



Hamamatsu Video Capture Library For LabVIEW

July 2016

Version 4.0

HAMAMATSU

Homepage Address <http://www.hamamatsu.com>

HAMAMATSU PHOTONICS K.K., Systems Division

812 Joko-cho, Hamamatsu City, 431-3196, Japan, Telephone: (81)53-431-0124, Fax: (81)53-435-1574, E-mail: export@sys.hpk.co.jp

U.S.A. and Canada: Hamamatsu Photonic Systems, 360 Foothill Road, Bridgewater, N.J. 08807-0910, U.S.A., Telephone: (1)908-231-1116, Fax: (1)908-231-0852, E-mail: usa@hamamatsu.com

Germany: Hamamatsu Photonics Deutschland GmbH, Arzbergerstr. 10, D-82211 Herrsching am Ammersee, Germany, Telephone: (49)8152-375-0, Fax: (49)8152-2658, E-mail: info@hamamatsu.de

France: Hamamatsu Photonics France S.A.R.L.: 8, Rue du Saule Trapu, Parc du Moulin de Massy, 91882 Massy Cedex, France, Telephone: (33)1 69 53 71 00, Fax: (33)1 69 53 71 10, E-mail: infos@hamamatsu.fr

United Kingdom: Hamamatsu Photonics UK Limited, 2 Howard Court, 10 Tewin Road Welwyn Garden City Hertfordshire AL7 1BW U.K., Telephone: (44)0 1707-294888, Fax: (44)0 1701-325777, E-mail: info@hamamatsu.co.uk

North Europe: Hamamatsu Photonics Norden AB, Smidesvägen 12, SE-171-41 Solna, Sweden, Telephone: (46)8-509-031-00, Fax: (46)8-509-031-01, E-mail: info@hamamatsu.se

Italy: Hamamatsu Photonics Italia S.R.L.: Strada della Moia, 1/E 20020 Arese (Milano), Italy, Telephone: (39)02-935 81 733, Fax: (39)02-935 81 741, E-mail: info@hamamatsu.it

Overview	4
System Requirements.....	4
What's New In Version 4	4
Installation	5
Hamamatsu Video Capture Installation	5
Camera Driver Installation.....	5
General Functions.....	6
TM_INITIALIZE_40	6
TM_DEINITIALIZE_40	6
TM_OPENCAMERA_40	7
TM_CLOSECAMERA_40	7
TM_GETCAMERAINFO_40.....	7
TM_SETPARAMETER_40.....	8
TM_GETPARAMETER_40	8
TM_GETPARAMETERLIMITS_40	8
TM_GETPARAMETERLIST_40.....	9
TM_SETAREA_40	9
TM_GETAREA_40.....	9
TM_SETINPUTTRIGGER_40.....	10
TM_SETOUTPUTTRIGGER_40	10
TM_FIRETRIGGER_40.....	11
TM_PREPARECAPTURE_40	11
TM_UNPREPARECAPTURE_40.....	11
TM_STARTCAPTURE_40	11
TM_STOPCAPTURE_40	12
TM_WAITNEXTFRAME_40.....	12
TM_GETCAPTUREINFO_40	12
TM_GETFRAME16_40.....	12
Advance Functions	13
TM_GETFRAME8_40.....	13
TM_GETFRAMES16_40.....	13
TM_SETWVIEWAREA_40	13
TM_SETEMPROTECTION_40.....	13
TM_GETEMPROTECTSTATUS_40.....	14
TM_SETMASTERPULSE_40	14
TM_GETELECTRONINFO_40	14
TM_RECURSIVEAVERAGING_40.....	14
TM_FRAMEAVERAGING_40	15
TM_SETBACKGROUNDFRAME_40.....	15
TM_SETSHADINGFRAME_40	15
TM_STOREBACKGROUNDFRAME_40.....	15
TM_STORESHADINGFRAME_40	15

Hard Disk Recording Functions.....	16
TM_STARTRECORDER_40.....	16
TM_STOPRECORDER_40.....	16
TM_GETRECORDERSTATUS_40	16
TM_OPENDCIMGFILE_40.....	17
TM_CLOSEDCIMGFILE_40	17
TM_GETDCIMGFRAME_40	17
TM_WRITEMETADATA_40	18
TM_READMETADATA_40	18
 Samples	 20
Single Frame Acquisition - TM410_SNAP.VI.....	20
Acquire Images First Then Display - TM412_SNAPSERIES.VI	21
Continuous Capture - TM420_SEQUENCE.VI	22
Polling – TM421_POLLING.VI.....	22
Software Trigger Synchronization – TM422_FIRETRIGGER.VI	23
Capture From Two Cameras - TM423_DUALCAMERASYNC.VI	24
Function Enumeration - TM430_FUNCTIONS.VI	25
Region Of Interest – TM431_SUBARRAY.VI.....	26
Recording Direct To Disk – TM450_DCIMGRECORDER.VI.....	27
Reading A DCIMG File – TM460_DCIMGREADER.VI.....	28
Splitview (W-View) Mode - TM481_WVIEW.VI.....	29
Electron Conversion – TM482_ELECTRONCOEFFICIENT.VI	30
 Parameter Reconfigure.....	 31
 HVCC Debug Log	 32
 Parameter Definitions	 33

Overview

The Hamamatsu Video Capture (HVC) library is a collection of LabVIEW VI functions designed to control a Hamamatsu camera and gather image data from within LabVIEW. Because this is built using Hamamatsu's DCAMAPI, applications created with this library will be compatible with many of the Hamamatsu line of cameras. And because it is made for LabVIEW, a programmer will be able to create large virtual instruments in a relatively short amount of time.

It is understood that the user is familiar with the Hamamatsu camera(s) and its functions. If you are unfamiliar with some of the capabilities of the camera, please refer to your camera manual. You will find technical specifications and installation instructions. If you need further assistance on the use of your Hamamatsu camera, please contact our support group.

It is also understood that the user is familiar with National Instruments LabVIEW. If you need further assistance with LabVIEW, please refer to your LabVIEW manual or contact National Instruments.

System Requirements

- Microsoft Windows 7 or newer
- DCAMAPI v16.2 or newer
- National Instruments LabVIEW 2011 or newer
- Hamamatsu camera with a compatible DCAMAPI module

What's New In Version 4

Changes have been made to this version of the HVC library to help you solve problems with your instruments and to get more out of your camera. Some of the major additions include a new logging feature and a dynamic parameter list system.

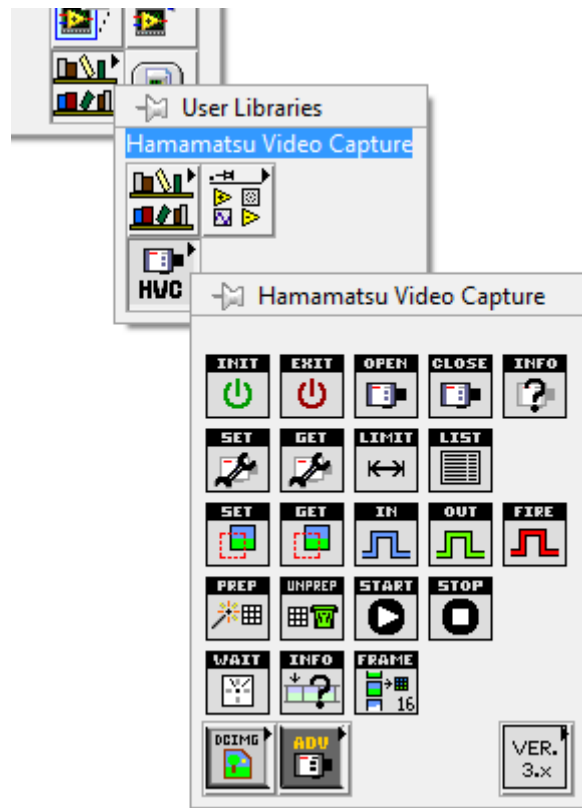
The new logging system will record all of the functions that have been called in a session. The generated log will be an ASCII text file which will allow you to easily open and understand the information. This can help assist you in debugging any problems you may encounter.

A dynamic parameter list system was developed for our growing line of cameras. Because the available parameters from one camera can be radically different from the available parameters of another camera, having one set of parameters is not ideal. This can cause confusion for customers trying to set parameters that do not exist for their device. With the dynamic parameter list system, you can now automatically reconfigure your TM_SETPARAMETER, TM_GETPARAMETER, and TM_GETPARAMETERLIMITS functions to list only the parameters that are available with your camera.

Installation

Hamamatsu Video Capture Installation

1. Be sure that LabVIEW is not running. If so, close all LabVIEW windows. Also be sure that your camera drivers have been properly installed. If not, please see the Camera Driver Installation below.
2. Run the SETUP.EXE program that comes with the installation software. For the 64-bit version of LabVIEW, run the SETUP_X64.EXE program instead to install the 64-bit version.
3. Follow all of the prompts during the installation process.
4. If you have LabVIEW 2011 or newer, the setup will install all of the proper files.
5. To access your new video capture functions, just follow the following diagram to locate the library.

