

DexelaDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

 BusScanner	This class is used to scan the different interfaces and give information about devices found.
 DetStatus	Structure to hold the detector current status.
 DevInfo	A structure to hold device information.
 DexelaDetector	This class is used to control any interface-type Detector and acquire images from it. It will provide all the basic functionality required for all different Dexela detectors. For interface specific functionality please see the interface specific classes (e.g. DexelaDetectorGE , DexelaDetectorCL).
 DexelaDetectorCL	This class is used to control CameraLink Type Detectors. It will give access to functions that are not available to other interface-type detectors. Note: For all standard detector function calls please see the DexelaDetector class (these

	<p>functions are also available to DexelaDetectorCL objects)</p>
 DexelaDetectorGE	<p>This class is used to control GigE Type Detectors. It will give access to functions that are not available to other interface-type detectors.</p> <p>Note: For all standard detector function calls please see the DexelaDetector class (these functions are also available to DexelaDetectorGE objects)</p>
 DexelaException	<p>This class contains information about any possible error's in the API. In the event of a problem a DexelaException will be thrown.</p> <p>Note: It is suggested that you wrap your code in a try-catch block to ensure that if any errors occur you can detect (and properly handle them) in your code.</p>
 DexImage	<p>This class is used to store and handle the images acquired from a detector.</p>
 GeometryCorrectionParams	<p>A structure used to specify the new image dimensions for geometry correction</p>

DexelaDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members
BusScanner Class Reference			Public Member Functions Friends List of all members

This class is used to scan the different interfaces and give information about devices found. [More...](#)

```
#include <BusScanner.h>
```

Public Member Functions

BusScanner (void)

Constructor for **BusScanner**. More...

~BusScanner (void)

Destructor for **BusScanner**. More...

int **EnumerateDevices** ()

This method will enumerate all devices (regardless of interface) currently connected to the system. More...

int **EnumerateGEDevices** ()

This method will enumerate all GigE devices currently connected to the system. More...

int **EnumerateCLDevices** ()

This method will enumerate all CameraLink devices currently connected to the system. More...

DevInfo **GetDevice** (int index)

This method will return a **DevInfo** object for the device at the specified index. This object contains all necessary information for establishing a connection with the detector (**DexelaDetector(DevInfo &defInfo)**).

Note: To determine how many devices are currently connected to the system you can call **EnumerateDevices**

Note2: This method will call **EnumerateDevices** automatically if it has not previously been called More...

DevInfo **GetDeviceGE** (int index)

This method will return a **DevInfo** object for the GigE device at the specified index. This object contains all necessary information for establishing a connection with the detector (**DexelaDetector(DevInfo &defInfo)**).

Note: To determine how many GigE devices are currently connected to the system you can call

EnumerateGEDevices

Note2: This method will call **EnumerateGEDevices** automatically if it has not previously been called [More...](#)

DevInfo **GetDeviceCL** (int index)

This method will return a **DevInfo** object for the CameraLink device at the specified index. This object contains all necessary information for establishing a connection with the detector (**DexelaDetector(DevInfo &defInfo)**).

Note: To determine how many CL devices are currently connected to the system you can call

EnumerateCLDevices

Note2: This method will call **EnumerateCLDevices** automatically if it has not previously been called [More...](#)

Friends

class **ScanMockSetter**

Detailed Description

This class is used to scan the different interfaces and give information about devices found.

Constructor & Destructor Documentation

BusScanner::BusScanner (void)

Constructor for **BusScanner**.

BusScanner::~BusScanner (void)

Destructor for **BusScanner**.

Member Function Documentation

int BusScanner::EnumerateCLDevices ()

This method will enumerate all CameraLink devices currently connected to the system.

Returns

The count of the number of CL devices found

Exceptions

[DexelsException](#)

int BusScanner::EnumerateDevices ()

This method will enumerate all devices (regardless of interface) currently connected to the system.

Returns

The count of the number of devices found

Exceptions

[DexelsException](#)

int BusScanner::EnumerateGEDevices ()

This method will enumerate all GigE devices currently connected to the system.

Returns

The count of the number of GigE devices found

Exceptions

DexelaException

DevInfo BusScanner::GetDevice (int index)

This method will return a **DevInfo** object for the device at the specified index. This object contains all necessary information for establishing a connection with the detector
([DexelaDetector\(DevInfo &defInfo\)](#)).

Note: To determine how many devices are currently connected to the system you can call [EnumerateDevices](#)

Note2: This method will call [EnumerateDevices](#) automatically if it has not previously been called

Parameters

index The index of the device to access. If this index is out of bounds an exception will be thrown.

Returns

The **DevInfo** object with the detector at the desired index

Exceptions

[DexelaException](#)

DevInfo BusScanner::GetDeviceCL (int index)

This method will return a **DevInfo** object for the CameraLink device at the specified index. This object contains all necessary information for establishing a connection with the detector
([DexelaDetector\(DevInfo &defInfo\)](#)).

Note: To determine how many CL devices are currently connected to the system you can call [EnumerateCLDevices](#)

Note2: This method will call [EnumerateCLDevices](#) automatically if it has not previously been called

Parameters

index The index of the device to access. If this index is out of bounds an exception will be thrown.

Returns

The **DevInfo** object with the detector at the desired index

Exceptions

DexelaException

DevInfo BusScanner::GetDeviceGE (int index)

This method will return a **DevInfo** object for the GigE device at the specified index. This object contains all necessary information for establishing a connection with the detector
(**DexelaDetector(DevInfo &defInfo)**).

Note: To determine how many GigE devices are currently connected to the system you can call **EnumerateGEDevices**

Note2: This method will call **EnumerateGEDevices** automatically if it has not previously been called

Parameters

index The index of the device to access. If this index is out of bounds an exception will be thrown.

Returns

The **DevInfo** object with the detector at the desired index

Exceptions

DexelaException

The documentation for this class was generated from the following files:

- **BusScanner.h**
- **BusScanner.cpp**

DexelsDetector API

Main Page	Classes	Files			
Class List	Class Index	Class Hierarchy	Class Members	Public Attributes List of all members	
<h2>DetStatus Struct Reference</h2>					

Structure to hold the detector current status. [More...](#)

```
#include <DexDefs.h>
```

Public Attributes

ExposureModes **exposureMode**
The currently set exposure mode [More...](#)

FullWellModes **fullWellMode**
The currently set Full Well mode [More...](#)

float **exposureTime**
The currently set exposure time [More...](#)

bins **binLevel**
The currently set bin level [More...](#)

ExposureTriggerSource **triggerSource**
The currently set Trigger Source [More...](#)

BOOL **testMode**
True if the detector test mode is set to on
[More...](#)

Detailed Description

Structure to hold the detector current status.

Member Data Documentation

bins DetStatus::binLevel

The currently set bin level

ExposureModes DetStatus::exposureMode

The currently set exposure mode

float DetStatus::exposureTime

The currently set exposure time

FullWellModes DetStatus::fullWellMode

The currently set Full Well mode

BOOL DetStatus::testMode

True if the detector test mode is set to on

ExposureTriggerSource DetStatus::triggerSource

The currently set Trigger Source

The documentation for this struct was generated from the following file:

- **DexDefs.h**
-

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members
Public Attributes List of all members			

DevInfo Struct Reference

A structure to hold device information. [More...](#)

```
#include <DexDefs.h>
```

Public Attributes

int **model**

The Device Model Number [More...](#)

int **serialNum**

The Device Serial Number [More...](#)

DetectorInterface **iface**

char **param** [50]

Pointer to the parameter needed for opening
detector [More...](#)

int **unit**

Unit number for cameralink detectors [More...](#)

TransportLib **transport**

Low level tranport library

Detailed Description

A structure to hold device information.

Member Data Documentation

DetectorInterface DevInfo::iface

The interface type (e.g. GigE or CameraLink)

int DevInfo::model

The Device Model Number

char DevInfo::param[50]

Pointer to the parameter needed for opening detector

int DevInfo::serialNum

The Device Serial Number

int DevInfo::unit

Unit number for cameralink detectors

The documentation for this struct was generated from the following file:

- [DexDefs.h](#)
-

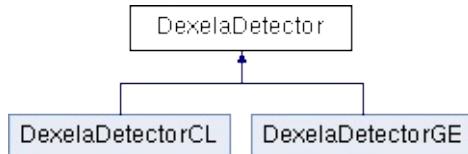
DexelsDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members
			Classes Public Member Functions Protected Attributes Friends List of all members
DexelsDetector Class Reference			

This class is used to control any interface-type Detector and acquire images from it. It will provide all the basic functionality required for all different Dexela detectors. For interface specific functionality please see the interface specific classes (e.g. [DexelsDetectorGE](#), [DexelsDetectorCL](#)). [More...](#)

```
#include <DexelsDetector.h>
```

Inheritance diagram for DexelaDetector:



Public Member Functions

DexelaDetector (DevInfo &devInfo)
Constructor for **DexelaDetector**. This version uses the **DevInfo** struct returned from a **GetDevice**, **GetDeviceGE** or **GetDeviceCL** call. [More...](#)

DexelaDetector (DetectorInterface transport, int unit, const char *params)
Constructor for **DexelaDetector**. This version assumes you know the interface and the correct parameters to connect to the detector. [More...](#)

virtual ~DexelaDetector (void)
Destructor for **DexelaDetector**. [More...](#)

virtual void OpenBoard ()
Opens the connection to the detector. Every open should be matched with a close to free resources. [More...](#)

void OpenBoard (int NumBufs)
Opens the connection to the detector and sets the number of buffers to use/allocate. Every open should be matched with a close to free resources. [More...](#)

void CloseBoard ()
Closes the connection to the detector. [More...](#)

int GetBufferXdim (void)
Get the x dimension of the transport buffer (in bytes) [More...](#)

int [GetBufferYdim](#) (void)

Get the y dimension of the transport buffer (in pixels) [More...](#)

int [GetNumBuffers](#) (void)

Get the number of internal buffers that are currently allocated for the detector.

[More...](#)

int [GetCapturedBuffer](#) (void)

Gets the number of the buffer just captured. This can be used to determine which buffer to read-out. [More...](#)

int [GetFieldCount](#) (void)

Gets the number of fields(frames) captured so far. [More...](#)

void [ReadBuffer](#) (int bufNum, byte *buffer)

Reads the specified transport buffer into the passed in buffer (byte*).

Note: [GetCapturedBuffer](#) can be used to get the number of the lastest buffer to be filled. [More...](#)

void [ReadBuffer](#) (int bufNum, DexImage &img, int iZ=0)

Reads the specified transport buffer into the passed in **DexImage** object at the passed in plane.

Note: [GetCapturedBuffer](#) can be used to get the number of the lastest buffer to be filled. [More...](#)

void [WriteBuffer](#) (int bufNum, byte *buffer)

Writes data to the specified transport buffer. [More...](#)

void **SetFullWellMode** (**FullWellModes** fwm)
Sets the full well mode parameter of the detector. [More...](#)

void **SetExposureMode** (**ExposureModes** mode)
Sets the ExposureMode parameter of the detector [More...](#)

void **SetExposureTime** (float timems)
Sets the exposure time parameter of the detector [More...](#)

void **SetBinningMode** (**bins** flag)
Sets the binning mode of the detector. [More...](#)

void **SetTestMode** (BOOL SetTestOn)
Enables/disables test mode. The detector will output a generated test pattern if this mode is turned on. [More...](#)

void **SetTriggerSource**
(**ExposureTriggerSource** ets)
Sets the trigger source setting on the detector [More...](#)

void **SetNumberOfExposures** (int num)
Sets the number of exposures to acquire after a trigger. This is only relevant in **Sequence_Exposure** and **Frame_Rate_exposure** modes of operation. [More...](#)

int **GetNumberOfExposures** ()
Gets the number of exposures setting from the detector. This is only relevant in **Sequence_Exposure** and

Frame_Rate_exposure modes of operation. [More...](#)

void SetGapTime (float timems)
Sets the gap-time setting of the detector. When run in **Frame_Rate_exposure** mode the detector will insert this gap period between consecutive frames in an image sequence.
Note: The minimum time for the gap-time setting is the current readout-time for the detector. Attempting to write anything smaller to the detector will result in a gap-time equal to the readout-time. [More...](#)

float GetGapTime ()
Gets the current gap-time setting of the detector. When run in **Frame_Rate_exposure** mode the detector will insert this gap period between consecutive frames in an image sequence.
Note: The minimum time for the gap-time setting is the current readout-time for the detector. Attempting to write anything smaller to the detector will result in a gap-time equal to the readout-time. [More...](#)

bool IsConnected ()
Check to see if the connection to the detector is open (i.e. **OpenBoard**) [More...](#)

ExposureModes GetExposureMode ()
Gets the ExposureMode parameter of the detector. [More...](#)

float GetExposureTime ()

Gets the exposure time parameter of the detector (ms). [More...](#)

DetStatus **GetDetectorStatus ()**
Returns the current settings of the detector in the form of a **DetStatus** object. [More...](#)

ExposureTriggerSource **GetTriggerSource ()**
Gets the current trigger source setting from the detector [More...](#)

BOOL **GetTestMode ()**
Gets the current state of the detector test mode (on/off) [More...](#)

FullWellModes **GetFullWellMode ()**
Gets the current detector well-mode. [More...](#)

bins **GetBinningMode ()**
Gets the current state of the detector binning mode [More...](#)

int **GetSerialNumber ()**
Gets the detector serial number. [More...](#)

int **GetModelNumber ()**
Gets the detector model number. [More...](#)

int **GetFirmwareVersion ()**
Gets the detector firmware version number. [More...](#)

void **GetFirmwareBuild (int &iDayAndMonth,
int &iYear, int &iTime)**
Gets the detector firmware build date.

Note: This feature may not be supported on older detectors [More...](#)

DetectorInterface **GetTransportMethod ()**
Returns the communication method (i.e. interface) for the detector object. [More...](#)

double **GetReadOutTime ()**
This method will return the read-out time of the detector (in ms) for it's current binning mode. [More...](#)

bool **IsCallbackActive ()**
This method will inform the user if the callback mode (i.e. background thread) is currently active. [More...](#)

bool **IsLive ()**
This method will inform the user if detector is currently in Live mode. [More...](#)

void **Snap (int buffer, int timeout)**
Snaps an image into the specified buffer.
Note: If the detector trigger source is set to **Internal_Software**, this call will automatically trigger the detector. [More...](#)

int **ReadRegister (int address, int sensorNum=1)**
Reads the specified register from the detector. The sensor number corresponds to the desired sensor from which to read the register. The SensorNumber will default to 1 (master-sensor) if not specified otherwise by the user. [More...](#)

void **WriteRegister (int address, int value, int sensorNum=0)**

Writes the value to the specified register from the detector. The sensor number corresponds to the desired sensor to which the value will be written. The SensorNumber will default to 0 (broadcast to all sensors) if not specified otherwise by the user. [More...](#)

void [ClearCameraBuffer](#) (int i)
Clears (i.e. zero-out) the specified camera buffer. [More...](#)

void [ClearBuffers](#) ()
Clears all the camera buffers [More...](#)

void [LoadSensorConfigFile](#) (char *filename)
Loads the sensor configuration file into the detector. This file will write values to the ADC offset registers for each sensor in the detector. [More...](#)

void [SoftReset](#) (void)
Cycles the power on the detector [More...](#)

void [GoLiveSeq](#) (int start, int stop, int numBuf)
Sets the host computer up to be ready to receive images into the specified buffer range. [More...](#)

void [GoLiveSeq](#) ()
Sets the host computer up to be ready to receive images. This call will use all available buffers in a circular fashion (i.e. ring-buffer). [More...](#)

void [GoUnLive](#) ()
Exits live mode. The host computer will

no longer be ready to receive transmitted images. [More...](#)

void **SoftwareTrigger ()**

Sends a trigger to the detector (will only work if the trigger source is set to [Internal_Software](#)) [More...](#)

void **EnablePulseGenerator (float frequency)**

This function will enable the pulse generator software trigger signal. In this mode the software trigger can be continuously sent to the detector at the desired frequency.

Note: In order to use this mode the trigger source should be set to Internal_Software

Note2: To actually enable the pulse train you must call [ToggleGenerator](#). A [SoftwareTrigger](#) call will not work when in this mode. [More...](#)

void **EnablePulseGenerator ()**

This function is identical to [EnablePulseGenerator](#), except that the frequency of the pulse train is set automatically. The frequency will be set such as to ensure continuous image acquisition for the current detector/binning mode.

Note: In order to use this mode the trigger source should be set to Internal_Software

Note2: To actually enable the pulse train you must call [ToggleGenerator](#). A [SoftwareTrigger](#) call will not work when in this mode. [More...](#)

void DisablePulseGenerator ()

This function will disable Pulse Generator mode. After calling this you should be able to use the **SoftwareTrigger** call.

[More...](#)

void ToggleGenerator (BOOL onOff)

This function will control the pulse train.

Note: In order to use this mode the pulse generator must be enabled. See **EnablePulseGenerator**. [More...](#)

void WaitImage (int timeout)

This function will wait for the specified amount of time for an image to arrive.

Note: If the image arrives before then it will return as soon as it does (i.e. it won't wait for the duration of the timeout period). If the image does not arrive in the specified time a **DexelaException** will be thrown. [More...](#)

void SetCallback (IMAGE_CALLBACK func)

Sets the user defined callback function to be called for every image arrival event.

[More...](#)

void StopCallback ()

This function will terminate the callback loop (i.e. will wait for all spawned threads to finish executing). [More...](#)

void CheckForCallbackError ()

This function will check to see if any errors have occurred in the background thread that is running when using callbacks. This thread is activated after a call to **SetCallback** and terminated with a call to **StopCallback**. If no error has

occurred this method will just return. if an error has occurred a **DexelaException** will be thrown. [More...](#)

void CheckForLiveError ()

This function will check to see if any errors have occurred in the background thread that is running when using live-mode. This thread is activated after a call to **GoLiveSeq** and terminated with a call to **GoUnLive**. If no error has occurred this method will just return. if an error has occurred a **DexelaException** will be thrown. [More...](#)

void SetPreProgrammedExposureTimes (int numExposures, float *exposuretimes_ms)

This method will set the exposure times for pre-programmed exposure mode.

[More...](#)

void SetROICoordinates (unsigned short usStartColumn, unsigned short usStartRow, unsigned short usROIWidth, unsigned short usROIHeight)

This method set the coordinates of the ROI when detector runs in ROI mode.

Parameters

usStartColumn Sensor column to start the ROI read out from.

usStartRow Sensor width to start the ROI read out from.

usROIWidth Width (number of columns) of the ROI.

usROIHeight Height (number of columns) of the ROI.

Exceptions

DexelaException

GetROICoordinates (unsigned short &usStartColumn, unsigned short &usStartRow, unsigned short &usROIWidth, unsigned short &usROIHeight)
This method retrieves the coordinates of the region of interest (ROI) set within the detector. [More...](#)

void EnableROIMode (bool bEnableROI)
This method activates or deactivates the ROI mode of the detector. [More...](#)

bool GetROIState ()
This method retrieves the enabled state of the region of interest of the detector. [More...](#)

unsigned short GetSensorHeight (unsigned short uiSensorID=1)
Gets the height of the sensor in pixels. [More...](#)

unsigned short GetSensorWidth (unsigned short uiSensorID=1)
Gets the width of the sensor in pixels. [More...](#)

bool IsFrameCntWithinImage ()
Checks if framecounter is displayed within the image data (being the 2nd pixel).
[More...](#)

void EnableFrameCntWithinImage (unsigned short usEnable)
Enables displaying the framecounter in the image (being the 2nd pixel). [More...](#)

void SetSlowed (bool flag)
This method can specify to the api that the detector being used is a slowed-down detector (e.g. mammo detector). This should not be necessary as the API should be able to determine most of the time whether the firmwarwe version is a slowed down one. However, for certain older detectors/firmwares this may not be possible. In this case this method can be used to inform the library that the firmware is slowed down and it will use the correct read-out times. [More...](#)

void SetReadoutMode (ReadoutModes mode)
Sets the ReadoutMode parameter of the detector [More...](#)

ReadoutModes GetReadoutMode ()
Gets the ReadoutModes parameter of the detector. [More...](#)

int QueryReadoutMode (ReadoutModes mode)
Query the detector to see if the desired readout mode is present (i.e. available).
Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that

it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

int **QueryExposureMode (ExposureModes mode)**

Query the detector to see if the desired exposure mode is present (i.e. available). **Note:** Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

int **QueryTriggerSource (ExposureTriggerSource ets)**

Query the detector to see if the desired trigger source is present (i.e. available). **Note:** Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

`int QueryFullWellMode (FullWellModes fwm)`

Query the detector to see if the desired full-well mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

`int QueryBinningMode (bins flag)`

Query the detector to see if the desired binning mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

Protected Attributes

boost::shared_ptr< baseDetector > **base**

boost::shared_ptr< gigEDetector > **gigeDet**

boost::shared_ptr
< camLinkDetector > **clDet**

Friends

class **baseBusScanner**

class **MockSetter**

class **DexelsDetectorPy**

class **Dex_CL**

Detailed Description

This class is used to control any interface-type Detector and acquire images from it. It will provide all the basic functionality required for all different Dexela detectors. For interface specific functionality please see the interface specific classes (e.g. [DexelaDetectorGE](#), [DexelaDetectorCL](#)).

Constructor & Destructor Documentation

DexelaDetector::DexelaDetector ([DevInfo](#) & [devInfo](#))

Constructor for [DexelaDetector](#). This version uses the [DevInfo](#) struct returned from a [GetDevice](#), [GetDeviceGE](#) or [GetDeviceCL](#) call.

Parameters

devInfo The [DevInfo](#) object for the desired detector.

Exceptions

[DexelaException](#)

DexelaDetector::DexelaDetector ([DetectorInterface](#) [transport](#), [int](#) [unit](#), [const char *](#) [params](#))

Constructor for [DexelaDetector](#). This version assumes you know the interface and the correct parameters to connect to the detector.

Parameters

transport The [DetectorInterface](#) for the detector (i.e. CL or GIGE)

unit The unit number for CL type detectors. For GIGE detectors this can be set to 0

params The parameter string for connection to the detector. For GIGE detectors this should be the detector IP address. For CL detectors this parameter will be ignored.

Exceptions

DexelsException

DexelsDetector::~DexelsDetector (void)

virtual

Destructor for **DexelsDetector**.

Member Function Documentation

void DexelaDetector::CheckForCallbackError()

This function will check to see if any errors have occurred in the background thread that is running when using callbacks. This thread is activated after a call to **SetCallback** and terminated with a call to **StopCallback**. If no error has occurred this method will just return. if an error has occurred a **DexelaException** will be thrown.

Exceptions

DexelaException

void DexelaDetector::CheckForLiveError()

This function will check to see if any errors have occurred in the background thread that is running when using live-mode. This thread is activated after a call to **GoLiveSeq** and terminated with a call to **GoUnLive**. If no error has occurred this method will just return. if an error has occurred a **DexelaException** will be thrown.

Exceptions

DexelaException

void DexelaDetector::ClearBuffers()

Clears all the camera buffers

Exceptions

DexelaException

void DexelaDetector::ClearCameraBuffer (int **i)**

Clears (i.e. zero-out) the specified camera buffer.

Parameters

i The buffer number to clear.

Exceptions

[DexelaException](#)

void DexelaDetector::CloseBoard ()

Closes the connection to the detector.

Exceptions

[DexelaException](#)

void DexelaDetector::DisablePulseGenerator ()

This function will disable Pulse Generator mode. After calling this you should be able to use the [SoftwareTrigger](#) call.

Exceptions

[DexelaException](#)

void DexelaDetector::EnableFrameCntWithinImage (unsigned short **us**

Enables displaying the framecounter in the image (being the 2nd pixel).

Parameters

usEnable Enables / disabled frame counter within image.

Exceptions

DexelaException

void DexelaDetector::EnablePulseGenerator (float frequency)

This function will enable the pulse generator software trigger signal. In this mode the software trigger can be continuously sent to the detector at the desired frequency.

Note: In order to use this mode the trigger source should be set to Internal_Software

Note2: To actually enable the pulse train you must call **ToggleGenerator**. A **SoftwareTrigger** call will not work when in this mode.

Parameters

frequency The frequency that the software trigger signal will be run at.

Exceptions

DexelaException

void DexelaDetector::EnablePulseGenerator ()

This function is identical to **EnablePulseGenerator**, except that the frequency of the pulse train is set automatically. The frequency will be set such as to ensure continuous image acquisition for the current detector/binning mode.

Note: In order to use this mode the trigger source should be set to Internal_Software

Note2: To actually enable the pulse train you must call **ToggleGenerator**. A **SoftwareTrigger** call will not work when in this mode.

Exceptions

DexelaException

void DexelaDetector::EnableROIMode (bool bEnableROI)

This method activates or deactivates the ROI mode of the detector.

Parameters

bEnableROI This parameter can have the following values:
value 0: Disable ROI mode
value 1: Enable ROI mode

Exceptions

[DexelaException](#)

bins DexelaDetector::GetBinningMode ()

Gets the current state of the detector binning mode

Returns

A member of the [bins](#) enumeration detailing the current detector binning mode.

Exceptions

[DexelaException](#)

int DexelaDetector::GetBufferXdim (void)

Get the x dimension of the transport buffer (in bytes)

Returns

The x dimension of the transport buffer (in bytes)

Exceptions

[DexelaException](#)

int DexelaDetector::GetBufferYdim (void)

Get the y dimension of the transport buffer (in pixels)

Returns

The y dimension of the transport buffer (in pixels)

Exceptions

[DexelsException](#)

int DexelaDetector::GetCapturedBuffer (void)

Gets the number of the buffer just captured. This can be used to determine which buffer to read-out.

Returns

The number of the last buffer, which the last captured image was written to.

