
M_GRAB_FIELD_END_EVEN_THREAD_ID	Identifier of the thread in which the function(s) hooked to M_GRAB_FIELD_END_EVEN will run. If M_GRAB_START_MODE is set to M_FIELD_START_ODD, this is also the identifier for the thread in which the function(s) hooked to M_GRAB_END, M_GRAB_FIELD_END, and M_GRAB_FRAME_END will run.

InquireType	Description	
M_GRAB_FIELD_END_ODD_THREAD_ID	Identifier of the thread in which the function(s), hooked to M_GRAB_FIELD_END_ODD, will run. If M_GRAB_START_MODE is set to M_FIELD_START_EVEN, this is also the identifier for the thread in which the function(s) hooked to M_GRAB_END, M_GRAB_FIELD_END, and M_GRAB_FRAME_END will run.	
M_GRAB_IN_PROGRESS	Current grab state: M_YES or M_NO.	
M_GRAB_INPUT_GAIN	The gain that is applied to the input signal: M_GAIN0, M_GAIN1, M_GAIN2, M_GAIN3.	
M_GRAB_LUT_PALETTE	Input-LUT palette: M_LUT_PALETTE0, M_LUT_PALETTE1, M_LUT_PALETTE2, M_LUT_PALETTE3. No palette is available when grabbing in digital mode.	
M_GRAB_TIMEOUT	The maximum time to wait for the end of the grab before generating an error: M_INFINITE, or a value in msec.	
M_INPUT_SIGNAL_SOURCE	Input-signal source: M_HARDWARE_PORT0 (analog input connector) or M_HARDWARE_PORT1 (digital input connector).	
M_THREAD_PRIORITY	The hook function priority under Windows.	
	11-15	High priority.
	16 or 22-26	Real-time priority.
M_USER_BIT+(0...5)	Current state of the input bit of the digital input connector: M_ON (1) or M_OFF (0).	
	User Bit	Matrox Pulsar TTL/RS422 digital connector signal output
	0	CTRL_USR0 on RS-422 module connector.
	1	CTRL_USR1 (TTL, or RS-422 if digital module present)
	2	CTRL_USR2 (TTL, or RS-422 if digital module present)
	3	TTL_USR0 on RS-422 module connector.
	4	TLL_USR1 on RS-422 module connector.
	5	TLL_USR2 on RS-422 module connector.

InquireType	Description
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 data, input mode, and destination image buffer:

Input data	Input mode	Image buffer	Size of the LUT buffer		Notes
8-bit	digital ¹	8-bit	256-entry	8-bit	
		16-bit (in LSB)	256-entry	16-bit	This is a true 8-bit LUT.
		RGB 8:8:8	3x256-entry	8-bit	
10-bit	analog	8-bit	256-entry	8-bit or 1024-entry	- For 256-entry, the 8 MSB of the 10 bits are used. - For 1024-entry, each value must occupy 4 consecutive entries of the LUT buffer.
		16-bit (in LSB)	1024-entry	16-bit	This is a true 10-bit LUT.
		RGB 8:8:8	3x1024-entry	8-bit	
	digital ¹	8-bit	1024-entry	8-bit	By default, the 8 MSB of the 10 bits are used.
		16-bit (in LSB)	1024-entry	16-bit	This is a true 10-bit LUT.
		RGB 8:8:8	3x1024-entry	8-bit	
12-bit	digital ¹	8-bit	4096-entry	8-bit	By default, the 8 MSB of the 12 bits are used.
		16-bit (in LSB)	4096-entry	16-bit	This is a true 12-bit LUT.
		RGB 8:8:8	3x4096-entry	8-bit	

Input data	Input mode	Image buffer	Size of the LUT buffer	Notes
16-bit or 14-bit	digital ¹	16-bit (in MSB or 14-bit (in LSB)	256-entry 16-bit	The most(16-bit)/least(14-bit) significant byte of this LUT buffer maps the most/least significant byte of the input data. ²
¹ When in digital mode, only one LUT is possible at any time. ² This occurs because, in this mode, Matrox Pulsar's LUTs are configured as two 256-entry 8-bit input LUTs (one for each byte), not one 64K 16-bit input LUT.				

MdigReference()

- Since Matrox Pulsar does not support color grabbing, the valid **ReferenceType** settings do **not** include:
M_BRIGHTNESS_REF, M_CONTRAST_REF, M_HUE_REF, M_SATURATION_REF.
- When grabbing, you can control the black and white reference levels.
- For M_BLACK_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 1.25 V.
- For M_WHITE_REF, note the meaning of the **ReferenceLevel** parameter settings:
 - The minimum voltage level (M_MIN_LEVEL) corresponds to 1.25 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 32 distinct adjustment (adjustments of 4.88 mV each).

MdispAlloc()

- When operating under Windows in single-screen mode, set **DispFormat** to the string "M_DEFAULT". This sets the main (underlay) frame buffer display format to that of the on-board overlay (VGA) frame buffer display.

When operating under dual screen mode, set **DispFormat** to the required display resolution for the VGA display. Possible settings are:

Display Format	Display Resolution
"320x200x8PP"	320x200 at 60Hz
"640x400x8PP"	640x400 at 60Hz
"640x480x8PP"	640x480 at 60Hz
"800x600x8PP"	800x600 at 60Hz
"1024x768x8PP"	1024x768 at 60Hz
"1280x1024x8PP"	1280x1024 at 60Hz
"1600x1200x8PP"	1600x1200 at 60Hz

- On the standard Matrox Pulsar board, in dual-screen mode, additional settings are available:

Display Format	Display Resolution
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 \MIL (user-specified) directory for the current list of video configuration format (VCF) files.

MdispLut()

- In single screen mode, a LUT can only be associated with a display that has the M_OVR attribute.

- You can associate a display with a monochrome (1 band x 8 bit x 256 entries) or a pseudo-color (3 bands x 8bit x 256 entries) LUT buffer.

MdispPan()

- Software panning is fully supported in windowed mode.
- The following restrictions apply, when performing a hardware pan (in non-windowed mode, this is the only type of panning that is supported. In windowed mode, you can enable hardware panning with *MdispControl()*):
 - The hardware pan (XOffset) must be a multiple of 4 (for example, 0, 4, 8, 12, or 16).
 - Any region beyond the image buffer's boundaries cannot be displayed.
 - Although panning is supported in single screen mode, it follows that of the VGA; that is, when you pan the Matrox Pulsar display, the VGA is panned as well, and vice versa.

MdispSelect()

Although Matrox Pulsar has an 8-bit display resolution, you can display a 16-bit buffer by extracting 8 bits from a 16-bit image, copying them into a monochrome 8-bit display buffer (for example, with *MimShift()*), and then displaying that 8-bit buffer. Alternatively, you can display all 16-bits, represented as two side-by-side 8-bit pixels or display any consecutive 8-bit group within the 16 bits. These options are available using the M_VIEW_MODE control type (refer to *MdispControl()*).

MdispZoom()

- Software zooming is fully supported in windowed mode.
- The following restrictions apply, when performing a hardware zoom (in non-windowed mode, this is the only type of zoom that is supported. In windowed mode, you can enable hardware zoom with *MdispControl()*):
 - In the x and y direction, only hardware zoom factors of 1, 2, and 4 are supported.

- Although hardware zoom is supported in single screen mode, it follows that of the VGA, that is, if you select hardware zoom (using the *MdispControl()* function), the VGA shares the same zoom, and vice versa.
- If you are using single-screen mode under Windows:
 - You cannot use the zoom factor of 4 in the 640 x 480 and 800 x 600 resolutions.
 - Due to hardware limitations, when using the zoom factor of 4 with a 1152 x 882 x 8 resolution, a 32-pixel band of spurious data appears at the right border of the screen.
- In multi-head mode, only the screen where the cursor is positioned appears zoomed. To zoom the other screen, move the cursor to that screen.