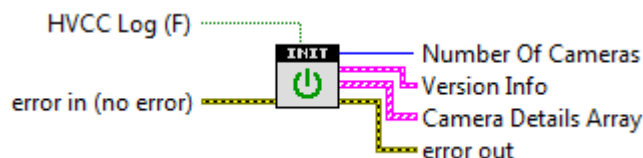


Camera Driver Installation

1. If you do not have a DCAM-API driver disk, you can download the camera driver from <http://www.dcamapi.com> and unzip the contents to a new directory.
2. Launch SETUP.EXE from the root directory of the installation.
3. Select and click the button from the menu of the interface that your camera is using.
4. Follow the on-screen instructions to install the driver you selected.

General Functions

TM_INITIALIZE_40



This is the first function that you need to run before you can use any other camera control functions in this library. Its purpose is to initialize the required resources for the video capture library and to find all supported cameras on the system. Any camera you wish to use with your system must be connected to your system and powered on before this function is called. When this function returns, it will report the number of supported cameras that were found, the version of this capture library and the DCAM version, and it will also provide the model number, serial number, and firmware version of each camera found.

This function allows you the option to record a log to assist in debugging any problems you may encounter in your software. If enabled, all function calls to this library will be recorded to an ASCII text file until TM_DEINITIALIZE is called. Please use caution. If TM_INITIALIZE is called with the HVC log enabled, it will overwrite any existing HVC log file.

If the capture library has already been initialized and TM_DEINITIALIZE has not been called yet, calling this function again will simply return the number of cameras, version info, and camera details. The resources for the capture library will not be reinitialized. Also, the HVCC Log input will be ignored.

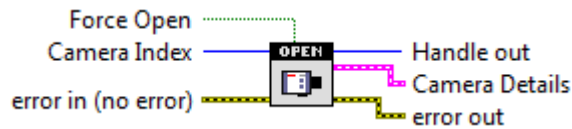
TM_DEINITIALIZE_40



If all connected devices have been closed, this will cleanly free all resources used to by the video capture library. However, if there are still some cameras opened, this will not force close those other devices, and you will be able to reopen the camera again without calling TM_INITIALIZE again. It is strongly recommended that you close all open devices before calling this function.

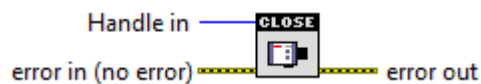
If the HVCC log is enabled through TM_INITIALIZE, this function will close the log regardless if there are cameras still open.

TM_OPENCAMERA_40



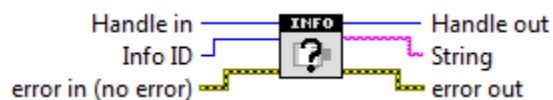
This will activate a camera for operation at the specified Camera Index and will return a handle for that camera as well as some details about the camera. These details include the model number, serial number, and firmware version of the camera. This handle must be used by the other camera control functions to determine the camera to be controlled. The index values start at 0, so if there are 3 cameras available on the system, the valid input for Camera Index would be 0, 1, and 2. This function can only be used to activate cameras that were found by TM_INITIALIZE. If the camera is already open in another VI or was not closed properly, this function will fail. If you set Force Open to TRUE, the function will attempt to close any open session with the camera before opening the camera.

TM_CLOSECAMERA_40



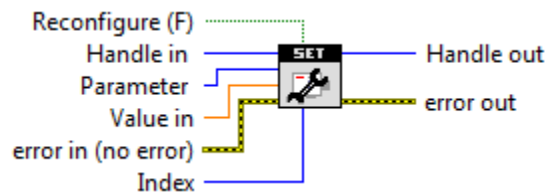
This will close the camera associated with the handle and free resources that it used. Once this function is called, the handle will become invalid and cannot be used anymore. The camera can still be opened again with TM_OPENCAMERA.

TM_GETCAMERAINFO_40



This function will return information about the camera. The possible items to get information are the camera model, camera ID, camera bus, vendor, driver version, camera version, module version, and DCAMAPI version. This function can also be used on unopened cameras by using the camera index instead of the camera handle.

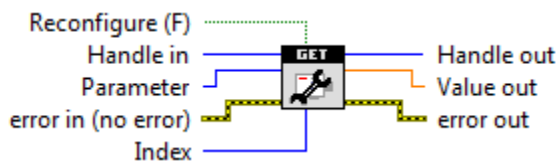
TM_SETPARAMETER_40



This function can adjust the parameters of the camera. These settings may include gain, offset, exposure time, binning, etc. While you can adjust some settings during a capture, settings that can change the image resolution cannot be changed during a capture. Please refer to your camera manual to determine the capabilities of your camera.

The Reconfigure (F) option must remain false unless you are using the Parameter Reconfigure tool. Please see the Parameter Reconfigure section for more information.

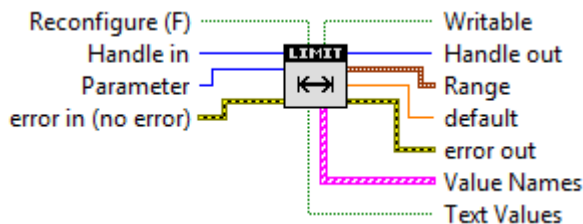
TM_GETPARAMETER_40



This function gets the current value of a selected camera parameter.

The Reconfigure (F) option must remain false unless you are using the Parameter Reconfigure tool. Please see the Parameter Reconfigure section for more information.

TM_GETPARAMETERLIMITS_40



This function returns the range of the input parameter. The values you will receive will include minimum, maximum, stepping, and default values. If the parameter is a mode function, it will have associated text values and the Text Values output will return true. If that value returns true, then Value Names will contain an array of text values for this function along with their associated number values.

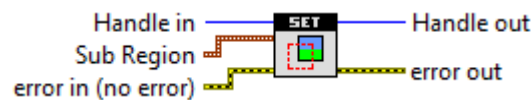
The Reconfigure (F) option must remain false unless you are using the Parameter Reconfigure tool. Please see the Parameter Reconfigure section for more information.

TM_GETPARAMETERLIST_40



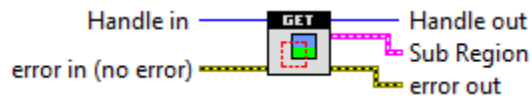
The Hamamatsu camera line is expanding with an ever growing number of functions. This VI will output an array which contains a list of the supported functions available to the camera. This array can be connected directly to the StringsAndValues[] property of a menu ring. The list of supported functions will vary depending on the camera. Additional information of these functions can be obtained from TM_GETPARAMETERLIMITS. And the function values can be controlled with TM_GETPARAMETER and TM_SETPARAMETER.

TM_SETAREA_40



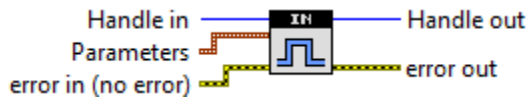
By default, your camera will use the whole sensor when capturing data. With this function, you will be able to select a smaller portion of that sensor. This will help to reduce the size of the output data when you do not need to use the whole sensor. And depending on the camera and settings that you use, you can also decrease the readout time of an exposure which can lead to faster frame rates. The Sub Region cluster takes four values. They are Horizontal Offset, Vertical Offset, Horizontal Size, and Vertical Size. Because of the limitations of the different cameras, the values you enter may not be the same values that were set. It is important to use TM_GETAREA to see the values that were set to the camera. This should not be called while capturing images.

TM_GETAREA_40



This function will return the value of the current Sub Region. This function will also return the current minimum, maximum, and increment values of each variable. These limits can change whenever you set a new sub-array or binning. It is always a good idea to call this function to get the current valid limits.

TM_SETINPUTTRIGGER_40

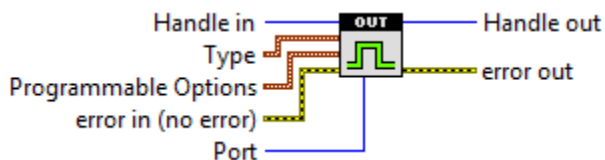


This function will allow you to set how the camera will trigger for new images. Parameters contains four settings you can adjust. Mode will allow you to set one of the following trigger modes

- **Internal** - This will set the camera to do its own triggering. This will also produce the fastest frame rate.
- **External edge** – The camera will wait for an external trigger signal to go active.
- **External level** – The camera will wait for an external trigger signal to go active and continue exposing until that signal switches to inactive.
- **Software** – The camera will wait for TM_FIRETRIGGER to be called.
- **Fast Repetition** – This is a special mode. Not available for all cameras.
- **TDI** – This is a special mode. Not available for all cameras.
- **Trigger Start** – The camera will wait for an external trigger signal to go active and then proceeds to work in internal trigger mode.
- **Sync Readout** – The camera will wait for the first trigger to start an exposure. All of the following triggers will stop the current exposure and start the next one.
- **Master Pulse** – This is a special mode. Not available for all cameras.

Polarity will allow you to set the external trigger to negative or positive. Times is used with Sync Readout. Delay determines the length of time until the camera will react to a trigger. This is only available on some cameras.

