Introduction

The **CRi Partner MSI API** is a C interface for hardware device control and image processing operations. The software consists of two components: a software development kit (SDK) used to develop application programs, and a redistributable library DLL file(s) which is used by application programs at run-time. Windows XP SP2 is required to run the software.

This is a multi-threaded API that is compiled with Visual Studio 7.1 as a Windows DLL library. The Partner MSI API includes the following components:

- Debug and Release static library versions of the API:
- Debug and Release versions of the DLLs
- All necessary include files:

  `CriAcquireDefines.h` – Definitions and support structures for camera, filter, and image utilities. Includes error definitions and explanations.

  `CriAcquireAPI.h` – Camera, Filter, and Image operations including cube acquisition.

  `CriSpectrumDefines.h` – Definitions and support structures for spectra.

  `CriSpectrumAPI.h` – Spectra analysis operations including Spectrum IO, Unmix, and spectra sampling.

  `CriTransmissionAPI.h` – Transmission analysis operations including **`FlatField`**, **`DarkCurrentSubstract`**, **`ConvertToTransmission`**, and **`ConvertToOpticalDensity`**.

Library Descriptions

These libraries export C functions so that they can be readily used in a variety of programming environments.

CriAcquireDefines: Acquisition Data Types

`struct cri_Int8Image`
`struct cri_Int16Image`
`struct cri_Float32Image`

Single plane image with 8 or 16 bit integer pixels, or 32 bit float pixels, when **`numberOfChannels`** is 1. Image cube with 8 or 16 bit integer images or 32 bit float images when **`numberOfChannels`** is > 1. Each plane in the cube is associated with a wavelength. The image pixels are in row-major order (columns vary the fastest; this is the standard “C” order).

`struct cri_FilterState`

Stores how the filters are tuned.

`struct cri_AutoExposeParameters`

Parameters for the autoexposure algorithm.

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com

CriAcquireAPI: Acquisition Functions

General Functions

`cri_Shutdown`

At application shutdown, callers must call **`cri_Shutdown`**.

`cri_GetDescription`

Get documentary information on the version for the Partner MSI API (CriAcquireAPI) library. The description parameter should be preallocated to **`cri_MAX_STRING_LENGTH`**.

`cri_GetLastError`

Query the last error code and string description. See **`CriAcquireDefines.h`** file for a list of error codes including descriptions and interfaces that return specific errors.

Note: Not all error codes have a corresponding string description.

`cri_ClearLastError`

Clear the last error code and description. See **`CriAcquireDefines.h`** file for a list of error codes including descriptions and interfaces that return specific errors.

Acquisition Functions

`cri_GetNumberAvailableCameras`

Return the number of available cameras.

Note: This routine should not be called when a camera is open.

`cri_FindAvailableCameras`

Find all the available camera ids. The maximum number of camera ids is defined by the **`maxCameras`** parameter. The **`cameraIds`** parameter must be preallocated by the value of the **`maxCameras`** parameter. The function returns the number of actual camera ids put into the array. It does not open the camera.

Note: This routine should not be called when a camera is open.

`cri_GetNumberAvailableFilters`

Return the number of available filters.

`cri_FindAvailableFilters`

Find all the available filter ids. The maximum number of filter ids is defined by the **`maxFilters`** parameter. The **`filterIds`** parameter must be preallocated by the value of the **`maxFilters`** parameter. The function returns the number of actual filter ids put into the array. It does not open the filter.

Note: This routine should not be called when a filter is open.

`cri_OpenCamera`

Open the camera. The handle parameter is set to a valid camera on success. It is set to **`cri_InvalidCameraHandle`** on error. Returns the error code **`cri_NoError`** on success.