Exceptions

[DexelsException](#)

DetStatus DexelaDetector::GetDetectorStatus ()

Returns the current settings of the detector in the form of a [DetStatus](#) object.

Returns

A [DetStatus](#) structure containing settings retrieved.

Exceptions

[DexelsException](#)

ExposureModes DexelaDetector::GetExposureMode ()

Gets the ExposureMode parameter of the detector.

Returns

The [ExposureModes](#) enumeration member that the detector is

currently set to.

Exceptions

[DexelaException](#)

float DexelaDetector::GetExposureTime ()

Gets the exposure time parameter of the detector (ms).

Returns

The exposure time (ms) that the detector is currently set to.

Exceptions

[DexelaException](#)

int DexelaDetector::GetFieldCount (void)

Gets the number of fields(frames) captured so far.

Returns

The number of fields captured.

Exceptions

[DexelaException](#)

void DexelaDetector::GetFirmwareBuild (int & iDayAndMonth, int & iYear, int & iTIME)

Gets the detector firmware build date.

Note: This feature may not be supported on older detectors

Parameters

iDayAndMonth The day and month of the firmware build

(DDMM format)

iYear The year of the firmware build (YYYY format).

iTime The time of the firmware build (hhmm format).

Exceptions

[DexelaException](#)

int DexelaDetector::GetFirmwareVersion ()

Gets the detector firmware version number.

Returns

The detector firmware version number as read from the detector.

Exceptions

[DexelaException](#)

FullWellModes DexelaDetector::GetFullWellMode ()

Gets the current detector well-mode.

Returns

A member of the [FullWellModes](#) enumeration detailing the current detector Full-Well mode.

Exceptions

[DexelaException](#)

float DexelaDetector::GetGapTime ()

Gets the current gap-time setting of the detector. When run in [Frame_Rate_exposure](#) mode the detector will insert this gap period between consecutive frames in an image sequence.

Note: The minimum time for the gap-time setting is the current

readout-time for the detector. Attempting to write anything smaller to the detector will result in a gap-time equal to the readout-time.

Returns

The gap-time in ms.

Exceptions

[DexelaException](#)

int DexelaDetector::GetModelNumber ()

Gets the detector model number.

Returns

The detector model number as read from the detector.

Exceptions

[DexelaException](#)

int DexelaDetector::GetNumBuffers (void)

Get the number of internal buffers that are currently allocated for the detector.

Returns

The number of internal buffers allocated

Exceptions

[DexelaException](#)

int DexelaDetector::GetNumOfExposures ()

Gets the number of exposures setting from the detector. This is only relevant in [Sequence_Exposure](#) and [Frame_Rate_exposure](#) modes of operation.

Returns

The number of exposures that will be acquired each trigger received.

Exceptions

[DexelaException](#)

ReadoutModes DexelaDetector::GetReadoutMode ()

Gets the ReadoutModes parameter of the detector.

Returns

The [ReadoutModes](#) enumeration member that the detector is currently set to.

Exceptions

[DexelaException](#)

double DexelaDetector::GetReadOutTime ()

This method will return the read-out time of the detector (in ms) for it's current binning mode.

Returns

The read-out time (in ms) for the current settings of the detector.

Exceptions

[DexelaException](#)

```
void  
DexelaDetector::GetROICoordinates ( unsigned short & usStartCo  
                                  unsigned short & usStartRo  
                                  unsigned short & usROIWid  
                                  unsigned short & usROIHeig  
                                )
```

This method retrieves the coordinates of the region of interest (ROI) set within the detector.

Parameters

- usStartColumn** Index of first column of ROI.
- usROIWidth** Height (number of columns) of the ROI.
- usStartRow** Index of first row of ROI.
- usROIHeight** Height (number of rows) of the ROI.

Exceptions

- DexelaException**

bool DexelaDetector::GetROIState ()

This method retrieves the enabled state of the region of interest of the detector.

Parameters

- bEnableROI** This parameter can have the following values:
 - value 0: ROI mode disabled
 - value 1: ROI mode enabled

Exceptions

- DexelaException**

unsigned short

DexelaDetector::GetSensorHeight (unsigned short uiSensorID = 1)

Gets the height of the sensor in pixels.

Parameters

- usSensorID** ID of sensor FPGA to send the query to. For "global reads" sensor FPGA 1 has to be queried, using usSensorID=1/param>

Exceptions

DexelaException

unsigned short

DexelaDetector::GetSensorWidth (unsigned short uiSensorID = 1)

Gets the width of the sensor in pixels.

Parameters

usSensorID ID of sensor FPGA to send the query to. For "global reads" sensor FPGA 1 has to be queried, using usSensorID=1/param>

Exceptions

DexelaException

int DexelaDetector::GetSerialNumber ()

Gets the detector serial number.

Returns

The detector serial number as read from the detector.

Exceptions

DexelaException

BOOL DexelaDetector::GetTestMode ()

Gets the current state of the detector test mode (on/off)

Returns

Boolean value containing the current state of the detector test mode.

Exceptions

DexelaException

DetectorInterface DexelaDetector::GetTransportMethod ()

Returns the communication method (i.e. interface) for the detector object.

Returns

The **DetectorInterface** enumeration used by the detector

Exceptions

DexelaException

ExposureTriggerSource DexelaDetector::GetTriggerSource ()

Gets the current trigger source setting from the detector

Returns

A member of the **ExposureTriggerSource** enumeration detailing the current trigger source of the detector.

Exceptions

DexelaException

```
void DexelaDetector::GoLiveSeq ( int start,  
                                int stop,  
                                int numBuf  
                            )
```

Sets the host computer up to be ready to receive images into the specified buffer range.

Parameters

start Number of the first buffer to use for acquisition

stop Number of the last buffer to use for acquisition

numBuf Number of frames to acquire. If this is set to 0 the buffer will be circular (i.e. ring-buffer).

Exceptions

[DexelaException](#)

void DexelaDetector::GoLiveSeq()

Sets the host computer up to be ready to receive images. This call will use all available buffers in a circular fashion (i.e. ring-buffer).

Exceptions

[DexelaException](#)

void DexelaDetector::GoUnLive()

Exits live mode. The host computer will no longer be ready to receive transmitted images.

Exceptions

[DexelaException](#)

bool DexelaDetector::IsCallbackActive()

This method will inform the user if the callback mode (i.e. background thread) is currently active.

Returns

A boolean value indicating whether the callback mode is active.

Exceptions

[DexelaException](#)

bool DexelaDetector::IsConnected()

Check to see if the connection to the detector is open (i.e. [OpenBoard](#))

Returns

A boolean indicating whether the connection to detector is open.

Exceptions

[DexelaException](#)

bool DexelaDetector::IsFrameCntWithinImage ()

Checks if framecounter is displayed within the image data (being the 2nd pixel).

Parameters

usEnable 0 if not enabled; 1 if enabled

Exceptions

[DexelaException](#)

bool DexelaDetector::IsLive ()

This method will inform the user if detector is currently in Live mode.

Returns

A boolean value indicating whether the detector is in Live mode.

Exceptions

[DexelaException](#)

void DexelaDetector::LoadSensorConfigFile (char * filename)

Loads the sensor configuration file into the detector. This file will write values to the ADC offset registers for each sensor in the detector.

Parameters

filename The path to the sensor configuration to use.

Exceptions

[DexelaException](#)

void DexelaDetector::OpenBoard ()

virtual

Opens the connection to the detector. Every open should be matched with a close to free resources.

Exceptions

[DexelaException](#)

Reimplemented in [DexelaDetectorCL](#), and [DexelaDetectorGE](#).

void DexelaDetector::OpenBoard (int NumBufs)

Opens the connection to the detector and sets the number of buffers to use/allocate. Every open should be matched with a close to free resources.

Parameters

NumBufs Number of buffers to use/allocate

Exceptions

[DexelaException](#)

int DexelaDetector::QueryBinningMode (bins flag)

Query the detector to see if the desired binning mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may

still be possible to use the feature in this case but no guarantees can be made on whether or not it will work.

Parameters

mode The [bins](#) enumeration member to be checked for.

Returns

An integer value representing whether the feature is present. A value of 1 indicates that the feature is present. A value of 0 indicates that the feature is not present. A value of -1 indicates that it is unknown whether the feature is present.

Exceptions

[DexelaException](#)

int

DexelaDetector::QueryExposureMode (ExposureModes mode)

Query the detector to see if the desired exposure mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work.

Parameters

mode The [ExposureModes](#) enumeration member to be checked for.

Returns

An integer value representing whether the feature is present. A value of 1 indicates that the feature is present. A value of 0 indicates that the feature is not present. A value of -1 indicates that it is unknown whether the feature is present.

Exceptions

DexelaException

`int DexelaDetector::QueryFullWellMode (FullWellModes fwm)`

Query the detector to see if the desired full-well mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work.

Parameters

mode The `FullWellModes` enumeration member to be checked for.

Returns

An integer value representing whether the feature is present. A value of 1 indicates that the feature is present. A value of 0 indicates that the feature is not present. A value of -1 indicates that it is unknown whether the feature is present.

Exceptions

`DexelaException`

`int DexelaDetector::QueryReadoutMode (ReadoutModes mode)`

Query the detector to see if the desired readout mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work.

Parameters

mode The [ReadoutModes](#) enumeration member to be checked for.

Returns

An integer value representing whether the feature is present. A value of 1 indicates that the feature is present. A value of 0 indicates that the feature is not present. A value of -1 indicates that it is unknown whether the feature is present.

Exceptions

[DexelaException](#)

int

DexelaDetector::QueryTriggerSource ([ExposureTriggerSource](#) ets

Query the detector to see if the desired trigger source is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work.

Parameters

mode The [ExposureTriggerSource](#) enumeration member to be checked for.

Returns

An integer value representing whether the feature is present. A value of 1 indicates that the feature is present. A value of 0 indicates that the feature is not present. A value of -1 indicates that it is unknown whether the feature is present.

Exceptions

[DexelaException](#)

```
void DexelaDetector::ReadBuffer ( int bufNum,
                                byte * buffer
                                )
```

Reads the specified transport buffer into the passed in buffer (byte*).
Note: [GetCapturedBuffer](#) can be used to get the number of the lastest buffer to be filled.

Parameters

bufNum The index of the transport buffer to read from.
buffer The user-created (byte*) buffer to write the image to.

Exceptions

[DexelaException](#)

```
void DexelaDetector::ReadBuffer ( int bufNum,
                                DexImage & img,
                                int iZ = 0
                                )
```

Reads the specified transport buffer into the passed in [DexImage](#) object at the passed in plane.

Note: [GetCapturedBuffer](#) can be used to get the number of the lastest buffer to be filled.

Parameters

bufNum The index of the transport buffer to read from.
img The [DexImage](#) object that the image will be written into
iZ The plane that the image should be written to in the [DexImage](#) object (defaults to 0).

Exceptions

[DexelaException](#)

```
int DexelaDetector::ReadRegister ( int address,  
                                  int sensorNum = 1  
                            )
```

Reads the specified register from the detector. The sensor number corresponds to the desired sensor from which to read the register. The SensorNumber will default to 1 (master-sensor) if not specified otherwise by the user.

Parameters

address The address of the desired register to read.
sensorNum The sensor number from which to read the register (defaults to master-sensor).

Returns

The integer value of the register.

Exceptions

[DexelaException](#)

```
void DexelaDetector::SetBinningMode ( bins flag )
```

Sets the binning mode of the detector.

Parameters

flag The [bins](#) enumeration member to be set.

Exceptions

[DexelaException](#)

```
void DexelaDetector::SetCallback ( IMAGE_CALLBACK func )
```

Sets the user defined callback function to be called for every image arrival event.

Parameters

func The call back function (**IMAGE_CALLBACK**) to be called for every image arrival event.

Exceptions

DexelaException

void DexelaDetector::SetExposureMode (ExposureModes mode)

Sets the ExposureMode parameter of the detector

Parameters

mode The **ExposureModes** enumeration member to be set.

Exceptions

DexelaException

void DexelaDetector::SetExposureTime (float timems)

Sets the exposure time parameter of the detector

Parameters

timems The exposure time (in milliseconds) to be set. **Note:** if you attempt to set an exposure time smaller then the current read-out time for the detector, it will be set to minimum.

Exceptions

DexelaException

void DexelaDetector::SetFullWellMode (FullWellModes fwm)

Sets the full well mode parameter of the detector.

Parameters

fwm The [FullWellModes](#) enumeration member to be set.

Exceptions

[DexelaException](#)

void DexelaDetector::SetGapTime (float timems)

Sets the gap-time setting of the detector. When run in [Frame_Rate_exposure](#) mode the detector will insert this gap period between consecutive frames in an image sequence.

Note: The minimum time for the gap-time setting is the current readout-time for the detector. Attempting to write anything smaller to the detector will result in a gap-time equal to the readout-time.

Parameters

timems The gap-time in ms.

Exceptions

[DexelaException](#)

void DexelaDetector::SetNumberOfExposures (int num)

Sets the number of exposures to acquire after a trigger. This is only relevant in [Sequence_Exposure](#) and [Frame_Rate_exposure](#) modes of operation.

Parameters

num The number of exposues to acquire for each trigger received.

Exceptions

[DexelaException](#)

void DexelaDetector::SetPreProgrammedExposureTimes (int numE

```
float * expos  
)
```

This method will set the exposure times for pre-programmed exposure

Parameters

numExposures	The number of exposures to set. This number correspond to the number of exposures that will be acquired in pre-programmed exposure mode. It must be between 2-4.
exposuretimes_ms	An array of floating point numbers representing exposure times for pre-programmed exposure. The number of exposure times should correspond to the numExposures parameter.

Exceptions

[DexelsException](#)

void DexelaDetector::SetReadoutMode ([ReadoutModes mode](#))

Sets the ReadoutMode parameter of the detector

Parameters

mode The [ReadoutModes](#) enumeration member to be set.

Exceptions

[DexelsException](#)

void DexelaDetector::SetSlowed (bool flag)

This method can specify to the API that the detector being used is a slowed-down detector (e.g. mammo detector). This should not be necessary as the API should be able to determine most of the time whether the firmware version is a slowed down one. However, for certain older detectors/firmwares this may not be possible. In this case this method can be used to inform the library that the firmware

is slowed down and it will use the correct read-out times.

Parameters

numExposures A boolean flag indicating whether the detector is slowed down or not.

Exceptions

DexelaException

void DexelaDetector::SetTestMode (BOOL SetTestOn)

Enables/disables test mode. The detector will output a generated test pattern if this mode is turned on.

Parameters

SetTestOn if set to true The test pattern is turned on.

Exceptions

DexelaException

void DexelaDetector::SetTriggerSource (ExposureTriggerSource ets)

Sets the trigger source setting on the detector

Parameters

ets A member of the **ExposureTriggerSource** enumeration to be set to the detector

Exceptions

DexelaException

void DexelaDetector::Snap (int buffer, int timeout)

Snaps an image into the specified buffer.

Note: If the detector trigger source is set to [Internal_Software](#), this call will automatically trigger the detector.

Parameters

- buffer** The buffer number to snap to. The number of available buffers can be found by calling the [GetNumBuffers](#) method.
- timeout** The amount of time (in ms) that the library will wait for an image before throwing a timeout exception.

Exceptions

[DexelaException](#)

void DexelaDetector::SoftReset (void)

Cycles the power on the detector

Exceptions

[DexelaException](#)

void DexelaDetector::SoftwareTrigger ()

Sends a trigger to the detector (will only work if the trigger source is set to [Internal_Software](#))

Exceptions

[DexelaException](#)

void DexelaDetector::StopCallback ()

This function will terminate the callback loop (i.e. will wait for all spawned threads to finish executing).

Exceptions

[DexelaException](#)

void DexelaDetector::ToggleGenerator (BOOL onOff)

This function will control the pulse train.

Note: In order to use this mode the pulse generator must be enabled. See [EnablePulseGenerator](#).

Parameters

onOff Boolean that will control the state of the pulse train (true = on, false = off).

Exceptions

[DexelaException](#)

void DexelaDetector::WaitImage (int timeout)

This function will wait for the specified amount of time for an image to arrive.

Note: If the image arrives before then it will return as soon as it does (i.e. it won't wait for the duration of the timeout period). If the image does not arrive in the specified time a [DexelaException](#) will be thrown.

Parameters

timeout The timeout period (in ms) for which to wait before throwing a [DexelaException](#).

Exceptions

[DexelaException](#)

void DexelaDetector::WriteBuffer (int bufNum, byte * buffer)

Writes data to the specified transport buffer.

Parameters

- bufNum** The index of the transport buffer to write to.
- buffer** The user-created data (byte*) buffer to write to the transport buffer.

Exceptions

[DexelaException](#)

```
void DexelaDetector::WriteRegister ( int address,
                                    int value,
                                    int sensorNum = 0
)
```

Writes the value to the specified register from the detector. The sensor number corresponds to the desired sensor to which the value will be written. The SensorNumber will default to 0 (broadcast to all sensors) if not specified otherwise by the user.

Parameters

- address** The address of the desired register to write to.

+

Parameters

- value** The value to write into the register
- sensorNum** The sensor number that the register will be written to. This defaults to 0 which is a broadcast to all detector sensors.

Exceptions

[DexelaException](#)

The documentation for this class was generated from the following

files:

- **DexelsDetector.h**
- DexelaDetector.cpp

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members
Public Member Functions List of all members			

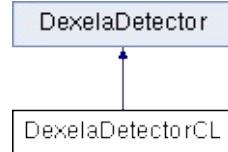
DexelsDetectorCL Class Reference

This class is used to control CameraLink Type Detectors. It will give access to functions that are not available to other interface-type detectors.

Note: For all standard detector function calls please see the [DexelsDetector](#) class (these functions are also available to [DexelsDetectorCL](#) objects) [More...](#)

```
#include <DexelsDetectorCL.h>
```

Inheritance diagram for DexelaDetectorCL:



Public Member Functions

DexelaDetectorCL (DetectorInterface

transport, int unit, const char *params)

Constructor for **DexelaDetectorCL**. This version assumes you know the interface and the correct parameters to connect to the detector. [More...](#)

DexelaDetectorCL (DevInfo &devInfo)

Constructor for **DexelaDetectorCL**.

Identical to the **DexelaDetector constructor**, except with an additional check for the correct (CameraLink) interface.

Note: A

Exceptions

is thrown if the interface of the **DexelaException DevInfo** object is not correct (i.e. CL)

[More...](#)

virtual ~DexelaDetectorCL (void)

Destructor for **DexelaDetectorCL**. [More...](#)

void PowerCLInterface (bool flag)

Function to turn the CameraLink interface on and off [More...](#)

void OpenBoard ()

Identical to the **OpenBoard** call. The only difference is a check to make sure the detector has the correct (CL) interface.

[More...](#)

void **OpenBoard** (int NumBufs)

Identical to the **OpenBoard** call. The only difference is a check to make sure the detector has the correct (CL) interface.

[More...](#)

▶ **Public Member Functions inherited from DexelaDetector**

DexelaDetector (DevInfo &devInfo)

Constructor for **DexelaDetector**. This version uses the **DevInfo** struct returned from a **GetDevice**, **GetDeviceGE** or **GetDeviceCL** call. [More...](#)

DexelaDetector (DetectorInterface

transport, int unit, const char *params)

Constructor for **DexelaDetector**. This version assumes you know the interface and the correct parameters to connect to the detector. [More...](#)

virtual ~**DexelaDetector** (void)

Destructor for **DexelaDetector**. [More...](#)

void **OpenBoard** (int NumBufs)

Opens the connection to the detector and sets the number of buffers to use/allocate. Every open should be matched with a close to free resources. [More...](#)

void **CloseBoard** ()

Closes the connection to the detector.

[More...](#)

int **GetBufferXdim** (void)

Get the x dimension of the transport buffer (in bytes) [More...](#)

int [GetBufferYdim](#) (void)

Get the y dimension of the transport buffer (in pixels) [More...](#)

int [GetNumBuffers](#) (void)

Get the number of internal buffers that are currently allocated for the detector.

[More...](#)

int [GetCapturedBuffer](#) (void)

Gets the number of the buffer just captured. This can be used to determine which buffer to read-out. [More...](#)

int [GetFieldCount](#) (void)

Gets the number of fields(frames) captured so far. [More...](#)

void [ReadBuffer](#) (int bufNum, byte *buffer)

Reads the specified transport buffer into the passed in buffer (byte*).

Note: [GetCapturedBuffer](#) can be used to get the number of the lastest buffer to be filled. [More...](#)

void [ReadBuffer](#) (int bufNum, DexImage &img, int iZ=0)

Reads the specified transport buffer into the passed in **DexImage** object at the passed in plane.

Note: [GetCapturedBuffer](#) can be used to get the number of the lastest buffer to be filled. [More...](#)

void [WriteBuffer](#) (int bufNum, byte *buffer)

Writes data to the specified transport

buffer. More...

void **SetFullWellMode** (**FullWellModes** fwm)
Sets the full well mode parameter of the detector. [More...](#)

void **SetExposureMode** (**ExposureModes** mode)
Sets the ExposureMode parameter of the detector [More...](#)

void **SetExposureTime** (float timems)
Sets the exposure time parameter of the detector [More...](#)

void **SetBinningMode** (**bins** flag)
Sets the binning mode of the detector.
[More...](#)

void **SetTestMode** (BOOL SetTestOn)
Enables/disables test mode. The detector will output a generated test pattern if this mode is turned on. [More...](#)

void **SetTriggerSource**
(**ExposureTriggerSource** ets)
Sets the trigger source setting on the detector [More...](#)

void **SetNumberOfExposures** (int num)
Sets the number of exposures to acquire after a trigger. This is only relevant in **Sequence_Exposure** and **Frame_Rate_exposure** modes of operation. [More...](#)

int **GetNumberOfExposures** ()
Gets the number of exposures setting

from the detector. This is only relevant in **Sequence_Exposure** and **Frame_Rate_exposure** modes of operation. [More...](#)

void SetGapTime (float timems)

Sets the gap-time setting of the detector. When run in **Frame_Rate_exposure** mode the detector will insert this gap period between consecutive frames in an image sequence.

Note: The minimum time for the gap-time setting is the current readout-time for the detector. Attempting to write anything smaller to the detector will result in a gap-time equal to the readout-time. [More...](#)

float GetGapTime ()

Gets the current gap-time setting of the detector. When run in **Frame_Rate_exposure** mode the detector will insert this gap period between consecutive frames in an image sequence.

Note: The minimum time for the gap-time setting is the current readout-time for the detector. Attempting to write anything smaller to the detector will result in a gap-time equal to the readout-time. [More...](#)

bool IsConnected ()

Check to see if the connection to the detector is open (i.e. **OpenBoard**) [More...](#)

ExposureModes GetExposureMode ()

Gets the ExposureMode parameter of the detector. [More...](#)

float **GetExposureTime ()**

Gets the exposure time parameter of the detector (ms). [More...](#)

DetStatus GetDetectorStatus ()

Returns the current settings of the detector in the form of a **DetStatus** object. [More...](#)

ExposureTriggerSource GetTriggerSource ()

Gets the current trigger source setting from the detector [More...](#)

BOOL **GetTestMode ()**

Gets the current state of the detector test mode (on/off) [More...](#)

FullWellModes GetFullWellMode ()

Gets the current detector well-mode. [More...](#)

bins GetBinningMode ()

Gets the current state of the detector binning mode [More...](#)

int **GetSerialNumber ()**

Gets the detector serial number. [More...](#)

int **GetModelNumber ()**

Gets the detector model number. [More...](#)

int **GetFirmwareVersion ()**

Gets the detector firmware version number. [More...](#)

void **GetFirmwareBuild (int &iDayAndMonth,**
int &iYear, int &iTime)

Gets the detector firmware build date.
Note: This feature may not be supported on older detectors [More...](#)

DetectorInterface **GetTransportMethod ()**
Returns the communication method (i.e. interface) for the detector object. [More...](#)

double **GetReadOutTime ()**
This method will return the read-out time of the detector (in ms) for it's current binning mode. [More...](#)

bool **IsCallbackActive ()**
This method will inform the user if the callback mode (i.e. background thread) is currently active. [More...](#)

bool **IsLive ()**
This method will inform the user if detector is currently in Live mode. [More...](#)

void **Snap (int buffer, int timeout)**
Snares an image into the specified buffer.
Note: If the detector trigger source is set to **Internal_Software**, this call will automatically trigger the detector. [More...](#)

int **ReadRegister (int address, int sensorNum=1)**
Reads the specified register from the detector. The sensor number corresponds to the desired sensor from which to read the register. The SensorNumber will default to 1 (master-sensor) if not specified otherwise by the user. [More...](#)

void WriteRegister (int address, int value, int sensorNum=0)

Writes the value to the specified register from the detector. The sensor number corresponds to the desired sensor to which the value will be written. The SensorNumber will default to 0 (broadcast to all sensors) if not specified otherwise by the user. [More...](#)

void ClearCameraBuffer (int i)

Clears (i.e. zero-out) the specified camera buffer. [More...](#)

void ClearBuffers ()

Clears all the camera buffers [More...](#)

void LoadSensorConfigFile (char *filename)

Loads the sensor configuration file into the detector. This file will write values to the ADC offset registers for each sensor in the detector. [More...](#)

void SoftReset (void)

Cycles the power on the detector [More...](#)

void GoLiveSeq (int start, int stop, int numBuf)

Sets the host computer up to be ready to receive images into the specified buffer range. [More...](#)

void GoLiveSeq ()

Sets the host computer up to be ready to receive images. This call will use all available buffers in a circular fashion (i.e. ring-buffer). [More...](#)

void GoUnLive ()

Exits live mode. The host computer will no longer be ready to receive transmitted images. [More...](#)

void [SoftwareTrigger \(\)](#)

Sends a trigger to the detector (will only work if the trigger source is set to **Internal_Software**) [More...](#)

void [EnablePulseGenerator \(float frequency\)](#)

This function will enable the pulse generator software trigger signal. In this mode the software trigger can be continuously sent to the detector at the desired frequency.

