Matrox Imaging Library (MIL) 10.0 Update 30
Release Notes (MIL’s USB3 Vision system)
September, 2016

This document outlines what is new with MIL’s USB3 Vision system and explains the current limitations and particularities.

It also presents last-minute information that did not make it into the manual or on-line help. Note that this help file serves to complement your manual. The information found in this file overrides your formally documented material.

Contents

1. Matrox Imaging Library’s USB3 Vision system

1.1 Differences between MIL 10 Update 30 and MIL 10 Update 19

Various bug fixes were applied to MIL's USB3 Vision system. Note that MIL Update 30 is a cumulative update, including all content from MIL Update 1 and Update 19.

1.2 What's new in MIL 10 Update 30

1.2.1 New features summary

- No new Features.

1.2.2 Bug Fixes

- Fixed an error in MdigAlloc(M_GC_DEVICE_USER_NAME) caused when the name was programmed without power cycling.

- Fixed a synchronization lost event at the end of MdigProcess().

- Improved packed and YUV444 pixel format support.

- Fixed camera access errors in MdigHookFunction(M_CAMERA_PRESENT).

- Fixed MdigInquire (M_SOURCE_SIZEX/Y) such that the camera Width/Height can be returned without a scaling factor.

- Fixed race condition when grabbing using multiple Host controllers. This previously caused a driver exception.

- Fixed acquisition using multiple cameras (n>4) with an event enabled. This previously caused a driver exception.

1.2.3 Additions to the command reference

- Additions to MdigInquireFeature()/MdigControlFeature()
  - The FeatureType parameter has been changed to UserVarType. This was done to simplify writing code with MdigControl/InquireFeature(). UserVarType must always reflect the type of the pointer passed to the UserVarPtr parameter. Legacy code is transparently supported, but we recommend you update your code. Note that M_TYPE_REGISTER now becomes M_TYPE_UINT8, M_TYPE_ENUMERATION now becomes M_TYPE_INT64 or M_TYPE_STRING, and M_TYPE_COMMAND now becomes M_DEFAULT. Data type conversions are made, whenever possible, in cases where the feature's "native" data type is different than the UserVarType supplied. Regardless of a feature's "native" data type it can always be read as a string. See Board-specific examples for details.

    The following is a list of example calls using the new UserVatType:

    - MdigControlFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("Width"), M_TYPE_INT64, &Int64Var)

    - MdigControlFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("Gain"), M_TYPE_DOUBLE, &DoubleVar)

    - MdigControlFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("ReverseX"), M_TYPE_BOOLEAN, &BoolVar)

    - MdigControlFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("PixelFormat"), M_TYPE_STRING, MIL_TEXT("Mono8"))

    - MdigControlFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("LUTValueAll"), M_TYPE_UINT8, Uint8Array)

    - MdigControlFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("AcquisitionStart"), M_DEFAULT, M_NULL)

    - MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("Width"), M_TYPE_INT64, &Int64Var)

- MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("Gain"), M_TYPE_DOUBLE, &DoubleVar)

- MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("ReverseX"), M_TYPE_BOOLEAN, &BoolVar)

- MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE + M_STRING_SIZE, MIL_TEXT ("PixelFormat"), M_TYPE_MIL_INT, &MilIntVar)

- MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("PixelFormat"), M_TYPE_STRING, MilTextCharArray)

- MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("LUTValueAll"), M_TYPE_UINT8, Uint8Array)

- M_FEATURE_USER_ARRAY_SIZE() can now be used with MdigInquireFeature when the data type returned is a string or an array of bytes (register). The M_FEATURE_USER_ARRAY_SIZE() macro is used to pass the size of the user-allocated buffer passed to MdigInquireFeature's UserVarPtr parameter. M_FEATURE_USER_ARRAY_SIZE() is passed using the UserVarType parameter. See MilGige board specific example for sample usage.

  The following is a list of example calls using M_FEATURE_USER_ARRAY_SIZE():

  - MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("PixelFormat"), M_TYPE_STRING + M_FEATURE_USER_ARRAY_SIZE(N), MilTextCharArray); N being equal to the number of MIL_TEXT_CHAR in the MilTextCharArray.

  - MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("LUTValueAll"), M_TYPE_UINT8 + M_FEATURE_USER_ARRAY_SIZE(N), Uint8Array); N being equal to the number of Uint8 in the Uint8Array.