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com

cri_CloseCamera

Close the camera. The handle is no longer valid after calling **cri_CloseCamera**. Returns the error code **cri_NoError** on success.

cri_OpenFilter

Open the filter. The handle parameter is set to a valid filter on success. It is set to **cri_InvalidFilterHandle** on error. Returns the error code **cri_NoError** on success.

cri_CloseFilter

Close the filter. The handle is no longer valid after calling **cri_CloseFilter**. Returns the error code **cri_NoError** on success.

cri_CameraReady

Check if the camera specified by the handle is ready. The handle parameter must already be opened. The ready parameter is set to **true** if the camera is ready. Returns the error code **cri_NoError** on success.

cri_FilterReady

Check if the filter specified by the handle is ready. The handle parameter must already be opened. The ready parameter is set to **true** if the filter is ready. Returns the error code **cri_NoError** on success.

cri_IsCameraStreaming

Check if the camera specified by the handle is streaming. The handle parameter must already be opened. The ready parameter is set to **true** if the camera is streaming. Returns the error code **cri_NoError** on success.

cri_IsCameraAcquiring

Check if the camera specified by the handle is acquiring. The handle parameter must already be opened. The ready parameter is set to **true** if the camera is acquiring. An error code of **cri_CameraBusy** and acquiring parameter will be **false** if camera is busy. Returns the error code **cri_NoError** on success.

cri_StartInt8Stream
cri_StartInt16Stream

Start streaming from the given camera. Each time a new frame is acquired, the **streamCallback** function will be called. If an error is encountered, the **errorCallback** function will be called. If this is set to **NULL** it is ignored. It is an error to start streaming a camera that is already streaming. Returns the error code **cri_NoError** on success.

cri_StopStream

Tell the camera to stop streaming. If the synchronous parameter is **true**, the function will not return until the camera has stopped streaming. If the synchronous parameter is **false**, the function may return before the camera actually stops streaming. It is not an error to stop streaming a camera that is not streaming. Returns the error code **cri_NoError** on success.

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com

cri_GetCurrentCameraSettings

Query the camera's width, height, bit depth and binning. Returns the error code **cri_NoError** on success.

cri_SnapInt8
cri_SnapInt16

Snap a single image at the current filter and camera settings. Image must be preallocated (use **cri_GetCurrentCameraSettings** to find the correct width and height). If the synchronous parameter is **true**, the function will not return until image is acquired. If the synchronous parameter is **false**, the function will start an acquisition and return. The callback parameter will be called when the acquisition finishes. It will be called even for synchronous snaps. If the callback parameter is **NULL** it is ignored. Returns the error code **cri_NoError** on success.

cri_AcquireInt8Cube
cri_AcquireInt16Cube

Acquire an image cube. This will tune the filter to the given filter states, set the exposure time to the given exposure times, and collect the images. All the images must be preallocated (use **cri_GetCurrentCameraSettings** to find the correct width and height). If the synchronous parameter is **true**, the function will not return until the entire cube is acquired. If the synchronous parameter is **false**, the function will start an acquisition and return. The callback parameter will be called each time a new image is acquired. It will be called even for synchronous calls. The **errorCallback** will be called if an error occurs during acquisition. If either callback parameter is **NULL** it is ignored. Returns the error code **cri_NoError** on success.

cri_AutoExposePlane

Find the best exposure for the current camera and filter settings. The **exposureTimeMS** parameter is the calculated exposure time. Returns the error code **cri_NoError** on success.

cri_AutoExposeCube

Find the best exposure for the entire cube. If the **singleExposureTimeCube** parameter is **true**, the algorithm will find a single exposure time to use across the entire cube. If this parameter is **false**, it will calculate exposure times for each wavelength. Returns the error code **cri_NoError** on success.

Setting and Getting Various Camera Parameters

cri_GetCameraBinning
cri_SetCameraBinning

Camera binning controls how CCD sensor elements are combined to form image pixels. Increasing the binning can affect the ROI (downward). Call **cri_GetCameraRoi** afterward to detect if ROI changed.

cri_GetCameraBitDepth
cri_SetCameraBitDepth

Camera bit depth controls the bits per pixel in an image. (Images acquired at 12 bits are stored in a 16 bit data type).