Note: In order to use this mode the trigger source should be set to Internal_Software

Note2: To actually enable the pulse train you must call **ToggleGenerator**. A **SoftwareTrigger** call will not work when in this mode. [More...](#)

void [EnablePulseGenerator \(\)](#)

This function is identical to **EnablePulseGenerator**, except that the frequency of the pulse train is set automatically. The frequency will be set such as to ensure continuous image acquisition for the current detector/binning mode.

Note: In order to use this mode the trigger source should be set to Internal_Software

Note2: To actually enable the pulse train you must call **ToggleGenerator**. A **SoftwareTrigger** call will not work when in this mode. [More...](#)

void DisablePulseGenerator ()

This function will disable Pulse Generator mode. After calling this you should be able to use the **SoftwareTrigger** call.

[More...](#)

void ToggleGenerator (BOOL onOff)

This function will control the pulse train.

Note: In order to use this mode the pulse generator must be enabled. See **EnablePulseGenerator**. [More...](#)

void WaitImage (int timeout)

This function will wait for the specified amount of time for an image to arrive.

Note: If the image arrives before then it will return as soon as it does (i.e. it won't wait for the duration of the timeout period). If the image does not arrive in the specified time a **DexelaException** will be thrown. [More...](#)

void SetCallback (IMAGE_CALLBACK func)

Sets the user defined callback function to be called for every image arrival event.

[More...](#)

void StopCallback ()

This function will terminate the callback loop (i.e. will wait for all spawned threads to finish executing). [More...](#)

void CheckForCallbackError ()

This function will check to see if any errors have occurred in the background thread that is running when using

callbacks. This thread is activated after a call to **SetCallback** and terminated with a call to **StopCallback**. If no error has occurred this method will just return. if an error has occurred a **DexelaException** will be thrown. [More...](#)

void **CheckForLiveError ()**

This function will check to see if any errors have occurred in the background thread that is running when using live-mode. This thread is activated after a call to **GoLiveSeq** and terminated with a call to **GoUnLive**. If no error has occurred this method will just return. if an error has occurred a **DexelaException** will be thrown. [More...](#)

void **SetPreProgrammedExposureTimes (int numExposures, float *exposuretimes_ms)**

This method will set the exposure times for pre-programmed exposure mode.
[More...](#)

void **SetROICoordinates (unsigned short usStartColumn, unsigned short usStartRow, unsigned short usROIWidth, unsigned short usROIHeight)**

This method set the coordinates of the ROI when detector runs in ROI mode.

Parameters

usStartColumn Sensor column to start the ROI read out from.

usStartRow Sensor width to start the ROI read out from.

usROIWidth Width (number of

columns) of the ROI.

usROIHeight Height (number of columns) of the ROI.

Exceptions

DexelaException

GetROICoordinates (unsigned short &usStartColumn, unsigned short &usStartRow, unsigned short &usROIWidth, unsigned short &usROIHeight)
This method retrieves the coordinates of the region of interest (ROI) set within the detector. [More...](#)

EnableROIMode (bool bEnableROI)
This method activates or deactivates the ROI mode of the detector. [More...](#)

GetROIState ()
This method retrieves the enabled state of the region of interest of the detector.
[More...](#)

GetSensorHeight (unsigned short uiSensorID=1)
Gets the height of the sensor in pixels.
[More...](#)

GetSensorWidth (unsigned short uiSensorID=1)
Gets the width of the sensor in pixels.
[More...](#)

bool IsFrameCntWithinImage ()
Checks if framecounter is displayed within the image data (being the 2nd pixel).
[More...](#)

void EnableFrameCntWithinImage (unsigned short usEnable)
Enables displaying the framecounter in the image (being the 2nd pixel). [More...](#)

void SetSlowed (bool flag)
This method can specify to the api that the detector being used is a slowed-down detector (e.g. mammo detector). This should not be necessary as the API should be able to determine most of the time whether the firmwarwe version is a slowed down one. However, for certain older detectors/firmwares this may not be possible. In this case this method can be used to inform the library that the firmware is slowed down and it will use the correct read-out times. [More...](#)

void SetReadoutMode (ReadoutModes** mode)**
Sets the ReadoutMode parameter of the detector [More...](#)

ReadoutModes GetReadoutMode ()
Gets the ReadoutModes parameter of the detector. [More...](#)

int QueryReadoutMode (ReadoutModes** mode)**
Query the detector to see if the desired readout mode is present (i.e. available).
Note: Older detectors may not support

the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

int **QueryExposureMode** ([ExposureModes](#) mode)

Query the detector to see if the desired exposure mode is present (i.e. available). **Note:** Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

int **QueryTriggerSource** ([ExposureTriggerSource](#) ets)

Query the detector to see if the desired trigger source is present (i.e. available). **Note:** Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

int **QueryFullWellMode (FullWellModes fwm)**

Query the detector to see if the desired full-well mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

int **QueryBinningMode (bins flag)**

Query the detector to see if the desired binning mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

Additional Inherited Members

▶ Protected Attributes inherited from DexelaDetector

boost::shared_ptr< baseDetector > **base**

boost::shared_ptr< gigEDetector > **gigeDet**

boost::shared_ptr
< camLinkDetector > **cIDet**

Detailed Description

This class is used to control CameraLink Type Detectors. It will give access to functions that are not available to other interface-type detectors.

Note: For all standard detector function calls please see the [DexelaDetector](#) class (these functions are also available to [DexelaDetectorCL](#) objects)

Constructor & Destructor Documentation

```
DexelaDetectorCL::DexelaDetectorCL ( DetectorInterface transpor  
                                      int               unit,  
                                      const char *      params  
                                    )
```

Constructor for **DexelaDetectorCL**. This version assumes you know the interface and the correct parameters to connect to the detector.

Parameters

- transport** The **DetectorInterface** for the detector (i.e. CL)
- unit** The unit number for CL type detectors.
- params** The parameter string for connection to the detector. For CL detectors this parameter will be ignored.

Exceptions

DexelaException

DexelaDetectorCL::DexelaDetectorCL (DevInfo & devInfo)

Constructor for **DexelaDetectorCL**. Identical to the **DexelaDetector constructor**, except with an additional check for the correct (CameraLink) interface.

Note: A

Exceptions

DexelaException is thrown if the interface of the **DevInfo** object is not correct (i.e. CL)

Parameters

- devInfo** The **DevInfo** object for the desired detector. This can be obtained from the **GetDeviceCL** method

Exceptions

DexelsException

DexelsDetectorCL::~DexelsDetectorCL (void)

virtual

Destructor for **DexelsDetectorCL**.

Member Function Documentation

void DexelaDetectorCL::OpenBoard()

virtual

Identical to the [OpenBoard](#) call. The only difference is a check to make sure the detector has the correct (CL) interface.

Exceptions

[DexelaException](#)

Reimplemented from [DexelaDetector](#).

void DexelaDetectorCL::OpenBoard(int NumBufs)

Identical to the [OpenBoard](#) call. The only difference is a check to make sure the detector has the correct (CL) interface.

Parameters

NumBufs Number of buffers to use/allocate

Exceptions

[DexelaException](#)

void DexelaDetectorCL::PowerCLInterface(bool flag)

Function to turn the CameraLink interface on and off

Parameters

flag If flag is true the interface will be turned on otherwise it will be turned off

Exceptions

DexelaException

The documentation for this class was generated from the following files:

- [DexelaDetectorCL.h](#)
 - DexelaDetectorCL.cpp
-

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by [doxygen](#) 1.8.7

DexelaDetector API

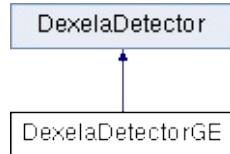
Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members
Public Member Functions List of all members			
DexelaDetectorGE			Class Reference

This class is used to control GigE Type Detectors. It will give access to functions that are not available to other interface-type detectors.

Note: For all standard detector function calls please see the [DexelaDetector](#) class (these functions are also available to [DexelaDetectorGE](#) objects) [More...](#)

```
#include <DexelaDetectorGE.h>
```

Inheritance diagram for DexelaDetectorGE:



Public Member Functions

DexelaDetectorGE (DevInfo &devInfo)

Constructor for **DexelaDetectorGE**.

Identical to the **DexelaDetector**

constructor, except with an additional check for the correct (GIGE) interface.

Note: A

Exceptions

is thrown if the interface of the **DexelaException DevInfo** object is not correct (i.e. GIGE)

[More...](#)

DexelaDetectorGE (DetectorInterface

transport, int unit, const char *params)

Constructor for **DexelaDetectorGE**. This version assumes you know the interface and the correct parameters to connect to the detector. [More...](#)

virtual **~DexelaDetectorGE (void)**

Destructor for **DexelaDetectorGE**.

[More...](#)

void **SetPersistentIPAddress (int firstByte, int secondByte, int thirdByte, int fourthByte)**

Function for setting a new persistent IP address for the detector. After power-cycling the detector it should come up with the desired IP address [More...](#)

void **OpenBoard ()**

Identical to the **OpenBoard** call. The only difference is a check to make sure the detector has the correct (GIGE) interface.
[More...](#)

void OpenBoard (int NumBufs)

Identical to the **OpenBoard** call. The only difference is a check to make sure the detector has the correct (GIGE) interface.
[More...](#)

► Public Member Functions inherited from **DexelaDetector**

DexelaDetector (DevInfo &devInfo)
Constructor for **DexelaDetector**. This version uses the **DevInfo** struct returned from a **GetDevice**, **GetDeviceGE** or **GetDeviceCL** call.
[More...](#)

DexelaDetector (DetectorInterface transport, int unit, const char *params)
Constructor for **DexelaDetector**. This version assumes you know the interface and the correct parameters to connect to the detector.
[More...](#)

virtual ~DexelaDetector (void)

Destructor for **DexelaDetector**.
[More...](#)

void OpenBoard (int NumBufs)

Opens the connection to the detector and sets the number of buffers to use/allocate. Every open should be matched with a close to free resources.
[More...](#)

void CloseBoard ()

Closes the connection to the detector.
[More...](#)

- int [GetBufferXdim](#) (void)**
Get the x dimension of the transport buffer (in bytes) [More...](#)
- int [GetBufferYdim](#) (void)**
Get the y dimension of the transport buffer (in pixels) [More...](#)
- int [GetNumBuffers](#) (void)**
Get the number of internal buffers that are currently allocated for the detector.
[More...](#)
- int [GetCapturedBuffer](#) (void)**
Gets the number of the buffer just captured. This can be used to determine which buffer to read-out. [More...](#)
- int [GetFieldCount](#) (void)**
Gets the number of fields(frames) captured so far. [More...](#)
- void [ReadBuffer](#) (int bufNum, byte *buffer)**
Reads the specified transport buffer into the passed in buffer (byte*).
Note: [GetCapturedBuffer](#) can be used to get the number of the lastest buffer to be filled. [More...](#)
- void [ReadBuffer](#) (int bufNum, DexImage &img, int iZ=0)**
Reads the specified transport buffer into the passed in **DexImage** object at the passed in plane.
Note: [GetCapturedBuffer](#) can be used to get the number of the lastest buffer to be filled. [More...](#)

void **WriteBuffer** (int bufNum, byte *buffer)
Writes data to the specified transport buffer. [More...](#)

void **SetFullWellMode** (**FullWellModes** fwm)
Sets the full well mode parameter of the detector. [More...](#)

void **SetExposureMode** (**ExposureModes** mode)
Sets the ExposureMode parameter of the detector [More...](#)

void **SetExposureTime** (float timems)
Sets the exposure time parameter of the detector [More...](#)

void **SetBinningMode** (**bins** flag)
Sets the binning mode of the detector.
[More...](#)

void **SetTestMode** (BOOL SetTestOn)
Enables/disables test mode. The detector will output a generated test pattern if this mode is turned on. [More...](#)

void **SetTriggerSource** (**ExposureTriggerSource** ets)
Sets the trigger source setting on the detector [More...](#)

void **SetNumberOfExposures** (int num)
Sets the number of exposures to acquire after a trigger. This is only relevant in **Sequence_Exposure** and **Frame_Rate_exposure** modes of operation. [More...](#)

int [GetNumOfExposures \(\)](#)

Gets the number of exposures setting from the detector. This is only relevant in **Sequence_Exposure** and **Frame_Rate_exposure** modes of operation. [More...](#)

void [SetGapTime \(float timems\)](#)

Sets the gap-time setting of the detector. When run in **Frame_Rate_exposure** mode the detector will insert this gap period between consecutive frames in an image sequence.

Note: The minimum time for the gap-time setting is the current readout-time for the detector. Attempting to write anything smaller to the detector will result in a gap-time equal to the readout-time. [More...](#)

float [GetGapTime \(\)](#)

Gets the current gap-time setting of the detector. When run in **Frame_Rate_exposure** mode the detector will insert this gap period between consecutive frames in an image sequence.

Note: The minimum time for the gap-time setting is the current readout-time for the detector. Attempting to write anything smaller to the detector will result in a gap-time equal to the readout-time. [More...](#)

bool [IsConnected \(\)](#)

Check to see if the connection to the detector is open (i.e. **OpenBoard**) [More...](#)

ExposureModes **GetExposureMode ()**
Gets the ExposureMode parameter of the detector. [More...](#)

float **GetExposureTime ()**
Gets the exposure time parameter of the detector (ms). [More...](#)

DetStatus **GetDetectorStatus ()**
Returns the current settings of the detector in the form of a **DetStatus** object. [More...](#)

ExposureTriggerSource **GetTriggerSource ()**
Gets the current trigger source setting from the detector [More...](#)

BOOL **GetTestMode ()**
Gets the current state of the detector test mode (on/off) [More...](#)

FullWellModes **GetFullWellMode ()**
Gets the current detector well-mode. [More...](#)

bins **GetBinningMode ()**
Gets the current state of the detector binning mode [More...](#)

int **GetSerialNumber ()**
Gets the detector serial number. [More...](#)

int **GetModelNumber ()**
Gets the detector model number. [More...](#)

int **GetFirmwareVersion ()**
Gets the detector firmware version

number. [More...](#)

**void GetFirmwareBuild (int &iDayAndMonth,
int &iYear, int &iTime)**

Gets the detector firmware build date.

Note: This feature may not be supported
on older detectors [More...](#)

DetectorInterface GetTransportMethod ()

Returns the communication method (i.e.
interface) for the detector object. [More...](#)

double GetReadOutTime ()

This method will return the read-out time
of the detector (in ms) for it's current
binning mode. [More...](#)

bool IsCallbackActive ()

This method will inform the user if the
callback mode (i.e. background thread) is
currently active. [More...](#)

bool IsLive ()

This method will inform the user if
detector is currently in Live mode. [More...](#)

void Snap (int buffer, int timeout)

Snaps an image into the specified buffer.

Note: If the detector trigger source is set
to **Internal_Software**, this call will
automatically trigger the detector. [More...](#)

**int ReadRegister (int address, int
sensorNum=1)**

Reads the specified register from the
detector. The sensor number corresponds
to the desired sensor from which to read
the register. The SensorNumber will

default to 1 (master-sensor) if not specified otherwise by the user. [More...](#)

void WriteRegister (int address, int value, int sensorNum=0)
Writes the value to the specified register from the detector. The sensor number corresponds to the desired sensor to which the value will be written. The SensorNumber will default to 0 (broadcast to all sensors) if not specified otherwise by the user. [More...](#)

void ClearCameraBuffer (int i)
Clears (i.e. zero-out) the specified camera buffer. [More...](#)

void ClearBuffers ()
Clears all the camera buffers [More...](#)

void LoadSensorConfigFile (char *filename)
Loads the sensor configuration file into the detector. This file will write values to the ADC offset registers for each sensor in the detector. [More...](#)

void SoftReset (void)
Cycles the power on the detector [More...](#)

void GoLiveSeq (int start, int stop, int numBuf)
Sets the host computer up to be ready to receive images into the specified buffer range. [More...](#)

void GoLiveSeq ()
Sets the host computer up to be ready to receive images. This call will use all

available buffers in a circular fashion (i.e. ring-buffer). [More...](#)

void [GoUnLive \(\)](#)

Exits live mode. The host computer will no longer be ready to receive transmitted images. [More...](#)

void [SoftwareTrigger \(\)](#)

Sends a trigger to the detector (will only work if the trigger source is set to **Internal_Software**) [More...](#)

void [EnablePulseGenerator \(float frequency\)](#)

This function will enable the pulse generator software trigger signal. In this mode the software trigger can be continuously sent to the detector at the desired frequency.

Note: In order to use this mode the trigger source should be set to Internal_Software

Note2: To actually enable the pulse train you must call **ToggleGenerator**. A **SoftwareTrigger** call will not work when in this mode. [More...](#)

void [EnablePulseGenerator \(\)](#)

This function is identical to **EnablePulseGenerator**, except that the frequency of the pulse train is set automatically. The frequency will be set such as to ensure continuous image acquisition for the current detector/binning mode.

Note: In order to use this mode the trigger source should be set to Internal_Software

Note2: To actually enable the pulse train you must call **ToggleGenerator**. A **SoftwareTrigger** call will not work when in this mode. [More...](#)

void DisablePulseGenerator ()

This function will disable Pulse Generator mode. After calling this you should be able to use the **SoftwareTrigger** call. [More...](#)

void ToggleGenerator (BOOL onOff)

This function will control the pulse train. **Note:** In order to use this mode the pulse generator must be enabled. See **EnablePulseGenerator**. [More...](#)

void WaitImage (int timeout)

This function will wait for the specified amount of time for an image to arrive. **Note:** If the image arrives before then it will return as soon as it does (i.e. it won't wait for the duration of the timeout period). If the image does not arrive in the specified time a **DexelaException** will be thrown. [More...](#)

void SetCallback (IMAGE_CALLBACK func)

Sets the user defined callback function to be called for every image arrival event.

[More...](#)

void StopCallback ()

This function will terminate the callback loop (i.e. will wait for all spawned threads to finish executing). [More...](#)

void CheckForCallbackError ()

This function will check to see if any errors have occurred in the background thread that is running when using callbacks. This thread is activated after a call to **SetCallback** and terminated with a call to **StopCallback**. If no error has occurred this method will just return. if an error has occurred a **DexelaException** will be thrown. [More...](#)

void **CheckForLiveError ()**

This function will check to see if any errors have occurred in the background thread that is running when using live-mode. This thread is activated after a call to **GoLiveSeq** and terminated with a call to **GoUnLive**. If no error has occurred this method will just return. if an error has occurred a **DexelaException** will be thrown. [More...](#)

void **SetPreProgrammedExposureTimes (int numExposures, float *exposuretimes_ms)**

This method will set the exposure times for pre-programmed exposure mode.
[More...](#)

void **SetROICoordinates (unsigned short usStartColumn, unsigned short usStartRow, unsigned short usROIWidth, unsigned short usROIHeight)**

This method set the coordinates of the ROI when detector runs in ROI mode.

Parameters

usStartColumn Sensor column to start the ROI read out from.

usStartRow Sensor width to

start the ROI read out from.

usROIWidth Width (number of columns) of the ROI.

usROIHeight Height (number of columns) of the ROI.

Exceptions

DexelaException

GetROICoordinates (unsigned short &usStartColumn, unsigned short &usStartRow, unsigned short &usROIWidth, unsigned short &usROIHeight)
This method retrieves the coordinates of the region of interest (ROI) set within the detector. [More...](#)

void EnableROIMode (bool bEnableROI)
This method activates or deactivates the ROI mode of the detector. [More...](#)

bool GetROIState ()
This method retrieves the enabled state of the region of interest of the detector. [More...](#)

unsigned short GetSensorHeight (unsigned short uiSensorID=1)
Gets the height of the sensor in pixels. [More...](#)

unsigned short **GetSensorWidth** (unsigned short uiSensorID=1)
Gets the width of the sensor in pixels.
[More...](#)

bool **IsFrameCntWithinImage** ()
Checks if framecounter is displayed within the image data (being the 2nd pixel).
[More...](#)

void **EnableFrameCntWithinImage** (unsigned short usEnable)
Enables displaying the framecounter in the image (being the 2nd pixel). [More...](#)

void **SetSlowed** (bool flag)
This method can specify to the api that the detector being used is a slowed-down detector (e.g. mammo detector). This should not be necessary as the API should be able to determine most of the time whether the firmwarwe version is a slowed down one. However, for certain older detectors/firmwares this may not be possible. In this case this method can be used to inform the library that the firmware is slowed down and it will use the correct read-out times. [More...](#)

void **SetReadoutMode** (**ReadoutModes** mode)
Sets the ReadoutMode parameter of the detector [More...](#)

ReadoutModes **GetReadoutMode** ()
Gets the ReadoutModes parameter of the detector. [More...](#)

`int QueryReadoutMode (ReadoutModes mode)`

Query the detector to see if the desired readout mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

`int QueryExposureMode (ExposureModes mode)`

Query the detector to see if the desired exposure mode is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

`int QueryTriggerSource (ExposureTriggerSource ets)`

Query the detector to see if the desired trigger source is present (i.e. available).

Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the

feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

int **QueryFullWellMode (FullWellModes fwm)**
Query the detector to see if the desired full-well mode is present (i.e. available).
Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

int **QueryBinningMode (bins flag)**
Query the detector to see if the desired binning mode is present (i.e. available).
Note: Older detectors may not support the querying of features. In this case a value of -1 will be returned indicating that it is unknown whether the feature is present. In this case it is possible that the feature is present but that the detector is unable to report so. It may still be possible to use the feature in this case but no guarantees can be made on whether or not it will work. [More...](#)

Additional Inherited Members

▶ Protected Attributes inherited from DexelaDetector

boost::shared_ptr< baseDetector > **base**

boost::shared_ptr< gigEDetector > **gigeDet**

boost::shared_ptr
< camLinkDetector > **cIDet**

Detailed Description

This class is used to control GigE Type Detectors. It will give access to functions that are not available to other interface-type detectors.

Note: For all standard detector function calls please see the [DexelaDetector](#) class (these functions are also available to [DexelaDetectorGE](#) objects)

Constructor & Destructor Documentation

DexelaDetectorGE::DexelaDetectorGE (DevInfo & devInfo)

Constructor for [DexelaDetectorGE](#). Identical to the [DexelaDetector constructor](#), except with an additional check for the correct (GIGE) interface.

Note: A

Exceptions

DexelaException is thrown if the interface of the **DevInfo** object is not correct (i.e. GIGE)

Parameters

devInfo The [DevInfo](#) object for the desired detector. This can be obtained from the [GetDeviceGE](#) method

Exceptions

DexelaException

Constructor for **DexelaDetectorGE**. This version assumes you know the interface and the correct parameters to connect to the detector.

Parameters

transport The **DetectorInterface** for the detector (i.e. GIGE)
unit For GIGE detectors this can be set to 0
params The parameter string for connection to the detector. For GIGE detectors this should be the detector IP address.

Exceptions

DexelsException

DexelsDetectorGE::~DexelsDetectorGE (void)

virtual

Destructor for **DexelsDetectorGE**.

Member Function Documentation

void DexelaDetectorGE::OpenBoard ()

virtual

Identical to the [OpenBoard](#) call. The only difference is a check to make sure the detector has the correct (GIGE) interface.

Exceptions

[DexelaException](#)

Reimplemented from [DexelaDetector](#).

void DexelaDetectorGE::OpenBoard (int NumBufs)

Identical to the [OpenBoard](#) call. The only difference is a check to make sure the detector has the correct (GIGE) interface.

Parameters

NumBufs Number of buffers to use/allocate

Exceptions

[DexelaException](#)

```
void  
DexelaDetectorGE::SetPersistentIPAddress ( int firstByte,  
                                         int secondByte,  
                                         int thirdByte,  
                                         int fourthByte  
 )
```

Function for setting a new persistent IP address for the detector.
After power-cycling the detector it should come up with the desired

IP address

Parameters

- firstByte** The first byte of the desired IP address (e.g. 169 for the address 169.254.70.3)
- secondByte** The second byte of the desired IP address (e.g. 254 for the address 169.254.70.3)
- thirdByte** The third byte of the desired IP address (e.g. 70 for the address 169.254.70.3)
- fourthByte** The fourth byte of the desired IP address (e.g. 3 for the address 169.254.70.3)

Exceptions

- DexelsException**

The documentation for this class was generated from the following files:

- [DexelsDetectorGE.h](#)
- DexelaDetectorGE.cpp

DexelsDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members
			Public Member Functions Static Public Member Functions List of all members

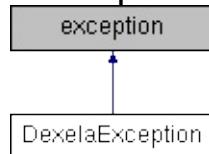
DexelsException Class Reference

This class contains information about any possible error's in the API. In the event of a problem a [DexelsException](#) will be thrown.

Note: It is suggested that you wrap your code in a try-catch block to ensure that if any errors occur you can detect (and properly handle them) in your code. [More...](#)

```
#include <DexelsException.h>
```

Inheritance diagram for DexelaException:



Public Member Functions

DexelaException (const char *message, **Derr** code, int line, const char *filename, const char *function, int transportEr, const char *transportMessage)
Constructor for the **DexelaException** Class. [More...](#)

DexelaException (const **DexelaException** &ex, const char *function)
Copy constructor for the **DexelaException** Class.
[More...](#)

~DexelaException (void) throw ()
DexelaException destructor. [More...](#)

const char * **what** () const throw ()
Function for retrieving the exception's error message.
[More...](#)

Derr **GetCode** ()
Function for retrieving the exception's **Derr** code.
[More...](#)

int **GetTransportError** ()
Function for retrieving the exception's low-level transport error code.
Note: This code (along with line-number, and file-name) can be sent to PerkinElmer support for further information about possible causes of the exception.
[More...](#)

const char * **GetFileName** ()
Function for retrieving the name of the (low-level) source file from which the exception was thrown.
Note: This information (along with line-number, and transport error) can be sent to PerkinElmer support for further information about possible causes of the

exception. [More...](#)

int [GetLineNumber \(\)](#)

Function for retrieving line-number of the source of the (low-level) exception throw.