TM_SETOUTPUTTRIGGER_40



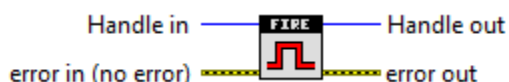
This function will allow you to set the programmable output triggers of your camera.

Parameters contains four settings that you can adjust. Mode will set the type of output.

- **Low** – Sets the output to low regardless of what the camera is doing.
- **Global Exposure** – Sets the output active when all of the pixels are exposing at the same time.
- **Programmable** – Sets the output to active for a specified period after a specified delay.
- **Trigger Ready** – Sets the output to active when the camera is ready to receive another input trigger.
- **High** – Sets the output to high regardless of what the camera is doing.

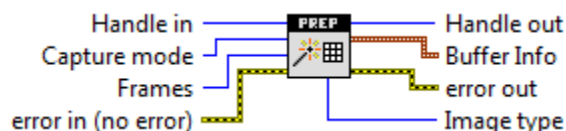
Polarity determines which direction is active. Period sets the length of time the output will remain active during Programmable trigger mode. Delay sets the length of time the output will wait before setting the output to active during Programmable trigger mode. The index option determines which output trigger these settings will be applied to.

TM_FIRETRIGGER_40



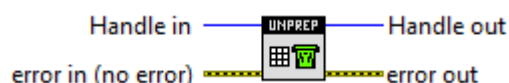
If your camera is set to software trigger mode, no images will be acquired until a software trigger is fired. When a software trigger is fired with this function, the camera will begin exposing a new image. This function is very useful when synchronizing the camera with application events.

TM_PREPARECAPTURE_40



This function prepares the driver and camera for image capture. There are two inputs available to configure your capture function. CAPTURE MODE allows you to adjust the behavior of the capturing cycle. In Snap mode, the camera will capture the specified number of frames and will stop at the end of the cycle. In Sequence mode, the camera will collect data continuously without stopping at the end of the cycle. Once it reaches the end of a cycle of frames, it will continue on the first frame. FRAMES allow you to adjust the number of frames of images you wish to capture. The minimum amount of frames you may set in Snap mode is 1 frame. The minimum amount of frames you may set in Sequence mode is 3 frames. If you set frames less than the minimum, the driver will set the frames to the minimum value. This function will also return an Image Type value which is used by IMAQ Vision to determine the image data type. This function does not start the camera to capture images.

TM_UNPREPARECAPTURE_40



This function will free all resources created with TM_PREPARECAPTURE. This step is important if you wish to change certain camera settings such as the binning or region of interest. If the camera is capturing data, you must stop it first with TM_STOPCAPTURE before calling this function.

TM_STARTCAPTURE_40



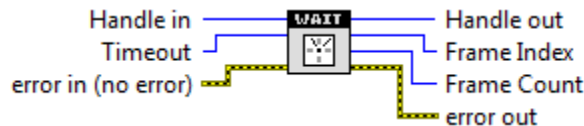
This begins capturing of image data from the camera to your computer. You will need to have called TM_PREPARECAPTURE to setup the driver for capture before calling this function.

TM_STOPCAPTURE_40



This function will stop the current capture session. The current capture mode and internal buffers create will remain intact so you will be able to start a new capture session immediately with TM_STARTCAPTURE.

TM_WAITNEXTFRAME_40



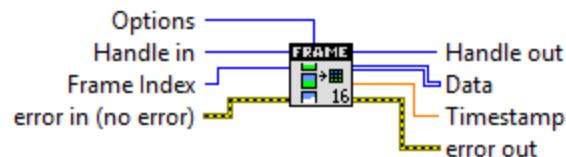
This function will wait for the next frame to become available. When a frame arrives, this function will return and you are able to work with the data. By default, the timeout is set to 5000 ms. The timeout value can be adjusted by setting the timeout input with a new value. If you set a timeout of -1, this function will wait indefinitely until a new frame arrives. The frame index is the buffer index location where the data was stored. Frame count is the number of frames captured since calling TM_STARTCAPTURE.

TM_GETCAPTUREINFO_40



This function will return the current frame index and frame count of the capture sequence. The frame index is the buffer index location where the data is stored. Frame count is the number of frames captured since calling TM_STARTCAPTURE.

TM_GETFRAME16_40

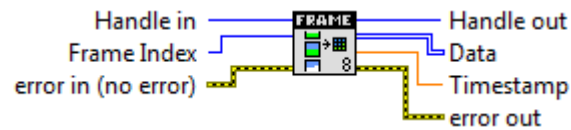


This function will retrieve the frame in the buffer at the specified frame index. You can get the current frame index from either TM_WAITNEXTFRAME or TM_GETCAPTUREINFO. The output of the function is unsigned 16-bit data. If your camera outputs a different datatype, you must use the appropriate function to retrieve the data. This function will also output the timestamp of the frame.

Advance Functions

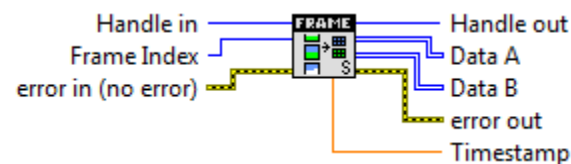
Some cameras have functions not available to other cameras. Your camera may or may not have the ability to use these functions. Please check your camera manual to determine if your camera has these capabilities.

TM_GETFRAME8_40



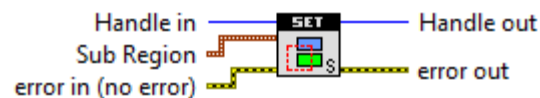
Similar to TM_GETFRAME16, this function will return the image data for the provided frame index. However, this function should be used specifically for older cameras that support 8-bit data. If the camera is outputting 16-bit data and this function is used, this function will return incorrect data.

TM_GETFRAMES16_40



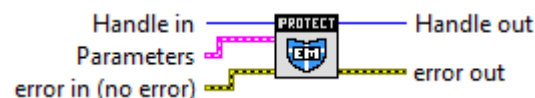
Similar to TM_GETFRAME16, this function will return the image data for the provided frame index. However, this function is to be used when the camera is set to splitview (W-View) mode which is available on a few camera models. This function will return the two halves of the image data as separate outputs.

TM_SETWVIEWAREA_40



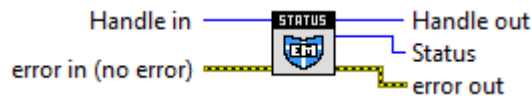
Splitview (W-View) mode is a special mode available to some cameras. If the camera is in this mode, this function can be used to set the region of interest.

TM_SETEMPROTECTION_40



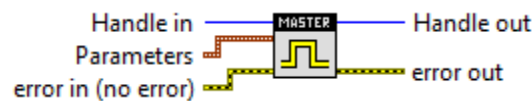
The EM Protection system is available to our EM cameras. This allows you to set the level of protection of your camera to help keep your camera working in case the cooling system should fail.

TM_GETEMPROTECTSTATUS_40



You can monitor the status of the EM Protection system through this function.

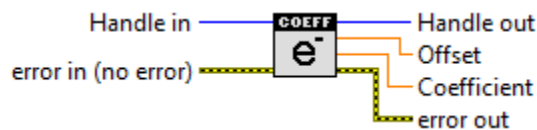
TM_SETMASTERPULSE_40



The Master Pulse allows you to configure an internal pulse generator of the camera for either internal or external trigger uses.

- **Continuous** – The master pulse will start automatically and continue to fire indefinitely according to the specified interval.
- **Start** – The master pulse will not start automatically. It will wait for a signal from the specified source. Once the signal is received, it will fire indefinitely according to the specified interval.
- **Burst** - The master pulse will not start automatically. It will wait for a signal from the specified source. Once the signal is received, it will fire triggers for the number of frames specified according to the specified interval.

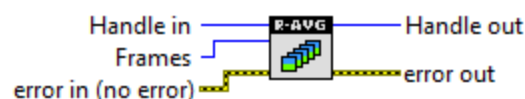
TM_GETELECTRONINFO_40



For newer cameras, the electron conversion information is provided to get the actual photon count of the individual pixels of the image. Here is the formula to convert the intensity value to electrons.

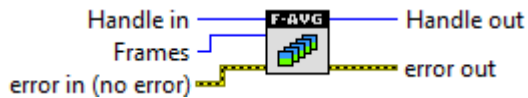
$$e^- = (\text{intensity} - \text{offset}) * \text{coefficient}$$

TM_RECURSIVEAVERAGING_40



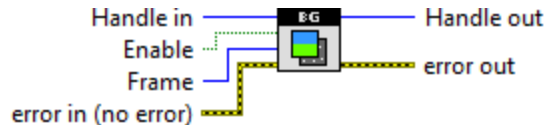
This DSP function allows you enable recursive filtering. Each output image of the camera will be the average of the number of frames specified by the Frames input. You can disable this function by setting Frames to 1.