- M_FEATURE_ENUM_ENTRY_DISPLAY_NAME can now be used to inquire possible enumeration string entry to use for display purposes. See M_FEATURE_ENUM_ENTRY_NAME in the MIL documentation.

- M_FEATURE_VALUE_AS_STRING is now deprecated.

  - To read a feature's value as a string and get the required string length use:

    - MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE + M_STRING_SIZE, MIL_TEXT("Width"), M_TYPE_MIL_INT, &MilIntVar);

  - To read a feature's value as a string use:

    - MdigInquireFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("Width"), M_TYPE_STRING+M_FEATURE_USER_ARRAY_SIZE(ArraySize), MilTextCharArray);

  - To write a feature's value from a string use:

    - MdigControlFeature(MilDigitizer, M_FEATURE_VALUE, MIL_TEXT("Width"), M_TYPE_STRING, MIL_TEXT("1024"));

- M_FEATURE_CHANGE_HOOK. Identifies the specified FeatureName to trigger the M_FEATURE_CHANGE hook callback. You must be hooked to the M_FEATURE_CHANGE hook type using MdigHookFunction().

- Additions to MdigHookFunction()
  - When hooking to a GenICam feature change event (see enumfeatures.cpp board-specific example):

    - M_GC_FEATURE_CHANGE can be used as a hook type. The hook triggers when a GenICam feature is invalidated. This usually occurs when a feature or a dependent feature is written.

- Additions to MdigGetHookInfo()
  - When hooking to a GenICam feature change event (see enumfeatures.cpp board-specific example):

    - M_GC_FEATURE_CHANGE_NAME can be used from a M_GC_FEATURE_CHANGE hook function. The function returns the name of the GenICam feature that triggered the hook. UserVarPtr must point to a user allocated array of type MIL_TEXT_CHAR.

    - M_GC_FEATURE_CHANGE_NAME_SIZE can be used from a M_GC_FEATURE_CHANGE hook function. The function returns the number of characters in the string returned by M_GC_FEATURE_CHANGE_NAME. UserVarPtr must point to a MIL_INT.

1.2.4 Behavior change

- Default color buffer type changed from RGB24 to BGR32 for an optimal display performance.

- In the case where a synchronization lost occurs, the driver automatically attempts to resynchronize the image stream with the camera. This resets the BlockID to 0.

1.2.5 Standard compliance

- GenCP 1.1.

- USB3Vision 1.0.1.

- GenICam 3.0.

1.3 Differences between MIL 10 Update 19 and MIL 10 Update 1

- Camera Assistant can now be used to manipulate USB3 Vision devices and to diagnose some setup problems.

- The MIL Help now has information about MIL’s USB3 Vision system.

- New features and bug fixes to MIL's USB3 Vision system.

1.4 What's new in MIL 10 Update 19

1.4.1 New features summary

- Added support for MIL’s USB3 Vision system in Capture Assistant.

- Added support for streaming Monochrome and Bayer data of 10- and 12 bit packed pixel format.

- Added support for MdigAlloc(M_GC_DEVICE_USER_NAME).

- Added support for MdigAlloc(M_GC_MANIFEST_ENTRY()).

- Added support for MdigAlloc(M_GC_XML_DOWNLOAD_SKIP and M_GC_XML_FORCE_DOWNLOAD).

- Added support for MdigInquire(M_CAMERA_PRESENT).

- Added support for MsysHookFunction(M_CAMERA_PRESENT).

- Added support for MdigHookFunction(M_CAMERA_PRESENT).

- Added support for MdigInquire/MsysGetHookInfo(M_GC_UNIQUE_ID_STRING).

- Added MultiCamera hardware-specific example.

- Added acquisition statistics to be displayed in Camera Assistant.

1.4.2 Behavior change

- If two cameras are used and M_DEV0 camera is removed from the PC, M_DEV1 camera will be remapped as M_DEV0, as if M_DEV1 and M_DEV0 were not allocated on their associated cameras.

- MdigGetHookInfo(M_GC_FRAME_BLOCK_ID) Called from within MdigProcess callback is now reporting the data trailer packet's BlockId instead of the data leader packet's BlockId.

1.4.3 Bug Fixes

- Fixed MdigInquire(M_PROCESS_FRAME_MISSED) which previously always returned 0.

- Fixed MdigAlloc() failure with certain cameras.

- Fixed issue regarding cameras sometimes not being detected after falling into hibernate/sleep mode.