Note: This number (along with transport-error and file-name) can be sent to PerkinElmer support for further information about possible causes of the exception.

[More...](#)

const char * [GetFunctionName \(\)](#)

Function for retrieving the name of the (top-level) function from which the error was thrown.

Note: This should help to find the function that is causing the exception. [More...](#)

const char * [GetTransportMessage \(\)](#)

Function for retrieving the (low-level) message from the transport library.

Note: This message (along with transport-error, line number and file-name) can be sent to PerkinElmer support for further information about possible causes of the exception. [More...](#)

Static Public Member Functions

static void **LoadErrorStrings** (const char *filename)

Detailed Description

This class contains information about any possible error's in the API. In the event of a problem a **DexelaException** will be thrown.

Note: It is suggested that you wrap your code in a try-catch block to ensure that if any errors occur you can detect (and properly handle them) in your code.

Constructor & Destructor Documentation

Constructor for the **DexelaException** Class.

Parameters

message	The error message, detailing the source of the exception.
code	The Derr error code associated with the exception.
line	The (low-level) line-number of the source of the exception.
filename	The (low-level) name of the source file that was the source of the exception.
function	The (top-level) name of the function in which the exception was thrown.
transportEr	The (low-level) transport error that may be associated with the exception.
transportMessage	The (low-level) transport layer error message. This may be empty depending on the source of the error.

)

Copy constructor for the **DexelaException** Class.

Parameters

ex The **DexelaException** object from which to copy the low-level information about the source of the exception.

function The top-level name of the function in which the exception thrown.

```
DexelaException::~DexelaException( void )
    throw(
        )
```

DexelaException destructor.

Member Function Documentation

Derr DexelaException::GetCode ()

Function for retrieving the exception's **Derr** code.

Returns

A member of the **Derr** enumeration detailing the exception's error code.

const char * DexelaException::GetFileName ()

Function for retrieving the name of the (low-level) source file from which the exception was thrown.

Note: This information (along with line-number, and transport error) can be sent to PerkinElmer support for further information about possible causes of the exception.

Returns

A string containing the name of the (low-level) source file from which the exception was thrown.

const char * DexelaException::GetFunctionName ()

Function for retrieving the name of the (top-level) function from which the error was thrown.

Note: This should help to find the function that is causing the exception.

Returns

A string containing the name of the (top-level) function from which the exception was thrown.

int DexelaException::GetLineNumber()

Function for retrieving line-number of the source of the (low-level) exception throw.

Note: This number (along with transport-error and file-name)can be sent to PerkinElmer support for further information about possible causes of the exception.

Returns

An integer detailing the exception's low-level line number.

int DexelaException::GetTransportError()

Function for retrieving the exception's low-level transport error code.

Note: This code (along with line-number, and file-name)can be sent to PerkinElmer support for further information about possible causes of the exception.

Returns

An integer detailing the exception's low-level transport error code.

const char * DexelaException::GetTransportMessage()

Function for retrieving the (low-level) message from the transport library.

Note: This message (along with transport-error, line number and file-name)can be sent to PerkinElmer support for further information about possible causes of the exception.

Returns

A string containing the low-level transport layer error message.

const char * DexelaException::what() const throw(

)

Function for retrieving the exception's error message.

Returns

A string containing the exception's error message.

The documentation for this class was generated from the following files:

- [DexelsException.h](#)
- DexelaException.cpp

DexelaDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members
Public Member Functions List of all members			

DexImage Class Reference

This class is used to store and handle the images acquired from a detector. [More...](#)

```
#include <DexImage.h>
```

Public Member Functions

DexImage (void)

DexImage constructor. Creates a new (empty) image. [More...](#)

DexImage (const char *filename)

DexImage constructor. Creates a new image by reading in from the specified file. [More...](#)

DexImage (const **DexImage** &input)

DexImage copy constructor. Creates a new

DexImage object by copying the input **DexImage** object. [More...](#)

void **operator=** (const **DexImage** &input)

DexImage assignment operator. Creates a new

DexImage object by copying the input **DexImage** object. [More...](#)

~DexImage (void)

DexImage destructor. [More...](#)

void **ReadImage** (const char *filename)

Reads an image in from the specified file [More...](#)

void **WritelImage** (const char *filename)

Writes the image data to the specified file (SMV, HIS or TIF)

Note: This will write the entire image stack out.

Note2: See the function **WritelImage** for writing out a single (user specified) plane from the stack. [More...](#)

void **WritelImage** (const char *filename, int iZ)

Writes out a single image plane (user specified)

from the stack (SMV, HIS or TIF) [More...](#)

void **Build** (int iWidth, int iHeight, int iDepth, **pType** iPxType)
Builds an image using the specified dimensions and pixel type [More...](#)

void **Build** (int model, **bins** binFmt, int iDepth)
Builds an image using the specified detector model and binning format [More...](#)

void * **GetDataPointerToPlane** (int iZ=0)
Gets the pointer to the image data for the specified plane. [More...](#)

int **GetImageXdim** ()
Gets the image x dimension (width) in pixels.
[More...](#)

int **GetImageYdim** ()
Gets the image y dimension (height) in pixels.
[More...](#)

int **GetImageDepth** ()
Gets the image depth. [More...](#)

pType **GetImagePixelType** ()
Gets the image pixel type. [More...](#)

float **PlaneAvg** (int iZ=0)
Calculates the average pixel value of the input image for the specified plane. [More...](#)

void **FixFlood** ()
This image fixes the input flood image. This means that the reciprocal of the image is taken and normalized about 1. This is done to speed up

the flood correction procedure (multiplication is faster than division).

Note: The input flood image should be a median image (i.e. should have a depth of 1). This can be obtained by using the **FindMedianofPlanes** method. [More...](#)

void [FindMedianofPlanes \(\)](#)

This calculates the median image from the input image stack.

Note: This function will replace the input stack of images with a single (median) image [More...](#)

void [FindAverageofPlanes \(\)](#)

This calculates the average image from the input image stack.

Note: This function will replace the input stack of images with a single (average) image of type float
[More...](#)

void [LinearizeData \(\)](#)

Linearizes the pixel values of the image. This is done using a piece-wise linear approximation where the sections are defined by an array of integers (linearization starts). This allows the output of the detector to be made linear

Note: A default set of linearization starts will be used unless the user specifies their own using the **SetLinearizationStarts** method. [More...](#)

void [SubtractDark \(\)](#)

Subtracts a dark image from an input image.

Note: The dark images must first be loaded before calling this function (see **LoadDarkImage**).

Note2: A dark offset value will be added to the resulting image to prevent any negative numbers. This offset is set to 300 by default but can be changed using the **SetDarkOffset** method. [More...](#)

void FloodCorrection ()
Performs flood correction on the input image using the passed in fixed-flood image.
Note: The flood and dark images must first be loaded before calling this function (see [LoadFloodImage](#), [LoadDarkImage](#)). [More...](#)

void DefectCorrection (int DefectFlags=31)
Function for performing defect corrections on the image.
Note: The flood, dark and defect map images must first be loaded before calling this function (see [LoadFloodImage](#), [LoadDarkImage](#), [LoadDefectMap](#)). [More...](#)

void SubImageDefectCorrection (int startCol, int startRow, int width, int height, int CorrectionsFlag=31)

void FullCorrection ()
This function performs the full correction (dark/offset, flood/gain and defect).
Note: The flood, dark and defect map images must first be loaded before calling this function (see [LoadFloodImage](#), [LoadDarkImage](#), [LoadDefectMap](#)). [More...](#)

void UnscrambleImage ()
This function unscrambles (sorts) a raw image acquired from a detector.
Note: The model number and the binning mode of the detector that the image was captured from must be specified (using [SetImageParameters](#)) before calling this method. [More...](#)

void AddImage ()
Adds another image (plane) to the stack. [More...](#)

`void LoadDarkImage (const DexImage &dark)`

Loads the dark image from the specified **DexImage** object.

Note: This dark image will then automatically be used for the various corrections.

Note2: This image should be a single plane (e.g. median) image. If it's not then the median image will be calculated (using **FindMedianofPlanes**) and stored.

Note3: This image should be of type **Offset**.

[More...](#)

`void LoadDarkImage (const char *filename)`

Loads the dark image from the specified file.

Note: This dark image will then automatically be used for the various corrections.

Note2: This image should be a single plane (e.g. median) image. If it's not then the median image will be calculated (using **FindMedianofPlanes**) and stored.

Note3: This image should be of type **Offset**.

[More...](#)

`void LoadFloodImage (const DexImage &flood)`

Loads the flood image from the specified **DexImage** object.

Note: This flood image will then automatically be used for offset corrections.

Note2: This image should be a single plane, fixed floating point image. If it's not then the median image will be calculated (using **FindMedianofPlanes**), then the image will be fixed (using **FixFlood**) and stored.

Note3: This image should be of type **Gain**. [More...](#)

`void LoadFloodImage (const char *filename)`

Loads the flood image from the specified file.

Note: This flood image will then automatically be used for gain corrections.

Note2: This image should be a single plane, fixed floating point image. If it's not then the median image will be calculated (using [FindMedianofPlanes](#)), then the image will be fixed (using [FixFlood](#)) and stored.

Note3: This image should be of type [Gain](#). More...

void [LoadDefectMap](#) (const [DexImage](#) &defect)
Loads the defect-map image from the specified [DexImage](#) object.
Note: This defect-map image will then automatically be used for defect corrections.
Note2: This image should be of type [Defect](#).
More...

void [LoadDefectMap](#) (const char *filename)
Loads the defect-map image from the specified file.
Note: This defect-map image will then automatically be used for defect corrections.
Note2: This image should be of type [Defect](#).
More...

[DexImage](#) [GetDarkImage](#) ()
Gets the dark image (which is used in offset/dark correction). [More...](#)

[DexImage](#) [GetFloodImage](#) ()
Gets the flood image (which is used in gain/flood correction). [More...](#)

[DexImage](#) [GetDefectMap](#) ()
Gets the defect-map image (which is used in defect correction). [More...](#)

[DexImage](#) [GetImagePlane](#) (int iZ)
Creates a new [DexImage](#) object from the data at the specified plane. [More...](#)

DexImageTypes **GetImageType ()**
Gets the image type (e.g. offset, gain, data, defect map). [More...](#)

void **SetImageType (DexImageTypes type)**
Sets the image type to the desired type. [More...](#)

void **SetDarkOffset (int offset)**
Sets the dark offset value to be used for various corrections (e.g. dark correction). This value is used as an offset to prevent from the possibility of getting negative pixel values. [More...](#)

int **GetDarkOffset ()**
Gets the current dark offset value. This value is used as an offset to prevent from the possibility of getting negative pixel values. [More...](#)

void **SetLinearizationStarts (unsigned int *msArray, int msLength)**
Sets the linearization section numbers that are used for the linearization correction ([LinearizeData](#)). [More...](#)

unsigned int * **GetLinearizationStarts (int &msLength)**
Gets the linearization section numbers that are used for the linearization correction([LinearizeData](#)). [More...](#)

void **SetImageParameters (bins binningMode, int modelNumber)**
Sets the model number and the binning mode of the detector that was used to acquire the image. This is used for the data-sorting ([UnscrambleImage](#)). [More...](#)

int **GetImageModel ()**

Gets the model number of the detector that was used to acquire the image. [More...](#)

bins **GetImageBinning ()**

Gets the binning mode of the detector that was used to acquire the image. [More...](#)

bool **IsEmpty ()**

This method returns whether the image is empty or not. [More...](#)

void **SetScrambledFlag (bool onOff)**

This method can be used to manually set the scrambled flag of the image. This flag stores whether the data from the detector has already been unscrambled or not. This flag is automatically set when an image is unscrambled and consequently this method should not be required for most use-cases. [More...](#)

void **SetROIParameters (unsigned short usStartColumn, unsigned short usStartRow, unsigned short usROIWidth, unsigned short usROIHeight)**

Detailed Description

This class is used to store and handle the images acquired from a detector.

Constructor & Destructor Documentation

DexImage::DexImage (void)

DexImage constructor. Creates a new (empty) image.

Exceptions

DexelsException

DexImage::DexImage (const char * filename)

DexImage constructor. Creates a new image by reading in from the specified file.

Parameters

filename Path to file to read image in from

Exceptions

DexelsException

DexImage::DexImage (const DexImage & input)

DexImage copy constructor. Creates a new **DexImage** object by copying the input **DexImage** object.

Parameters

input DexImage object to copy from.

Exceptions

DexelsException

DexImage::~DexImage (void)

DexImage destructor.

Member Function Documentation

void DexImage::AddImage()

Adds another image (plane) to the stack.

Exceptions

DexelaException

```
void DexImage::Build ( int      iWidth,
                      int      iHeight,
                      int      iDepth,
                      pType   iPxType
                    )
```

Builds an image using the specified dimensions and pixel type

Parameters

iWidth Desired width (in pixels) for the image.

iHeight Desired height (in pixels) for the image.

iDepth Desired depth for the image.

iPxType A member of the **pType** enumeration representing the pixel type for the image.

Exceptions

DexelaException

```
void DexImage::Build ( int    model,
                      bins   binFmt,
                      int    iDepth
                    )
```

Builds an image using the specified detector model and binning format

Parameters

model Detector model type corresponding to the image.

binFmt A member of the **bins** enumeration representing the binning mode corresponding to the image.

iDepth Desired depth for the image.

Exceptions

DexelaException

void DexImage::DefectCorrection (int DefectFlags = 31)

Function for performing defect corrections on the image.

Note: The flood, dark and defect map images must first be loaded before calling this function (see [LoadFloodImage](#), [LoadDarkImage](#), [LoadDefectMap](#)).

Exceptions

DexelaException

void DexImage::FindAverageofPlanes ()

This calculates the average image from the input image stack.

Note: This function will replace the input stack of images with a single (average) image of type float

Exceptions

DexelaException

void DexImage::FindMedianofPlanes ()

This calculates the median image from the input image stack.

Note: This function will replace the input stack of images with a single (median) image

Exceptions

[DexelaException](#)

void DexImage::FixFlood()

This image fixes the input flood image. This means that the reciprocal of the image is taken and normalized about 1. This is done to speed up the flood correction procedure (multiplication is faster than division).

Note: The input flood image should be a median image (i.e. should have a depth of 1). This can be obtained by using the [FindMedianofPlanes](#) method.

Exceptions

[DexelaException](#)

void DexImage::FloodCorrection()

Performs flood correction on the input image using the passed in fixed-flood image.

Note: The flood and dark images must first be loaded before calling this function (see [LoadFloodImage](#), [LoadDarkImage](#)).

Exceptions

[DexelaException](#)

void DexImage::FullCorrection()

This function performs the full correction (dark/offset, flood/gain and defect).

Note: The flood, dark and defect map images must first be loaded before calling this function (see [LoadFloodImage](#), [LoadDarkImage](#),

`LoadDefectMap).`

Exceptions

`DexelsException`

DexImage DexImage::GetDarkImage ()

Gets the dark image (which is used in offset/dark correction).

Returns

The `DexImage` object that is used for offset/dark corrections

Exceptions

`DexelsException`

int DexImage::GetDarkOffset ()

Gets the current dark offset value. This value is used as an offset to prevent from the possibility of getting negative pixel values.

Returns

The offset value (in ADU).

Exceptions

`DexelsException`

void * DexImage::GetDataPointerToPlane (int iZ = 0)

Gets the pointer to the image data for the specified plane.

Parameters

`iZ` The number of the desired image plane.

Returns

Pointer to the image data for the specified plane.

Exceptions

[DexelaException](#)

DexImage DexImage::GetDefectMap ()

Gets the defect-map image (which is used in defect correction).

Returns

The [DexImage](#) object that is used for defect corrections

Exceptions

[DexelaException](#)

DexImage DexImage::GetFloodImage ()

Gets the flood image (which is used in gain/flood correction).

Returns

The [DexImage](#) object that is used for gain/flood corrections

Exceptions

[DexelaException](#)

bins DexImage::GetImageBinning ()

Gets the binning mode of the detector that was used to acquire the image.

Returns

A member of the [bins](#) enumeration representing the binning mode of the detector.

Exceptions

[DexelaException](#)

int DexImage::GetImageDepth ()

Gets the image depth.

Returns

Image depth.

Exceptions

[DexelaException](#)

int DexImage::GetImageModel ()

Gets the model number of the detector that was used to acquire the image.

Returns

The detector model number.

Exceptions

[DexelaException](#)

pType DexImage::GetImagePixelType ()

Gets the image pixel type.

Returns

A member of the [pType](#) enumeration specifying the image pixel type.

Exceptions

[DexelaException](#)

DexImage DexImage::GetImagePlane (int iZ)

Creates a new [DexImage](#) object from the data at the specified

plane.

Parameters

iZ The index of the plane to get the data from.

Returns

A new **DexImage** object that consists of the data from the desired plane

Exceptions

DexelsException

DexImageTypes DexImage::GetImageType ()

Gets the image type (e.g. offset, gain, data, defect map).

Returns

A member of the **DexImageTypes** enumeration specifying the image type.

Exceptions

DexelsException

int DexImage::GetImageXdim ()

Gets the image x dimension (width) in pixels.

Returns

Image x dimension (width) in pixels.

Exceptions

DexelsException

int DexImage::GetImageYdim ()

Gets the image y dimension (height) in pixels.

Returns

Image y dimension (height) in pixels.

Exceptions

DexelaException

unsigned int *

DexImage::GetLinearizationStarts

(int & msLength)

Gets the linearization section numbers that are used for the linearization correction ([LinearizeData](#)).

Parameters

msLength An integer that will be set to the length of the linearization section numbers array.

Returns

An array of unsigned integers representing the section numbers used for linearization correction.

Exceptions

DexelaException

bool DexImage::IsEmpty ()

This method returns whether the image is empty or not

Returns

A boolean indicating whether the image is empty.

Exceptions

DexelaException

void DexImage::LinearizeData()

Linearizes the pixel values of the image. This is done using a piece-wise linear approximation where the sections are defined by an array of integers (linearization starts). This allows the output of the detector to be made linear

Note: A default set of linearization starts will be used unless the user specifies their own using the **SetLinearizationStarts** method.

Exceptions

DexelaException

void DexImage::LoadDarkImage (const DexImage & dark)

Loads the dark image from the specified **DexImage** object.

Note: This dark image will then automatically be used for the various corrections.

Note2: This image should be a single plane (e.g. median) image. If it's not then the median image will be calculated (using **FindMedianofPlanes**) and stored.

Note3: This image should be of type **Offset**.

Parameters

dark The **DexImage** object that should be used for dark corrections.

Exceptions

DexelaException

void DexImage::LoadDarkImage (const char * filename)

Loads the dark image from the specified file.

Note: This dark image will then automatically be used for the various corrections.

Note2: This image should be a single plane (e.g. median) image. If it's not then the median image will be calculated (using

`FindMedianofPlanes`) and stored.

Note3: This image should be of type `Offset`.

Parameters

filename The path to the image file that should be read in and used for dark corrections.

Exceptions

`DexelaException`

`void DexImage::LoadDefectMap (const DexImage & defect)`

Loads the defect-map image from the specified `DexImage` object.

Note: This defect-map image will then automatically be used for defect corrections.

Note2: This image should be of type `Defect`.

Parameters

defect The `DexImage` object that should be used for defect corrections.

Exceptions

`DexelaException`

`void DexImage::LoadDefectMap (const char * filename)`

Loads the defect-map image from the specified file.

Note: This defect-map image will then automatically be used for defect corrections.

Note2: This image should be of type `Defect..`

Parameters

filename The path to the image file that should be read in and used for defect corrections.

Exceptions

DexelaException

void DexImage::LoadFloodImage (const DexImage & flood)

Loads the flood image from the specified **DexImage** object.

Note: This flood image will then automatically be used for offset corrections.

Note2: This image should be a single plane, fixed floating point image. If it's not then the median image will be calculated (using **FindMedianofPlanes**), then the image will be fixed (using **FixFlood**) and stored.

Note3: This image should be of type **Gain**.

Parameters

flood The **DexImage** object that should be used for flood corrections.

Exceptions

DexelaException

void DexImage::LoadFloodImage (const char * filename)

Loads the flood image from the specified file.

Note: This flood image will then automatically be used for gain corrections.

Note2: This image should be a single plane, fixed floating point image. If it's not then the median image will be calculated (using **FindMedianofPlanes**), then the image will be fixed (using **FixFlood**) and stored.

Note3: This image should be of type **Gain**.

Parameters

flood The path to the image file that should be read in and used for flood corrections.

Exceptions

DexelaException

`void DexImage::operator= (const DexImage & input)`

DexImage assignment operator. Creates a new **DexImage** object by copying the input **DexImage** object.

Parameters

input **DexImage** object to copy from.

Exceptions

DexelaException

`float DexImage::PlaneAvg (int iZ = 0)`

Calculates the average pixel value of the input image for the specified plane.

Parameters

iZ The plane of the image to work on

Returns

A floating point number that is the average pixel value of the specified image plane.

Exceptions

DexelaException

`void DexImage::ReadImage (const char * filename)`

Reads an image in from the specified file

Parameters

filename The path to the file to read image from. Currently this is limited to SMV, HIS and TIF file types.

Exceptions

[DexelaException](#)

void DexImage::SetDarkOffset (int offset)

Sets the dark offset value to be used for various corrections (e.g. dark correction). This value is used as an offset to prevent from the possibility of getting negative pixel values.

Parameters

offset The offset value (in ADU).

Exceptions

[DexelaException](#)

void DexImage::SetImageParameters (bins binningMode, int modelNumber)

Sets the model number and the binning mode of the detector that was used to acquire the image. This is used for the data-sorting ([UnscrambleImage](#)).

Parameters

binningMode A member of the [bins](#) enumeration representing the binning mode of the detector.

modelNumber The model number of the detector

Exceptions

[DexelaException](#)

void DexImage::SetImageType (DexImageTypes type)

Sets the image type to the desired type.

Parameters

type A member of the [DexImageTypes](#) enumeration that the image type should be set to.

Exceptions

[DexelsException](#)

```
void
DexImage::SetLinearizationStarts ( unsigned int * msArray,
                                  int               msLength
                                )
```

Sets the linearization section numbers that are used for the linearization correction ([LinearizeData](#)).

Parameters

msArray An array of unsigned integers representing the section numbers used for linearization correction.

msLength The length of the array

Exceptions

[DexelsException](#)

```
void DexImage::SetScrambledFlag ( bool onOff )
```

This method can be used to manually set the scrambled flag of the image. This flag stores whether the data from the detector has already been unscrambled or not. This flag is automatically set when an image is unscrambled and consequently this method should not be required for most use-cases.

Parameters

onOff A boolean representing the desired state of the flag.

Exceptions

[DexelsException](#)

void DexImage::SubtractDark()

Subtracts a dark image from an input image.

Note: The dark images must first be loaded before calling this function (see [LoadDarkImage](#)).

Note2: A dark offset value will be added to the resulting image to prevent any negative numbers. This offset is set to 300 by default but can be changed using the [SetDarkOffset](#) method.

Exceptions

[DexelaException](#)

void DexImage::UnscrambleImage()

This function unscrambles (sorts) a raw image acquired from a detector.

Note: The model number and the binning mode of the detector that the image was captured from must be specified (using [SetImageParameters](#)) before calling this method.

Exceptions

[DexelaException](#)

void DexImage::WritelImage (const char * filename)

Writes the image data to the specified file (SMV, HIS or TIF)

Note: This will write the entire image stack out.

Note2: See the function [WritelImage](#) for writing out a single (user specified) plane from the stack.

Parameters

filename The path to the file to write image to. Currently this is limited to SMV, HIS and TIF file types.

Exceptions

DexelaException

```
void DexImage::WriteImage ( const char * filename,  
                           int           iZ  
                         )
```

Writes out a single image plane (user specified) from the stack (SMV, HIS or TIF)

Parameters

filename The path to the file to write image to. Currently this is limited to SMV, HIS and TIF file types.
iZ Image plane to write out

Exceptions

DexelaException

The documentation for this class was generated from the following files:

- [DexImage.h](#)
- [DexImage.cpp](#)

DexelsDetector API

[Main Page](#)[Classes](#)[Files](#)[Class List](#)[Class Index](#)[Class Hierarchy](#)[Class Members](#)[Public Attributes](#) | [List of all members](#)

GeometryCorrectionParams Struct Reference

A structure used to specify the new image dimensions for geometry correction [More...](#)

```
#include <DexDefs.h>
```

Public Attributes

int **iRefX**

The new image width More...

int **iRefY**

The new image height More...