MsysAlloc()

- To allocate a Pulsar system, you must select `M_SYSTEM_PULSAR` as the **SystemTypePtr** parameter. This selection opens communication with the board and will automatically enable the system to use some Host memory, if necessary.
- You can disable the use of interrupts by setting **InitFlag** to `M_NO_INTERRUPT` when allocating the MIL Pulsar system. Note that, by doing so, you will run with limited grab functionality.
- ❖ You can use this flag if your Matrox Pulsar board does not function properly during grab operations. This will allow you to determine if there is a conflict with the assigned interrupts. Refer to the *Troubleshooting* appendix of the *Matrox Pulsar Hardware and Installation Manual* for more information.

MsysControl()

- To modify the transfer rate and performance of the PCI bus, the Pulsar system supports these additional combinations for the **ControlType** and **ControlValue** parameters:

ControlType	Description and ControlValue
M_PCI_BRIDGE_LATENCY or M_PCI_LATENCY	Controls the length of the PCI bus transfer-time slice that is allocated to Matrox Pulsar. Increasing it can improve transfer time when there is heavy traffic on the PCI bus. However, you should be aware that this can affect the performance of the other bus masters (including the main CPU). Possible values are: 0 - 127. Use <i>MsysInquire()</i> to determine the default value on your system. It is recommended that the chosen value be one near the default value.
M_FAST_PCI_TO_MEM	The M_FAST_PCI_TO_MEM type (available on some PC systems), when enabled, ensures the fastest transfer between the PCI bus and Host memory. If fastest transfer was not already established automatically by the PC's system BIOS at Host start-up, enabling this control can result in system unreliability. Therefore, you should disable this control when it is no longer needed. Possible value are: M_ENABLE or M_DISABLE.

- MIL can perform a continuous grab operation live in the overlay (VGA) frame buffer when the display is in windowed mode. When necessary, the grab will switch to pseudo-live (simulating a live grab by grabbing into the Host buffer and updating the display) to prevent the grab from overwriting another window. If there is an instance when automatic live-to-pseudo-live switching does not happen or you want to override the default behavior, you can use the following

ControlType and **ControlValue** parameters settings.
When grabbing into the underlay, the following control types should be left to their default setting.

ControlType	ControlValue
M_LIVE_GRAB	Set whether to perform a live grab whenever possible or to force a pseudo-live grab into displayable image buffers.
	M_ENABLE Live whenever possible (default).
	M_DISABLE Force pseudo-live.
M_PSEUDO_LIVE_GRAB_WHEN_OVERLAPPED	Set whether to pause the grab or switch to a pseudo-live grab when the live grab is interrupted due to one of the above-mentioned conditions (for example, if the window displaying the grab is overlapped by a menu).
	M_ENABLE Pseudo-live (default).
	M_DISABLE Pause grab.
M_STOP_LIVE_GRAB_WHEN_DISABLED	Set whether or not to switch to a pseudo-live grab while the display window is disabled (for example, when a pop-up dialog box is opened):
	M_ENABLE Pseudo-live while the display window is disabled (default).
	M_DISABLE Force live.
M_STOP_LIVE_GRAB_WHEN_INACTIVE	Set whether or not to switch to a pseudo-live grab while the display window is inactive (that is, while it does not have the focus):
	M_ENABLE Pseudo-live while the display window is inactive (default).
	M_DISABLE Force live.
M_STOP_LIVE_GRAB_WHEN_MENU	Set whether or not to switch to a pseudo-live grab while an opened menu overlaps the display window:
	M_ENABLE Pseudo-live while menu overlaps the display window (default).
	M_DISABLE Force live.

- When the display is in windowed mode (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.

- In multi-head mode, you can force pseudo-live grabbing when a grab is displayed from a board other than the one performing the grab. To do so, set the following **ControlType** to M_ENABLE:

ControlType	ControlValue	
M_FORCE_PSEUDO_IN_NON_UNDERLAY_DISPLAYS	Can be set to:	
	M_ENABLE	Force pseudo.
	M_DISABLE	Grab live whenever possible (default).

- 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	Use double buffering when displaying a pseudo-live grab operation under Windows:	
	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 grab window is displaced, the grab is stopped and restarted at every displacement. When grabbing from a triggered camera, a trigger is probably not issued as often as the window is displaced. To avoid having an empty window, a copy is performed from the old location to the new before restarting the grab. To override this default, set the following **ControlType** to M_DISABLE:

ControlType	ControlValue	
M_LIVE_GRAB_MOVE_UPDATE	Can be set to:	
	M_ENABLE	Perform a copy between the windows. Default for triggered cameras.
	M_DISABLE	Don't perform a copy between the windows. Default for non-triggered cameras.

- 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	Can be set to:	
	M_ENABLE	Default for hardware 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.

- During a live grab operation in windowed mode, the MIL driver keeps track of the grab window. This default can be overridden by setting the following **ControlType** to M_DISABLE:

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

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_PULSAR, M_PULSAR_WITH_RS422 or M_PULSAR_RS422_J16.
M_DISPLAY_DOUBLE_BUFFERING	The state of double buffering.
M_FAST_PCI_TO_MEM	State of the fast-PCI-to-memory flag.
M_FORCE_PSEUDO_IN_NON_UNDERLAY_DISPLAYS	Pseudo-live grab performed in non-Pulsar-driven screen: M_ENABLE or M_DISABLE.
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	A live grab (not pseudo-live) is performed: M_ENABLE or M_DISABLE.
M_LIVE_GRAB_END_TRIGGER	Generate automatic trigger at end of grab: M_ENABLE or M_DISABLE.

InquireType	Description
M_LIVE_GRAB_MOVE_UPDATE	Copy performed between windows on update: M_ENABLE or M_DISABLE.
M_LIVE_GRAB_TRACK	Whether or not MIL keeps track of the live-grab window: M_ENABLE or M_DISABLE.
M_NATIVE_ID	The native identifier (handle) of the system.
M_PCI_BRIDGE_LATENCY or M_PCI_LATENCY	PCI latency count.
M_PHYSICAL_ADDRESS_UNDERLAY	The physical address of the underlay frame buffer surface.
M_PSEUDO_LIVE_GRAB_WHEN_OVERLAPPED	A switch is made to a pseudo-live grab when the display window is overlapped by another window: M_ENABLE or M_DISABLE.
M_STOP_LIVE_GRAB_WHEN_DISABLED	Grabbing is frozen when the display window is disabled: M_ENABLE or M_DISABLE.
M_STOP_LIVE_GRAB_WHEN_INACTIVE	Grabbing is frozen when the display window is inactive: M_ENABLE or M_DISABLE.
M_STOP_LIVE_GRAB_WHEN_MENU	Grabbing is frozen when the display window is overlapped by a menu: M_ENABLE or M_DISABLE.

Chapter 7: MIL and the Matrox 4Sight platform

This section discusses features of MIL that are distinct to the Matrox 4Sight platform and ways that optimize the board's performance.

Matrox 4Sight-specific features

Matrox 4Sight is a self-contained platform that integrates 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.

