Software Developpement Kit
for

Crius1280

version 2.3.0

Provided by

Xenics
EXOSENS GROUP

Friday 10th November, 2023, 14:39

Contents

Chapter 1

Crius1280 SDK.

1.1 Introduction

Xenics Exosens provides a set of functions in a Library to communicate with *Crius1280*.
This Library allows customers to use it's owns programming language (*C* ...) or tools (*Labview*, *Matlab* ...).

Provided SmartViewer GUI is based on features exposed by this Library.

1.1.1 Windows

Library kit provides the following items :

- DALProxy1280_12USB.dll
- DALProxy1280_12USB_x64.dll
- DALProxy1280_12USB.h
- DALProxy1280_12USBDef.h

These files must remain together.

Requirement

Microsoft Visual Studio 2010 Runtime is also required. This runtime is installed during *CriusViewer* setup.

1.1.2 Linux :

Library kit provides the following items :

- DALProxy1280_12USB.so.x.y.z (shared library)
- DALProxy1280_12USB.so.x.y (symbolic link)
- DALProxy1280_12USB.so.x (symbolic link)
- DALProxy1280_12USB.so (symbolic link)
- libusb-1.0.so.0 (shared library)
- libusb-1.0.so (symbolic link)
- DALProxy1280_12USB.h
- DALProxy1280_12USBDef.h

(x.y.z is the version number)

Requirement

At least gcc 5.

1.2 Installation

DALProxy1280_12USB Library was designed for Microsoft Windows Vista/Seven/8.X/10 and Linux.
Crius1280 must be properly installed on system, plugged and powered.

1.2.1 Windows Installation :

In order to use the Crius1280 module : install drivers. Use the library files to build programs you can share with msi drivers installer.

1.2.2 Linux Installation :

Don't forget to put the .so files into correct environment folder, or configure environment to point to the .so files.

1.3 Overview

Before using this Library, please take few minutes to read these Technical Notes .

Next, have a look at How to integrate this Library for details on how to use this Library in your favorite tools.

Then, the SDK came with some examples code, you can give a look at them to show how simple is this SDK.

Mode details about Library processing chain may be found on Processing Chain.

Calibration is a important part of IR image processing. To understand how to use Crius1280 Calibration functions, please, give a look at Calibration Process.

1.4 Library Overview

Library provides a sub-set of functions :

- Crius1280 Management
- Crius1280 Processing
- Crius1280 Image
- Crius1280 Storage
- Crius1280 Calibration
- Function return code

Chapter 2

Technical Notes

Important notes on using these functions.

2.1 Module connection

Several *Crius1280* may be plugged into Workstation.
Due to image flow design, **only one application can be connected to *Crius1280***. Application must release it (disconnect) to make it available to another application.
On another side, a single application can connect to several *Crius1280*, and get images from them.

Note

Even if only one module may be connected to one application, functions provide by DALProxy1280_12USB Library can be used in a thread. For example, get image from a single module can be called from a separate thread, having one thread per *Crius1280*. In the meanwhile, main thread (usually GUI) can call *Settings* functions for parameters update.

2.2 Data

DALProxy1280_12USB Library was designed to be easily use by any programming langage or software able to use Library. Functions perform single task, and did not require special knowledge.

Functions provide by this Library use common C type :

- signed or unsigned char (1 byte).
- signed or unsigned short (2 bytes).
- signed or unsigned int (4 bytes).
- float (4 bytes).
- C Style string (null terminate array).

Note

Most functions use HANDLE type. This type is void$*$.

2.3 Memory management

Note

In this section, caller refer as program calling Library function.

Library was designed to exchange many data with caller.

To simplify memory management, Library involves this single rule : **It's caller responsability to handle parameter placeholder**.

For example, when caller want to set a new table of Gain for NUC processing, caller allocates Gain table for values, and fills it. Then, it calls appropriate function, and passes pointer on this table as function parameter.

The same schema apply when caller want to retrieve table of Gain from NUC processing. Caller allocates Gain table for values, and passes it as parameter to appropriate function.

Warning

Most function use pre-defined table size (Gain, Offset and Image). It's caller responsibility to ensure table is large enough. Otherwise, memory corruption or even crash may occur.

Most table are the same number of element, i.e. image's dimension (`1280 x 1024 = 1310K` values). But, depending on single value memory size (short vs float), memory allocation may be different.