Detailed Description

A structure used to specify the new image dimensions for geometry correction

Member Data Documentation

int GeometryCorrectionParams::iRefX

The new image width

int GeometryCorrectionParams::iRefY

The new image height

The documentation for this struct was generated from the following file:

- [DexDefs.h](#)
-

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members

Class Index

[B](#) | [D](#) | [G](#)

[B](#)

[D](#)

[DevInfo](#)

[DexelsDetectorGE](#)

[DexelsDetector](#)

[DexelsException](#)

[BusScanner](#)

[DetStatus](#)

[DexelsDetectorCL](#)

[DexImage](#)

[B](#) | [D](#) | [G](#)

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by [doxygen](#) 1.8.7

DexelaDetector API

Main Page	Classes	Files
Class List	Class Index	Class Hierarchy
Class Members		

Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

[detail level [1](#) [2](#)]

 BusScanner	This class is used to scan the different interfaces and give information about devices found.
 DetStatus	Structure to hold the detector current status.
 DevInfo	A structure to hold device information.
 DexelaDetector	This class is used to control any interface-type Detector and acquire images from it. It will provide all the basic functionality required for all different Dexela detectors. For interface specific functionality please see the interface specific classes (e.g. DexelaDetectorGE , DexelaDetectorCL).
 DexelaDetectorCL	This class is used to control CameraLink Type Detectors. It will give access to functions that are not available to other interface-type detectors. Note: For all standard detector function calls please see the DexelaDetector class (these

	<p>functions are also available to DexelaDetectorCL objects)</p>
 DexelaDetectorGE	<p>This class is used to control GigE Type Detectors. It will give access to functions that are not available to other interface-type detectors.</p> <p>Note: For all standard detector function calls please see the DexelaDetector class (these functions are also available to DexelaDetectorGE objects)</p>
 DexImage	<p>This class is used to store and handle the images acquired from a detector.</p>
 ▾  exception	
 DexelaException	<p>This class contains information about any possible error's in the API. In the event of a problem a DexelaException will be thrown.</p> <p>Note: It is suggested that you wrap your code in a try-catch block to ensure that if any errors occur you can detect (and properly handle them) in your code.</p>
 GeometryCorrectionParams	<p>A structure used to specify the new image dimensions for geometry correction</p>

DexelsDetector API

The screenshot shows a navigation bar for a software API. At the top, there are three main tabs: "Main Page", "Classes", and "Files". The "Classes" tab is currently selected and highlighted in blue. Below these are four sub-tabs: "Class List", "Class Index", "Class Hierarchy", and "Class Members", with "Class Members" also being highlighted in blue. Under "Class Members", there are three categories: "All", "Functions", and "Variables", with "All" being the active choice. At the bottom of the bar is a horizontal menu with letters: a, b, c, d, e, f, g, i, l, m, o, p, q, r, s, t, u, w, ~. The letter 'a' is highlighted in blue, indicating the current section of the alphabetical index.

Here is a list of all documented class members with links to the class documentation for each member:

- a -

- AddImage() : [DexImage](#)

- b -

- binLevel : [DetStatus](#)
- Build() : [DexImage](#)
- BusScanner() : [BusScanner](#)

- c -

- CheckForCallbackError() : [DexelsDetector](#)
- CheckForLiveError() : [DexelsDetector](#)
- ClearBuffers() : [DexelsDetector](#)
- ClearCameraBuffer() : [DexelsDetector](#)
- CloseBoard() : [DexelsDetector](#)

- d -

- DefectCorrection() : [DexImage](#)
- DexelaDetector() : [DexelsDetector](#)
- DexelaDetectorCL() : [DexelsDetectorCL](#)
- DexelaDetectorGE() : [DexelsDetectorGE](#)
- DexelaException() : [DexelsException](#)
- DexImage() : [DexImage](#)
- DisablePulseGenerator() : [DexelsDetector](#)

- e -

- EnableFrameCntWithinImage() : DexelaDetector
- EnablePulseGenerator() : DexelaDetector
- EnableROIMode() : DexelaDetector
- EnumerateCLDevices() : BusScanner
- EnumerateDevices() : BusScanner
- EnumerateGDEDevices() : BusScanner
- exposureMode : DetStatus
- exposureTime : DetStatus

- f -

- FindAverageofPlanes() : DexImage
- FindMedianofPlanes() : DexImage
- FixFlood() : DexImage
- FloodCorrection() : DexImage
- FullCorrection() : DexImage
- fullWellMode : DetStatus

- g -

- GetBinningMode() : DexelaDetector
- GetBufferXdim() : DexelaDetector
- GetBufferYdim() : DexelaDetector
- GetCapturedBuffer() : DexelaDetector
- GetCode() : DexelaException
- GetDarkImage() : DexImage
- GetDarkOffset() : DexImage
- GetDataPointerToPlane() : DexImage
- GetDefectMap() : DexImage
- GetDetectorStatus() : DexelaDetector
- GetDevice() : BusScanner
- GetDeviceCL() : BusScanner
- GetDeviceGE() : BusScanner
- GetExposureMode() : DexelaDetector
- GetExposureTime() : DexelaDetector
- GetFieldCount() : DexelaDetector
- GetFileName() : DexelaException
- GetFirmwareBuild() : DexelaDetector

- GetFirmwareVersion() : DexelaDetector
- GetFloodImage() : DexImage
- GetFullWellMode() : DexelaDetector
- GetFunctionName() : DexelaException
- GetGapTime() : DexelaDetector
- GetImageBinning() : DexImage
- GetImageDepth() : DexImage
- GetImageModel() : DexImage
- GetImagePixelType() : DexImage
- GetImagePlane() : DexImage
- GetImageType() : DexImage
- GetImageXdim() : DexImage
- GetImageYdim() : DexImage
- GetLinearizationStarts() : DexImage
- GetLineNumber() : DexelaException
- GetModelNumber() : DexelaDetector
- GetNumBuffers() : DexelaDetector
- GetNumOfExposures() : DexelaDetector
- GetReadoutMode() : DexelaDetector
- GetReadOutTime() : DexelaDetector
- GetROICoordinates() : DexelaDetector
- GetROIState() : DexelaDetector
- GetSensorHeight() : DexelaDetector
- GetSensorWidth() : DexelaDetector
- GetSerialNumber() : DexelaDetector
- GetTestMode() : DexelaDetector
- GetTransportError() : DexelaException
- GetTransportMessage() : DexelaException
- GetTransportMethod() : DexelaDetector
- GetTriggerSource() : DexelaDetector
- GoLiveSeq() : DexelaDetector
- GoUnLive() : DexelaDetector

- i -

- iface : DevInfo
- iRefX : GeometryCorrectionParams
- iRefY : GeometryCorrectionParams
- IsCallbackActive() : DexelaDetector
- IsConnected() : DexelaDetector

- IsEmpty() : **DexImage**
- IsFrameCntWithinImage() : **DexelaDetector**
- IsLive() : **DexelaDetector**

- | -

- LinearizeData() : **DexImage**
- LoadDarkImage() : **DexImage**
- LoadDefectMap() : **DexImage**
- LoadFloodImage() : **DexImage**
- LoadSensorConfigFile() : **DexelaDetector**

- m -

- model : **DevInfo**

- o -

- OpenBoard() : **DexelaDetector** , **DexelaDetectorCL** ,
DexelaDetectorGE
- operator=() : **DexImage**

- p -

- param : **DevInfo**
- PlaneAvg() : **DexImage**
- PowerCLInterface() : **DexelaDetectorCL**

- q -

- QueryBinningMode() : **DexelaDetector**
- QueryExposureMode() : **DexelaDetector**
- QueryFullWellMode() : **DexelaDetector**
- QueryReadoutMode() : **DexelaDetector**
- QueryTriggerSource() : **DexelaDetector**

- r -

- ReadBuffer() : **DexelaDetector**

- ReadImage() : DexImage
- ReadRegister() : DexelaDetector

- S -

- serialNum : DevInfo
- SetBinningMode() : DexelaDetector
- SetCallback() : DexelaDetector
- SetDarkOffset() : DexImage
- SetExposureMode() : DexelaDetector
- SetExposureTime() : DexelaDetector
- SetFullWellMode() : DexelaDetector
- SetGapTime() : DexelaDetector
- SetImageParameters() : DexImage
- SetImageType() : DexImage
- SetLinearizationStarts() : DexImage
- SetNumOfExposures() : DexelaDetector
- SetPersistentIPAddress() : DexelaDetectorGE
- SetPreProgrammedExposureTimes() : DexelaDetector
- SetReadoutMode() : DexelaDetector
- SetROICoordinates() : DexelaDetector
- SetScrambledFlag() : DexImage
- SetSlowed() : DexelaDetector
- SetTestMode() : DexelaDetector
- SetTriggerSource() : DexelaDetector
- Snap() : DexelaDetector
- SoftReset() : DexelaDetector
- SoftwareTrigger() : DexelaDetector
- StopCallback() : DexelaDetector
- SubtractDark() : DexImage

- t -

- testMode : DetStatus
- ToggleGenerator() : DexelaDetector
- transport : DevInfo
- triggerSource : DetStatus

- u -

- unit : [DevInfo](#)
- UnscrambleImage() : [DexImage](#)

- W -

- WaitImage() : [DexelaDetector](#)
- what() : [DexelaException](#)
- WriteBuffer() : [DexelaDetector](#)
- WriteImage() : [DexImage](#)
- WriteRegister() : [DexelaDetector](#)

- ~ -

- ~BusScanner() : [BusScanner](#)
- ~DexelaDetector() : [DexelaDetector](#)
- ~DexelaDetectorCL() : [DexelaDetectorCL](#)
- ~DexelaDetectorGE() : [DexelaDetectorGE](#)
- ~DexelaException() : [DexelaException](#)
- ~DexImage() : [DexImage](#)

DexelsDetector API

Main Page	Classes	Files															
Class List	Class Index	Class Hierarchy	Class Members														
All	Functions	Variables															
a	b	c	d	e	f	g	i	l	o	p	q	r	s	t	u	w	~

- a -

- AddImage() : DexImage

- b -

- Build() : DexImage
- BusScanner() : BusScanner

- c -

- CheckForCallbackError() : DexelaDetector
- CheckForLiveError() : DexelaDetector
- ClearBuffers() : DexelaDetector
- ClearCameraBuffer() : DexelaDetector
- CloseBoard() : DexelaDetector

- d -

- DefectCorrection() : DexImage
- DexelaDetector() : DexelaDetector
- DexelaDetectorCL() : DexelaDetectorCL
- DexelaDetectorGE() : DexelaDetectorGE
- DexelaException() : DexelaException
- DexImage() : DexImage
- DisablePulseGenerator() : DexelaDetector

- e -

- EnableFrameCntWithinImage() : DexelaDetector
- EnablePulseGenerator() : DexelaDetector
- EnableROIMode() : DexelaDetector
- EnumerateCLDevices() : BusScanner
- EnumerateDevices() : BusScanner
- EnumerateGDEDevices() : BusScanner

- f -

- FindAverageofPlanes() : DexImage
- FindMedianofPlanes() : DexImage
- FixFlood() : DexImage
- FloodCorrection() : DexImage
- FullCorrection() : DexImage

- g -

- GetBinningMode() : DexelaDetector
- GetBufferXdim() : DexelaDetector
- GetBufferYdim() : DexelaDetector
- GetCapturedBuffer() : DexelaDetector
- GetCode() : DexelaException
- GetDarkImage() : DexImage
- GetDarkOffset() : DexImage
- GetDataPointerToPlane() : DexImage
- GetDefectMap() : DexImage
- GetDetectorStatus() : DexelaDetector
- GetDevice() : BusScanner
- GetDeviceCL() : BusScanner
- GetDeviceGE() : BusScanner
- GetExposureMode() : DexelaDetector
- GetExposureTime() : DexelaDetector
- GetFieldCount() : DexelaDetector
- GetFileName() : DexelaException
- GetFirmwareBuild() : DexelaDetector
- GetFirmwareVersion() : DexelaDetector
- GetFloodImage() : DexImage
- GetFullWellMode() : DexelaDetector

- GetFunctionName() : DexelaException
- GetGapTime() : DexelaDetector
- GetImageBinning() : DexImage
- GetImageDepth() : DexImage
- GetImageModel() : DexImage
- GetImagePixelType() : DexImage
- GetImagePlane() : DexImage
- GetImageType() : DexImage
- GetImageXdim() : DexImage
- GetImageYdim() : DexImage
- GetLinearizationStarts() : DexImage
- GetLineNumber() : DexelaException
- GetModelNumber() : DexelaDetector
- GetNumBuffers() : DexelaDetector
- GetNumOfExposures() : DexelaDetector
- GetReadoutMode() : DexelaDetector
- GetReadOutTime() : DexelaDetector
- GetROICoordinates() : DexelaDetector
- GetROIState() : DexelaDetector
- GetSensorHeight() : DexelaDetector
- GetSensorWidth() : DexelaDetector
- GetSerialNumber() : DexelaDetector
- GetTestMode() : DexelaDetector
- GetTransportError() : DexelaException
- GetTransportMessage() : DexelaException
- GetTransportMethod() : DexelaDetector
- GetTriggerSource() : DexelaDetector
- GoLiveSeq() : DexelaDetector
- GoUnLive() : DexelaDetector

- i -

- IsCallbackActive() : DexelaDetector
- IsConnected() : DexelaDetector
- IsEmpty() : DexImage
- IsFrameCntWithinImage() : DexelaDetector
- IsLive() : DexelaDetector

- | -

- `LinearizeData()` : **DexImage**
- `LoadDarkImage()` : **DexImage**
- `LoadDefectMap()` : **DexImage**
- `LoadFloodImage()` : **DexImage**
- `LoadSensorConfigFile()` : **DexelaDetector**

- o -

- `OpenBoard()` : **DexelaDetector** , **DexelaDetectorCL** ,
DexelaDetectorGE
- `operator=()` : **DexImage**

- p -

- `PlaneAvg()` : **DexImage**
- `PowerCLInterface()` : **DexelaDetectorCL**

- q -

- `QueryBinningMode()` : **DexelaDetector**
- `QueryExposureMode()` : **DexelaDetector**
- `QueryFullWellMode()` : **DexelaDetector**
- `QueryReadoutMode()` : **DexelaDetector**
- `QueryTriggerSource()` : **DexelaDetector**

- r -

- `ReadBuffer()` : **DexelaDetector**
- `ReadImage()` : **DexImage**
- `ReadRegister()` : **DexelaDetector**

- s -

- `SetBinningMode()` : **DexelaDetector**
- `SetCallback()` : **DexelaDetector**
- `SetDarkOffset()` : **DexImage**
- `SetExposureMode()` : **DexelaDetector**
- `SetExposureTime()` : **DexelaDetector**
- `SetFullWellMode()` : **DexelaDetector**

- SetGapTime() : DexelaDetector
- SetImageParameters() : DexImage
- SetImageType() : DexImage
- SetLinearizationStarts() : DexImage
- SetNumOfExposures() : DexelaDetector
- SetPersistentIPAddress() : DexelaDetectorGE
- SetPreProgrammedExposureTimes() : DexelaDetector
- SetReadoutMode() : DexelaDetector
- SetROICoordinates() : DexelaDetector
- SetScrambledFlag() : DexImage
- SetSlowed() : DexelaDetector
- SetTestMode() : DexelaDetector
- SetTriggerSource() : DexelaDetector
- Snap() : DexelaDetector
- SoftReset() : DexelaDetector
- SoftwareTrigger() : DexelaDetector
- StopCallback() : DexelaDetector
- SubtractDark() : DexImage

- t -

- ToggleGenerator() : DexelaDetector

- u -

- UnscrambleImage() : DexImage

- w -

- WaitImage() : DexelaDetector
- what() : DexelaException
- WriteBuffer() : DexelaDetector
- WriteImage() : DexImage
- WriteRegister() : DexelaDetector

- ~ -

- ~BusScanner() : BusScanner
- ~DexelaDetector() : DexelaDetector

- ~DexelaDetectorCL() : **DexelaDetectorCL**
 - ~DexelaDetectorGE() : **DexelaDetectorGE**
 - ~DexelaException() : **DexelaException**
 - ~DexImage() : **DexImage**
-

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members
All	Functions	Variables	

- binLevel : **DetStatus**
- exposureMode : **DetStatus**
- exposureTime : **DetStatus**
- fullWellMode : **DetStatus**
- iface : **DevInfo**
- iRefX : **GeometryCorrectionParams**
- iRefY : **GeometryCorrectionParams**
- model : **DevInfo**
- param : **DevInfo**
- serialNum : **DevInfo**
- testMode : **DetStatus**
- transport : **DevInfo**
- triggerSource : **DetStatus**
- unit : **DevInfo**

DexelsDetector API

Main Page

Classes

Files

File List

File Members

File List

Here is a list of all documented files with brief descriptions:

 BusScanner.h	
 DexDefines.h	
 DexDefs.h	
 DexelsDetector.h	
 DexelsDetectorCL.h	
 DexelsDetectorGE.h	
 DexelsException.h	
 DexImage.h	
 DexImage/resource.h	
 BusScanner/resource.h	
 DexelsException/resource.h	
 resource1.h	

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page	Classes	Files
File List	File Members	
DexelsDetector	BusScanner	Classes

BusScanner.h File Reference

```
#include "DexDefs.h" #include "DexelsDetector.h"  
#include <vector>  
#include <boost/shared_ptr.hpp>
```

[Go to the source code of this file.](#)

Classes

class **BusScanner**

This class is used to scan the different interfaces and give information about devices found. [More...](#)

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page	Classes	Files
File List	File Members	
DexelsDetector	DexelsException	
Macros Typedefs		

DexDefines.h File Reference

Go to the source code of this file.

Macros

```
#define TransMsgSize 1024
```

```
#define DllExport __declspec( dllexport )
```

```
#define DllExportC __declspec( dllexport )
```

```
#define MAX_PIXEL_VAL 16383
```

Maximum value for any pixel in detector output (14 bit)

[More...](#)

```
#define MIN_PIXEL_VAL 0
```

Minimum allowable pixel value [More...](#)

```
#define minTimeIncrement 0.01F
```

The minimum time increment in ms for expose and read mode [More...](#)

```
#define minTimeIncrement2 195.2F
```

The minimum time increment in ns for line delay mode
[More...](#)

```
#define ExposureSleepTimems 10
```

Variable to hold a time in ms used for sleeping threads in streaming mode. [More...](#)

```
#define TimingResolution 100
```

The resolution of the exposure time settings 1 for ms 100 for 0.01 ms [More...](#)

```
#define RETURN_CHAR_LENGTH_CONST 50
```

```
#define DarkPixelXOffset 2
```

The offset in pixels in x when reading dark pixel data [More...](#)

```
#define DarkPixelYOffset 4
```

The offset in pixels in y when reading dark pixel data [More...](#)

```
#define AddrFPGANumber 126
```

Register address for the FPGA version number of the detector [More...](#)

```
#define AddrSerialNumber 125
```

Register address for the serial number of the detector [More...](#)

```
#define AddrModelNumber 124
```

Register address for the model number of the detector [More...](#)

```
#define AddrGapTime 18
```

Register address for the gap time used in Frame Rate mode [More...](#)

```
#define AddrNumberOfFrames 17
```

Register address of number of frames register for use with sequence modes [More...](#)

```
#define AddrFirmwareVersion 127
```

Register address of firmware verion information [More...](#)

```
#define AddrTriggerSource 0
```

Register address of the Trigger Source bits [More...](#)

```
#define AddrExposureTimeLow 11
```

```
#define AddrExposureTimeHigh 12
```

```
#define AddrExposureTime 12
```

Register address of exposure time information in low res system [More...](#)

```
#define AddrExposureTime2 13
    Register address of exposure time information in low res
    system for Line_Delay mode More...
```

```
#define AddrExposureTime2Low 13
    Register address of low bytes of exposure time information
    for line delay mode More...
```

```
#define AddrExposureTime2High 14
    Register address of high bytes of exposure time information
    for line delay mode More...
```

```
#define AddrHorizontalBinReg 10
    Address of horizontal binning register More...
```

```
#define AddrVerticalBinReg 9
    Address of vertical binning register More...
```

```
#define AddrControlReg 0
    Address of control register More...
```

```
#define AddrPPExposreTime1Low 27
    Register address of low bytes of pre-programmed exposure
    time 1 information More...
```

```
#define AddrPPExposreTime1High 28
    Register address of high bytes of pre-programmed
    exposure time 1 information More...
```

```
#define AddrPPExposreTime2Low 29
    Register address of low bytes of pre-programmed exposure
    time 2 information More...
```

```
#define AddrPPExposreTime2High 30
    Register address of high bytes of pre-programmed
    exposure time 2 information More...
```

```
#define AddrPPExposreTime3Low 31
    Register address of low bytes of pre-programmed exposure
    time 3 information More...
```

```
#define AddrPPExposreTime3High 32
    Register address of high bytes of pre-programmed
    exposure time 3 information More...
```

```
#define AddrPPExposreTime4Low 33
    Register address of low bytes of pre-programmed exposure
    time 4 information More...
```

```
#define AddrPPExposreTime4High 34
    Register address of high bytes of pre-programmed
    exposure time 4 information More...
```

```
#define SerialNumberReg1 0
    Address of serial number register 1 More...
```

```
#define SerialNumberReg2 0
    Address of serial number register 2 More...
```

```
#define SerialNumberReg3 0
    Address of serial number register 3 More...
```

```
#define TemperatureReg 0
    Address of temperature register More...
```

```
#define AddrWellReg 3
    Address of Well register More...
```

```
#define AddrWellHigh 4
    Address of High Fullwell register More...
```

```
#define AddrWellLow 65531
    Address of Low Fullwell register More...
```

-
- ```
#define AddrSensorBinReg 3
 Address of sensor bin register More...
```
- 
- ```
#define AddrSensorBinReg2 5
      Address of sensor bin register 2 More...
```
-
- ```
#define AddrNumLines 7
 Address of Numer of lines per sensor register More...
```
- 
- ```
#define AddrNumPixels 8
      Address of Number of pixels (per line) per sensor register
      More...
```
-
- ```
#define SensorBinClear 65087
 Constant for clearing binning command More...
```
- 
- ```
#define DigitalBinBit 65533
      Constant for digital binning bit More...
```
-
- ```
#define Sensor1x1 0
 Constant for 1x1 binning command More...
```
- 
- ```
#define Sensor1x2 0
      Constant for 1x2 binning command More...
```
-
- ```
#define Sensor1x4 64
 Constant for 1x4 binning command More...
```
- 
- ```
#define Sensor2x1 128
      Constant for 2x1 binning command More...
```
-
- ```
#define Sensor2x2 128
 Constant for 2x2 binning command More...
```
- 
- ```
#define Sensor2x4 192
```

Constant for 2x4 binning command [More...](#)

#define **Sensor4x1** 256

Constant for 4x1 binning command [More...](#)

#define **Sensor4x2** 256

Constant for 4x2 binning command [More...](#)

#define **Sensor4x4** 320

Constant for 4x4 binning command [More...](#)

#define **BinCommit** 514

Constant for binning commit command [More...](#)

#define **AddrReadOutTime** 410

Readout time for the sensor, dynamically adapted to the read out mode (binning, ROI). Dependent on the implementation the value has to be multiplied by the factor ReadoutTimeFactor [More...](#)

#define **ReadoutTimeFactor1313** 2

ReadoutTimeFactor for Dexela 1313 [More...](#)

#define **AddrROIStartColumn** 404

Register address for the ROI OFFSET / first column: [More...](#)

#define **AddrROIwidth** 405

Register address for the width of the ROI stripe [More...](#)

#define **AddrROIStartRow** 402

Register address for the ROI OFFSET / first row: [More...](#)

#define **AddrROIheight** 403

Register address for the height of the ROI stripe [More...](#)

#define **AddrFrameCounter** 63

Register address for 16bit framecounter [More...](#)

#define **AddrFramePackingMode_ImageCountPerBlock** 64

Register address for frame packing mode: number of images per block [More...](#)

#define **AddrFramePackingMode_BlockHeightInRows** 65

Register address for frame packing mode: number of images per block [More...](#)

#define **AddrBuildDayAndMonth** 38

Register address storing day and month of the current firmware build [More...](#)

#define **AddrBuildYear** 39

Register address storing the year of the current firmware built [More...](#)

#define **AddrBuildTime** 40

Register address storing the time of the current firmware built [More...](#)

#define **AddrReadOutTimeLow** 55

Register address storing the first 16bit of detector read-out time will be retrieved in ticks (1 tick = ReadOutTimeBase1313 ns) [More...](#)

#define **AddrReadOutTimeHigh** 56

Register address storing the second 16bit of detector read-out time will be retrieved in ticks (1 tick = ReadOutTimeBase1313 ns) [More...](#)

#define **AddrControlReg1** 1

Address of control register 1 [More...](#)

#define **AddrFeaturesReg0** 36

Address of features register 0 [More...](#)

```
#define AddrFeaturesReg1 37
Address of features register 1 More...

#define AVGERAGED_FLAG 1

#define FIXED_FLAG 2

#define LINEARIZED_FLAG 4

#define SORTED_FLAG 8

#define CLEAR_SORTED_FLAG 0xFFFF7

#define OPERATION_KNOWN_FLAG 0x8000

#define CLEAR_OPERATION_KNOWN_FLAG 0x7FFF

#define NOOP_FLAG 0x0

#define XIS_OFFSET_CORRECTED_FLAG 1

#define XIS_GAIN_CORRECTED_FLAG 2

#define XIS_DEFECT_CORRECTED_FLAG 4

#define XIS_MULTIGAIN_CORRECTED 8 /*this is not currently
used appart from in XIS*/

#define DEX_OFFSET_CORRECTED_FLAG 16 /*Dexela versions
of the corrections*/

#define DEX_GAIN_CORRECTED_FLAG 32

#define DEX_DEFECT_CORRECTED_FLAG 64

DEX_EXTRA_PARAMS_FLAG 0x4000 /*this flag will
```

```
#define indicate the presence of new parameters (e.g. model,  
binning, operations) in the HIS header*/  
  
#define CORRECTION_KNOWN_FLAG 0x8000  
  
#define UNCORRECTED_FLAG 0x0  
  
#define TIFFTAG_DEX_CORRECTION_FLAGS 34595 /* New tiff-  
tag for storing correction flags parameter */  
  
#define TIFFTAG_DEX_OPERATION_FLAGS 34596 /* New tiff-tag  
for storing operation flags parameter */  
  
#define TIFFTAG_DEX_IMAGE_TYPE 34597 /* New tiff-tag for  
storing image-type parameter */  
  
#define DEX_DATA_IMAGE 0 /* regular data image */  
  
#define DEX_OFFSET_IMAGE 1 /* offset data image */  
  
#define DEX_GAIN_IMAGE 2 /* gain data image */  
  
#define DEX_DEFECT_MAP 3 /* defect map image */  
  
#define DEX_UNKONWN_TYPE_IMAGE 0xFF /* type of image is  
unknown */  
  
#define TIFFTAG_DEX_MODEL_NUM 34598 /* New tiff-tag for  
storing image-type parameter */  
  
#define TIFFTAG_DEX_BIN_FMT 34599  
  
#define TIFFTAG_ROI_START_COL 34600  
  
#define TIFFTAG_ROI_START_ROW 34601  
  
#define TIFFTAG_DEFECT_FLAGS 34602
```

```
#define MAX_REG_ADDR 999
```

```
#define MAX_REG_VALUE 0xFFFF
```