Matrox 4Sight is compliant with the PC-104/*Plus*[™] form factor. The integrated unit version of Matrox 4Sight encloses a Matrox Meteor-II /Standard or Meteor-II /Multi-Channel frame grabber. Both frame grabbers support the optional Matrox Meteor-II MJPEG module for PC/104-*Plus*. Refer to the *MIL and the Matrox Meteor-II platform* chapter in this manual for details on the capabilities of Matrox Meteor-II frame grabbers for PC/104-*Plus*.

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). The video assist unit in the processor's companion chip allows for non-destructive, live graphics and video overlay display. Therefore, the Matrox 4Sight supports the DirectDraw underlay-surface display architecture. When using the DirectDraw underlay-surface display architecture, Matrox 4Sight can dynamically allocate an underlay frame buffer surface that is in either RGB16 or YUV16 packed format.

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

Matrox 4Sight has discrete digital and serial I/Os for operating a PLC or motion control unit. Matrox 4Sight also includes two RS-232 serial ports for I/O, as well as 20 auxiliary I/O pins with interrupt-generation capabilities.

The Matrox 4Sight motherboard integrates an IEEE 1394 serial bus controller with three IEEE 1394 ports. Currently, serial data transfer rates of up to 400 Mbits per second (bps)

are possible. Despite being an integrated component on the motherboard, this 1394 serial bus controller provides identical functionality as the Matrox Meteor-II /1394. Refer to the *MIL and the Matrox Meteor-II platform* chapter in this manual for details.

See Appendix A for a data flow diagram.

- ❖ It is important to note that Windows 98 is not supported on the Matrox 4Sight platform, although Windows NT/2000 is supported on the Matrox 4Sight platform.

Using Matrox 4Sight with MIL

MIL treats Matrox 4Sight as a normal PC platform. Accordingly, you don't allocate Matrox 4Sight as a system. However, to use a Matrox Meteor-II frame grabber in a Matrox 4Sight, you must first allocate the board as a Meteor-II system. To use 1394-compliant cameras attached to the 1394 serial ports, you must allocate Meteor-II /1394 systems. Then assign a camera from the pool of 1394 cameras to a particular system, by allocating it as a digitizer on that system. See *Chapter 4: MIL and the Matrox Meteor-II platform* for details on allocating a Meteor-II /1394 system, Meteor-II /1394 digitizer, and managing multiple 1394 cameras.

Once you allocate the appropriate system(s) for Matrox 4Sight, you establish a MIL system environment in which MIL uses the Matrox Meteor-II board, Host computer memory, the integrated VGA, and, in the case of Meteor-II /1394, the system's assigned set of 1394-compliant cameras.

This chapter relates the Matrox 4Sight platform to the following: *Auxiliary input/outputs*, *Particularities of existing MIL functions on Matrox 4Sight*, and *Matrox 4Sight-specific MIL functions*.

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

Auxiliary inputs/outputs

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 platform. For example, on Matrox 4Sight, 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 system. 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, carry out 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 User Guide* 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

Certain functions have special features or functionality on a Matrox 4Sight. This table provides an overview of the affected commands. Details and procedures are described in the individual command descriptions, which follow. Note, for information on the integrated 1394 serial bus controller and ports, refer to the *Matrox Meteor-II /1394* features described in the *Matrox Meteor-II platform* chapter for details. In addition, for details on the particularities of the optional Matrox Meteor-II boards and Matrox Meteor-II MJPEG module, refer to *Chapter 4: MIL and the Matrox-Meteor-II platform*.

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

MsysControl()

To control a specific user bit or a group of user bits on Matrox 4Sight, 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 + (0-19) 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 ControlValue bit set to 1.	Set the user bit to 1.
	M_OFF or ControlValue bit set to 0.	Set the user bit to 0. (default)
M_USER_BIT_MODE + (0-19) or + M_BIT_MASK(bit-encoded mask value)	Set the system's specified user bit(s) input/output mode.	
	M_INPUT or ControlValue bit set to 0.	Set the user bit to input. (default)
	M_OUTPUT or ControlValue bit set to 1.	Set the user bit to output.
M_USER_BIT_INTERRUPT_STATE + (0-19) 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 ControlValue bit set to 1.	Enable interrupt.
	M_DISABLE or ControlValue bit set to 0.	Disable interrupt. (default)

ControlType	ControlValue	Description
M_USER_BIT_INTERRUPT_MODE + (0-19) 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 ControlValue bit set to 0.	Set interrupt mode to rising edge mode. (default)
	M_EDGE_FALLING or ControlValue bit set to 1.	Set interrupt mode to falling edge mode.

MsysInquire()

To inquire the state of a specific user bit or a group of user bits on Matrox 4Sight, specify the appropriate **InquireType** value, and read its inquire value from the user variable (**UserVarPtr**). To inquire about a single user bit, add its 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. If a group of user bits is specified, a bit-encoded inquire value is returned in the **UserVarPtr** parameter; in this case, **UserVarPtr** must be a pointer to a long. Note that for these inquire types, only M_DEFAULT_HOST or a Host system can be used as a target system (**SystemId**).

InquireType	Description
M_USER_BIT_VALUE + (0-19) 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.
M_USER_BIT_MODE + (0-19) 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.

InquireType	Description	
M_USER_BIT_INTERRUPT_STATE + (0-19) 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 + (0-19) 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-specific MIL functions

There are Matrox 4Sight-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 particularities
MsysGetHookInfo()	4Sight-specific hook event inquire.
MsysHookFunction()	4Sight-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()**).

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. The possible values are 0 to 19, each corresponding to one of the 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 (MsysHookFunction()) will be executed for each interrupt.

The **UserVarPtr** parameter specifies the address of the variable in which the requested information is to be written.

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.

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:

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 MsysControl()). 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 MsysGetHookInfo() from within your hook-handler function.

HookType	Description
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 MsysGetHookInfo() 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 8: MIL and the VGA platform

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

MIL functions under Windows

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.

Mdig...()

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

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

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

Using the Windows API functions

Using the Windows API function 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.

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 **MdispInquire()**. 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 function, 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 `\MIL\EXAMPLES` directory.

VGA board-specific functions

The following describes the MIL VGA system-specific functions. These functions are an extension to the regular MIL function set, and are only available under Windows. You must include the *windows.h* file before using these functions.

These functions have not been replaced by regular functions in the **Mdisp** module. They are 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.

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, 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.

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 MdispSelect() or MDispSelectWindow()).
M_MODIFIED_DIB_DESTRUCTION	An image has been disassociated from the display (using MdispDeselect()).
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.

Returned 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:

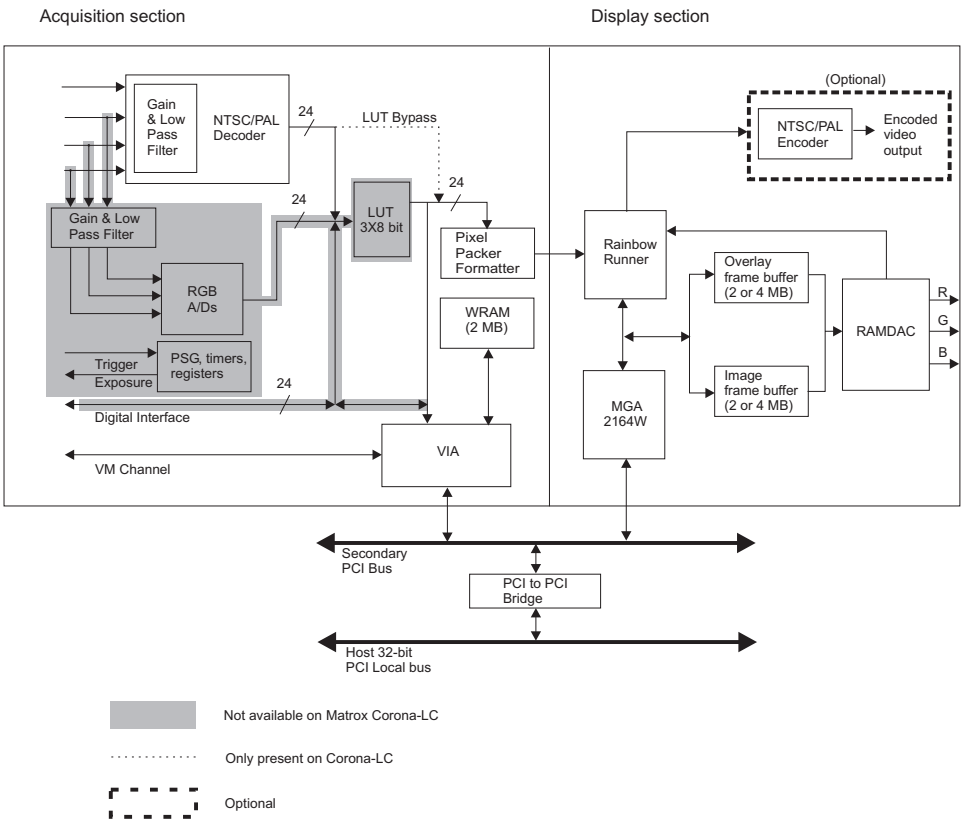
MVGAHOOKFCTPTR OldHookFctPtr;

See also MdispSelect(), MdispSelectWindow()

Appendix A: Board flow diagrams

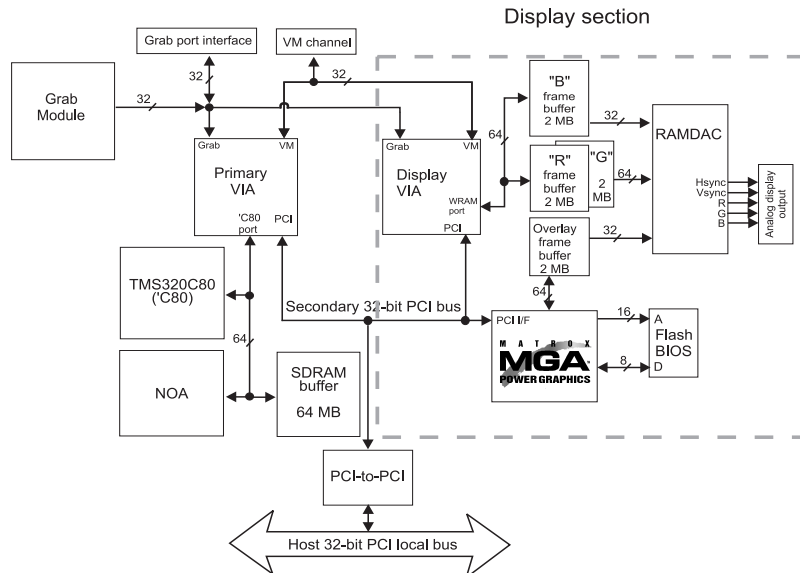
This section contains data flow diagrams for all Matrox frame grabbers.