On a final note about memory management : Caller don't need to hold memory allocation after calling Library fonction. Library fonction don't take ownership of parameters.

2.4 Dimension

This last point remind array dimension use by the Library. Image is 1280 width by 1024 height. Image data storage is `1280 x 1024 = 1310KPixels = 2621KB` values. The same dimension apply to :

- Gain values table.
- Offset values table.

Bad pixel table is limited to 6143 elements. Place holder for bad pixels retrieval must be large enough.

Crius1280 provides 5 slots for Gain or Offset table.

Chapter 3

Processing Chain

Which steps are performed before the delivery of images.

Crius1280 include an Image Processing Chain.
For more information about *processing chain*, see *User's Guide - Image processing*.
This processing chain is composed of the following steps :

- Non Uniformity Correction.
- Bad Pixel Correction.
- Automatic Gain Correction

These 3 steps are disabled by default. Each of them may be disabled using Proxy1280_12USB_SetProcessing() function.

3.1 Non Uniformity Correction

NUC require Gain and Offset values for each image pixel. Default Gain and Offset at Library startup provide a neutral NUC, i.e. do not modified raw image. Gain values may be set using Proxy1280_12USB_SetTableGain(), and Offset values my be set using Proxy1280_12USB_SetTableOffset().
Current Gain and Offset table may be query using Proxy1280_12USB_GetTableGain() and Proxy1280_12USB_↩ GetTableOffset().
Crius1280 can store Gain and Offset table into slot. See Crius1280 Storage.

User can provide his own values, or use calibration process. See Crius1280 Calibration.

Note

See *User's Guide - Full Calibration* for more details about calibration.

3.2 Bad Pixel Correction

This processing fixes bad pixel from *Crius1280*. According bad pixel's position, some may not be fixable. Proxy1280_12USB_GetPixelMask() build an image mask of pixel status. See function documentation for details. It required a list of pixel position (x, y), inside image ([0-1279],[0-1023]). To set bad pixel list, use Proxy1280_12↩ USB_SetBadPixels(). Proxy1280_12USB_GetBadPixels() is used to retrieve current pixel list. *Crius1280* can store pixel list using Proxy1280_12USB_SaveBadPixels(), and retrieve it with Proxy1280_12USB_LoadBadPixels().

Note

Bad pixel list is limited to 6143 pixels.

3.3 Bad Lines and Bad Columns

Entire lines and columns of pixels can be encoded using a special [column,line] encoding where the first index codes the type and the second index the index (line or column):

- [10000, line index] - the entire line at "line index" is bad pixels
- [12000, column index] - the entire column at "column index" is bad pixels

Note

The functions Proxy1280_12USB_LoadBadPixels() and Proxy1280_12USB_SaveBadPixels() are compatible with this special bad pixel format.

3.4 Automatic Gain Correction

Images can be processed with gain correction, with different automatic method. Its purpose is to maximize the image dynamic. To configure Automatic Gain Correction, use Proxy1280_12USB_SetAGCProcessing. The possibility are Disable (eNoAGC), Histogram (eAGCEqHisto), Enhanced rendering (eAGCEnhanced), Linear (eAGCLinear). Proxy1280_12USB_GetAGCProcessing is used to retrieve current AGC processing.

Note

see *User's Guide - Gain control* for more details about AGC.

Chapter 4

How to integrate this Library

Communicate with *Crius1280* using your favorite tool.

DALProxy1280_12USB Library may be used with a programming langage (like *C*), or tool (like *Labview*, *Matlab* ...).

Note

Functions provide in Library are C standard style.

4.1 Tools.

Tools like *Labview* from National Instruments allows library function call. Use `Call library function` node, and configure it to match function prototype using this documentation.

See also

`National Instruments online help for Call library function node`.

Note

This DLL was designed for 32bit *Labview* version only.

Matlab from *Mathwork* use shared library in a way similar to C language. First, you have to load library using `loadlibrary()`, and call a function using `calllib()`. *Matlab* will require functions header definition.

See also

`Mathwork online help for C Shared Library functions`.

4.2 Linux.

Here is a compilation example for linux x64, with required options :

```
g++ -std=c++11 main.c -lDALProxy1280_12USB_x64 -pthread -lusb-1.0
```

Chapter 5

Calibration Process

How to perform calibration for *Crius1280*.