Typedefs

```
typedef unsigned short ushort
```

```
typedef unsigned long ulong
```

```
typedef unsigned char byte
```

Macro Definition Documentation

#define AddrBuildDayAndMonth 38

Register address storing day and month of the current firmware build

#define AddrBuildTime 40

Register address storing the time of the current firmware built

#define AddrBuildYear 39

Register address storing the year of the current firmware built

#define AddrControlReg 0

Address of control register

#define AddrControlReg1 1

Address of control register 1

#define AddrExposureTime 12

Register address of exposure time information in low res system

#define AddrExposureTime2 13

Register address of exposure time information in low res system for Line_Delay mode

#define AddrExposureTime2High 14

Register address of high bytes of exposure time information for line delay mode

#define AddrExposureTime2Low 13

Register address of low bytes of exposure time information for line delay mode

#define AddrFeaturesReg0 36

Address of features register 0

#define AddrFeaturesReg1 37

Address of features register 1

#define AddrFirmwareVersion 127

Register address of firmware verion information

#define AddrFPGANumber 126

Register address for the FPGA version number of the detector

REGISTER CONSTANTS//////////

#define AddrFrameCounter 63

Register address for 16bit framecounter

#define AddrFramePackingMode_BlockHeightInRows 65

Register address for frame packing mode: number of images per block

#define AddrFramePackingMode_ImageCountPerBlock 64

Register address for frame packing mode: number of images per block

#define AddrGapTime 18

Register address for the gap time used in Frame Rate mode

#define AddrHorizontalBinReg 10

Address of horizontal binning register

#define AddrModelNumber 124

Register address for the model number of the detector

#define AddrNumberOfFrames 17

Register address of number of frames register for use with sequence modes

#define AddrNumLines 7

Address of Numer of lines per sensor register

#define AddrNumPixels 8

Address of Number of pixels (per line) per sensor register

#define AddrPPExposreTime1High 28

Register address of high bytes of pre-programmed exposure time 1 information

#define AddrPPExposreTime1Low 27

Register address of low bytes of pre-programmed exposure time 1 information

#define AddrPPExposreTime2High 30

Register address of high bytes of pre-programmed exposure time 2 information

#define AddrPPExposreTime2Low 29

Register address of low bytes of pre-programmed exposure time 2 information

#define AddrPPExposreTime3High 32

Register address of high bytes of pre-programmed exposure time 3 information

#define AddrPPExposreTime3Low 31

Register address of low bytes of pre-programmed exposure time 3 information

#define AddrPPExposreTime4High 34

Register address of high bytes of pre-programmed exposure time 4 information

#define AddrPPExposreTime4Low 33

Register address of low bytes of pre-programmed exposure time 4 information

#define AddrReadOutTime 410

Readout time for the sensor, dynamically adapted to the read out mode (binning, ROI). Dependent on the implementation the value has to be multiplied by the factor ReadoutTimeFactor

#define AddrReadOutTimeHigh 56

Register address storing the second 16bit of detector read-out time will be retrieved in ticks (1 tick = ReadOutTimeBase1313 ns)

#define AddrReadOutTimeLow 55

Register address storing the first 16bit of detector read-out time will be retrieved in ticks (1 tick = ReadOutTimeBase1313 ns)

#define AddrROIheight 403

Register address for the height of the ROI stripe

#define AddrROIStartColumn 404

Register address for the ROI OFFSET / first column:

#define AddrROIStartRow 402

Register address for the ROI OFFSET / first row:

#define AddrROIwidth 405

Register address for the width of the ROI stripe

#define AddrSensorBinReg 3

Address of sensor bin register

#define AddrSensorBinReg2 5

Address of sensor bin register 2

#define AddrSerialNumber 125

Register address for the serial number of the detector

#define AddrTriggerSource 0

Register address of the Trigger Source bits

#define AddrVerticalBinReg 9

Address of vertical binning register

#define AddrWellHigh 4

Address of High Fullwell register

#define AddrWellLow 65531

Address of Low Fullwell register

#define AddrWellReg 3

Address of Well register

#define BinCommit 514

Constant for binning commit command

#define DarkPixelXOffset 2

The offset in pixels in x when reading dark pixel data

d

#define DarkPixelYOffset 4

The offset in pixels in y when reading dark pixel data

#define DigitalBinBit 65533

Constant for digital binning bit

#define ExposureSleepTimems 10

Variable to hold a time in ms used for sleeping threads in streaming mode.

#define MAX_PIXEL_VAL 16383

Maximum value for any pixel in detector output (14 bit)

#define MIN_PIXEL_VAL 0

Minimum allowable pixel value

#define minTimeIncrement 0.01F

The minimum time increment in ms for expose and read mode

#define minTimeIncrement2 195.2F

The minimum time increment in ns for line delay mode

```
#define ReadoutTimeFactor1313 2
```

ReadoutTimeFactor for Dexela 1313

```
#define Sensor1x1 0
```

Constant for 1x1 binning command

```
#define Sensor1x2 0
```

Constant for 1x2 binning command

```
#define Sensor1x4 64
```

Constant for 1x4 binning command

```
#define Sensor2x1 128
```

Constant for 2x1 binning command

```
#define Sensor2x2 128
```

Constant for 2x2 binning command

```
#define Sensor2x4 192
```

Constant for 2x4 binning command

#define Sensor4x1 256

Constant for 4x1 binning command

#define Sensor4x2 256

Constant for 4x2 binning command

#define Sensor4x4 320

Constant for 4x4 binning command

#define SensorBinClear 65087

Constant for clearing binning command

#define SerialNumberReg1 0

Address of serial number register 1

#define SerialNumberReg2 0

Address of serial number register 2

#define SerialNumberReg3 0

Address of serial number register 3

#define TemperatureReg 0

Address of temperature register

#define TimingResolution 100

The resolution of the exposure time settings 1 for ms 100 for 0.01
ms

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page	Classes	Files
File List	File Members	
DexelsDetector	DexelsException	

Classes | Enumerations

DexDefs.h File Reference

```
#include "DexDefines.h" #include "windows.h"
```

Go to the source code of this file.

Classes

struct **DevInfo**

A structure to hold device information. [More...](#)

struct **GeometryCorrectionParams**

A structure used to specify the new image dimensions for geometry correction [More...](#)

struct **DetStatus**

Structure to hold the detector current status. [More...](#)

Enumerations

enum **DetectorInterface** { **CL**, **GIGE** }

An enumeration of detector interface types. [More...](#)

enum **TransportLib** { **Pleora**, **Epix** }

An enumeration of detector interface types. [More...](#)

enum **bins** {
 x11 = 1, **x12**, **x14**, **x21**,
 x22, **x24**, **x41**, **x42**,
 x44, **ix22**, **binsError**
}

An enumeration of the different bin levels available [More...](#)

enum **FileType** { **SMV**, **TIF**, **HIS**, **UNKNOWN** }

An enumeration of file types [More...](#)

enum **Derr** {
 SUCCESS, **NULL_IMAGE**, **WRONG_TYPE**,
WRONG_DIMS,
 BAD_PARAM, **BAD_COMMs**, **BAD_TRIGGER**,
BAD_COMMs_OPEN,
 BAD_COMMs_WRITE, **BAD_COMMs_READ**,
BAD_FILE_IO, **BAD_BOARD**,
 OUT_OF_MEMORY, **EXPOSURE_FAILED**,
BAD_BIN_LEVEL
}

Enumeration for error codes returned from the API functions
[More...](#)

enum **FullWellModes** { **Low** =0, **High**, **FullWellModesError** }

An enumeration of the available full well modes [More...](#)

enum **pType** { **u16** = 2, **flt** = 4, **u32** = 6 }

Enumeration of pixel types [More...](#)

enum **ExposureModes** { **Expose_and_read**,
Sequence_Exposure, **Frame_Rate_exposure**,
Preprogrammed_exposure }
An enumeration of exposure modes. [More...](#)

enum **ExposureTriggerSource** { **Ext_neg_edge_trig**,
Internal_Software, **Ext_Duration_Trig** }
An enumeration of exposure trigger sources. [More...](#)

enum **ReadoutModes** { **ContinuousReadout**, **IdleMode**,
ReadoutModeError }
An enumeration of ReadOut modes. [More...](#)

enum **ResolutionModes** { **pixelsize50micron** = 1,
pixelsize100micron = 0, **ResolutionModesError** }
An enumeration of the available resolution modes [More...](#)

enum **DexImageTypes** {
 Data = 0, **Offset** = 1, **Gain** = 2, **Defect** = 3,
 UnknownType = 0xFF
}
An enumeration of the different image types. [More...](#)

Enumeration Type Documentation

enum bins

An enumeration of the different bin levels available

Enumerator	
x11	Unbinned
x12	Binned vertically by 2
x14	Binned vertically by 4
x21	Binned horizontally by 2
x22	Binned horizontally by 2 and vertically by 2
x24	Binned horizontally by 2 and vertically by 4
x41	Binned horizontally by 4
x42	Binned horizontally by 4 and vertically by 2
x44	Binned horizontally by 4 and vertically by 4
ix22	Digital 2x2 binning
binsError	Indicates an error

enum Derr

Enumeration for error codes returned from the API functions

Enumerator	
SUCCESS	The operation was successful
NULL_IMAGE	The image pointer was NULL
WRONG_TYPE	The image pixel type was wrong for the operation requested
WRONG_DIMS	The image dimensions were wrong for the operation requested

	operation requested
BAD_PARAM	One or more parameters were incorrect
BAD_COMMS	The communications channel is not open or could not be opened
BAD_TRIGGER	An invalid trigger source was requested
BAD_COMMS_OPEN	The communications channel failed to open
BAD_COMMS_WRITE	A failure in a detector write command occurred
BAD_COMMS_READ	A failure in a detector read command occurred
BAD_FILE_IO	An error occurred opening or reading from a file
BAD_BOARD	The software failed to open the PC driver or frame grabber
OUT_OF_MEMORY	A function call was not able to reserve the memory it required
EXPOSURE_FAILED	Exposure Acquisition failed
BAD_BIN_LEVEL	Incorrect bin level specified

enum DetectorInterface

An enumeration of detector interface types.

Enumerator	
CL	CameraLink
GIGE	Gigabit Ethernet

enum DexImageTypes

An enumeration of the different image types.

Enumerator

Data	A data image
Offset	An offset (dark) image
Gain	An gain (flood) image
Defect	A defect-map image
UnknownType	The type of the image is not known

enum ExposureModes

An enumeration of exposure modes.

Enumerator	
Expose_and_read	The detector should clear the sensor and wait for exposure time to pass before reading the detector image.
Sequence_Exposure	The detector should take a sequence of images with no gaps.
Frame_Rate_exposure	The detector should take a sequence of images with a specified gap no less than the minimum exposure time for the bin level.
Preprogrammed_exposure	The detector should take a number of images with preset exposure times without a gap..

enum ExposureTriggerSource

An enumeration of exposure trigger sources.

Enumerator	
Ext_neg_edge_trig	Trigger on negative edge
Internal_Software	Trigger using software
Ext_Duration_Trig	Detector exposure duration and trigger controlled externally

enum FileType

An enumeration of file types

Enumerator	
SMV	SMV
TIF	TIFF
HIS	HIS
UNKNOWN	Unknown/Unsupported

enum FullWellModes

An enumeration of the available full well modes

Enumerator	
Low	The low noise reduced dynamic range mode
High	The normal full well mode
FullWellModesError	Indicates an error

enum pType

Enumeration of pixel types

Enumerator	
u16	A pixel type of 16-bit unsigned short
flt	A pixel type of 32-bit floating point
u32	A pixel type of 32-bit unsigned int </summary>

enum ReadoutModes

An enumeration of ReadOut modes.

Enumerator	
ContinuousReadout	The sensor is continuously read-out using the minimum read-out time. On request an image will be transmitted. A frame request can be an external trigger pulse, internal trigger or software trigger
IdleMode	The sensor is only read out (using the minimum frame time) on request. The read-out will be followed by the transmission of the image. A frame request can be an external trigger pulse, internal trigger or software trigger
ReadoutModeError	Indicates an error

enum ResolutionModes

An enumeration of the available resolution modes

Enumerator	
pixelsize50micron	The 100micron mode
pixelsize100micron	The 50micron mode
ResolutionModesError	Indicates an error

enum TransportLib

An enumeration of detector interface types.

Enumerator	
Pleora	Pleora type GigE transport library
Epix	Epix FG CameraLink transport library

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page	Classes	Files
File List	File Members	
DexelsDetector	DexelsDetector	Classes Typedefs

DexelsDetector.h File Reference

```
#include <vector> #include "DexDefs.h"  
#include "DexImage.h"  
#include "DexelsException.h"  
#include <boost/shared_ptr.hpp>
```

Go to the source code of this file.

Classes

class **DexelaDetector**

This class is used to control any interface-type Detector and acquire images from it. It will provide all the basic functionality required for all different Dexela detectors. For interface specific functionality please see the interface specific classes (e.g. **DexelaDetectorGE**, **DexelaDetectorCL**). More...

Typedefs

```
typedef void(* IMAGE_CALLBACK )(int fc, int buf, DexelaDetector  
*det)
```

Image callback function signature. This is the
signature that any user-passed callback functions
must adhere to. [More...](#)

Typedef Documentation

typedef void(* IMAGE_CALLBACK)(int fc, int buf, DexelaDetector *det)

Image callback function signature. This is the signature that any user-passed callback functions must adhere to.

Parameters

- fc** The field count associated with the image that just arrived
- buf** The buffer number where the image was written to (this can be used by [DexelaDetector::ReadBuffer\(\)](#) to read-out the image data).
- det** The [DexelaDetector](#) object that sent the image.

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > DexelaDetectorCL.h

Classes

DexelaDetectorCL.h File Reference

```
#include "dexeladetector.h"
```

Go to the source code of this file.

Classes

class **DexelsDetectorCL**

This class is used to control CameraLink Type Detectors. It will give access to functions that are not available to other interface-type detectors.

Note: For all standard detector function calls please see the **DexelsDetector** class (these functions are also available to **DexelsDetectorCL** objects) [More...](#)

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > DexelaDetectorGE.h

Classes

DexelaDetectorGE.h File Reference

```
#include "DexelaDetector.h"
```

Go to the source code of this file.

Classes

class **DexelaDetectorGE**

This class is used to control GigE Type Detectors. It will give access to functions that are not available to other interface-type detectors.

Note: For all standard detector function calls please see the **DexelaDetector** class (these functions are also available to **DexelaDetectorGE** objects) [More...](#)

DexelaDetector API

Main Page	Classes	Files
File List	File Members	
DexelaDetector	DexelaException	

[Classes | Macros](#)

DexelaException.h File Reference

```
#include "dexdefs.h" #include <exception>
```

[Go to the source code of this file.](#)

Classes

class **DexelsException**

This class contains information about any possible error's in the API. In the event of a problem a **DexelsException** will be thrown.

Note: It is suggested that you wrap your code in a try-catch block to ensure that if any errors occur you can detect (and properly handle them) in your code. [More...](#)

Macros

```
#define rethrowEr(EX) throw DexelaException(EX,__FUNCTION__  
  
#define DexelaException(MSG, CODE, TRANSER, TRANSMMSG) throw  
throwNewEr(MSG, CODE, TRANSER, TRANSMMSG);\\  
#define DexelaException(MSG, CODE, __LINE__, __FILE__, __FUNC__  
TRANSMMSG);\\
```

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by [doxygen](#) 1.8.7

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > DexImage

Classes

DexImage.h File Reference

```
#include "DexDefs.h" #include "DexImage.h"  
#include <vector>  
#include <boost/shared_ptr.hpp>
```

[Go to the source code of this file.](#)

Classes

class **DexImage**

This class is used to store and handle the images acquired from a detector. [More...](#)

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page Classes **Files**

File List File Members

DexelsDetector > DexImage

DexImage/resource.h

```
1 //{{NO_DEPENDENCIES}}
2 // Microsoft Visual C++ generated include
file.
3 // Used by DexImageResources.rc
4
5 // Next default values for new objects
6 //
7 #ifdef APSTUDIO_INVOKED
8 #ifndef APSTUDIO_READONLY_SYMBOLS
9 #define _APS_NEXT_RESOURCE_VALUE      101
10 #define _APS_NEXT_COMMAND_VALUE       40001
11 #define _APS_NEXT_CONTROL_VALUE       1001
12 #define _APS_NEXT_SYMED_VALUE        101
13 #endif
14 #endif
```

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > BusScanner

BusScanner/resource.h

```
1 //{{NO_DEPENDENCIES}}
2 // Microsoft Visual C++ generated include
file.
3 // Used by BusScannerResources.rc
4
5 // Next default values for new objects
6 //
7 #ifdef APSTUDIO_INVOKED
8 #ifndef APSTUDIO_READONLY_SYMBOLS
9 #define _APS_NEXT_RESOURCE_VALUE      101
10 #define _APS_NEXT_COMMAND_VALUE       40001
11 #define _APS_NEXT_CONTROL_VALUE       1001
12 #define _APS_NEXT_SYMED_VALUE        101
13 #endif
14 #endif
```

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > DexelaException

DexelaException/resource.h

```
1 //{{NO_DEPENDENCIES}}
2 // Microsoft Visual C++ generated include
file.
3 // Used by DexelaExceptionResources.rc
4
5 // Next default values for new objects
6 //
7 #ifdef APSTUDIO_INVOKED
8 #ifndef APSTUDIO_READONLY_SYMBOLS
9 #define _APS_NEXT_RESOURCE_VALUE      101
10 #define _APS_NEXT_COMMAND_VALUE       40001
11 #define _APS_NEXT_CONTROL_VALUE       1001
12 #define _APS_NEXT_SYMED_VALUE        101
13 #endif
14 #endif
```

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > DexelaDetector

resource1.h

```
1 //{{NO_DEPENDENCIES}}
2 // Microsoft Visual C++ generated include
file.
3 // Used by DexelaDetectorResources.rc
4
5 // Next default values for new objects
6 //
7 #ifdef APSTUDIO_INVOKED
8 #ifndef APSTUDIO_READONLY_SYMBOLS
9 #define _APS_NEXT_RESOURCE_VALUE      101
10 #define _APS_NEXT_COMMAND_VALUE       40001
11 #define _APS_NEXT_CONTROL_VALUE       1001
12 #define _APS_NEXT_SYMED_VALUE        101
13 #endif
14 #endif
```

DexelaDetector API

The screenshot shows a navigation bar for a software API. At the top, there are three main tabs: "Main Page", "Classes", and "Files". The "Files" tab is currently selected and highlighted in blue. Below these, there are two sub-tabs: "File List" and "File Members", with "File Members" also being highlighted in blue. Further down, there are five categories: "All", "Typedefs", "Enumerations", "Enumerator", and "Macros". A large horizontal menu below these categories contains letters from 'a' to 'x', with each letter in its own box. The letter 'x' is highlighted in a dark blue box, indicating it is the current selection.

Here is a list of all documented file members with links to the documentation:

- a -

- AddrBuildDayAndMonth : [DexDefines.h](#)
- AddrBuildTime : [DexDefines.h](#)
- AddrBuildYear : [DexDefines.h](#)
- AddrControlReg : [DexDefines.h](#)
- AddrControlReg1 : [DexDefines.h](#)
- AddrExposureTime : [DexDefines.h](#)
- AddrExposureTime2 : [DexDefines.h](#)
- AddrExposureTime2High : [DexDefines.h](#)
- AddrExposureTime2Low : [DexDefines.h](#)
- AddrFeaturesReg0 : [DexDefines.h](#)
- AddrFeaturesReg1 : [DexDefines.h](#)
- AddrFirmwareVersion : [DexDefines.h](#)
- AddrFPGANumber : [DexDefines.h](#)
- AddrFrameCounter : [DexDefines.h](#)
- AddrFramePackingMode_BlockHeightInRows : [DexDefines.h](#)
- AddrFramePackingMode_ImageCountPerBlock : [DexDefines.h](#)
- AddrGapTime : [DexDefines.h](#)
- AddrHorizontalBinReg : [DexDefines.h](#)
- AddrModelNumber : [DexDefines.h](#)
- AddrNumberOfFrames : [DexDefines.h](#)
- AddrNumLines : [DexDefines.h](#)
- AddrNumPixels : [DexDefines.h](#)
- AddrPPExposreTime1High : [DexDefines.h](#)
- AddrPPExposreTime1Low : [DexDefines.h](#)
- AddrPPExposreTime2High : [DexDefines.h](#)

- AddrPPExposreTime2Low : **DexDefines.h**
- AddrPPExposreTime3High : **DexDefines.h**
- AddrPPExposreTime3Low : **DexDefines.h**
- AddrPPExposreTime4High : **DexDefines.h**
- AddrPPExposreTime4Low : **DexDefines.h**
- AddrReadOutTime : **DexDefines.h**
- AddrReadOutTimeHigh : **DexDefines.h**
- AddrReadOutTimeLow : **DexDefines.h**
- AddrROIheight : **DexDefines.h**
- AddrROIStartColumn : **DexDefines.h**
- AddrROIStartRow : **DexDefines.h**
- AddrROIwidth : **DexDefines.h**
- AddrSensorBinReg : **DexDefines.h**
- AddrSensorBinReg2 : **DexDefines.h**
- AddrSerialNumber : **DexDefines.h**
- AddrTriggerSource : **DexDefines.h**
- AddrVerticalBinReg : **DexDefines.h**
- AddrWellHigh : **DexDefines.h**
- AddrWellLow : **DexDefines.h**
- AddrWellReg : **DexDefines.h**

- b -

- BAD_BIN_LEVEL : **DexDefs.h**
- BAD_BOARD : **DexDefs.h**
- BAD_COMMs : **DexDefs.h**
- BAD_COMMs_OPEN : **DexDefs.h**
- BAD_COMMs_READ : **DexDefs.h**
- BAD_COMMs_WRITE : **DexDefs.h**
- BAD_FILE_IO : **DexDefs.h**
- BAD_PARAM : **DexDefs.h**
- BAD_TRIGGER : **DexDefs.h**
- BinCommit : **DexDefines.h**
- bins : **DexDefs.h**
- binsError : **DexDefs.h**

- C -

- CL : **DexDefs.h**
- ContinuousReadout : **DexDefs.h**

- d -

- DarkPixelIXOffset : **DexDefines.h**
- DarkPixelYOffset : **DexDefines.h**
- Data : **DexDefs.h**
- Defect : **DexDefs.h**
- Derr : **DexDefs.h**
- DetectorInterface : **DexDefs.h**
- DexImageTypes : **DexDefs.h**
- DigitalBinBit : **DexDefines.h**

- e -

- Epix : **DexDefs.h**
- Expose_and_read : **DexDefs.h**
- EXPOSURE_FAILED : **DexDefs.h**
- ExposureModes : **DexDefs.h**
- ExposureSleepTimems : **DexDefines.h**
- ExposureTriggerSource : **DexDefs.h**
- Ext_Duration_Trig : **DexDefs.h**
- Ext_neg_edge_trig : **DexDefs.h**

- f -

- FileType : **DexDefs.h**
- flt : **DexDefs.h**
- Frame_Rate_exposure : **DexDefs.h**
- FullWellModes : **DexDefs.h**
- FullWellModesError : **DexDefs.h**

- g -

- Gain : **DexDefs.h**
- GIGE : **DexDefs.h**

- h -

- High : **DexDefs.h**
- HIS : **DexDefs.h**

- i -

- IdleMode : **DexDefs.h**
- IMAGE_CALLBACK : **DexelsDetector.h**
- Internal_Software : **DexDefs.h**
- ix22 : **DexDefs.h**

- l -

- Low : **DexDefs.h**

- m -

- MAX_PIXEL_VAL : **DexDefines.h**
- MIN_PIXEL_VAL : **DexDefines.h**
- minTimeIncrement : **DexDefines.h**
- minTimeIncrement2 : **DexDefines.h**

- n -

- NULL_IMAGE : **DexDefs.h**

- o -

- Offset : **DexDefs.h**
- OUT_OF_MEMORY : **DexDefs.h**

- p -

- pixelsize100micron : **DexDefs.h**
- pixelsize50micron : **DexDefs.h**
- Pleora : **DexDefs.h**
- Preprogrammed_exposure : **DexDefs.h**
- pType : **DexDefs.h**

- r -

- ReadoutModeError : **DexDefs.h**
- ReadoutModes : **DexDefs.h**

- ReadoutTimeFactor1313 : **DexDefines.h**
- ResolutionModes : **DexDefs.h**
- ResolutionModesError : **DexDefs.h**