Matrox Corona

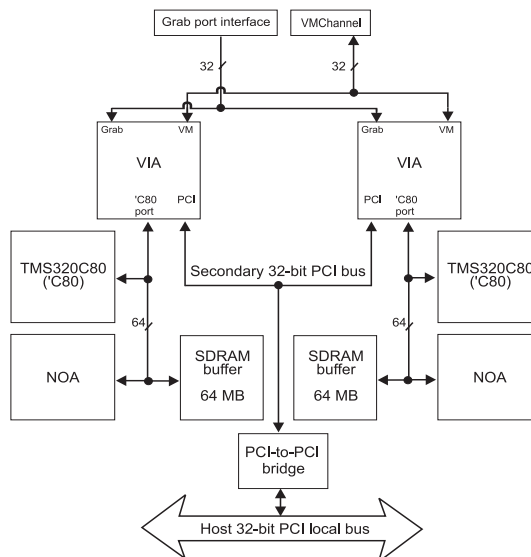


Matrox Genesis

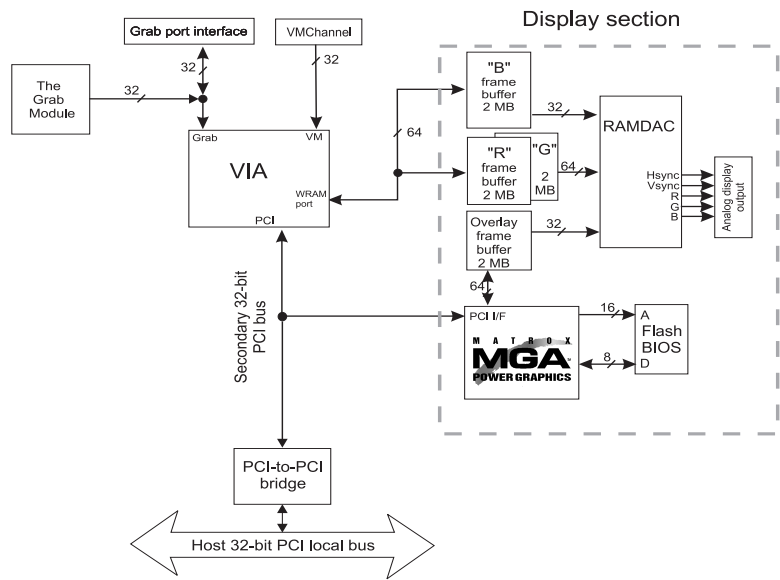
Genesis Main Board



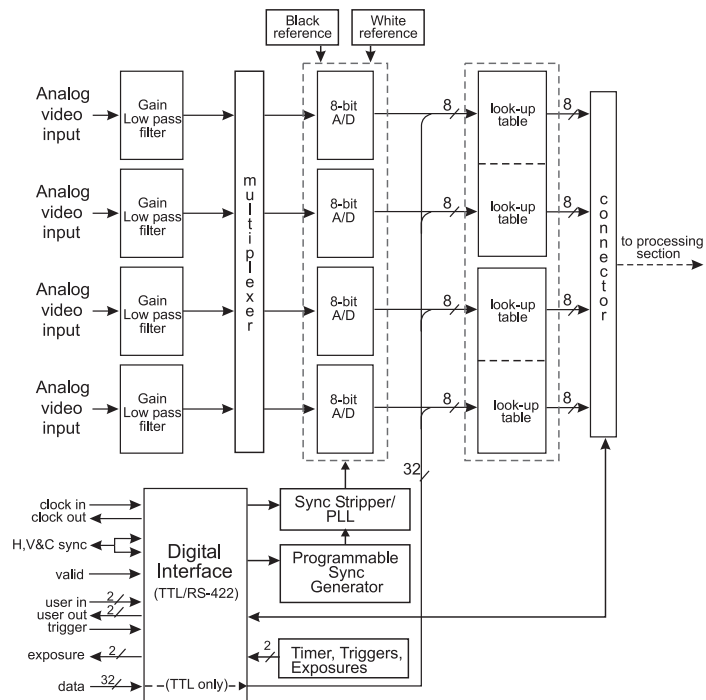
Genesis Processor Board



Genesis-LC

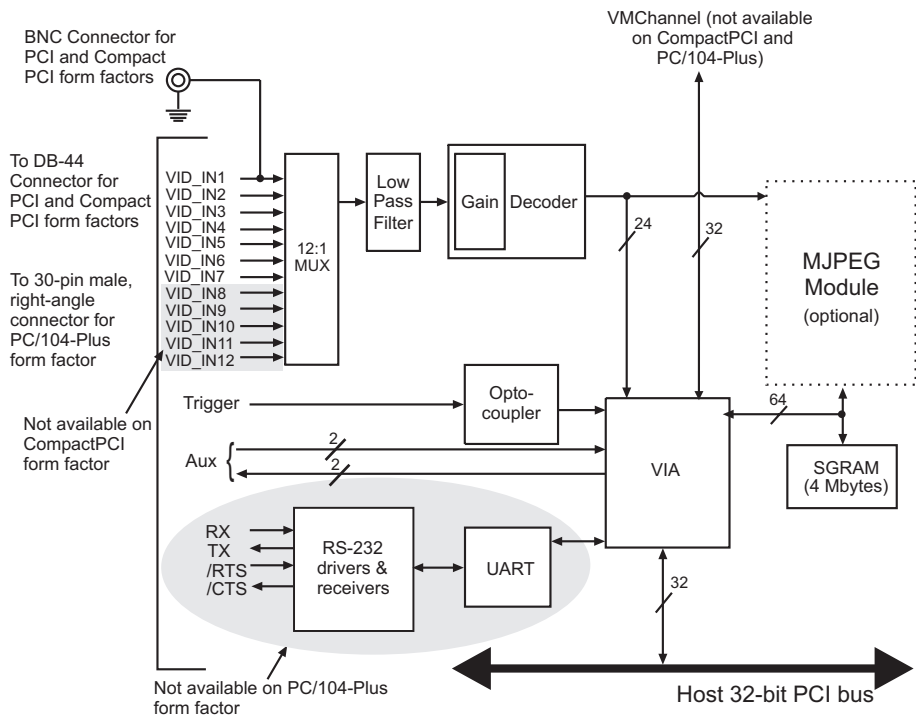


Grab module

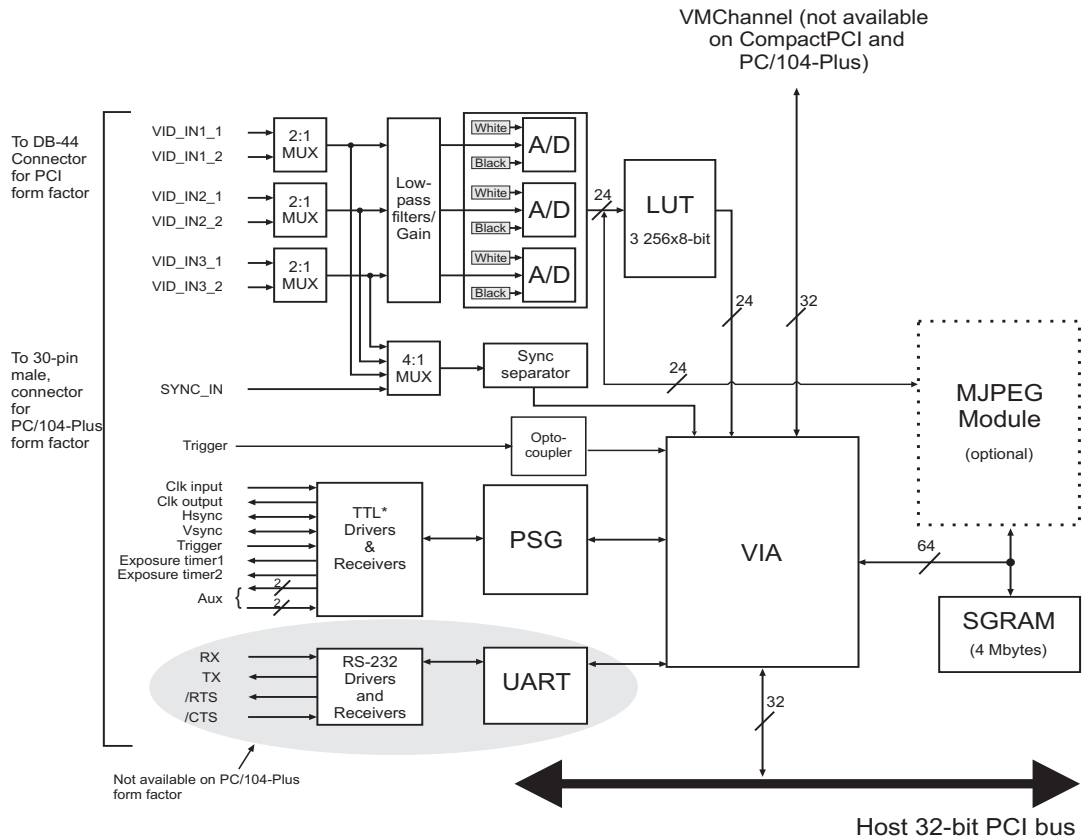


Matrox Meteor-II