The SDK provides functions for :

- Calibration for Shutter mode :
- Shutterless Calibrations

Note

See *User's Guide - Calibration for Shutter mode* and *Calibration for Shutterless mode* for more details.

5.1 Calibration for Shutter mode :

5.1.1 Full Calibration

This calibration, upon successfull with generate :

- Gain table,
- Offset table,
- Bad pixel list.

Warning

Upon completion, Gain, Offset and Bad pixel data currently in use will be overwritten.

To make this calibration, the correct sequencing is : In front of low temperature black body :

- Proxy1280_12USB_InitShutter2PtsCalibration for stage 1
- Call several times Proxy1280_12USB_StepShutter2PtsCalibration for stage 1 in order to capture low temperature images
- Proxy1280_12USB_FinishShutter2PtsCalibration for stage 1

In front of high temperature black body :

- Proxy1280_12USB_InitShutter2PtsCalibration for stage 2
- Call several times Proxy1280_12USB_StepShutter2PtsCalibration for stage 2 in order to capture high temperature images

- Proxy1280_12USB_FinishShutter2PtsCalibration for stage 2

At the end of the process, on success new calibration data is set.

The Proxy1280_12USB_StepShutter2PtsCalibration capture the current sensor image and must be called several times to reduce temporal noise.

Warning

During the Calibration sequence, all processing are disabled and must not be enabled again during calibration. At the end of the process, the processing settings are restored.

5.1.2 Fast Calibration

This calibration, also known as *Shutter Calibration*, or one point calibration will only produce new Offset values.

Note

Current Gain and Bad pipxel list will remain untouched.

To make this calibration, the correct sequencing is :

In front of black body or shutter:

- Proxy1280_12USB_InitShutterCalibration
- Call several times Proxy1280_12USB_StepShutterCalibration in order to capture images
- Proxy1280_12USB_FinishShutterCalibration

At the end of the process, on success new Offset are calculate and set, using current Gain values and Bad Pixels.

The Proxy1280_12USB_StepShutterCalibration add the current sensor image and must be called several times to reduce temporal noise.

Warning

During the Calibration sequence, NUC processing is disabled and must not be enable again during calibration. At the end of the process, the processing settings are restored.

5.2 Shutterless Calibrations

This calibration is based on existing bad pixel list.

5.2.1 Shutter less Calibration T0

This calibration, upon successful will generate :

- Shutter less data for FPA temperature T0.

Warning

Upon completion, shutter less data currently in use will be overwritten.

To make this calibration, the correct sequencing is :

In front of low temperature black body :

- Proxy1280_12USB_InitSLCalibrationT0 for stage 1
- Call several times Proxy1280_12USB_StepSLCalibrationT0 for stage 1 in order to capture low temperature images
- Proxy1280_12USB_FinishSLCalibrationT0 for stage 1

In front of high temperature black body :

- Proxy1280_12USB_InitSLCalibrationT0 for stage 2
- Call several times Proxy1280_12USB_StepSLCalibrationT0 for stage 2 in order to capture high temperature images
- Proxy1280_12USB_FinishSLCalibrationT0 for stage 2

At the end of the process, on success new calibration data is set.

The Proxy1280_12USB_StepSLCalibrationT0 capture the current sensor image and must be called several times to reduce temporal noise.

Warning

During the Calibration sequence, all processing are disabled and must not be enable again during calibration. At the end of the process, the processing settings are restored.

5.2.2 Shutterless Calibration T1

This calibration, upon successful will generate :

- Shutter less data for FPA temperature T1.

To make this calibration, the correct sequencing is :

In front of black body or shutter :

- Proxy1280_12USB_InitSLCalibrationT1
- Call several times Proxy1280_12USB_StepSLCalibrationT1 in order to capture temperature images
- Proxy1280_12USB_FinishSLCalibrationT1

At the end of the process, on success new calibration data is set.

The Proxy1280_12USB_StepSLCalibrationT1 capture the current sensor image and must be called several times to reduce temporal noise.

Warning

During the Calibration sequence, all processing are disabled and must not be enable again during calibration. At the end of the process, the processing setting are restored back.