TM_FRAMEAVERAGING_40



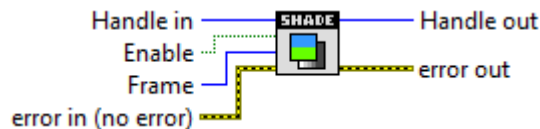
This DSP function allows you enable frame averaging. Each output image of the camera will be the average of the number of frames specified by the Frames input. You can disable this function by setting Frames to 1.

TM_SETBACKGROUNDFRAME_40



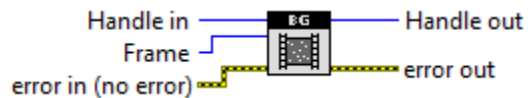
This DSP function allows you to enable a background image to be subtracted from the output data. You must specify the background frame you wish to subtract with the Frame input.

TM_SETSHADINGFRAME_40



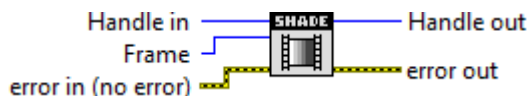
This DSP function allows you to enable a shading correction image to be applied to the output data. You must specify the correction frame you wish to apply with the Frame input.

TM_STOREBACKGROUNDFRAME_40



This DSP function will store a frame of data into the camera memory frame specified by the input Frame for background subtraction. This function will fail if the camera is capturing images.

TM_STORESHADINGFRAME_40

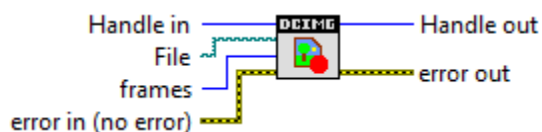


This DSP function will store a frame of data into the camera memory frame specified by the input Frame for shading correction. This function will fail if the camera is capturing images.

Hard Disk Recording Functions

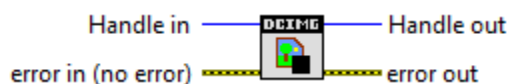
These functions allow you to capture image data directly to disk. This is especially useful if you are capturing a large amount of data but do not have enough RAM to store the data. The output data is written to a DCIMG file which is a proprietary Hamamatsu image stack format. In order to view images from a DCIMG file, you can use the DCIMG reader functions available with this capture library or you can use any other program designed to read DCIMG files such as HClmage. Please keep in mind that the storage disk is inherently slower than RAM. Your hard drive may not be fast enough to store data at the pace you are recording.

TM_STARTRECORDER_40



This function will enable the hard disk recorder. The recorder will write captured images directly to disk to the specified DCIMG file. This recorder will write up to the specified number of frames. Once that maximum number of frames have been reached, the recording will stop. This function must be called after TM_PREPARECAPTURE.

TM_STOPRECORDER_40



This function will end the current recorder session and close the DCIMG file. Once the DCIMG file is closed, it cannot be reopened for recording of more images. However, it may be opened by any DCIMG file reader to read all of the data that has been stored.

TM_GETRECORDERSTATUS_40



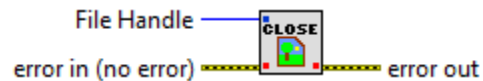
This function will provide the current index of the newest frame recorded as well as the total number of frames recorded.

TM_OPENDCIMGFILE_40



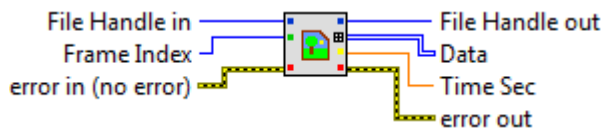
This function will open a DCIMG file for reading. It will return a file handle which will be used for all other DCIMG related functions. This file handle is different from the camera handle and should not be used for camera control functions. It will also return the total number of frames that are available in the DCIMG file. Once the DCIMG file is opened with this function, the data can be read by TM_GETDCIMGFRAME.

TM_CLOSEDCIMGFILE_40



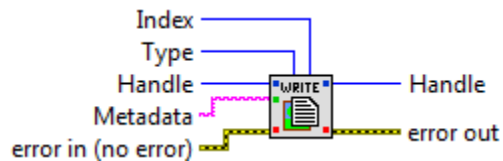
This function will release a DCIMG file and free any resources used for it. This will make the file handle invalid and unusable by the other functions.

TM_GETDCIMGFRAME_40



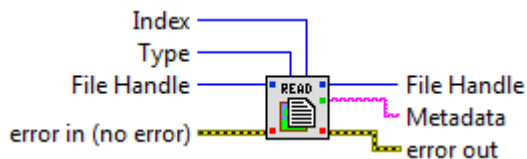
This function will retrieve the frame data from a DCIMG file specified by the file handle and the frame index. The file handle can be retrieved by TM_OPENDCIMGFILE. The Time Sec output is the timestamp in seconds of the selected frame relative to when the first frame in the file was captured.

TM_WRITEMETADATA_40



This function will allow you to write metadata to a DCIMG file currently open for recording. The Handle is the generated from TM_STARTRECORDER. With Type, you can specify if this is metadata for a specific frame or for the entire DCIMG file. Both types can be written to the DCIMG file. If you set Type to Frame, then Index will specify which frame the metadata will be written to. Be aware that once TM_STOPRECORDER is called, the DCIMG handle is destroyed and you will no longer be able to write metadata to the file.

TM_READMETADATA_40



This function will let you read the metadata that is stored in the DCIMG file specified by the file handle generated from TM_OPENDCIMGFILE. With Type, you can specify if this is metadata for a specific frame or for the entire DCIMG file. Both types can be written to the DCIMG file. If you set Type to Frame, then Index will specify which frame the metadata will be written to.

Error Code

When using this library, you may encounter an error when running a LabVIEW program. If the error was generated from one of the functions supplied in this library, then you will receive the error code as well as the particular VI that caused the error. This information will provide a clue into determining why the program failed and how it could be corrected. The following is a list of possible error codes that you may encounter.

TMERR_ABORT – The operation has been aborted.

TMERR_BUSY – The camera is currently busy and cannot execute your request. Please check that you are not changing resolution parameters during image capture.

TMERR_FAILOPENCAMERA – The system was unable to activate the camera. Check that the camera is properly connected to the computer. Power cycle the camera if necessary.

TMERR_FAILREADCAMERA – The camera could not read the command. Check that the camera is properly connected to the computer. Power cycle the camera if necessary.

TMERR_FAILWRITECAMERA – The camera could not change the parameter. Check that the camera is properly connected to the computer. Power cycle the camera if necessary.

TMERR_INVALIDCAMERA – The selected device is invalid. Check that a camera is connected at the specified index.

TMERR_INVALIDHANDLE – The specified camera handle is invalid. Check that a camera is open at the specified index.

TMERR_INVALIDVALUE – The specified value is invalid. Use only values used by the camera.

TMERR_INVALIDPARAMETER – The parameter ID is invalid. Use only parameters used by the camera.

TMERR_FAILEDOPENRECFILE – Failed to open the specified DCIMG file. Check that you have proper access to the specified directory.

TMERR_FAILEDREADDATA – Failed to read from the DCIMG file. Is the specified DCIMG file or frame index valid?

TMERR_FAILEDWRITEDATA – Failed to write to the DCIMG file. Check that you have write access to the specified directory.

TMERR_INVALIDRECHANDLE – The DCIMG file handle is invalid.

TMERR_INVALIDSUBARRAY – The subarray setting is invalid. Check the subarray settings.

TMERR_LOSTFRAME – A frame has been lost during transfer from the camera.

TMERR_NOCAMERA – No camera is available. Check your camera connections. Power cycle the camera if necessary.

TMERR_NOMEMORY – Your computer does not have enough memory for your request.

TMERR_NORESOURCE – Your computer does not have enough resources for your request.

TMERR_NOTREADY – The camera is not ready for the command.

TMERR_NOTWRITABLE – The parameter does not have a writable value.

TMERR_NOTREADABLE – The parameter does not have a readable value.

TMERR_NOTSUPPORT – The function is not supported by the camera.

TMERR_OUTOFRANGE – The specified value is out of range.

TMERR_TIMEOUT – The driver has timed out while waiting for the frame data.

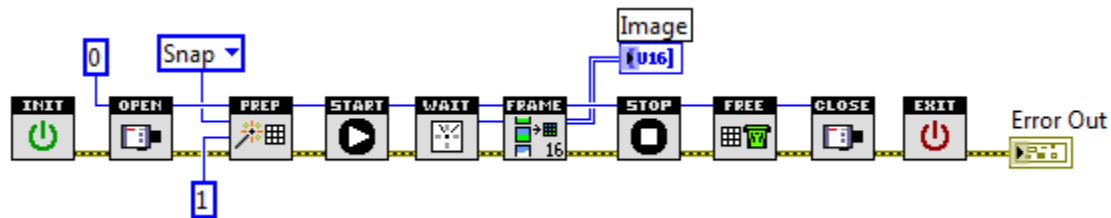
TMERR_UNKNOWNERROR – An unknown error has occurred.

Samples

The following is a quick description of some of the different sample VIs available with this library. This should cover all of the basic functions of cameras as well as some functions only available to certain cameras.

Single Frame Acquisition - TM410_SNAP.VI

In this example, we are capturing one image then displaying it to the screen.