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com

cri_GetCameraROI

Camera ROI controls the portion of the camera sensor that is read.

cri_SetCameraROI

Set ROI is a "request." The camera may set a different ROI from the one specified in the parameters. The actual ROI set is returned.

cri_GetCameraImageSize

Find the current image dimensions, taking the ROI and binning into account.

Gain Settings

Camera gain ranges can either be a discrete set of values or defined by a low to high range with a step size of 1. The below range interfaces will only return ranges as defined by the function name.

cri_GetCameraGainRange
cri_GetNumberCameraGainSparseRange
cri_GetCameraGainSparseRange
cri_GetCameraGain
cri_SetCameraGain

The function **cri_GetCameraGainRange** will return low and high range values if the range has a step size of 1. The function **cri_GetCameraGainSparseRange** will return a range with values that do not necessarily have a step size of 1. The error code **cri_NoError** can be used to determine if the range function provides the values. Otherwise the error code **cri_DataUnavailable** will be returned.

Offset Settings

Camera offset ranges can either be a discrete set of values or defined by a low to high range with a step size of 1. The below range interfaces will only return ranges as defined by the function name.

cri_GetCameraOffsetRange
cri_GetNumberCameraOffsetSparseRange
cri_GetCameraOffsetSparseRange
cri_GetCameraOffset
cri_SetCameraOffset

The function **cri_GetCameraOffsetRange** will return low and high range values if the range has a step size of 1. The function **cri_GetCameraOffsetSparseRange** will return a range with values that do not necessarily have a step size of 1. The error code **cri_NoError** can be used to determine if the range function provides the values. Otherwise the error code **cri_DataUnavailable** will be returned.

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com

Exposure Time

Camera exposure ranges can either be a discrete set of values or defined by a low to high range with a step size of 1ms. The below range interfaces will only return ranges as defined by the function name.

cri_GetCameraExposureRangeMs
cri_GetNumberCameraExposureMsSparseRange
cri_GetCameraExposureMsSparseRange
cri_GetCameraExposureMs
cri_SetCameraExposureMs

The function **cri_GetCameraExposureRangeMs** will return low and high range values if the range has a step size of 1ms. The function **cri_GetCameraExposureSparseRangeMs** will return a range with values that do not necessarily have a step size of 1ms. The error code **cri_NoError** can be used to determine if the range function provides the values. Otherwise the error code **cri_DataUnavailable** will be returned.

---

cri_GetCameraCoolerEnabled

Get temperature controls.

---

cri_GetCameraTemperature

Get temperature in Celsius.

---

cri_GetCameraSensorSize

Get the number of sensors in the camera.

---

cri_GetCameraDescription

Get documentary information on the camera. Strings should be preallocated to **cri_MAX_STRING_LENGTH**.

Setting and getting various filter parameters

---

cri_GetFilterState

Get wavelength information.

---

cri_SetFilterState

Tune the filter to the specified wavelength. If the synchronous parameter is **true**, the function will not return until the tuning is complete. If the parameter is **false**, the function will start tuning and return. Use the **cri_WaitForFilterTuneComplete** function to determine when the filter has finished tuning.

---

cri_WaitForFilterTuneComplete

This is useful when **cri_SetFilterState** is called asynchronously (synchronous flag set to **false**). This function will not return until the filter has completed tuning.

---

cri_GetFilterTemperature

Get temperature in Celsius.

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com

cri_GetFilterRangeAndStepSize

Get filter range and step size.

cri_GetFilterDescription

Get documentary information on the filter.

General Image Utility Functions

cri_FlipInt8Image
cri_FlipInt16Image
cri_FlipFloat32Image

Flip an image around its x, y, or x and y axis. The axis parameter determines what axis to flip (this may be bit-or'd to flip around both axes). We can do this in place by setting the **result** and **toFlip** parameters to the same image.

CriSpectrumDefines: Spectrum Data Types

struct cri_Spectrum

A spectrum is a collection of wavelengths and the magnitudes at those wavelengths.

struct cri_ImageRegionOfInterest

An image region of interest defines pixels in an image that certain algorithms restrict themselves to. The **originX**, **originY**, **width**, and **height** fields define a rectangle within a larger image. The **mask** field is a two-dimensional array of **width** and **height** (in row major order). A non-zero value in this mask marks the pixel as part of the region of interest.