Matrox Meteor-II /Standard

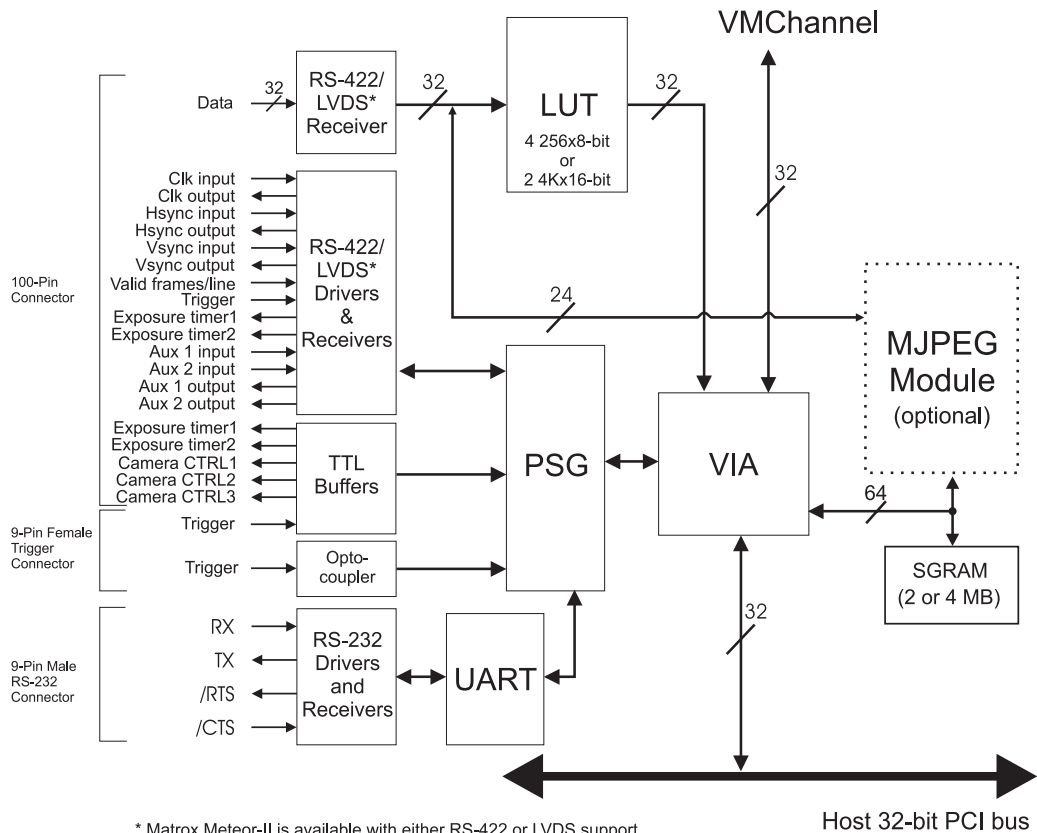


Matrox Meteor-II /Multi-Channel

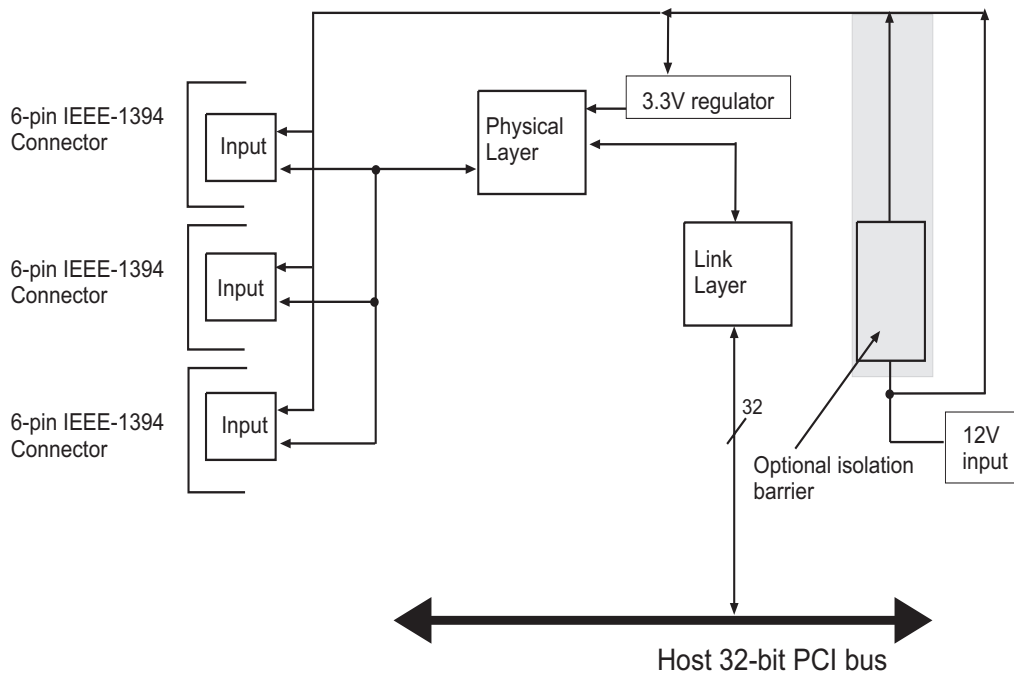


* RS-422 version of these signals are available on the optional RS-422 connector.
This connector is only available on Matrox Meteor-II /Multi-Channel in a PCI form factor.