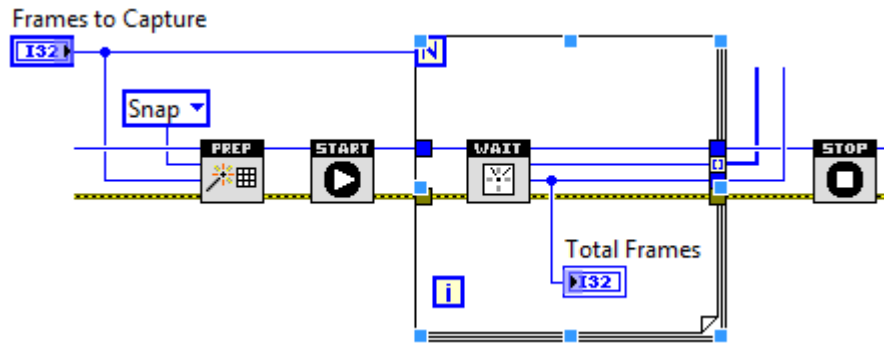
First, we are initializing the library by calling TM_INITIALIZE to find all cameras in the system. If it finds at least one camera, we call TM_OPENCAMERA with a camera index of 0 to activate the first camera that is found. If successful the function will return a handle to the camera. This handle will be used for all of the other functions.

To acquire a single image, TM_PREPARECAPTURE needs to be called to setup the capture. We set the capture mode to Snap and we set only 1 frame. Then we call TM_STARTCAPTURE to begin the capture session. TM_WAITNEXTFRAME is called to wait for a new frame to be available. Once that function returns, we call TM_GETFRAME16. We take the Frame Index output from TM_WAITNEXTFRAME to determine which frame we are going to retrieve from the capture library. The output 2D array will contain the data for the image. We then take this data and send it to the display.

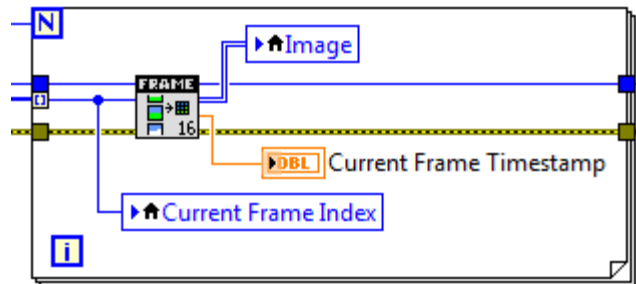
To clean up the acquisition, we first call TM_STOPCAPTURE to stop the capture process. Even though this is a snap of 1 image, this is still a necessary step. Then we call TM_UNPREPARECAPTURE to free the internal image buffers. Then TM_CLOSECAMERA is called to release the camera and all of the resources that were allocated for it. And finally TM_DEINITIALIZE is called to unload the capture library.

Acquire Images First Then Display - TM412_SNAPSERIES.VI

In this example, we are capturing a series of images that we intend on displaying later.



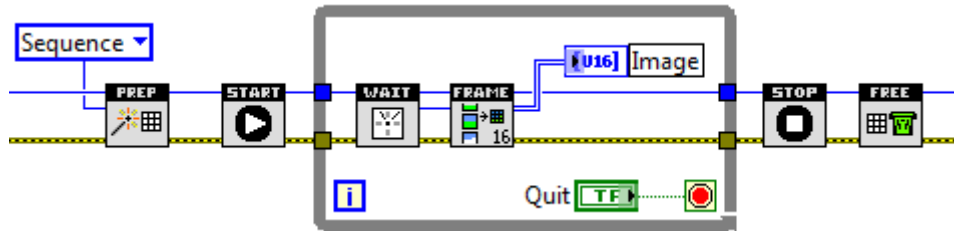
First, you will need to initialize the driver and open the camera as sample TM410. However, when you call TM_PREPARECAPTURE, you will need to specify the number of images you intend on capturing. After you start an acquisition with TM_STARTCAPTURE, you will need to wait for all of the images to be captured. The above example is one method to do this. Now that all of the images have been captured successfully, call TM_STOPCAPTURE to end the capture session. This does not release any of the frame data that you have captured.



Now that we have all of the images stored, we can go back to retrieve this data by calling TM_GETFRAME16 with the appropriate index values for the frames. TM_GETFRAME16 also provides

Continuous Capture - TM420_SEQUENCE.VI

In this example, we are capturing images continuously for a preview display.

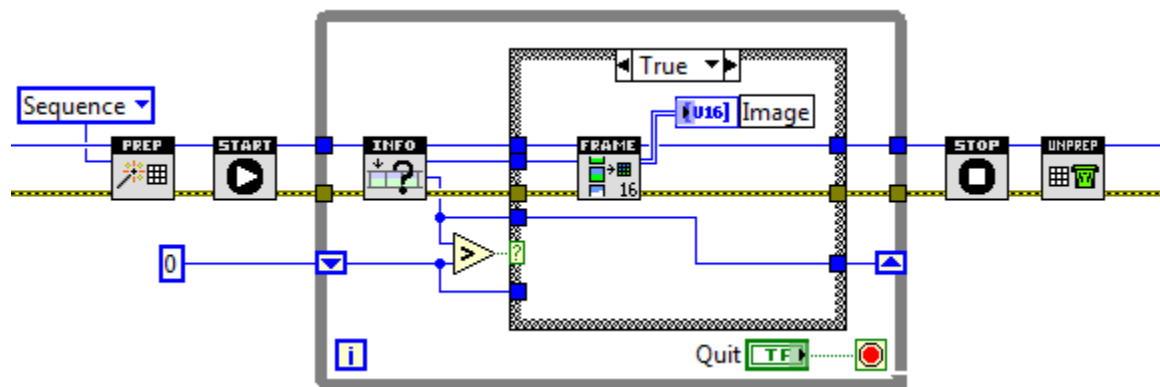


Setting up for a preview is very similar to acquiring a single image. After opening the camera with TM_OPENCAMERA, setup a sequence capture with TM_PREPARECAPTURE. For inputs, you need to set capture mode to Sequence. The default frame buffer for Sequence mode is 5, but you can set this value to what you like. However, 5 is usually enough for most cases. Once the buffers are prepared, start the capture session with TM_STARTCAPTURE. After the capture is started, we can enter our while loop so we can continuously capture images.

TM_WAITNEXTFRAME will wait for the next frame. TM_GETFRAME will retrieve the next frame from the buffer.

Polling - TM421_POLLING.VI

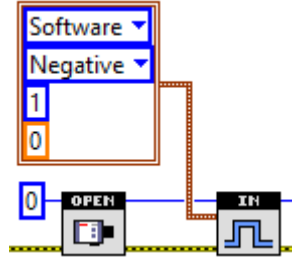
In this example, we are polling when new images are available instead of waiting for a new frame.



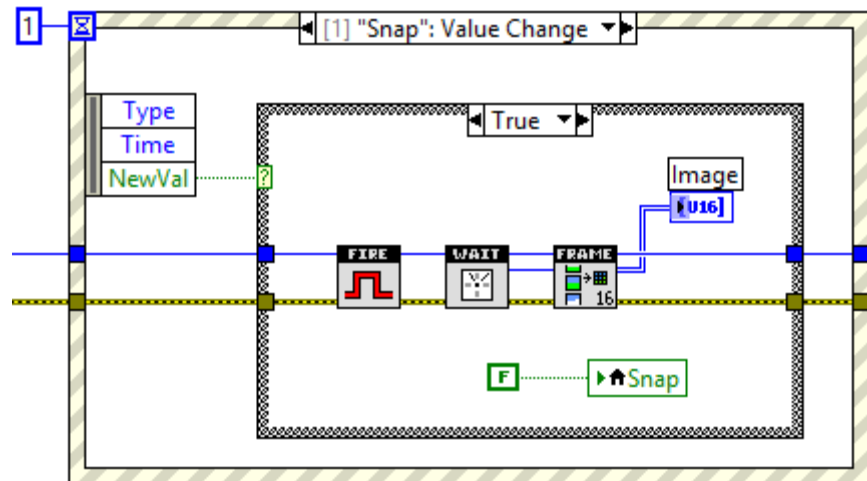
Instead of using TM_WAITNEXTFRAME to wait for a new image to arrive, you could use TM_GETCAPTUREINFO to see the current capture status. TM_GETCAPTUREINFO will give us the current frame index and the total frame count. This will only give us information, and it will not wait for a new frame. Only if the total frame count has increased do we get the new frame data. If you are using a relatively long exposure time, you may want to use this polling method instead as it will allow you more freedom to do other things while you wait for images to arrive.

Software Trigger Synchronization - TM422_FIRETRIGGER.VI

In this example, we are synchronizing the start of capture with the user event of pressing a button.



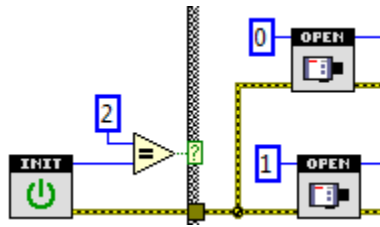
We first initialize the library and the camera as we did in TM420. However, after TM_OPENCAMERA is called, we can setup the camera for software trigger. In Software Trigger mode, the camera will not begin a new exposure until you fire a trigger through the software. This is done with TM_SETINPUTTRIGGER and setting Parameters.mode to Software. The other options in Parameters are ignored in this mode.