- Fixed the USB3 Vision Diagnostic Tool. Previously, the tool would enter an endless loop when dealing with certain cameras.

- Fixed MdigAlloc() operation that failed when one or more features listed in the specified DCF (Digitizer Configuration File) was not writeable in the camera.

- Fixed YUV 411 and 444 pixel formats that previously caused corrupted images.

- Fixed MdigGrab() with triggers enabled. Previously, this needed two triggers to grab one frame.

- Fixed a driver exception that was caused when starting a grab with certain cameras under Windows 7.

- Fixed MdigGrab() failure with an error message that was caused by an unaligned image size.

- Fixed MdigGetHookInfo(M_CORRUPTED_FRAME, M_GC_FRAME_BYTES_RECEIVED, M_GC_FRAME_LINE_COUNT, M_GC_FRAME_STATUS_CODE) which previously did not return the appropriate value.

1.4.4 Documentation

Added USB3 Vision-specific information to the MIL Help Reference and MIL Help Hardware-specific Notes.

1.4.5 Capture Assistant

- Added USB3 Vision support.

- Added support for DCF selection.

- Added a detailed Tree View; displays entire Ethernet and USB ecosystems.

- Added Statistics Report module. Generated Statistics reports are automatically included in a SysInfo.

- Added command-line dump mode. Used when generating a SysInfo.

- Added various diagnostics.

- Device triggers can now be controlled by the Feature Browser or by the triggers section of the Acquisition Properties tab.

- Added GigE Vision multicast monitor support.

- Added network adapter configuration parameters (Jumbo Packets, Receive Buffers, Interrupt Moderation) when selecting a network adapter from the Acquisition Device's tab.

- Network Adapter statistics are now generated using Windows performance counters.

2. Known Issues

- Matrox Inspector needs an image width that is a multiple of 4 pixels. Not following this rule will make Inspector crash or report errors.
- BGR or RGB, packed (10 or 12 bits) pixel format are not supported.
- If you are experiencing synchronization lost errors or image corruption under Windows 7 with Renesas USB 3.0 Host Controller, make sure you are using the Host controller driver version 2.1.39.0 or later for the 2.x series and 3.0.23.0 or later for 3.x series.
- When using events, if a camera disconnect (M_CAMERA_PRESENT hook gets called), in sequence MdigHook(M_UNHOOK), MdigFree()/MdigAlloc() MdigHook() must be called instead of only MdigProcess(M_START/M_STOP) as shown in the Hardware specific example MultiCamera.cpp.
- Acquisition using chunk mode with its image-part disabled is currently not working and will generate an error.
- You cannot allocate more than 34 cameras due to a problem within the driver (that is, M_DEV0 to M_DEV33).

3. Intellicam bug fixes

- With Intellicam, you can now modify a DCF while the feature browser is open, when working with a Camera Link camera, accessed using the GenICam CLProtocol. However, the feature browser will no longer work with Teledyne DALSA Camera Link cameras using the GenICam CLProtocol with Sapera software version 7.3 or lower. In this case, we recommend updating the Sapera software to version 7.4 or higher.
- Fixed possible Intellicam crash when clicking on "Dump state to DCF" in the camera configuration tab, when the camera contains a LUT.
- Fixed an issue where error messages would not be displayed when starting the feature browser.
- Fixed possible Intellicam crash when aborting a grab from a triggered camera that is not receiving any triggers.
- Fixed non-GenICam systems that could not grab after U1 was installed.
- Fixed Intellicam failure to load a DCF that was saved with "Dump state to DCF" button.

4. Supported operating systems

This section lists the supported operating systems.

- 32-bit Windows® 7[1]

- 64-bit Windows® 7[1]
- 32-bit Windows® 8.1
- 64-bit Windows® 8.1
- 32-bit Windows® 10
- 64-bit Windows® 10

[1] If using Windows 7, you must install the USB 3.0 host controller driver prior to using MIL's USB3 Vision system.

5. Location of examples (in the help file)

In the help file, the location information written at the top of examples might not be up-to-date. Use MIL Example Launcher to find an example on disk.

6. Troubleshooting

- Your camera must be USB3 Vision compliant in order to work with the Matrox USB3 Vision driver. For more information, you can refer to http://www.visiononline.org/vision-standards.cfm.
- Note that a diagnostic tool is available from MilConfig to troubleshoot the connection to USB3 Vision cameras. Capture assistant can also be used to diagnose USB issues and generate a usage statistic report.