5.2.3 Changing between Shutter Mode and Shutterless Mode Calibration

The *Crius1280* stores in its memory either a Shutter Mode calibration or a Shutterless Mode calibration, but not both. When an attempt to overwrite one type of calibration with the other, an erorr is returned which indicates that the format is wrong: eProxy1280_12USBFormatMismatch

If the user really needs to change the format of the saved calibration, the entry Proxy1280_12USB_ClearSavedCalibrationData() must be used to clear the type of data currently stored.

Chapter 6

Module Index

6.1 Modules

Here is a list of all modules:

Chapter 7

Module Documentation

7.1 Crius1280 Management

Etablish and manage communication with Crius1280.

Functions

- eDALProxy1280_12USBErr Proxy1280_12USB_GetModuleCount (int ∗paiCount)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetModuleName (int iIdx, char ∗paName, int iLen)
- eDALProxy1280_12USBErr Proxy1280_12USB_ConnectToModule (int iIdx, HANDLE ∗paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_IsConnectToModule (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_DisconnectFromModule (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_RunBIST (HANDLE paHandle, unsigned int ∗diagCode)

7.1.1 Detailed Description

Etablish and manage communication with Crius1280.

This set provides :

- Functions to enumerates and name plugged Crius1280.
- Function to connect and disconnect to Crius1280.

Application call Proxy1280_12USB_GetModuleCount() to know how many Crius1280 are plugged to workstation. First Crius1280 index is 0, and so on.

Calling Proxy1280_12USB_GetModuleCount() check Crius1280 count. So, call it will refresh Crius1280 list.

Before calling any other function's group, Application must connect to a Crius1280 using Proxy1280_12USB_ConnectToModule(). Once a Crius1280 is connected by an application, it's not available to another application. Application must release Crius1280 by calling Proxy1280_12USB_DisconnectFromModule().

Connection to Crius1280 will provide a *handle*. This *handle* is use by all functions addressing this Crius1280. It remains valid until Proxy1280_12USB_DisconnectFromModule() is called.

Application can connect several Crius1280, using different *handles*.

7.1.2 Function Documentation

7.1.2.1 Proxy1280_12USB_GetModuleCount()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetModuleCount (
            int * paiCount )
```

Retrieve current count of plugged module.

Parameters

out	*paiCount*	Number of plugged module.

7.1.2.2 Proxy1280_12USB_GetModuleName()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetModuleName (
            int iIdx,
            char * paName,
            int iLen )
```

Query Crius1280 name by index.

Parameters

in	*iIdx*	Module index.
out	*paName*	Crius1280 name from index.
in	*iLen*	paName storage size.

7.1.2.3 Proxy1280_12USB_ConnectToModule()

```
eDALProxy1280_12USBErr Proxy1280_12USB_ConnectToModule (
            int iIdx,
            HANDLE * paHandle )
```

Connect to Crius1280 by index.
This function will return a handle, which will be uses as Crius1280 identifier.
Connection may failed if Crius1280 is already connected by another application.

Parameters

in	*iIdx*	Module index. First Crius1280 index is 0.
out	*paHandle*	Crius1280 handle.

7.1.2.4 Proxy1280_12USB_IsConnectToModule()

```
eDALProxy1280_12USBErr Proxy1280_12USB_IsConnectToModule (
            HANDLE paHandle )
```

Check if handle connection. This function will check if handle is still valid, and then check connection with Crius1280.

Parameters

in	*paHandle*	Crius1280 handle.

Returns

eProxy1280_12USBSuccess on success, or error code.

7.1.2.5 Proxy1280_12USB_DisconnectFromModule()

```
eDALProxy1280_12USBErr Proxy1280_12USB_DisconnectFromModule (
            HANDLE paHandle )
```

Disconnect to Crius1280 by index. This function will release Crius1280 connection.

Parameters

in	*paHandle*	Crius1280 handle.

7.1.2.6 Proxy1280_12USB_RunBIST()

```
eDALProxy1280_12USBErr Proxy1280_12USB_RunBIST (
            HANDLE paHandle,
            unsigned int * diagCode )
```

Run the Crius1280 Built-In Self-Test. This function will check if handle is still valid, and then run the built-in self tests.

Parameters

in	*paHandle*	Crius1280 handle.
out	*diagCode*	Diagnostic code provided by the Crius1280. Value is 0 in case of success.

Returns

eProxy1280_12USBSuccess on success, or error code.