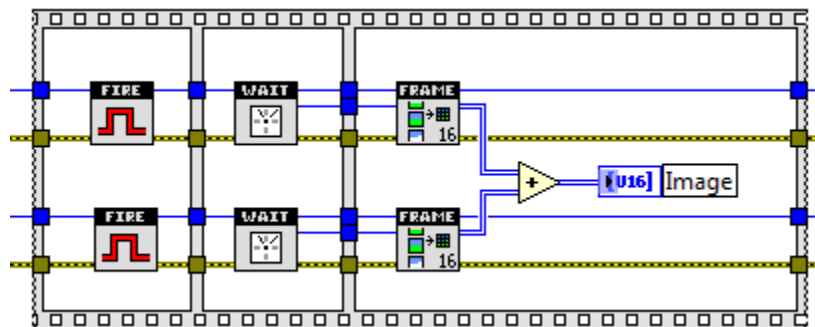
With the camera properly setup, we can begin our capture loop. Because we want to only fire a trigger when we press the SNAP button from the front panel, we use an event loop to detect when "SNAP" has changed. In the SNAP event, we call TM_FIRETRIGGER to send a software trigger to the camera. At that time, we can call TM_WAITNEXTFRAME to wait for the frame we just started. Then we call TM_GETFRAME to get the image data.

Capture From Two Cameras - TM423_DUALCAMERASYNC.VI

In this example, we are capturing data from two cameras simultaneously.



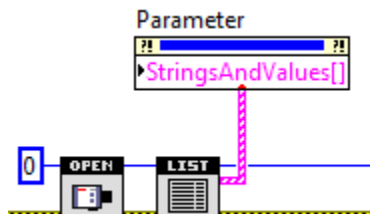
We initialize the library like usual, but now we check to see if there are two compatible cameras connected to the computer. If this is true, then we open both cameras to get the required camera handles. These camera handles are different and will allow us to control the two cameras separately.



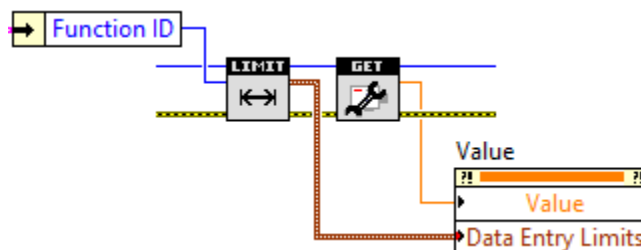
To synchronize the captures, TM_FIRETRIGGER is called for both cameras. If you are using an external trigger, you can skip this function call. Next, TM_WAITNEXTFRAME is called to wait for each camera to have a frame available. It will output the index of the frame when it is available. Finally, TM_GETFRAME16 is called with the frame index to get the frame data. In this example, we are adding the two images together into one image. Although it is not necessary to do so.

Function Enumeration - TM430_FUNCTIONS.VI

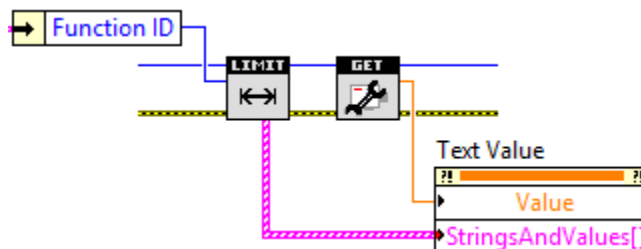
If you are creating a VI capable of handling different models of Hamamatsu cameras, you may choose to enumerate the available functions of the cameras. If you use TM_GETPARAMETERLIST with TM_GETPARAMETERLIMITS you will be able to see what functions are available to the camera.



In the above example, the output of TM_GETPARAMETERLIST is being used to set the StringsAndValues[] property of the menu ring item Function List. This will populate the menu ring with all of the functions available to this camera and associate them with the proper ID value. You may also use the dynamically created Function List menu ring with TM_GETPARAMETERLIMITS.



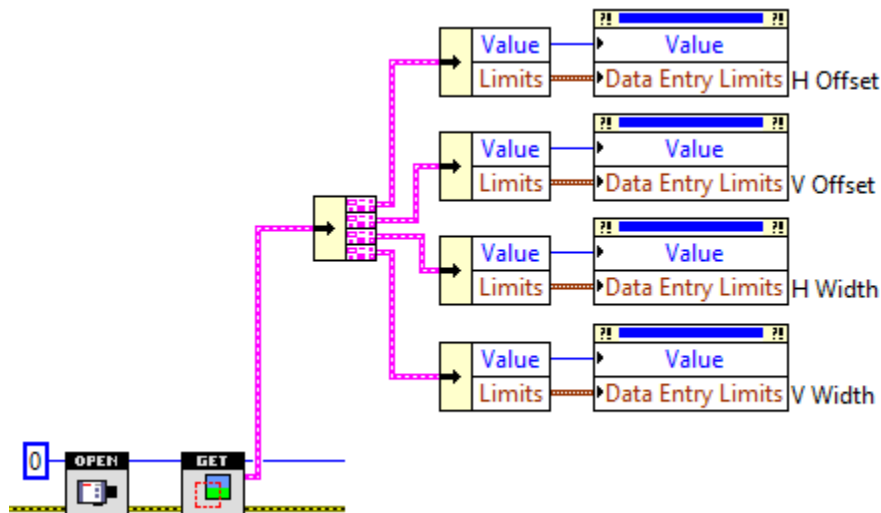
In addition, if the function is a mode function, you can get the available list of mode values for the function using TM_GETPARAMETERLIMITS. Just as in TM_GETNFUNCTIONLIST, this output can be used to set the StringsAndValues[] property of a menu ring.



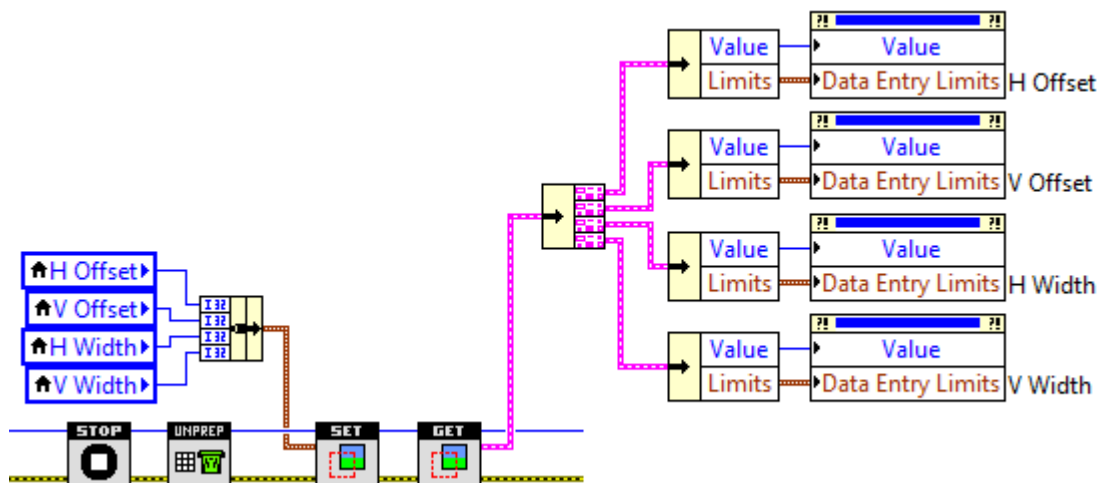
The dynamically created Function List and/or Value List menu rings can be used with TM_SETPARAMETER to control the camera. Be sure that you are using the correct Value List for the function that you are trying to control.

Region Of Interest - TM431_SUBARRAY.VI

In this example, we explain how to setup region of interest controls. And we show when and how you can change the region of interest.



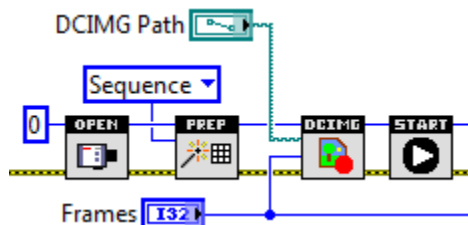
After the camera has been opened we call TM_GETAREA to get the current ROI size and limits. In this example, we take those values and plugged them directly to our controls.



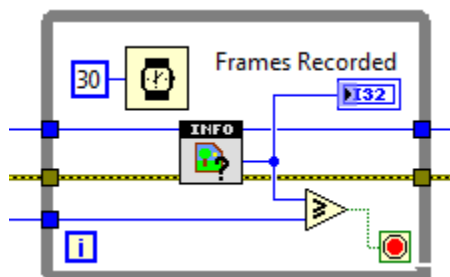
Because changing the region of interest changes the output data of the camera, we must make sure that the camera is not capturing data and we do not have the image buffer allocated. Once the capture has been stopped and buffer has been released, we can set new subarray values to the camera with TM_SETAREA. Please know that the software will auto-round your values if they do not fit in the camera limits. It is always good practice to call TM_GETAREA after you set your values to see what values the camera is actually using. Also, you will be able to get the new limits of the ROI as they would have likely changed.

