This document outlines what is new with Matrox Radient eV-CLHS 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. MIL Driver for Matrox Radient eV-CLHS

1.1 What's new in MIL 10 Update 21

1.1.1 Standards compliance

The MIL driver for Matrox Radient eV-CLHS supports the following standards:

- AIA (Automated Imaging Association) Camera Link HS™ version 1.0.
- GenICam™ Standard Version 2.3.1.
- GenICam™ GenCP Standard Version 1.0.

1.1.2 Summary of new features

The following features are new for this release:

- New API to latch information for each grabbed frame. See M_DATA_LATCH_*.
- New API to grab multiple frames for each grab command. This is useful when grabbing at very high frame rates. See M_GRAB_FRAME_BURST.
- Added a second pulse for timer signals. See M_TIMER_DURATION2 and M_TIMER_DELAY2.
- Added the following MIL hardware-specific example:
  - DataLatch.cpp. It demonstrates how the Data Latch API latches information at each grabbed frame (such as, timestamps and quadrature encoder positions). This example is located in ...\examples\board-specific\DataLatch\c++.
  - FrameBurst.cpp. It uses the frame burst API to aggregate multiple frames into each grab command. This example is located in …\examples\board-specific\FrameBurst\c++
  - Enumfeatures.cpp. This is a GenICam-specific example. It demonstrates how to enumerate all features in your GenICam compliant device in a MIL application. The example is located in ...\examples\board-specific\enumfeatures\c++.

1.1.3 API enhancements

- 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 MdigControl()/MdigInquire():

❖ Note that the following ControlTypes that can have + M_TIMERn are marked below. For information about M_TIMERn, refer to MdigControl() /MdigInquire().

M_TIMER_DELAY2 + M_TIMERn		Sets the delay between the end of the first active portion of the timer output signal and the start of the second pulse.
	M_DEFAULT	Specifies the default value. This is the same as specified in the DFC or, if not specified in the DCF, 0.
	Value > 0	Specifies the delay, in nsecs.
M_TIMER_DURATION2 + M_TIMERn		Sets the duration for the active portion of the second pulse of the timer output signal.
	M_DEFAULT	Specifies the default value. This is the same as specified in the DFC or, if not specified in the DCF, 0.
	Value > 0	Specifies the duration of the active portion of the second pulse of the timer output signal, in nsecs.
M_TL_TRIGGER_ACTIVATION		Sets the signal variation upon which to generate a trigger signal to the camera thru the transport layer interface.
	M_DEFAULT	Same as M_ANY_EDGE.
	M_ANY_EDGE	Specifies that a trigger will be generated both upon a high-to-low and a low-to-high signal transition.
	M_EDGE_RISING	Specifies that a trigger will be generated upon a low-to-high signal transition.
M_GRAB_FRAME_BURST_SIZE		Specifies the number of frames grabbed into the same buffer at each grab command. The size Y of the grab buffer must be equal to (height of the frame * M_GRAB_FRAME_BURST_SIZE).
	M_DEFAULT	Same as 1.
	1 <= Value <= 1023	Sets the number of frames grabbed.
M_GRAB_FRAME_BURST_MAX_TIME		Specifies the maximum amount of time to wait for all the frames to be grabbed in the multi-frame buffer. The timer starts when the first frame is grabbed. The number of frames in the buffer can be inquired using MdigGetHookInfo() with M_GRAB_FRAME_BURST_COUNT. This is useful when the camera stops sending frames and the multi-frame buffer is only partially full.
	M_DEFAULT	Same as 1.000 secs.
	0.000008 <= Value <= 1.000000	Specifies the maximum amount of time to wait, in secs.
	M_INFINITE	Specifies to wait indefinitely.
M_GRAB_FRAME_BURST_END_TRIGGER_SOURCE		Specifies the signal from which a rising edge signals the end of a multi-frame sequence. This is useful to force a partially completed multi-frame buffer to complete.
	M_DEFAULT	Same as M_AUX_IO0.
	M_AUX_IOn	Specifies to use auxiliary input signal n as the trigger source, where n is the number of the auxiliary signal. Note that the specified auxiliary signal can also be a bidirectional signal set to input (using M_IO_MODE set to M_INPUT).
M_GRAB_FRAME_BURST_END_TRIGGER_STATE		Enables the M_GRAB_FRAME_BURST_END_TRIGGER_SOURCE source.
	M_DEFAULT	Same as M_DISABLE.
	M_DISABLE	Disables the grab frame burst end trigger source.
	M_ENABLE	Enables the grab frame burst end trigger source.

- Additions to MdigHookFunction():
  - You can now hook to a GenICam feature change event.
  - M_GC_FEATURE_CHANGE. Hooks the function to the event that occurs when a GenICam feature value is changed on your camera. This usually occurs when a feature or a dependent feature is written.
- Additions to MdigGetHookInfo():

  The following allows you to retrieve information about grab frame burst events. Unless otherwise specified, you can retrieve these information types if you call this function from within a function hooked to any digitizer event using MdigHookFunction() or MdigProcess().

  - M_GRAB_FRAME_BURST_COUNT: Returns the number of frames grabbed in the multi-frame buffer.
  - M_GRAB_FRAME_BURST_END_SOURCE: Returns the type of event that generated the end of the frame burst. Multiple events can be set at the same time. Bitwise operators must be used to verify the presence of a specific returned value. Possible return values are:
    - M_BURST_MAX_TIME: Specifies that the frame burst has taken as much time to complete as the specified maximum frame burst time. To specify the maximum time for a frame burst to complete, use MdigControl with M_GRAB_FRAME_BURST_MAX_TIME.
    - M_BURST_TRIGGER: Specifies that a specified trigger signal generated the end of the burst sequence. To specify the source signal, use MdigControl() with M_GRAB_FRAME_BURST_END_TRIGGER_SOURCE.
    - M_BURST_COUNT: Specifies that the specified number of frames have been grabbed. To specify the number of frames in a frame burst, use MdigControl() with M_GRAB_FRAME_BURST_SIZE.

The following allows you to retrieve information about a GenICam SFNC-compliant event. The following information types are only available if MdigGetHookInfo() was called from a

function hooked to a GenICam event using M_GC_EVENT + M_GC_FEATURE_CHANGE. In addition, the GenICam event must be enabled using MdigControlFeature(), and the message channel must be supported by your camera.

- M_GC_FEATURE_CHANGE_NAME. Retrieves the name of the GenICam feature that changed. The UserVarPtr must point to a user allocated array of type MIL_TEXT_CHAR.
- M_GC_FEATURE_CHANGE_NAME_SIZE. Retrieves the size of the name of the GenICam feature that changed. The UserVarPtr must point to a MIL_INT.

2. Supported operating systems

This section lists all the operating systems that the Matrox Radient CLHS driver supports.

- 64-bit Windows® 7.
- 64-bit Windows® 8.

3. 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.