7.2 Crius1280 Processing

Control Crius1280 image processing. Query module connected to workstation, Open and close link.

Functions

- eDALProxy1280_12USBErr Proxy1280_12USB_SetCalibrationConfig (HANDLE paHandle, int paParam)
- eDALProxy1280_12USBErr Proxy1280_12USB_SetNUCProcessing (HANDLE paHandle, unsigned char paBadPixels, unsigned char paNUC)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetNUCProcessing (HANDLE paHandle, unsigned char *paBadPixels, unsigned char *paNUC)

- eDALProxy1280_12USBErr Proxy1280_12USB_SetShutterLessProcessing (HANDLE paHandle, bool b↩ Activate)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetShutterLessProcessing (HANDLE paHandle, bool ∗pb↩ IsActive)
- eDALProxy1280_12USBErr Proxy1280_12USB_SetAGCProcessing (HANDLE paHandle, unsigned char paeAGCProcessing)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetAGCProcessing (HANDLE paHandle, unsigned char ∗paeAGCProcessing)
- eDALProxy1280_12USBErr **Proxy1280_12USB_SetCurrentAGCLocal** (HANDLE panhandle, float pa↩ GlobalContrastStrength, float paLocalContrastStrength, int paSpeed)
- eDALProxy1280_12USBErr **Proxy1280_12USB_SetCurrentAGCLocal_devinternalonly** (HANDLE panhandle, unsigned int paAGCLocalMode, int pabins, int paeAGCNb_tiles_x, int paeAGCNb_tiles_y, int pae↩ AGCswitch_external, float paeAGCHot_details)
- eDALProxy1280_12USBErr Proxy1280_12USB_SetCurrentTableGain (HANDLE paHandle, float ∗paTable↩ Gains)
- eDALProxy1280_12USBErr Proxy1280_12USB_SetCurrentTableOffset (HANDLE paHandle, signed short ∗paTableOffsets)
- eDALProxy1280_12USBErr Proxy1280_12USB_SetCurrentBadPixels (HANDLE paHandle, unsigned short ∗paTableX, unsigned short ∗paTableY, unsigned short paCount)
- eDALProxy1280_12USBErr Proxy1280_12USB_SetCurrentShutterless (HANDLE paHandle, unsigned int ∗paShutterless)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentShutterlessSize (HANDLE paHandle, unsigned int ∗pSize)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentShutterless (HANDLE paHandle, unsigned int ∗paShutterless)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentTableGain (HANDLE paHandle, float ∗paTable↩ Gains)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentTableOffset (HANDLE paHandle, signed short ∗paTableOffsets)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentBadPixels (HANDLE paHandle, unsigned short ∗paTableX, unsigned short ∗paTableY, unsigned short ∗paCount)

7.2.1 Detailed Description

Control Crius1280 image processing. Query module connected to workstation, Open and close link.

This set of function provides control over image processing.

- Query and change processing step state (enable or disable).
- Query processing parameters.
- Set processing parameters.

Processing is compose of :

- Bad pixel correction.
- Non linearity correction.

See also

User's Guide or Processing Chain for details

7.2.2 Function Documentation

7.2.2.1 Proxy1280_12USB_SetCalibrationConfig()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetCalibrationConfig (
            HANDLE paHandle,
            int paParam )
```

Configure Internal Calibration.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paParam*	configuration to apply. • bit[0] Enable (1) or Disable (0) the automatic fast calibration associated with mechanical shutter • bit[1-31] Reserved.

7.2.2.2 Proxy1280_12USB_SetNUCProcessing()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetNUCProcessing (
            HANDLE paHandle,
            unsigned char paBadPixels,
            unsigned char paNUC )
```

Enable/Disable NUC processing steps. These are enabled by default at connection.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paBadPixels*	Enable(1)/Disable(0) bad pixels correction.
in	*paNUC*	Enable(1)/Disable(0) Non Uniformity Correction.

Returns

This return error eProxy1280_12USBFeatureNotAvailable if Shutterless is activated.