Recording Direct To Disk - TM450_DCIMRECORDER.VI

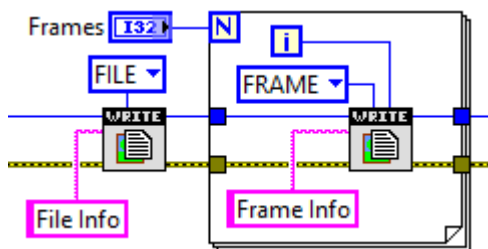
In this example, we are capturing data that is being recorded directly to the drive. This function is useful for situations when you are capturing more data that can fit in RAM at a high rate.



To start a recording, you must first prepare a capture session with TM_PREPARECAPTURE. It is best to use SEQUENCE mode when using the disk recorder. After that function is called, you may now call TM_STARTRECORDER with the file path and the number of frames to record. When images are being captured, they will be written directly to the file that you specified.



While it is capturing data, in order to determine the number of frames that were recorded, you will need to use TM_GETRECORDERSTATUS. This will provide the total number of frames recorded. This will help you to determine if the recording is completed.



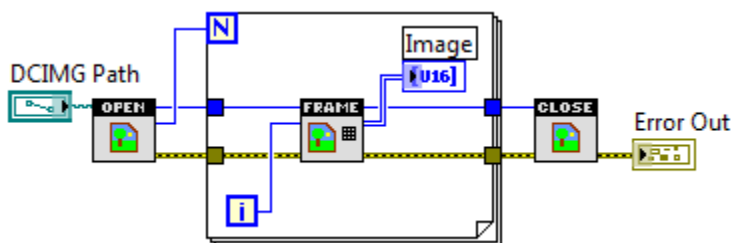
Aside from the image data, it is also possible to write metadata for the whole file and/or each individual frames. The metadata does not have to be written while the images are being recorded, but the metadata must be written before TM_STOPRECORDER is called.



Once the recording is done, you should call TM_STOPRECORDER. This will close the DCIMG file and allows it to be opened by a DCIMG reader. This function can be called anytime during capture.

Reading A DCIMG File - TM460_DCIMGREADER.VI

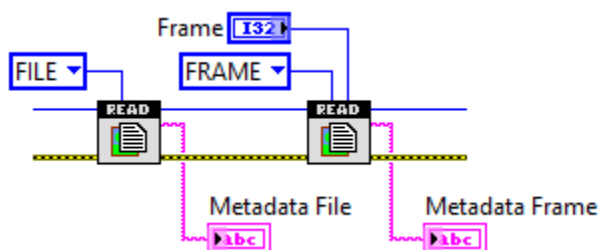
If you have data that you wish to read from a DCIMG file, you can use the functions available with this capture library. These functions are separate from the main camera control functions and can be used without a camera connected to the computer.



To read a DCIMG file, you will first need to open the file with TM_OPENDCIMGFILE and provide the complete file name and path. This will give you the file handle as well as the number of frames available in the DCIMG file. The file handle must be used with the other reader functions.

Once the file is open, you can begin reading the images that are contained in the file by calling TM_GETDCIMGFRAME. This function requires the file handle provided by TM_OPENDCIMGFILE and a frame index. The output will be a 2D array of the image data.

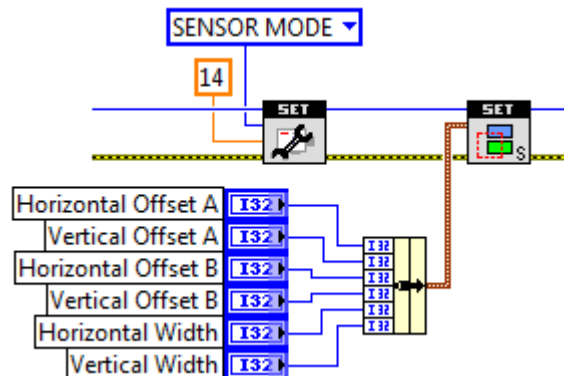
When you are finished with the DCIMG file, you can close the file with TM_CLOSEDCIMGFILE. This will free any resources used to open the file as well as allow any other DCIMG reader to open that file.



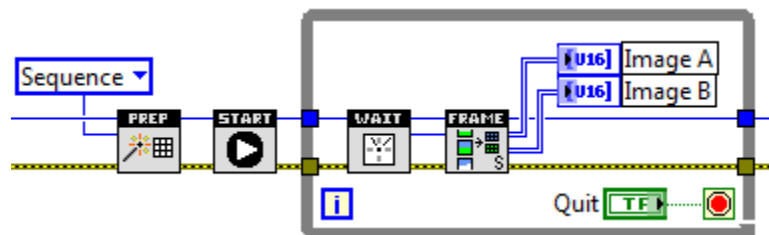
Aside from the image data, you can also read metadata that may be stored in the file. This metadata may be for the whole file and/or each individual frame.

Splitview (W-View) Mode - TM481_WVIEW.VI

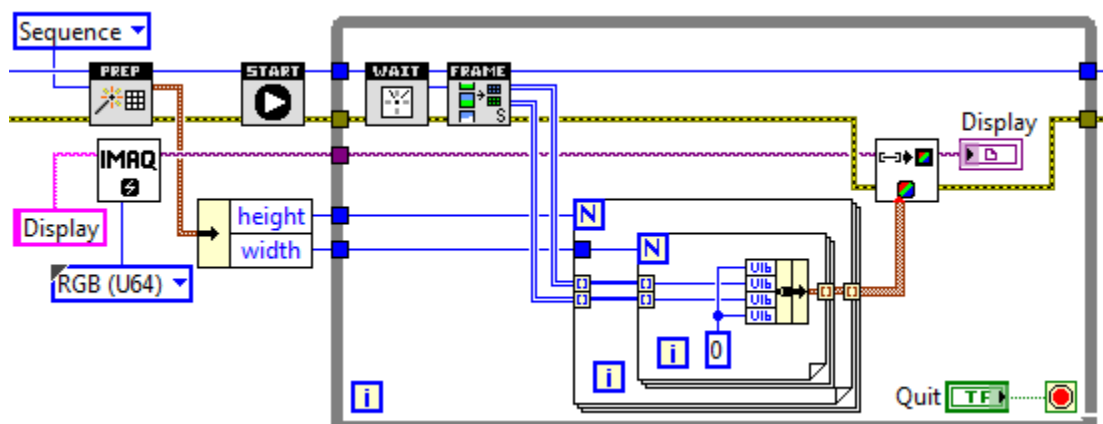
Some cameras are capable of Splitview mode. The two halves of the image could have independent settings from each other. This function is best used with the W-View Gemini optics.



In this example, we first have to setup the camera for Splitview. Splitview is a mode of the sensor, thus we use TM_SETPARAMETER and set SENSOR MODE to value 14 which is splitview mode. Once the SENSOR MODE is set, we are then able to change other parameters such as the subarray. For splitview, we use the TM_SETWVIEWAREA function to get set the subarray of both halves.



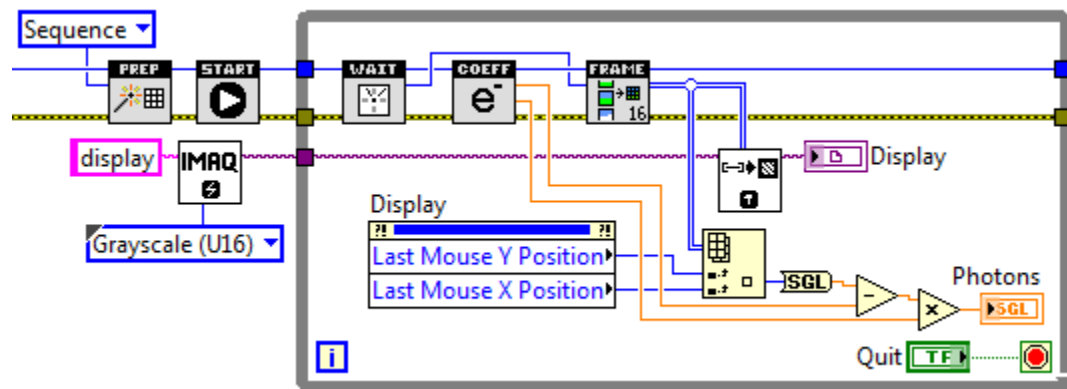
Capturing data is very simple. Instead of using TM_GETFRAME16, we will use TM_GETFRAMES16. This will give use both halves of the data as separate arrays. We could then send these two halves to separate displays.



Alternatively, you can merge both halves of the image and handle them as a single 64-bit RGB image with one half as one color, and the other half as a different color.

Electron Conversion - TM482_ELECTRONCOEFFICIENT.VI

Intensity values are useful, but some calculations require the number of photons that the sensor detected. In this example, we will show you how to convert the intensity value back to the photon count.