- S -

- Sensor1x1 : **DexDefines.h**
- Sensor1x2 : **DexDefines.h**
- Sensor1x4 : **DexDefines.h**
- Sensor2x1 : **DexDefines.h**
- Sensor2x2 : **DexDefines.h**
- Sensor2x4 : **DexDefines.h**
- Sensor4x1 : **DexDefines.h**
- Sensor4x2 : **DexDefines.h**
- Sensor4x4 : **DexDefines.h**
- SensorBinClear : **DexDefines.h**
- Sequence_Exposure : **DexDefs.h**
- SerialNumberReg1 : **DexDefines.h**
- SerialNumberReg2 : **DexDefines.h**
- SerialNumberReg3 : **DexDefines.h**
- SMV : **DexDefs.h**
- SUCCESS : **DexDefs.h**

- t -

- TemperatureReg : **DexDefines.h**
- TIF : **DexDefs.h**
- TimingResolution : **DexDefines.h**
- TransportLib : **DexDefs.h**

- u -

- u16 : **DexDefs.h**
- u32 : **DexDefs.h**
- UNKNOWN : **DexDefs.h**
- UnknownType : **DexDefs.h**

- w -

- **WRONG_DIMS** : **DexDefs.h**
- **WRONG_TYPE** : **DexDefs.h**

- X -

- x11 : **DexDefs.h**
- x12 : **DexDefs.h**
- x14 : **DexDefs.h**
- x21 : **DexDefs.h**
- x22 : **DexDefs.h**
- x24 : **DexDefs.h**
- x41 : **DexDefs.h**
- x42 : **DexDefs.h**
- x44 : **DexDefs.h**

DexelsDetector API

Main Page	Classes	Files			
File List	File Members				
All	TypeDefs	Enumerations	Enumerator	Macros	

- IMAGE_CALLBACK : DexelaDetector.h

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

Main Page	Classes	Files			
File List	File Members				
All	Typedefs	Enumerations	Enumerator	Macros	

- bins : [DexDefs.h](#)
- Derr : [DexDefs.h](#)
- DetectorInterface : [DexDefs.h](#)
- DexImageTypes : [DexDefs.h](#)
- ExposureModes : [DexDefs.h](#)
- ExposureTriggerSource : [DexDefs.h](#)
- FileType : [DexDefs.h](#)
- FullWellModes : [DexDefs.h](#)
- pType : [DexDefs.h](#)
- ReadoutModes : [DexDefs.h](#)
- ResolutionModes : [DexDefs.h](#)
- TransportLib : [DexDefs.h](#)

DexelaDetector API

Main Page	Classes	Files		
File List	File Members			
All	Typedefs	Enumerations	Enumerator	Macros
b c d e f g h i l n o p r s t u w x				

- b -

- BAD_BIN_LEVEL : [DexDefs.h](#)
- BAD_BOARD : [DexDefs.h](#)
- BAD_COMMs : [DexDefs.h](#)
- BAD_COMMs_OPEN : [DexDefs.h](#)
- BAD_COMMs_READ : [DexDefs.h](#)
- BAD_COMMs_WRITE : [DexDefs.h](#)
- BAD_FILE_IO : [DexDefs.h](#)
- BAD_PARAM : [DexDefs.h](#)
- BAD_TRIGGER : [DexDefs.h](#)
- binsError : [DexDefs.h](#)

- c -

- CL : [DexDefs.h](#)
- ContinuousReadout : [DexDefs.h](#)

- d -

- Data : [DexDefs.h](#)
- Defect : [DexDefs.h](#)

- e -

- Epix : [DexDefs.h](#)
- Expose_and_read : [DexDefs.h](#)
- EXPOSURE_FAILED : [DexDefs.h](#)

- Ext_Duration_Trig : **DexDefs.h**
- Ext_neg_edge_trig : **DexDefs.h**

- f -

- flt : **DexDefs.h**
- Frame_Rate_exposure : **DexDefs.h**
- FullWellModesError : **DexDefs.h**

- g -

- Gain : **DexDefs.h**
- GigE : **DexDefs.h**

- h -

- High : **DexDefs.h**
- HIS : **DexDefs.h**

- i -

- IdleMode : **DexDefs.h**
- Internal_Software : **DexDefs.h**
- ix22 : **DexDefs.h**

- l -

- Low : **DexDefs.h**

- n -

- NULL_IMAGE : **DexDefs.h**

- o -

- Offset : **DexDefs.h**
- OUT_OF_MEMORY : **DexDefs.h**

- p -

- pixelsize100micron : DexDefs.h
- pixelsize50micron : DexDefs.h
- Pleora : DexDefs.h
- Preprogrammed_exposure : DexDefs.h

- r -

- ReadoutModeError : DexDefs.h
- ResolutionModesError : DexDefs.h

- s -

- Sequence_Exposure : DexDefs.h
- SMV : DexDefs.h
- SUCCESS : DexDefs.h

- t -

- TIF : DexDefs.h

- u -

- u16 : DexDefs.h
- u32 : DexDefs.h
- UNKNOWN : DexDefs.h
- UnknownType : DexDefs.h

- w -

- WRONG_DIMS : DexDefs.h
- WRONG_TYPE : DexDefs.h

- x -

- x11 : DexDefs.h
- x12 : DexDefs.h
- x14 : DexDefs.h

- x21 : [DexDefs.h](#)
 - x22 : [DexDefs.h](#)
 - x24 : [DexDefs.h](#)
 - x41 : [DexDefs.h](#)
 - x42 : [DexDefs.h](#)
 - x44 : [DexDefs.h](#)
-

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by [doxygen](#) 1.8.7

DexelsDetector API

The screenshot shows a navigation bar for a software API. At the top, there are three main tabs: "Main Page", "Classes", and "Files". The "Files" tab is currently selected and highlighted in blue. Below these, there are two sub-tabs: "File List" and "File Members", with "File Members" being the active one. Further down, there are four more tabs: "All", "Typedefs", "Enumerations", and "Enumerator", followed by a "Macros" tab. At the bottom of the bar, there is a horizontal menu with letters "a", "b", "d", "e", "m", "r", "s", and "t", where "s" is the active selection.

- a -

- AddrBuildDayAndMonth : [DexDefines.h](#)
- AddrBuildTime : [DexDefines.h](#)
- AddrBuildYear : [DexDefines.h](#)
- AddrControlReg : [DexDefines.h](#)
- AddrControlReg1 : [DexDefines.h](#)
- AddrExposureTime : [DexDefines.h](#)
- AddrExposureTime2 : [DexDefines.h](#)
- AddrExposureTime2High : [DexDefines.h](#)
- AddrExposureTime2Low : [DexDefines.h](#)
- AddrFeaturesReg0 : [DexDefines.h](#)
- AddrFeaturesReg1 : [DexDefines.h](#)
- AddrFirmwareVersion : [DexDefines.h](#)
- AddrFPGANumber : [DexDefines.h](#)
- AddrFrameCounter : [DexDefines.h](#)
- AddrFramePackingMode_BlockHeightInRows : [DexDefines.h](#)
- AddrFramePackingMode_ImageCountPerBlock : [DexDefines.h](#)
- AddrGapTime : [DexDefines.h](#)
- AddrHorizontalBinReg : [DexDefines.h](#)
- AddrModelNumber : [DexDefines.h](#)
- AddrNumberOfFrames : [DexDefines.h](#)
- AddrNumLines : [DexDefines.h](#)
- AddrNumPixels : [DexDefines.h](#)
- AddrPPExposreTime1High : [DexDefines.h](#)
- AddrPPExposreTime1Low : [DexDefines.h](#)
- AddrPPExposreTime2High : [DexDefines.h](#)
- AddrPPExposreTime2Low : [DexDefines.h](#)

- AddrPPExposreTime3High : **DexDefines.h**
- AddrPPExposreTime3Low : **DexDefines.h**
- AddrPPExposreTime4High : **DexDefines.h**
- AddrPPExposreTime4Low : **DexDefines.h**
- AddrReadOutTime : **DexDefines.h**
- AddrReadOutTimeHigh : **DexDefines.h**
- AddrReadOutTimeLow : **DexDefines.h**
- AddrROIheight : **DexDefines.h**
- AddrROIStartColumn : **DexDefines.h**
- AddrROIStartRow : **DexDefines.h**
- AddrROIwidth : **DexDefines.h**
- AddrSensorBinReg : **DexDefines.h**
- AddrSensorBinReg2 : **DexDefines.h**
- AddrSerialNumber : **DexDefines.h**
- AddrTriggerSource : **DexDefines.h**
- AddrVerticalBinReg : **DexDefines.h**
- AddrWellHigh : **DexDefines.h**
- AddrWellLow : **DexDefines.h**
- AddrWellReg : **DexDefines.h**

- b -

- BinCommit : **DexDefines.h**

- d -

- DarkPixelIXOffset : **DexDefines.h**
- DarkPixelYOffset : **DexDefines.h**
- DigitalBinBit : **DexDefines.h**

- e -

- ExposureSleepTimems : **DexDefines.h**

- m -

- MAX_PIXEL_VAL : **DexDefines.h**
- MIN_PIXEL_VAL : **DexDefines.h**
- minTimeIncrement : **DexDefines.h**

- minTimeIncrement2 : [DexDefines.h](#)

- r -

- ReadoutTimeFactor1313 : [DexDefines.h](#)

- s -

- Sensor1x1 : [DexDefines.h](#)
- Sensor1x2 : [DexDefines.h](#)
- Sensor1x4 : [DexDefines.h](#)
- Sensor2x1 : [DexDefines.h](#)
- Sensor2x2 : [DexDefines.h](#)
- Sensor2x4 : [DexDefines.h](#)
- Sensor4x1 : [DexDefines.h](#)
- Sensor4x2 : [DexDefines.h](#)
- Sensor4x4 : [DexDefines.h](#)
- SensorBinClear : [DexDefines.h](#)
- SerialNumberReg1 : [DexDefines.h](#)
- SerialNumberReg2 : [DexDefines.h](#)
- SerialNumberReg3 : [DexDefines.h](#)

- t -

- TemperatureReg : [DexDefines.h](#)
- TimingResolution : [DexDefines.h](#)

DexelsDetector API

Main Page	Classes	Files
Class List	Class Index	Class Hierarchy

BusScanner Member List

This is the complete list of members for **BusScanner**, including all inherited members.

BusScanner(void)	BusScanner
EnumerateCLDevices()	BusScanner
EnumerateDevices()	BusScanner
EnumerateGEDevices()	BusScanner
GetDevice(int index)	BusScanner
GetDeviceCL(int index)	BusScanner
GetDeviceGE(int index)	BusScanner
ScanMockSetter (defined in BusScanner)	BusScanner friend
~BusScanner(void)	BusScanner

DexelaDetector API

Main Page	Classes	Files
File List	File Members	
DexelaDetector	BusScanner	

BusScanner.h

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef DEX_BUILD
4 #ifdef _DEBUG
5 #pragma comment(lib,"BusScanner-d.lib")
6 #else
7 #pragma comment(lib,"BusScanner.lib")
8 #endif
9 #endif
10
11
12 #include "DexDefs.h"
13 #include "DexelaDetector.h"
14 #include <vector>
15 #include <boost/shared_ptr.hpp>
16
17 using namespace std;
18
19
20
21
22
23 class DllExport BusScanner
24 {
25 public:
26     BusScanner(void);
27     ~BusScanner(void);
28
29     int EnumerateDevices();
30     int EnumerateGDEvices();
```

```
31     int EnumerateCLDevices();
32
33     DevInfo GetDevice(int index);
34     DevInfo GetDeviceGE(int index);
35     DevInfo GetDeviceCL(int index);
36
37     friend class ScanMockSetter;
38
39 #ifndef MOCK_TEST
40 private:
41 #endif
42     boost::shared_ptr<baseBusScanner>
43     baseScanner;
44 };
```

DexelsDetector API

Main Page	Classes	Files
Class List	Class Index	Class Hierarchy

DetStatus Member List

This is the complete list of members for **DetStatus**, including all inherited members.

binLevel	DetStatus
exposureMode	DetStatus
exposureTime	DetStatus
fullWellMode	DetStatus
testMode	DetStatus
triggerSource	DetStatus

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > DexelaException

DexDefs.h

Go to the documentation of this file.

```
1 #pragma once
2
4 #include "DexDefines.h"
5 #include "windows.h"
6
8
12 typedef enum //DetectorInterface
13 {
17     CL,
21     GIGE
22 }DetectorInterface;
23
27 typedef enum //TransportLib
28 {
32     Pleora,
36     Epix
37 }TransportLib;
38
42 typedef struct //DevInfo
43 {
47     int model;
51     int serialNum;
55     DetectorInterface iface;
59     char param[50];
63     int unit;
67     TransportLib transport;
```

```
68 }DevInfo;
69
73 typedef enum //bins
74 {
78     x11 = 1,
82     x12,
86     x14,
90     x21,
94     x22,
98     x24,
102    x41,
106    x42,
110    x44,
114    ix22,
118    binsError
119
120 }bins;
121
125 typedef enum //FileType
126 {
130     SMV,
134     TIF,
138     HIS,
142     UNKNOWN
143 }FileType;
144
145
147 typedef enum //Derr
148 {
150     SUCCESS,
152     NULL_IMAGE,
154     WRONG_TYPE,
156     WRONG_DIMS,
158     BAD_PARAM,
160     BAD_COMMs,
162     BAD_TRIGGER,
164     BAD_COMMs_OPEN,
```

```
166     BAD_COMMs_WRITE,
168     BAD_COMMs_READ,
170     BAD_FILE_IO,
172     BAD_BOARD,
174     OUT_OF_MEMORY,
176     EXPOSURE_FAILED,
178     BAD_BIN_LEVEL
179 }Derr;
180
181
185 typedef enum //FullWellModes
186 {
190     Low=0,
194     High,
198     FullWellModesError
199 }FullWellModes;
200
201
205 typedef enum //pType
206 {
210     u16 = 2,
214     flt = 4,
218     u32 = 6
219 }pType;
220
224 typedef enum //ExposureModes
225 {
229     Expose_and_read,
233     Sequence_Exposure,
237     Frame_Rate_exposure,
241     Preprogrammed_exposure
242 }ExposureModes;
243
244
248 typedef enum //ExposureTriggerSource
249 {
253     Ext_neg_edge_trig,
```

```
257     Internal_Software,
261     Ext_Duration_Trig
262 }ExposureTriggerSource;
263
267 typedef struct // GeometryCorrectionParams
268 {
272     int iRefX; //1536
276     int iRefY; //1944
277
278 }GeometryCorrectionParams;
279
283 typedef struct //DetStatus
284 {
285
289     ExposureModes exposureMode;
293     FullWellModes fullWellMode;
297     float exposureTime;
301     bins binLevel;
305     ExposureTriggerSource triggerSource;
309     BOOL testMode;
310 } DetStatus;
311
312
316 typedef enum //ReadoutModes
317 {
321     ContinuousReadout,
322
326     IdleMode,
327
331     ReadoutModeError
332 } ReadoutModes;
333
334
338 typedef enum //ResolutionModes
339 {
343     pixelsize50micron = 1,
347     pixelsize100micron = 0,
```

```
351     ResolutionModesError  
352 } ResolutionModes;  
353  
357 typedef enum //DexImageTypes  
358 {  
362     Data = 0,  
366     Offset = 1,  
370     Gain = 2,  
374     Defect = 3,  
378     UnknownType = 0xFF  
379 }DexImageTypes;  
380
```

DexelsDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members

DevInfo Member List

This is the complete list of members for [DevInfo](#), including all inherited members.

iface [DevInfo](#)

model [DevInfo](#)

param [DevInfo](#)

serialNum [DevInfo](#)

transport [DevInfo](#)

unit [DevInfo](#)

DexelsDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members

DexelsDetector Member List

This is the complete list of members for [DexelsDetector](#), including all inherited members.

[**base**](#) (defined in [DexelsDetector](#))

[**baseBusScanner**](#) (defined in [DexelsDetector](#))

[**CheckForCallbackError\(\)**](#)

[**CheckForLiveError\(\)**](#)

[**clDet**](#) (defined in [DexelsDetector](#))

[**ClearBuffers\(\)**](#)

[**ClearCameraBuffer\(int i\)**](#)

[**CloseBoard\(\)**](#)

[**Dex_CL**](#) (defined in [DexelsDetector](#))

[**DexelsDetector\(DevInfo &devInfo\)**](#)

[**DexelsDetector\(DetectorInterface transport, int unit, const char *params\)**](#)

[**DexelsDetectorPy**](#) (defined in [DexelsDetector](#))

[**DisablePulseGenerator\(\)**](#)

[**EnableFrameCntWithinImage\(unsigned short usEnable\)**](#)

[**EnablePulseGenerator\(float frequency\)**](#)

[**EnablePulseGenerator\(\)**](#)

[**EnableROIMode\(bool bEnableROI\)**](#)

[**GetBinningMode\(\)**](#)

[**GetBufferXdim\(void\)**](#)

[**GetBufferYdim\(void\)**](#)

[**GetCapturedBuffer\(void\)**](#)

GetDetectorStatus()

GetExposureMode()

GetExposureTime()

GetFieldCount(void)

GetFirmwareBuild(int &iDayAndMonth, int &iYear, int &iTime)

GetFirmwareVersion()

GetFullWellMode()

GetGapTime()

GetModelNumber()

GetNumBuffers(void)

GetNumOfExposures()

GetReadoutMode()

GetReadOutTime()

GetROICoordinates(unsigned short &usStartColumn, unsigned short &

GetROIState()

GetSensorHeight(unsigned short uiSensorID=1)

GetSensorWidth(unsigned short uiSensorID=1)

GetSerialNumber()

GetTestMode()

GetTransportMethod()

GetTriggerSource()

gigeDet (defined in **DexelaDetector**)

GoLiveSeq(int start, int stop, int numBuf)

GoLiveSeq()

GoUnLive()

IsCallbackActive()

IsConnected()

IsFrameCntWithinImage()

IsLive()

LoadSensorConfigFile(char *filename)

MockSetter (defined in **DexelaDetector**)

OpenBoard()

OpenBoard(int NumBufs)

QueryBinningMode(bins flag)

QueryExposureMode(ExposureModes mode)

QueryFullWellMode(FullWellModes fwm)

QueryReadoutMode(ReadoutModes mode)

QueryTriggerSource(ExposureTriggerSource ets)

ReadBuffer(int bufNum, byte *buffer)

ReadBuffer(int bufNum, DexImage &img, int iZ=0)

ReadRegister(int address, int sensorNum=1)

SetBinningMode(bins flag)

SetCallback(IMAGE_CALLBACK func)

SetExposureMode(ExposureModes mode)

SetExposureTime(float timems)

SetFullWellMode(FullWellModes fwm)

SetGapTime(float timems)

SetNumOfExposures(int num)

SetPreProgrammedExposureTimes(int numExposures, float *exposureTimes)

SetReadoutMode(ReadoutModes mode)

SetROICoordinates(unsigned short usStartColumn, unsigned short usStartRow)

SetSlowed(bool flag)

SetTestMode(BOOL SetTestOn)

SetTriggerSource(ExposureTriggerSource ets)

Snap(int buffer, int timeout)

SoftReset(void)

SoftwareTrigger()

StopCallback()

ToggleGenerator(BOOL onOff)

WaitImage(int timeout)

WriteBuffer(int bufNum, byte *buffer)

WriteRegister(int address, int value, int sensorNum=0)

~DexelaDetector(void)

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by  doxygen 1.8.7

DexelsDetector API

Main Page Classes **Files**

File List File Members

DexelsDetector > DexelaDetector

DexelsDetector.h

Go to the documentation of this file.

```
1 // DexelaDetector.h : main header file for
2 // the DexelaDetector DLL
3
4 #pragma once
5 #ifndef DEX_BUILD
6 #ifdef _DEBUG
7 #pragma comment(lib,"DexelaDetector-d.lib")
8 #else
9 #pragma comment(lib,"DexelaDetector.lib")
10#endif
11#endif
12
13#include <vector>
14#include "DexDefs.h"
15#include "DexImage.h"
16#include "DexelsException.h"
17#include <boost/shared_ptr.hpp>
18
19
20#pragma warning(disable: 4251)
21
22class DllExport DexelaDetector
23{
24    typedef void (*IMAGE_CALLBACK)(int
25        fc, int buf, DexelaDetector* det);
26}
```

```
31 | private:
32 |
33 |     void* pyData;
34 |     char param[50];
35 |     DexelaException* cbException;
36 |
37 |     bool callbackActive;
38 |     UINT CallbackCounterThread();
39 |     IMAGE_CALLBACK callback;
40 |
41 |     HANDLE* detHandle;
42 |
43 |     void* callbackWorker;
44 |
45 |     //struct containing relevant info
46 |     for threads
47 |         struct threadInfo
48 |         {
49 |             int FC;
50 |             int bufNum;
51 |             DexelaDetector* det;
52 |             threadInfo(int fc, int
53 | BufNum, DexelaDetector* Det)
54 |             {
55 |                 FC = fc;
56 |                 bufNum = BufNum;
57 |                 det = Det;
58 |             }
59 |         };
60 |
61 | protected:
62 |     boost::shared_ptr<baseDetector>
63 |     base;
64 |     boost::shared_ptr<gigEDetector>
65 |     gigeDet;
66 |     boost::shared_ptr<camLinkDetector>
67 |     clDet;
```

```
63 public:  
64  
65     DexelaDetector(DevInfo &devInfo);  
66     DexelaDetector(DetectorInterface  
67         transport, int unit, const char* params);  
68     virtual ~DexelaDetector(void);  
69  
70     virtual void OpenBoard();  
71     void OpenBoard(int NumBufs);  
72     void CloseBoard();  
73  
74     int GetBufferXdim(void);  
75     int GetBufferYdim(void);  
76     int GetNumBuffers(void);  
77     int GetCapturedBuffer(void);  
78     int GetFieldCount(void);  
79     void ReadBuffer(int bufNum, byte*  
80         buffer);  
81     void ReadBuffer(int bufNum,  
82         DexImage &img, int iZ=0);  
83     void WriteBuffer(int bufNum, byte*  
84         buffer);  
85  
86     void SetFullWellMode(FullWellModes  
87         fwm);  
88     void SetExposureMode(ExposureModes  
89         mode);  
90     void SetExposureTime(float timems);  
91     void SetBinningMode(bins flag);  
92     void SetTestMode(BOOL SetTestOn);  
93     void SetTriggerSource(  
94         ExposureTriggerSource ets);  
95     void SetNumOfExposures(int num);  
96     int GetNumOfExposures();  
97     void SetGapTime(float timems);  
98     float GetGapTime();  
99     bool IsConnected();
```

```
93
94     ExposureModes GetExposureMode();
95     float GetExposureTime();
96     DetStatus GetDetectorStatus();
97     ExposureTriggerSource
98         GetTriggerSource();
99         BOOL GetTestMode();
100        FullWellModes GetFullWellMode();
101        bins GetBinningMode();
102        int GetSerialNumber();
103        int GetModelNumber();
104        int GetFirmwareVersion();
105        void GetFirmwareBuild(int&
106            iDayAndMonth, int& iYear, int& iTime);
107        DetectorInterface
108            GetTransportMethod();
109            double GetReadOutTime();
110            //float GetReadOutTime();
111            bool IsCallbackActive();
112            bool IsLive();
113            void Snap(int buffer, int timeout);
114            int ReadRegister(int address, int
115                sensorNum=1);
116            void WriteRegister(int address, int
117                value, int sensorNum=0);
118            void ClearCameraBuffer(int i);
119            void ClearBuffers();
120            void LoadSensorConfigFile(char*
121                filename);
122            void SoftReset(void);
123
124            void GoLiveSeq(int start, int
125                stop,int numBuf);
126            void GoLiveSeq();
127            void GoUnLive();
```

```
123         void SoftwareTrigger();
124
125         void EnablePulseGenerator(float
126             frequency);
127         void EnablePulseGenerator();
128         void DisablePulseGenerator();
129         void ToggleGenerator(BOOL onoff);
130
131         void WaitImage(int timeout);
132
133         void SetCallback(IMAGE_CALLBACK
134             func);
135
136         void CheckForCallbackError();
137         void CheckForLiveError();
138
139         void
140             SetPreProgrammedExposureTimes(int
141                 numExposures, float* exposuretimes_ms);
142
143         void SetROICoordinates(unsigned
144             short usStartColumn, unsigned short
145                 usStartRow, unsigned short usROIWidth,
146                 unsigned short usROIHeight);
147
148         void GetROICoordinates(unsigned
149             short& usStartColumn, unsigned short&
150                 usStartRow, unsigned short& usROIWidth, unsigned
151                 short& usROIHeight);
152
153         void EnableROIMode(bool
154             bEnableROI);
155
156         bool GetROIState();
157
158         unsigned short
159             GetSensorHeight(unsigned short uiSensorID=1);
160
161         unsigned short
162             GetSensorWidth(unsigned short uiSensorID=1);
```

```
148         bool IsFrameCntWithinImage();
149         void
150         EnableFrameCntWithinImage(unsigned short
151             usEnable);
152         void SetSlowed(bool flag);
153         void SetReadoutMode(ReadoutModes
154             mode);
155         ReadoutModes GetReadoutMode();
156         int QueryReadoutMode(ReadoutModes
157             mode);
158         int QueryExposureMode(
159             ExposureModes mode);
160         int QueryTriggerSource(
161             ExposureTriggerSource ets);
162         int QueryFullWellMode(FullWellModes
163             fwm);
164         int QueryBinningMode(bins flag);
165
166         friend class baseBusScanner;
167         friend class MockSetter;
168         friend class DexelaDetectorPy;
169         friend class Dex_CL;
170     };
171
172     typedef void (*IMAGE_CALLBACK)(int fc, int
173         buf, DexelaDetector* det);
174 }
```

DexelaDetector API

Main Page	Classes	Files
Class List	Class Index	Class Hierarchy

DexelaDetectorCL Member List

This is the complete list of members for **DexelaDetectorCL**, including all inherited members.

base (defined in [DexelaDetector](#))

[CheckForCallbackError\(\)](#)

[CheckForLiveError\(\)](#)

clDet (defined in [DexelaDetector](#))

[ClearBuffers\(\)](#)

[ClearCameraBuffer\(int i\)](#)

[CloseBoard\(\)](#)

[DexelaDetector\(DevInfo &devInfo\)](#)

[DexelaDetector\(DetectorInterface transport, int unit, const char *params\)](#)

[DexelaDetectorCL\(DetectorInterface transport, int unit, const char *params\)](#)

[DexelaDetectorCL\(DevInfo &devInfo\)](#)

[DisablePulseGenerator\(\)](#)

[EnableFrameCntWithinImage\(unsigned short usEnable\)](#)

[EnablePulseGenerator\(float frequency\)](#)

[EnablePulseGenerator\(\)](#)

[EnableROIMode\(bool bEnableROI\)](#)

[GetBinningMode\(\)](#)

[GetBufferXdim\(void\)](#)

[GetBufferYdim\(void\)](#)

[GetCapturedBuffer\(void\)](#)

[GetDetectorStatus\(\)](#)

GetExposureMode()

GetExposureTime()

GetFieldCount(void)

GetFirmwareBuild(int &iDayAndMonth, int &iYear, int &iTime)

GetFirmwareVersion()

GetFullWellMode()

GetGapTime()

GetModelNumber()

GetNumBuffers(void)

GetNumOfExposures()

GetReadoutMode()

GetReadOutTime()

GetROICoordinates(unsigned short &usStartColumn, unsigned short &

GetROIState()

GetSensorHeight(unsigned short uiSensorID=1)

GetSensorWidth(unsigned short uiSensorID=1)

GetSerialNumber()

GetTestMode()

GetTransportMethod()

GetTriggerSource()

gigeDet (defined in **DexelaDetector**)

GoLiveSeq(int start, int stop, int numBuf)

GoLiveSeq()

GoUnLive()

IsCallbackActive()

IsConnected()

IsFrameCntWithinImage()

IsLive()

LoadSensorConfigFile(char *filename)

OpenBoard()

OpenBoard(int NumBufs)

PowerCLInterface(bool flag)

QueryBinningMode(bins flag)

QueryExposureMode(ExposureModes mode)

QueryFullWellMode(FullWellModes fwm)

QueryReadoutMode(ReadoutModes mode)

QueryTriggerSource(ExposureTriggerSource ets)

ReadBuffer(int bufNum, byte *buffer)

ReadBuffer(int bufNum, DexImage &img, int iZ=0)

ReadRegister(int address, int sensorNum=1)

SetBinningMode(bins flag)

SetCallback(IMAGE_CALLBACK func)

SetExposureMode(ExposureModes mode)

SetExposureTime(float timems)

SetFullWellMode(FullWellModes fwm)

SetGapTime(float timems)

SetNumOfExposures(int num)

SetPreProgrammedExposureTimes(int numExposures, float *exposureTimes)

SetReadoutMode(ReadoutModes mode)

SetROICoordinates(unsigned short usStartColumn, unsigned short usStartRow)

SetSlowed(bool flag)

SetTestMode(BOOL SetTestOn)

SetTriggerSource(ExposureTriggerSource ets)

Snap(int buffer, int timeout)

SoftReset(void)

SoftwareTrigger()

StopCallback()

ToggleGenerator(BOOL onOff)

WaitImage(int timeout)

WriteBuffer(int bufNum, byte *buffer)

WriteRegister(int address, int value, int sensorNum=0)

~DexelaDetector(void)

~DexelaDetectorCL(void)

Generated on Tue Nov 25 2014 10:22:44 for DexelaDetector API by  doxygen 1.8.7

DexelsDetector API

Main Page	Classes	Files
File List	File Members	
DexelsDetector	DexelsDetector	

DexelsDetectorCL.h

Go to the documentation of this file.