7.2.2.3 Proxy1280_12USB_GetNUCProcessing()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetNUCProcessing (
            HANDLE paHandle,
            unsigned char * paBadPixels,
            unsigned char * paNUC )
```

Query NUC processing steps status.

Parameters

in	*paHandle*	Crius1280 handle.
out	*paBadPixels*	bad pixels correction enable(1) or disable(0).
out	*paNUC*	Non Uniformity Correction enable(1) or disable(0).

Returns

This return error eProxy1280_12USBFeatureNotAvailable if Shutterless is activated.

7.2.2.4 Proxy1280_12USB_SetShutterLessProcessing()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetShutterLessProcessing (
            HANDLE paHandle,
            bool bActivate )
```

Enable/Disable ShutterLess processing.

Parameters

in	*paHandle*	Crius1280 handle.
in	*bActivate*	Enable(true)/Disable(false) Shutterless processing.

Returns

This function return eProxy1280_12USBFeatureNotAvailable error if Shutterless is unavailable on this module.

7.2.2.5 Proxy1280_12USB_GetShutterLessProcessing()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetShutterLessProcessing (
            HANDLE paHandle,
            bool * pbIsActive )
```

Query Shutterless processing status.

Parameters

in	*paHandle*	Crius1280 handle.
out	*pbIsActive*	shutterless processing enable(true) or disable(false).

7.2.2.6 Proxy1280_12USB_SetAGCProcessing()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetAGCProcessing (
            HANDLE paHandle,
            unsigned char paeAGCProcessing )
```

Set Auto Gain Control processing step. By default, No AGC processing set.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paeAGCProcessing*	see eAGCProcessingValue for values.

7.2.2.7 Proxy1280_12USB_GetAGCProcessing()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetAGCProcessing (
            HANDLE paHandle,
            unsigned char * paeAGCProcessing )
```

Query processing steps status.

Parameters

in	*paHandle*	Crius1280 handle.
out	*paeAGCProcessing*	see eAGCProcessingValue for values.

7.2.2.8 Proxy1280_12USB_SetCurrentTableGain()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetCurrentTableGain (
            HANDLE paHandle,
            float * paTableGains )
```

Set Gains values for NUC processing.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paTableGains*	New Gains values for NUC processing.

Note

Each pixel must have a value. So paTableGains must contains 1280 $*$ 1280 float values (4 bytes float).

7.2.2.9 Proxy1280_12USB_SetCurrentTableOffset()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetCurrentTableOffset (
            HANDLE paHandle,
            signed short * paTableOffsets )
```

Set Offset values for NUC processing.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paTableOffsets*	New offsets values for NUC processing.

Note

Each pixel must have a value. So paTableGains must contains 1280 ∗ 1024 values (2 bytes signed value).

7.2.2.10 Proxy1280_12USB_SetCurrentBadPixels()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetCurrentBadPixels (
            HANDLE paHandle,
            unsigned short * paTableX,
            unsigned short * paTableY,
            unsigned short paCount )
```

Set bad pixels position in image for bad pixels correction.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paTableX,paTableY*	Bad pixels position in image.
in	*paCount*	bad pixels count.

7.2.2.11 Proxy1280_12USB_SetCurrentShutterless()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetCurrentShutterless (
            HANDLE paHandle,
            unsigned int * paShutterless )
```

Set Shutterless data for restore purpose. Shutterless data must be considered as binary and must not be modified.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paShutterless*	values.

7.2.2.12 Proxy1280_12USB_GetCurrentShutterlessSize()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentShutterlessSize (
            HANDLE paHandle,
            unsigned int * pSize )
```

Get Shutterless size of the data for backup and restore purpose.

Parameters

in	*paHandle*	Crius1280 handle.
out	*pSize*	size of shutterless data in bytes.

7.2.2.13 Proxy1280_12USB_GetCurrentShutterless()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentShutterless (
            HANDLE paHandle,
            unsigned int * paShutterless )
```

Get Shutterless data for backup purpose. Shutterless data must be considered as binary and must not be modified.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paShutterless*	New Shutterless values.

7.2.2.14 Proxy1280_12USB_GetCurrentTableGain()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentTableGain (
            HANDLE paHandle,
            float * paTableGains )
```

Get Gains current values from NUC processing.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paTableGains*	New Gains values for NUC processing.

Note

Each pixel must have a value. So paTableGains must contains 1280 $*$ 1024 float values (4 bytes float).