In this example, we are capturing data in a sequence mode. However, we call TM_GETELECTRONINFO immediately after TM_WAITNEXTFRAME returns with a new frame. This is to ensure that we get the most accurate conversion values for the frame that was just captured. We then call TM_GETFRAME16 to get the image data.

In this example, we are using NI-IMAQ Vision for the display. This is because this display is capable of reading the mouse position. We use this mouse position to get a specific pixel from the image. And with that pixel data, we apply the electron conversion formula to the photon count of that pixel. You can use these numbers to convert the intensity value to electron. Here is the formula to convert the intensity value to electrons

$$e = (intensity - offset) * coefficient$$

Parameter Reconfigure

Some parameters available to one camera may not be available to another camera. It is important to configure the Hamamatsu Video Camera Control library for your specific camera before you begin development of your software. To configure your library, simply run Reconfigure.vi from the tools folder of the installer. It is necessary to have your camera on at the time.

The Reconfigure tool will run TM_SETPARAMETER, TM_GETPARAMETER, and TM_GETPARAMETERLIMITS with the reconfigure option set to TRUE. This will read the available functions for your camera, then rewrite Parameter control box with the appropriate functions. It is important to save these files before exiting LabVIEW. When executed, these functions will contain a list of the parameters available to your camera.

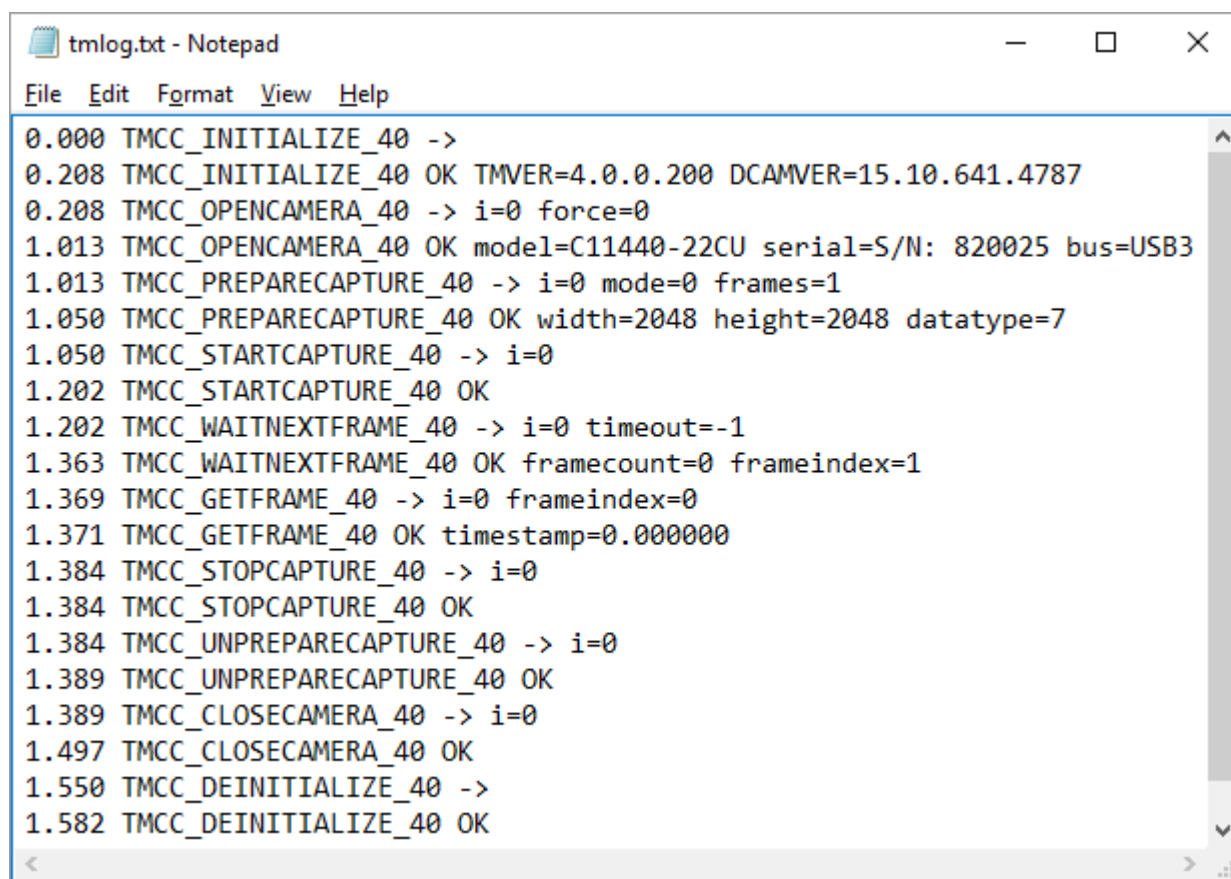
One very important thing to keep in mind, when you reconfigure the parameter list for one camera, you do not break the VIs for other camera models. This will only change the parameters visible in the combobox controls. Values that are not part of that list can still be used. Therefore, if you reconfigure your library for a Flash4.0 V2 camera, but later use an Imagem X2 with a LabVIEW VI that uses functions for that EM camera, it will still work. This reconfigure tool will only affect what parameters are visible.

HVC Debug Log

When enabled in TM_INITIALIZE, the HVC Log will record all of the functions that have been called in a session. Only HVC functions version 4.0 or higher will be recorded. And the log will be recorded to C:\temp\tmlog.txt. The generated log will be an ASCII text file which will allow you to easily open and understand the information.

Each HVC function called will have two lines recorded to the log. The first line will include the name of the function, a timestamp when the function began, and the values of any of the function inputs. The second line will include the name of the function, the timestamp when the function completed, and if the function succeeded. If the function succeeded, it will also show the values of the outputs. If the function failed, it will show the error code.

This tool can be useful when trying to diagnose a problem. However, discovering the root cause of a problem is not guaranteed. Depending on the problem you are experiencing, you may not find the answer this way.



```
tmlog.txt - Notepad
File Edit Format View Help
0.000 TMCC_INITIALIZE_40 ->
0.208 TMCC_INITIALIZE_40 OK TMVER=4.0.0.200 DCAMVER=15.10.641.4787
0.208 TMCC_OPENCAMERA_40 -> i=0 force=0
1.013 TMCC_OPENCAMERA_40 OK model=C11440-22CU serial=S/N: 820025 bus=USB3
1.013 TMCC_PREPARECAPTURE_40 -> i=0 mode=0 frames=1
1.050 TMCC_PREPARECAPTURE_40 OK width=2048 height=2048 datatype=7
1.050 TMCC_STARTCAPTURE_40 -> i=0
1.202 TMCC_STARTCAPTURE_40 OK
1.202 TMCC_WAITNEXTFRAME_40 -> i=0 timeout=-1
1.363 TMCC_WAITNEXTFRAME_40 OK framecount=0 frameindex=1
1.369 TMCC_GETFRAME_40 -> i=0 frameindex=0
1.371 TMCC_GETFRAME_40 OK timestamp=0.000000
1.384 TMCC_STOPCAPTURE_40 -> i=0
1.384 TMCC_STOPCAPTURE_40 OK
1.384 TMCC_UNPREPARECAPTURE_40 -> i=0
1.389 TMCC_UNPREPARECAPTURE_40 OK
1.389 TMCC_CLOSECAMERA_40 -> i=0
1.497 TMCC_CLOSECAMERA_40 OK
1.550 TMCC_DEINITIALIZE_40 ->
1.582 TMCC_DEINITIALIZE_40 OK
```


Parameter Definitions

Some of the capabilities of the Hamamatsu cameras are mode functions. These modes can be changed with this software, however the names of these modes are not always defined. Below is a list of the different mode functions and the definition of their values. Not all cameras are capable of each function or mode. You can get a listing of the camera functions and their value options with the TM_GETPARAMETERLIST and TM_GETPARAMETERLIMITS functions. Please refer to your camera manual for more information.

Sensor Mode

Area	1
Line	3
TDI	4
Progressive (Lightsheet)	12
Split View (W-View)	14

CCD Mode

Normal CCD	1
EM CCD	2

Binning

1x1	1
2x2	2
4x4	4
8x8	8
16x16	16

Light Mode

Low Light Mode	1
High Light Mode	2

High Dynamic Range Mode

Off	1
On	2

Sensor Cooler

Off	1
On	2

Sensor Cooler Fan

Off	1
On	2

Mechanical Shutter

Auto	1
Close	2

Open	3
------	---

Output Trigger Source

Exposure	1
----------	---

Readout End	2
-------------	---

VSync	3
-------	---

HSync	4
-------	---

Trigger	6
---------	---

Trigger Global Exposure

None	1
------	---

Always	2
--------	---

Delayed	3
---------	---

Emulate	4
---------	---

Global Reset	5
--------------	---

First Trigger Behavior

Start Exposure	1
----------------	---

Start Readout	2
---------------	---

Readout Direction

Forward	1
---------	---

Backward	2
----------	---