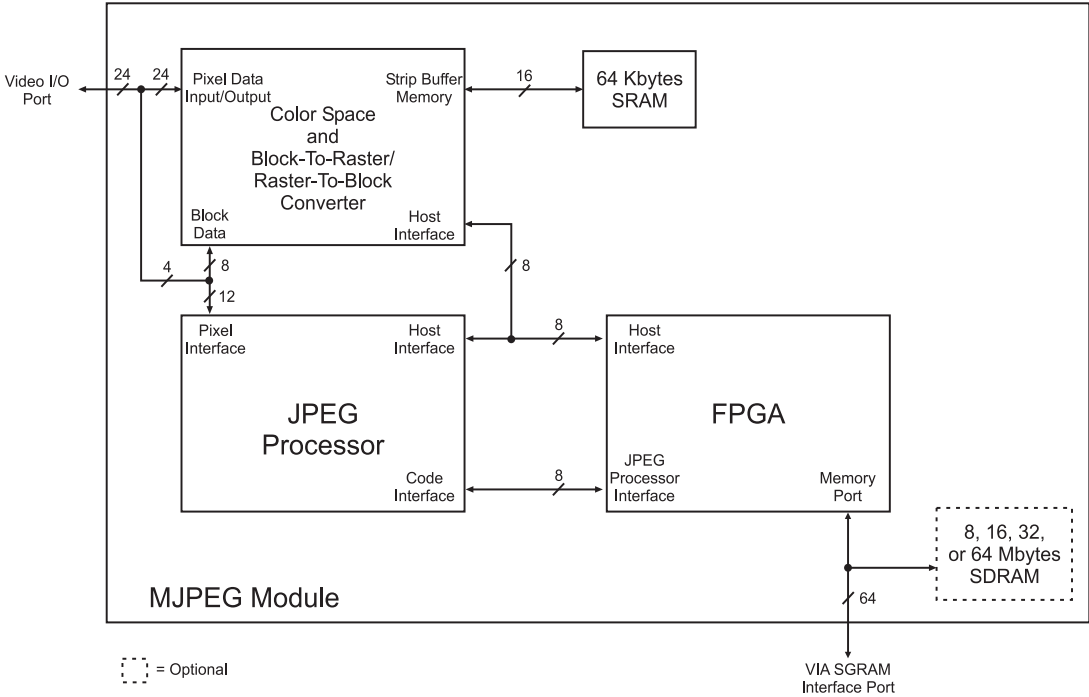
Matrox Meteor-II /Digital



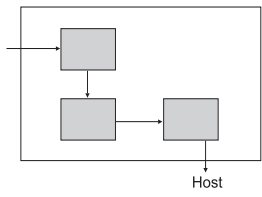
Matrox Meteor-II /1394



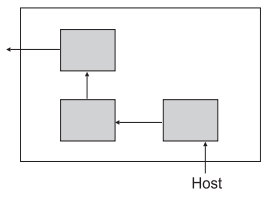
Matrox Meteor-II /MJPEG Module



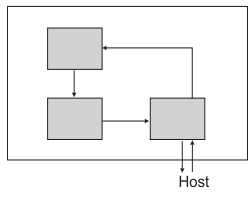
MJPEG compression



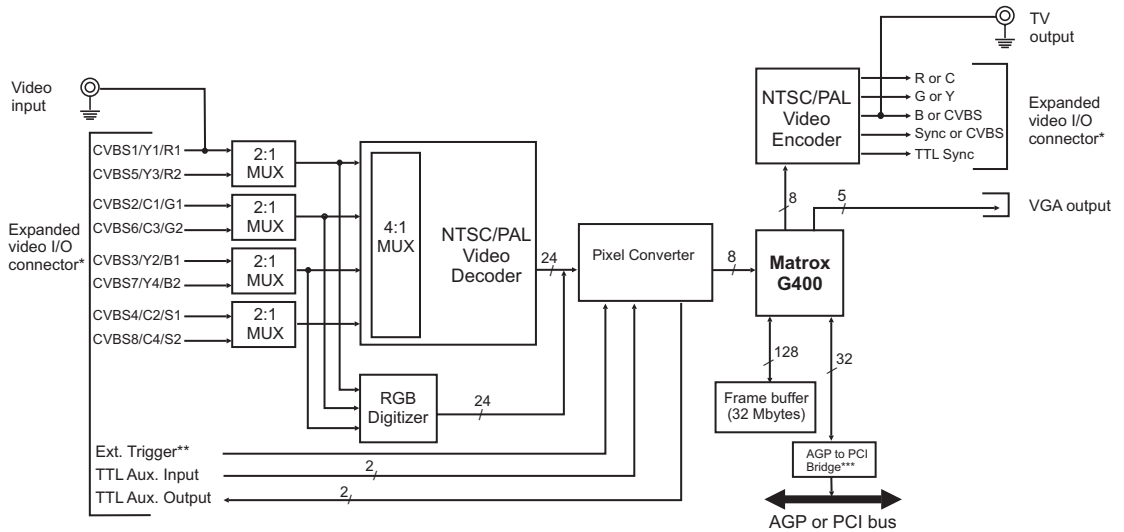
Decompression



JPEG compression



Matrox Orion

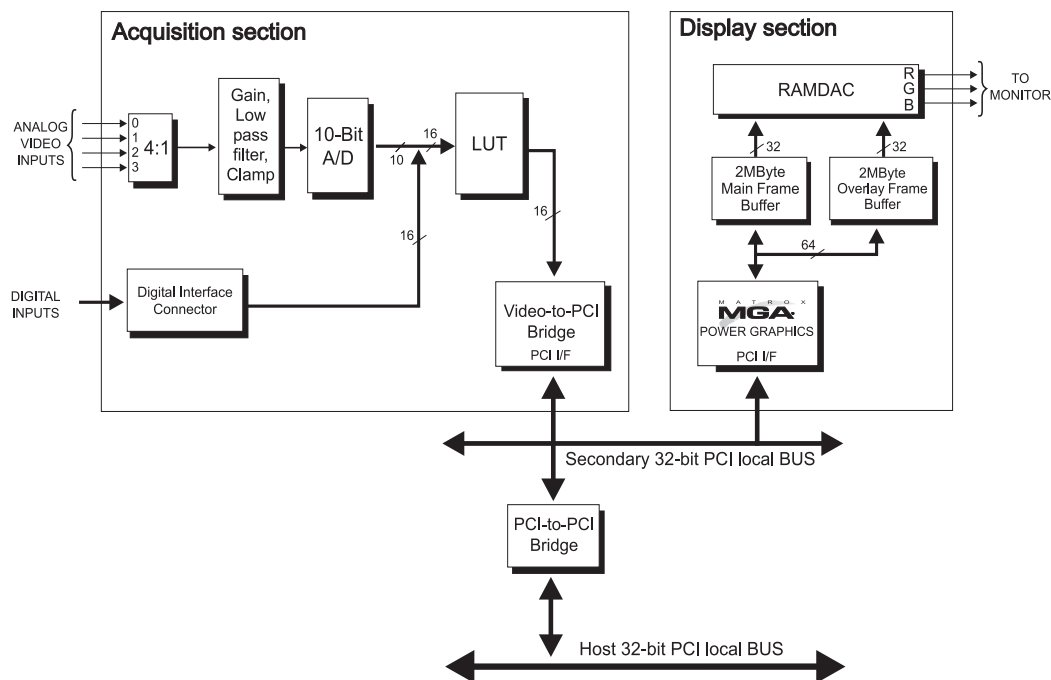


* The expanded video I/O connector is used for both inputs and outputs, and is located on a separate bracket.

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

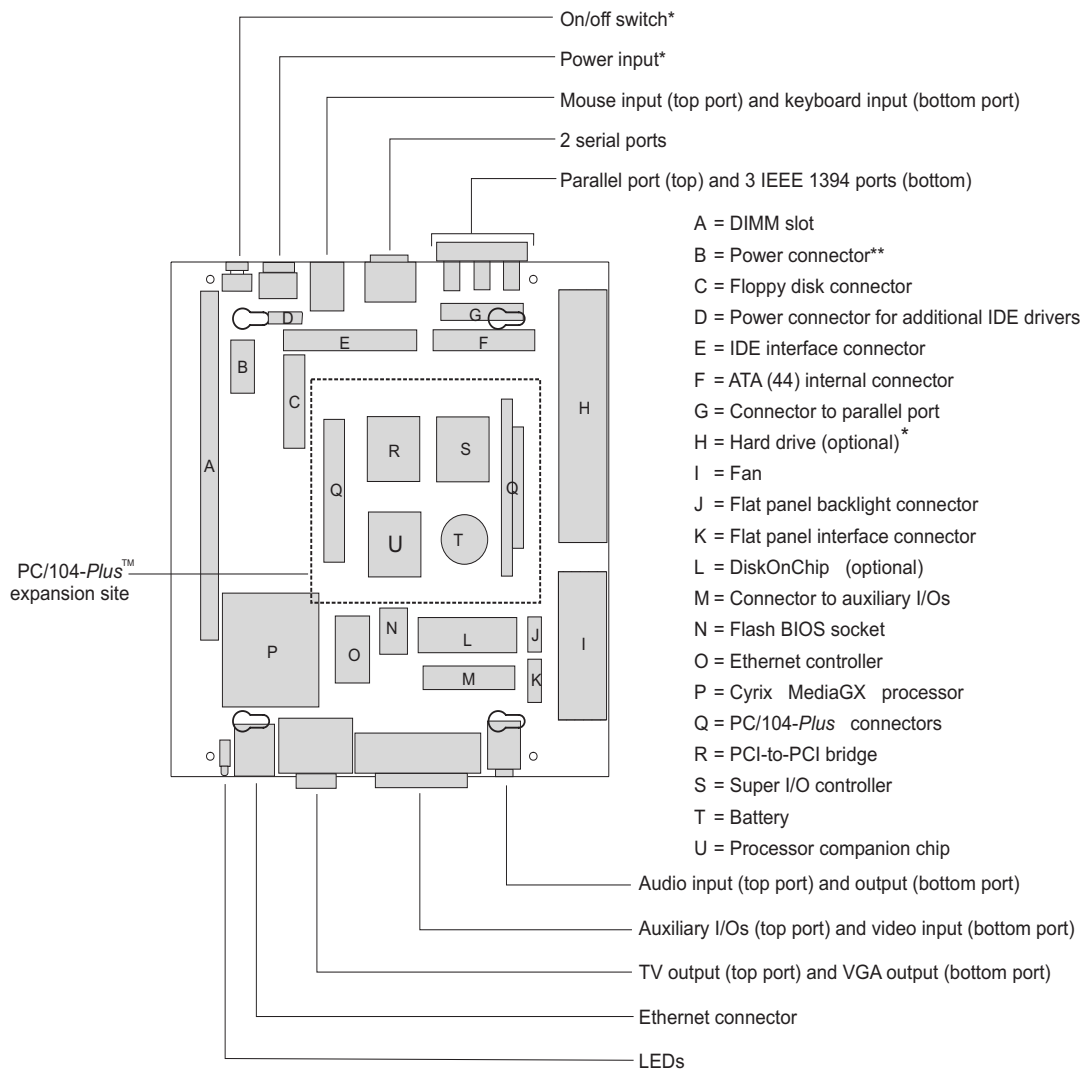
*** Only present on the PCI version.