CriSpectrumAPI: Spectrum Support: Unmixing, Compute Pure Spectra, etc.

cri_WriteSpectralLibraryTabDelimited

Save the spectra in a tab delimited file. Returns the error code **cri_NoError** on success.

cri_ReadSpectralLibraryTabDelimited

Read the spectra from a tab delimited file. The spectra array must be preallocated. The **wavelengths/magnitudes** fields for each spectrum must also be preallocated.

The **numberOfWavelengthsAllocated** parameter is the number of wavelengths allocated for the wavelengths/magnitudes in each spectrum.

The **numberOfWavelengths** parameter is set to the number of actual wavelengths/magnitudes defined in the file. If this parameter is greater than the allocated parameter, then the fields should be reallocated with this value and the function called again.

The **numberOfSpectraInLibrary** parameter is the number of spectra stored in the file. If it is less than **numberOfSpectraAllocated**, only the first **numberOfSpectraInLibrary** elements are valid. If the **numberOfSpectraInLibrary** is greater than **numberOfSpectraAllocated** then the spectra array should be reallocated so it has enough space to load all the spectra. Returns the error code cri_**NoError** on success.

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com

cri_UnmixInt8Cube
cri_UnmixInt16Cube
cri_UnmixFloat32Cube

Unmix an image cube into the component images using the given spectra. The **componentImages** parameter must be preallocated to the same width and height of the cube. There must be at least one spectra to unmix into. Returns the error code **cri_NoError** on success.

cri_ComputePureSpectra

Given a sample of a mixed spectrum (**aPlusB**) and a sample of a pure spectrum (**pureA**), compute the spectrum representing B (**pureB**). Set the **fitOffset** parameter to **true** if the algorithm should try to compensate for a constant (gray) offset in the image data. The **userFineTuning** parameter should be between 1 and 100 and lets users interactively fine tune the computed pure spectra. If you are not doing the computation interactively, set this parameter to 1.0F. On success, the **pureB** parameter's **magnitude** field will be allocated and populated with **numberOfWavelengths** entries. Returns the error code **cri_NoError** on success.

cri_GetSpectraSampleInt8Cube
cri_GetSpectraSampleInt16Cube
cri_GetSpectraSampleFloat32Cube

Given a region of interest mask and a cube, extract a sample spectrum from the cube. The **resultSpectrum** fields **wavelengths** and **magnitudes** must be preallocated to the number of planes in the cube. Returns the error code **cri_NoError** on success.

CriTransmissionAPI: Transmission Support: Convert to Transmission, Optical Density, etc.

cri_FlatFieldInt8Image
cri_FlatFieldInt16Image
cri_FlatFieldFloat32Image

Flat field correct an image. The white image should be an image taken against a uniform, white background. Correction is done in-place if the **toCorrect** parameter and the **correctedImage** parameter point to the same image. The **whiteImage** and **toCorrect** image must be the same size. If not correcting in-place, the **correctedImage** parameter must be preallocated to same size as the **toCorrect** image. Returns the error code **cri_NoError** on success.

cri_DarkCurrentSubtractInt8Image
cri_DarkCurrentSubtractInt16Image
cri_DarkCurrentSubtractFloat32Image

Subtract the dark current from each pixel in the image. The function **cri_DarkCurrentSubtractInt8Image** always clips to zero in all cases. The **cri_DarkCurrentSubtractInt16Image** and **cri_DarkCurrentSubtractFloat32Image** functions do so according to the **forceNonNegative** call argument. Subtraction is done in-place if one sets the **toCorrect** parameter and the **correctedImage** parameter to the same image. If not subtracting in-place, **correctedImage** parameter must be preallocated to same size as the **toCorrect** image. Returns the error code **cri_NoError** on success.