```
1  #pragma once
2  #include "dexeladetector.h"
3
4
5
10 class DllExport DexelaDetectorCL :
11     public DexelaDetector
12 {
13 public:
14     DexelaDetectorCL(DetectorInterface
15         transport, int unit, const char* params);
16     DexelaDetectorCL(DevInfo &devInfo);
17     virtual ~DexelaDetectorCL(void);
18     void PowerCLInterface(bool flag);
19
20     void OpenBoard();
21     void OpenBoard(int NumBufs);
22 };
```

DexelaDetector API

Main Page	Classes	Files
Class List	Class Index	Class Hierarchy
Class Members		

DexelaDetectorGE Member List

This is the complete list of members for **DexelaDetectorGE**, including all inherited members.

base (defined in [DexelaDetector](#))

[CheckForCallbackError\(\)](#)

[CheckForLiveError\(\)](#)

clDet (defined in [DexelaDetector](#))

[ClearBuffers\(\)](#)

[ClearCameraBuffer\(int i\)](#)

[CloseBoard\(\)](#)

[DexelaDetector\(DevInfo &devInfo\)](#)

[DexelaDetector\(DetectorInterface transport, int unit, const char *params\)](#)

[DexelaDetectorGE\(DevInfo &devInfo\)](#)

[DexelaDetectorGE\(DetectorInterface transport, int unit, const char *params\)](#)

[DisablePulseGenerator\(\)](#)

[EnableFrameCntWithinImage\(unsigned short usEnable\)](#)

[EnablePulseGenerator\(float frequency\)](#)

[EnablePulseGenerator\(\)](#)

[EnableROIMode\(bool bEnableROI\)](#)

[GetBinningMode\(\)](#)

[GetBufferXdim\(void\)](#)

[GetBufferYdim\(void\)](#)

[GetCapturedBuffer\(void\)](#)

[GetDetectorStatus\(\)](#)

GetExposureMode()

GetExposureTime()

GetFieldCount(void)

GetFirmwareBuild(int &iDayAndMonth, int &iYear, int &iTime)

GetFirmwareVersion()

GetFullWellMode()

GetGapTime()

GetModelNumber()

GetNumBuffers(void)

GetNumOfExposures()

GetReadoutMode()

GetReadOutTime()

GetROICoordinates(unsigned short &usStartColumn, unsigned short &

GetROIState()

GetSensorHeight(unsigned short uiSensorID=1)

GetSensorWidth(unsigned short uiSensorID=1)

GetSerialNumber()

GetTestMode()

GetTransportMethod()

GetTriggerSource()

gigeDet (defined in **DexelaDetector**)

GoLiveSeq(int start, int stop, int numBuf)

GoLiveSeq()

GoUnLive()

IsCallbackActive()

IsConnected()

IsFrameCntWithinImage()

IsLive()

LoadSensorConfigFile(char *filename)

OpenBoard()

OpenBoard(int NumBufs)

QueryBinningMode(bins flag)

QueryExposureMode(ExposureModes mode)

QueryFullWellMode(FullWellModes fwm)

QueryReadoutMode(ReadoutModes mode)

QueryTriggerSource(ExposureTriggerSource ets)

ReadBuffer(int bufNum, byte *buffer)

ReadBuffer(int bufNum, DexImage &img, int iZ=0)

ReadRegister(int address, int sensorNum=1)

SetBinningMode(bins flag)

SetCallback(IMAGE_CALLBACK func)

SetExposureMode(ExposureModes mode)

SetExposureTime(float timems)

SetFullWellMode(FullWellModes fwm)

SetGapTime(float timems)

SetNumOfExposures(int num)

SetPersistentIPAddress(int firstByte, int secondByte, int thirdByte, int fo

SetPreProgrammedExposureTimes(int numExposures, float *exposureT

SetReadoutMode(ReadoutModes mode)

SetROICoordinates(unsigned short usStartColumn, unsigned short usS

SetSlowed(bool flag)

SetTestMode(BOOL SetTestOn)

SetTriggerSource(ExposureTriggerSource ets)

Snap(int buffer, int timeout)

SoftReset(void)

SoftwareTrigger()

StopCallback()

ToggleGenerator(BOOL onOff)

WaitImage(int timeout)

WriteBuffer(int bufNum, byte *buffer)

WriteRegister(int address, int value, int sensorNum=0)

~DexelaDetector(void)

~DexelaDetectorGE(void)

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by  doxygen 1.8.7

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > DexelaDetector

DexelaDetectorGE.h

Go to the documentation of this file.

```
1 #pragma once
2 #include "DexelaDetector.h"
3
4
5 class DllExport DexelaDetectorGE :
6     public DexelaDetector
7 {
8 public:
9     DexelaDetectorGE(DevInfo &devInfo);
10    DexelaDetectorGE(DetectorInterface
11        transport, int unit, const char* params);
12    virtual ~DexelaDetectorGE(void);
13
14    void SetPersistentIPAddress(int
15        firstByte, int secondByte, int thirdByte, int
16        fourthByte);
17    void OpenBoard();
18    void OpenBoard(int NumBufs);
19
20};
```

DexelsDetector API

Main Page	Classes	Files
Class List	Class Index	Class Hierarchy

DexelsException Member List

This is the complete list of members for [DexelsException](#), including all inherited members.

[**DexelsException**\(const char *message, Derr code, int line, const char *\)](#)
[**DexelsException**\(const DexelaException &ex, const char *function\)](#)
[**GetCode\(\)**](#)
[**GetFileName\(\)**](#)
[**GetFunctionName\(\)**](#)
[**GetLineNumber\(\)**](#)
[**GetTransportError\(\)**](#)
[**GetTransportMessage\(\)**](#)
[**LoadErrorStrings**\(const char *filename\) \(defined in \[DexelsException\]\(#\)\)](#)
[**what\(\)** const](#)
[**~DexelsException**\(void\)](#)

DexelaDetector API

Main Page	Classes	Files
File List	File Members	
DexelaDetector	DexelaException	

DexelaException.h

Go to the documentation of this file.

```
1 #pragma once
2
3 #ifndef DEX_BUILD
4 #ifdef _DEBUG
5 #pragma comment(lib,"DexelaException-d.lib")
6 #else
7 #pragma comment(lib,"DexelaException.lib")
8 #endif
9 #endif
10
11
12 #include "dexdefs.h"
13 #include <exception>
14
15 using namespace std;
16
17 class DllExport DexelaException :
18     public exception
19 {
20 public:
21     DexelaException(const char*
22                     message, Derr code, int line, const char*
23                     filename, const char* function, int
24                     transportEr, const char* transportMessage);
25     DexelaException(const
26                     DexelaException& ex, const char* function);
```

```

27         ~DexelaException(void) throw();
28         const char* what() const throw ();
29         Derr GetCode();
30         int GetTransportError();
31         const char* GetFileName();
32         int GetLineNumber();
33         const char* GetFunctionName();
34         const char* GetTransportMessage();
35
36
37     static void LoadErrorStrings(const
38         char* filename);
39 private:
40         const char* _msg;
41         Derr _code;
42         const char* _filename;
43         int _line;
44         const char* _func;
45         int _transEr;
46         const char*_transMsg;
47     };
48
49 #define rethrowEr(EX) \
50     throw
51     DexelaException(EX, __FUNCTION__);\
52 #define
53     throwNewEr(MSG, CODE, TRANSER, TRANSMSG) \
54     throw
      DexelaException(MSG, CODE, __LINE__, __FILE__, __F
      UNCTION__, TRANSER, TRANSMSG);\

```

DexelaDetector API

Main Page	Classes	Files
Class List	Class Index	Class Hierarchy
Class Members		

DexImage Member List

This is the complete list of members for [DexImage](#), including all inherited members.

[AddImage\(\)](#)

[Build\(int iWidth, int iHeight, int iDepth, pType iPxType\)](#)

[Build\(int model, bins binFmt, int iDepth\)](#)

[DefectCorrection\(int DefectFlags=31\)](#)

[DexImage\(void\)](#)

[DexImage\(const char *filename\)](#)

[DexImage\(const DexImage &input\)](#)

[FindAverageofPlanes\(\)](#)

[FindMedianofPlanes\(\)](#)

[FixFlood\(\)](#)

[FloodCorrection\(\)](#)

[FullCorrection\(\)](#)

[GetDarkImage\(\)](#)

[GetDarkOffset\(\)](#)

[GetDataPointerToPlane\(int iZ=0\)](#)

[GetDefectMap\(\)](#)

[GetFloodImage\(\)](#)

[GetImageBinning\(\)](#)

[GetImageDepth\(\)](#)

[GetImageModel\(\)](#)

[GetImagePixelType\(\)](#)

GetImagePlane(int iZ)
GetImageType()
GetImageXdim()
GetImageYdim()
GetLinearizationStarts(int &msLength)
IsEmpty()
LinearizeData()
LoadDarkImage(const DexImage &dark)
LoadDarkImage(const char *filename)
LoadDefectMap(const DexImage &defect)
LoadDefectMap(const char *filename)
LoadFloodImage(const DexImage &flood)
LoadFloodImage(const char *filename)
operator=(const DexImage &input)
PlaneAvg(int iZ=0)
ReadImage(const char *filename)
SetDarkOffset(int offset)
SetImageParameters(bins binningMode, int modelNumber)
SetImageType(DexImageTypes type)
SetLinearizationStarts(unsigned int *msArray, int msLength)
SetROIParameters(unsigned short usStartColumn, unsigned short usStartRow, unsigned short usWidth, unsigned short usHeight)
SetScrambledFlag(bool onOff)
SubImageDefectCorrection(int startCol, int startRow, int width, int height)
SubtractDark()
UnscrambleImage()
WriteImage(const char *filename)
WriteImage(const char *filename, int iZ)
~DexImage(void)

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > DexImage

DexImage.h

Go to the documentation of this file.

```
1 // DexImage.h: Main header file for the
2 // DexImage object
3
4 #pragma once
5
6 #ifndef DEX_BUILD
7 #ifdef __DEBUG
8 #pragma comment(lib, "DexImage-d.lib")
9 #else
10 #pragma comment(lib, "DexImage.lib")
11 #endif
12 #endif
13 #endif
14
15 #include "DexDefs.h"
16 #include "DexImage.h"
17 #include <vector>
18 #include <boost/shared_ptr.hpp>
19
20 using namespace std;
21
22 #pragma warning(disable: 4251)
23
24 class DllExport DexImage
25 {
26
27     private:
```

```
    boost::shared_ptr<BaseImage> baseIm;
32
33 public:
34     DexImage(void);
35     DexImage(const char*
filename);
36     DexImage(const DexImage
&input);
37     void operator =(const
DexImage &input);
38     ~DexImage(void);
39
40     void ReadImage(const char*
filename);
41     void WriteImage(const
char* filename);
42     void WriteImage(const
char* filename, int iZ);
43     void Build(int iWidth, int
iHeight, int iDepth, pType iPxType);
44     void Build(int model, bins
binFmt, int iDepth);
45
46     void*
GetDataPointerToPlane(int iZ=0);
47     int GetImageXdim();
48     int GetImageYdim();
49     int GetImageDepth();
50     pType GetImagePixelType();
51     float PlaneAvg(int iZ=0);
52     void FixFlood();
53     void FindMedianofPlanes();
54     void
FindAverageofPlanes();
55     void LinearizeData();
56     void SubtractDark();
57     void FloodCorrection();
```

```
58 |                     void DefectCorrection(int
| DefectFlags=31);
59 |                     void
| SubImageDefectCorrection(int startCol, int
| startRow, int width, int height,int
| CorrectionsFlag=31);
60 |                     void FullCorrection();
61 |                     void UnscrambleImage();
62 |                     void AddImage();
63 |
64 |                     void LoadDarkImage(const
| DexImage &dark);
65 |                     void LoadDarkImage(const
| char* filename);
66 |                     void LoadFloodImage(const
| DexImage &flood);
67 |                     void LoadFloodImage(const
| char* filename);
68 |                     void LoadDefectMap(const
| DexImage &defect);
69 |                     void LoadDefectMap(const
| char* filename);
70 |
71 |                     DexImage GetDarkImage();
72 |                     DexImage GetFloodImage();
73 |                     DexImage GetDefectMap();
74 |                     DexImage GetImagePlane(int
| iZ);
75 |
76 |                     DexImageTypes
77 |                     GetImageType();
78 |                     void
79 |                     SetImageType(DexImageTypes type);
80 |                     void SetDarkOffset(int
| offset);
81 |                     int GetDarkOffset();
```

```
81
82             void
83             SetLinearizationStarts(unsigned int* msArray,
84                                     int msLength);
85             unsigned int*
86             GetLinearizationStarts(int& msLength);
87             void
88             SetImageParameters(bins binningMode, int
89                                 modelNumber);
90             int GetImageModel();
91             bins GetImageBinning();
92             bool IsEmpty();
93             void
94             SetScrambledFlag(bool
95             onOff);
96             void
97             SetROIParameters(unsigned short usStartColumn,
98                             unsigned short usStartRow, unsigned short
99                             usROIWidth, unsigned short usROIHeight);
100            };
```

DexelaDetector API

Main Page	Classes	Files	
Class List	Class Index	Class Hierarchy	Class Members

GeometryCorrectionParams Member List

This is the complete list of members for [GeometryCorrectionParams](#), including all inherited members.

[iRefX GeometryCorrectionParams](#)

[iRefY GeometryCorrectionParams](#)

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by [doxygen](#) 1.8.7

DexelaDetector API

Main Page Classes **Files**

File List File Members

DexelaDetector > DexelaException

DexDefines.h

Go to the documentation of this file.

```
1 #pragma once
2
4 typedef unsigned short ushort;
5 typedef unsigned long ulong;
6 typedef unsigned char byte;
7
8 #if _WIN32 || _WIN64
9 #if _WIN64
10#define _X64
11#else
12#define _X86
13#endif
14#endif
15
16#define TransMsgSize 1024
17
18//define import for applications using the
19//natveapi library.
20#ifndef DEX_BUILD
21#define DllExport __declspec( dllexport )
22#else
23#define DllExport __declspec( dllimport )
24#endif
25
26#ifndef DEX_BUILD_C
```

```
27 #define DllExportC __declspec( dllexport )
28 #else
29 #define DllExportC __declspec( dllimport )
30 #endif
31
36 #define MAX_PIXEL_VAL 16383
37 #define MIN_PIXEL_VAL 0
41 #define minTimeIncrement 0.01F
45 #define minTimeIncrement2 195.2F
49 #define ExposureSleepTimems 10
53 #define TimingResolution 100
57
58 #define RETURN_CHAR_LENGTH_CONST 50
59
60
61
65 #define DarkPixelXOffset 2
66 #define DarkPixelYOffset 4
70
71
76 #define AddrFPGANumber 126
77 #define AddrSerialNumber 125
81 #define AddrModelNumber 124
85 #define AddrGapTime 18
89 #define AddrNumberOfFrames 17
93 #define AddrFirmwareVersion 127
97 #define AddrTriggerSource 0
101 #define AddrExposureTimeLow 11
105 #define AddrExposureTimeHigh 12
109 #define AddrExposureTime 12
113 #define AddrExposureTime2 13
117 #define AddrExposureTime2Low 13
121 #define AddrExposureTime2High 14
125 #define AddrHorizontalBinReg 10
129 #define AddrVerticalBinReg 9
133 #define AddrControlReg 0
137 #define AddrPPExposreTime1Low 27
```

```
141 #define AddrPPExposreTime1High 28
145 #define AddrPPExposreTime2Low 29
149 #define AddrPPExposreTime2High 30
153 #define AddrPPExposreTime3Low 31
157 #define AddrPPExposreTime3High 32
161 #define AddrPPExposreTime4Low 33
165 #define AddrPPExposreTime4High 34
169 #define SerialNumberReg1 0
173 #define SerialNumberReg2 0
177 #define SerialNumberReg3 0
181 #define TemperatureReg 0
185 #define AddrWellReg 3
189 #define AddrWellHigh 4
193 #define AddrWellLow 65531
197 #define AddrSensorBinReg 3
201 #define AddrSensorBinReg2 5
205 #define AddrNumLines 7
209 #define AddrNumPixels 8
213 #define SensorBinClear 65087
217 #define DigitalBinBit 65533
221 #define Sensor1x1 0 //000 65087
225 #define Sensor1x2 0 //000 65087
229 #define Sensor1x4 64 //reset 65087 or 64
233 #define Sensor2x1 128 //128
237 #define Sensor2x2 128 //128
241 #define Sensor2x4 192 //192
245 #define Sensor4x1 256 //256
249 #define Sensor4x2 256 //256
253 #define Sensor4x4 320 //320
257 #define BinCommit 514
261 #define AddrReadOutTime
410
265 #define ReadoutTimeFactor1313
2
269 #define AddrROIStartColumn
404
273 #define AddrROIwidth
```

```
405
277 #define AddrROIStartRow
402
281 #define AddrROIheight
403
285 #define AddrFrameCounter
63
//0x3F
289 #define
AddrFramePackingMode_ImageCountPerBlock
64
//0x40
293 #define
AddrFramePackingMode_BlockHeightInRows
65
//0x41
297 #define AddrBuildDayAndMonth
38
301 #define AddrBuildYear
39
305 #define AddrBuildTime
40
309 #define AddrReadOutTimeLow
55
314 #define AddrReadOutTimeHigh
56
319 #define AddrControlReg1
1
323 #define AddrFeaturesReg0
36
327 #define AddrFeaturesReg1
37
331
332 #define AVGERAGED_FLAG
1
333 #define FIXED_FLAG
2
334 #define LINEARIZED_FLAG
4
```

```
335 #define SORTED_FLAG  
     8  
336 #define CLEAR_SORTED_FLAG  
     0xFFFF  
337 #define OPERATION_KNOWN_FLAG  
     0x8000  
338 #define CLEAR_OPERATION_KNOWN_FLAG  
     0x7FFF  
339 #define NOOP_FLAG  
     0x0  
340 #define XIS_OFFSET_CORRECTED_FLAG  
     1  
341 #define XIS_GAIN_CORRECTED_FLAG  
     2  
342 #define XIS_DEFECT_CORRECTED_FLAG  
     4  
343 #define XIS_MULTIGAIN_CORRECTED  
     8           /*this is not currently used  
      appart from in XIS*/  
344 #define DEX_OFFSET_CORRECTED_FLAG  
     16          /*Dexela versions of the  
      corrections*/  
345 #define DEX_GAIN_CORRECTED_FLAG  
     32  
346 #define DEX_DEFECT_CORRECTED_FLAG  
     64  
347 #define DEX_EXTRA_PARAMS_FLAG  
     0x4000    /*this flag will indicate the  
      presence of new parameters (e.g. model,  
      binning, operations) in the HIS header*/  
348 #define CORRECTION_KNOWN_FLAG  
     0x8000  
349 #define UNCORRECTED_FLAG  
     0x0  
350  
351 #define TIFFTAG_DEX_CORRECTION_FLAGS  
     34595    /* New tiff-tag for storing
```

```
    correction flags parameter */
352 #define TIFFTAG_DEX_OPERATION_FLAGS
34596 /* New tiff-tag for storing operation
flags parameter */
353 #define TIFFTAG_DEX_IMAGE_TYPE
34597 /* New tiff-tag for storing image-
type parameter */
354 #define DEX_DATA_IMAGE
0          /* regular data image */
355 #define DEX_OFFSET_IMAGE
1          /* offset data image */
356 #define DEX_GAIN_IMAGE
2          /* gain data image */
357 #define DEX_DEFECT_MAP
3          /* defect map image */
358 #define DEX_UNKONWN_TYPE_IMAGE 0xFF
/* type of image is unknown */
359 #define TIFFTAG_DEX_MODEL_NUM
34598 /* New tiff-tag for storing image-
type parameter */
360 #define TIFFTAG_DEX_BIN_FMT
34599
361 #define TIFFTAG_ROI_START_COL
34600
362 #define TIFFTAG_ROI_START_ROW
34601
363 #define TIFFTAG_DEFECT_FLAGS
34602
364
365
366 #define MAX_REG_ADDR
999
367 #define MAX_REG_VALUE
0xFFFF
368
369 #ifdef __cplusplus
370
```

```
371 class BaseImage;
372 class baseDetector;
373 class gigEDetector;
374 class camLinkDetector;
375 class PleoraLib;
376 class xclib;
377 class baseBusScanner;
378
379 #endif
```

DexelaDetector API

[Main Page](#)

[Classes](#)

[Files](#)

[DexelaDetector](#)

DexelaDetector Directory Reference

Directories

directory **BusScanner**

directory **DexelaDetector**

directory **DexelaException**

directory **DexImage**

DexelsDetector API

[Main Page](#)

[Classes](#)

[Files](#)

[DexelsDetector](#)

[BusScanner](#)

BusScanner Directory Reference

Files

file **BusScanner.cpp**

file **BusScanner.h** [code]

file **BusScanner/resource.h** [code]

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by **doxygen** 1.8.7

DexelsDetector API

[Main Page](#)

[Classes](#)

[Files](#)

[DexelsDetector](#)

[DexelsException](#)

DexelsException Directory Reference

Files

file [**DexDefines.h**](#) [code]

file [**DexDefs.h**](#) [code]

file [**DexelsException.cpp**](#)

file [**DexelsException.h**](#) [code]

file [**DexelsException/resource.h**](#) [code]

DexelaDetector API

[Main Page](#)

[Classes](#)

[Files](#)

[DexelaDetector](#)

[DexelaDetector](#)

DexelaDetector Directory Reference

Files

file **DexelsDetector.cpp**

file **DexelsDetector.h** [code]

file **DexelsDetectorCL.cpp**

file **DexelsDetectorCL.h** [code]

file **DexelsDetectorGE.cpp**

file **DexelsDetectorGE.h** [code]

file **resource1.h** [code]

DexelsDetector API

[Main Page](#)

[Classes](#)

[Files](#)

[DexelsDetector](#)

[DexImage](#)

DexImage Directory Reference

Files

file **DexImage.cpp**

file **DexImage.h** [code]

file **DexImage/resource.h** [code]

Generated on Tue Nov 25 2014 10:22:45 for DexelaDetector API by **doxygen** 1.8.7