Matrox Pulsar



Matrox 4Sight

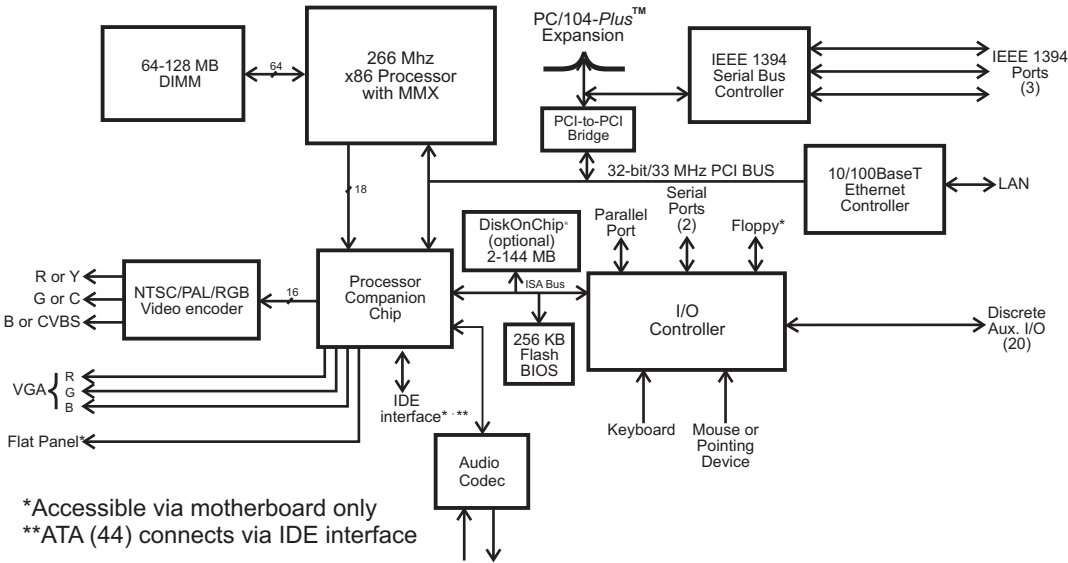
Matrox 4Sight components and connectors



* Available if you purchase the Matrox 4Sight integrated unit

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

Matrox 4Sight motherboard



Index

A

Auxiliary inputs/outputs
4Sight 198–199

B

buffers, Mbuf...()
Corona 22–23
Genesis 64, 66
Meteor-II 111
Orion 160–161
Pulsar 174–176

C

command reference
Corona 21, 111
Genesis 63
Meteor-II 110
Orion 159
Pulsar 173
VGA, under Windows 210
commands
VGA-specific 212
Corona
encoder 17
custom
window, VGA 210

D

DCF files
Corona 24
Genesis 67
Meteor-II 116
Orion 161
Pulsar 177
DIB info structure, VGA 210
digGrabWait()
Genesis 73
digitizer 161

digitizer, Mdig...()
Corona 24–26, 32–33, 37
Genesis 66–67, 73
Meteor-II 116, 134, 144
Orion 165–166, 168
Pulsar 177, 180–181, 184–185
display
16-bit buffer simulated, Genesis 54, 173
VGA system 210
Windows, VGA 210
display resolution
Orion 158
display, Mdisp...()
Corona 37, 39–41, 43
Genesis 44, 76
Meteor-II 146
Orion 168
Pulsar 186–187

E

edge
rising/falling 27
encoder 17
error reporting
hook, VGA 213

G

Genesis main board
general features 52
Genesis processor board
features 52
Genesis-LC 52–53
grabbing to Host 54
grab
double-buffering
Meteor-II /Digital 99
grabbing to Host
Genesis-LC 54
grab over-run 54–55, 99
Genesis-LC, solutions
buffer size 55
grabbing in on-board buffers 57
large frames of data 58

Meteor-II /Digital, solutions
buffer size 100
grabbing in on-board buffers 100
large frames of data 101

H

hook
error, VGA 213
trace, VGA 213
user-defined function, VGA 211

I

IEEE 1394 cameras
4Sight 197
inquire
system 205–206

M

M_DIB_INFO, VGA 210
MappHookFunction()
VGA 213
Matrox 4Sight
specific features 196
MblobInquire()
Genesis 64
MbufAlloc...()
Corona 22
Genesis 64
Meteor-II 111
Orion 160
MbufAlloc2d()
Genesis 64, 66
Pulsar 174
MbufAllocColor()
Pulsar 174
MbufControl()
Pulsar 174
MbufCopyCond()
Genesis 66
MbufCopyMask()
Genesis 66

MbufCreate...()
Corona 22
Meteor-II 112
Orion 160
Pulsar 175
MbufCreateColor()
Genesis 66
MbuffInquire()
Corona 23
Genesis 66
Meteor-II 113
Orion 161
Pulsar 176
Mdig...()
Orion 161
VGA 210
MdigAlloc()
Corona 24
Genesis 66
Meteor-II 114
Orion 161
Pulsar 177
MdigChannel()
Corona 25
Meteor-II 116
Orion 162
Pulsar 177
MdigControl()
Corona 26
Genesis 67
Meteor-II 119
Orion 163
Pulsar 177
MdigGrab()
Corona 32
Genesis 73
Meteor-II 134
Orion 165
Pulsar 180
MdigGrab()/MdigGrabContinuous()
Orion 165
MdigGrab()Wait()
Meteor-II 135
MdigGrabContinuous()
Corona 32
Meteor-II 134
Orion 165
Pulsar 181

MdigGrabWait()
 Genesis 73
MdigHookFunction()
 Corona 33
 Genesis 73
 Meteor-II 135
 Orion 165
MdigInquire()
 Corona 33
 Genesis 73
 Meteor-II 136
 Orion 166
 Pulsar 181
MdigLut()
 Genesis 75
 Meteor-II 144
 Pulsar 184
MdigReference()
 Corona 37
 Genesis 75
 Meteor-II 144
 Orion 168
 Pulsar 185
MdispAlloc()
 Corona 37
 Genesis 75
 Meteor-II 146
 Orion 168
 Pulsar 186
MdispControl()
 Corona 39
 Genesis 76
 Pulsar 186
MdispDeselect()
 VGA 220
MdispInquire()
 Corona 40
 Orion 168
MdispLut()
 Corona 41
 Genesis 77
 Meteor-II 146
 Pulsar 186
MdispOverlayKey()
 Corona 43

MdispPan()
 Corona 44
 Genesis 44
 Meteor-II 146
 Pulsar 187
MdispSelect()
 Pulsar 187
MdispZoom()
 Corona 44
 Genesis 78
 Meteor-II 146
 Pulsar 187
Meteor-II
 data compression/decompression
 Meteor-II, lossless 95
 Meteor-II, lossy 95
Meteor-II /Digital
 double-buffering 99
MgenWarpParameter()
 Genesis 79
MgraFontScale()
 Genesis 79
milcor.txt
 Corona 17
 Orion 159
milgen.txt
 Genesis 53
milmet2.txt
 Meteor-II 97, 197
milpul.txt
 Pulsar 172
MimConvolve()
 Genesis 79
MimResize()
 Genesis 79
MimRotate()
 Genesis 79
MimWarp()
 Genesis 79
MJPEG
 Meteor-II 95
MpatInquire()
 Genesis 79–80

MsysAlloc()

- Corona 45
- Genesis 80
- Meteor-II 147
- Orion 169
- Pulsar 188

MsysControl()

- 4Sight 198–199, 201
- Corona 45
- Genesis 80
- Meteor-II 148
- Pulsar 189

MsysGetHookInfor()

- 4Sight 205

MsysInquire() 205–206

- 4Sight 202
- Corona 49
- Genesis 85
- Meteor-II 153
- Orion 169
- Pulsar 193

MvgaAllocDIBInfo()

- VGA 220

MvgaDispFreeDIBInfo()

- VGA 210

MvgaDispSelectClientArea() 220

MvgaHookModifiedDIB() 211

N

number of channels

- Orion 158

R

redraw image, VGA 211

resolutions supported

- 4Sight 196

S

system

- inquire 205–206

system, Msys...()

- Corona 45, 49
- Genesis 80, 85
- Meteor-II 153
- Orion 169
- Pulsar 188–189, 193

T

trace

- hook, VGA 213

transfer rates

- IEEE 1394 196

U

UART

- Corona 16, 95

V

VGA, board-specific functions 212

Video Configuration Format

- Corona 38
- Genesis 76
- Pulsar 186

video formats supported

- 4Sight 196
- Corona 16
- Pulsar 172

W

Windows

- custom window, VGA 210
- redraw image, VGA 211

Product Assistance Request Form

[illegible]