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com

cri_ConvertToTransmissionInt8Image
cri_ConvertToTransmissionInt16Image
cri_ConvertToTransmissionFloat32Image

Convert a raw image to a transmission image. The **transmissionImage** parameter must be preallocated to the same size as the **rawImage**. If **whiteImage** is **NULL**, the conversion will use the largest signal in the raw image to measure the incident light. If **whiteImage** is not **NULL**, it must be the same size as **rawImage**, and each pixel in the **whiteImage** will be used as the measure of the incident light at that pixel. Returns the error code **cri_NoError** on success.

cri_ConvertToTransmissionInt8Cube
cri_ConvertToTransmissionInt16Cube
cri_ConvertToTransmissionFloat32Cube

These are convenience functions that convert raw data to transmission data one cube at a time. See the documentation for **cri_ConvertToTransmissionImage** for a description of the parameters.

cri_ConvertToOpticalDensityInt8Image
cri_ConvertToOpticalDensityInt16Image
cri_ConvertToOpticalDensityFloat32Image

Convert a raw image to an optical density image.

IMPORTANT: This takes a raw image as input, not a transmission image.

The **odImage** parameter must be preallocated to the same size as the **rawImage**. If **whiteImage** is **NULL**, the conversion will use the largest signal in the raw image to measure the incident light. If **whiteImage** is not **NULL**, it must be the same size as **rawImage**, and each pixel in the **whiteImage** will be used as the measure of the incident light at that pixel. Returns the error code **cri_NoError** on success.

cri_ConvertToOpticalDensityInt8Cube
cri_ConvertToOpticalDensityInt16Cube
cri_ConvertToOpticalDensityFloat32Cube

These are convenience functions that do raw to optical density conversions one cube at a time. See the documentation for **cri_ConvertToOpticalDensity** for a description of the parameters.

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com

Static Libraries

Release\CriMsiAPI-0_10.lib – Partner MSI API release static library

Debug\ CriMsiAPID-0_10.lib – Partner MSI API debug static library

Runtime DLL Libraries

Release\CriMsiAPI-0_10.dll – CriMsiAPI Dynamic Link Library

boost_thread-vc71-mt-1_32.dll – Boost Support Library

INetTrans.exe – InstallShield® Trialware Online Activation Utility

IsSvcInstCriMsiAPI-0_10.dll – System Level Service Installer

Debug\ CriMsiAPID-0_10.dll – CriMsiAPI Dynamic Link Library

boost_thread-vc71-mt-gd-1_32.dll – Boost Support Library

INetTrans.exe – InstallShield® Trialware Online Activation Utility

IsSvcInstCriMsiAPID-0_10.dll – System Level Service Installer

Microsoft Visual Studio Project Examples

Examples\API_C contains an example of a C++ Console program that uses the Partner API. The program performs the following operations:

- Open a camera port
- Open a filter port
- Display camera and filter characteristics
- Change the camera bit depth to 8 bit
- Auto Expose for camera snap operation
- Perform a camera snap operation asynchronously
- Flip the snapped image along the X and Y axis
- Start a camera stream asynchronous operation and capture 8 images
- Auto expose for acquiring a cube with 11 images
- Acquire cube
- Close filter port
- Close camera port

Examples\API_MFC contains an example of a Windows MFC C++ GUI program that uses the Partner API with MFC. The program displays buttons that perform the following operations:

- Open a camera port and filter port and then display camera and filter characteristics
- Perform an 8 bit 1x1 binning camera snap operation asynchronously and then flip the snapped image along the X and Y axis
- Start a camera stream asynchronous operation
- Stop a camera stream synchronously
- Perform an 8 bit 1x1 binning cube acquire operation asynchronously
- Unmix an 8 bit cube
- Close a camera and filter port

CRi, 35-B Cabot Road, Woburn, MA, 01801, USA :: (Toll-Free US) 1-800-383-7924, (P) +1-781-935-9099, (F) +1-781-935-3388 :: sales@cri-inc.com :: www.cri-inc.com