7.2.2.15 Proxy1280_12USB_GetCurrentTableOffset()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentTableOffset (
            HANDLE paHandle,
            signed short * paTableOffsets )
```

Get Offset current values from NUC processing.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paTableOffsets*	New offsets values for NUC processing.

Note

Each pixel must have a value. So paTableGains must contains 1280 ∗ 1024 values (2 bytes signed value).

7.2.2.16 Proxy1280_12USB_GetCurrentBadPixels()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetCurrentBadPixels (
            HANDLE paHandle,
            unsigned short * paTableX,
            unsigned short * paTableY,
            unsigned short * paCount )
```

Get current bad pixels position in image from bad pixels correction.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paTableX,paTableY*	Bad pixels position in image.
	paCount	Initial bad pixels array size, on return, bad pixel count.

Note

paCount must be init with paTableX / paTableY placeholder size (to avoid overflow), and will be modified by function with current bad pixel count.

7.3 Crius1280 Control

Set or Get module features. Refer to module user guide for details on feature, and SDK header file for paeFeature definition.

Functions

- eDALProxy1280_12USBErr Proxy1280_12USB_GetStringFeature (HANDLE paHandle, int paeFeature, char ∗paStr)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetUIntFeature (HANDLE paHandle, int paeFeature, unsigned int ∗paUInt)
- eDALProxy1280_12USBErr Proxy1280_12USB_GetFloatFeature (HANDLE paHandle, int paeFeature, float ∗paFloat)
- eDALProxy1280_12USBErr Proxy1280_12USB_SetStringFeature (HANDLE paHandle, int paeFeature, const char ∗paStr)
- eDALProxy1280_12USBErr Proxy1280_12USB_SetUIntFeature (HANDLE paHandle, int paeFeature, unsigned int paUInt)
- eDALProxy1280_12USBErr Proxy1280_12USB_SetFloatFeature (HANDLE paHandle, int paeFeature, float paFloat)

7.3.1 Detailed Description

Set or Get module features. Refer to module user guide for details on feature, and SDK header file for paeFeature definition.

7.3.2 Function Documentation

7.3.2.1 Proxy1280_12USB_GetStringFeature()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetStringFeature (
            HANDLE paHandle,
            int paeFeature,
            char * paStr )
```

Query string feature.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paeFeature*	Feature requested.
out	*paStr*	String from requested feature.

Warning

String Feature are 32 byte large, including null byte. Ensure paStr is large enougt.

7.3.2.2 Proxy1280_12USB_GetUIntFeature()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetUIntFeature (
            HANDLE paHandle,
            int paeFeature,
            unsigned int * paUInt )
```

Query integer feature.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paeFeature*	Feature requested.
out	*paUInt*	Integer value from requested feature.

7.3.2.3 Proxy1280_12USB_GetFloatFeature()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetFloatFeature (
            HANDLE paHandle,
            int paeFeature,
            float * paFloat )
```

Query float feature.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paeFeature*	Feature requested.

Parameters

out	*paFloat*	Float value from requested feature.

7.3.2.4 Proxy1280_12USB_SetStringFeature()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetStringFeature (
            HANDLE paHandle,
            int paeFeature,
            const char * paStr )
```

Set string feature.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paeFeature*	Feature written.
in	*paStr*	String for written feature.

Warning

String Feature are 32 byte large, including null byte. Ensure paStr is large enougt.

7.3.2.5 Proxy1280_12USB_SetUIntFeature()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetUIntFeature (
            HANDLE paHandle,
            int paeFeature,
            unsigned int paUInt )
```

Set integer feature.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paeFeature*	Feature written.
in	*paUInt*	Integer value for written feature.

7.3.2.6 Proxy1280_12USB_SetFloatFeature()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SetFloatFeature (
            HANDLE paHandle,
            int paeFeature,
            float paFloat )
```

Query float feature.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paeFeature*	Feature written.
in	*paFloat*	Float value for written feature.

7.4 Crius1280 Image

Query Image from Crius1280.

Functions

- eDALProxy1280_12USBErr Proxy1280_12USB_GetImage (HANDLE paHandle, unsigned short ∗paImage, int ∗paMeta, int paiTimeout)

7.4.1 Detailed Description

Query Image from Crius1280.

This set provides a single function to query current Crius1280 image. Calling it will block application until an image is available, or timeout occurs.

Application may provide image storage for new IR image. Image nature (Raw or Fixed) depend on processing settings (see Crius1280 Processing).

IR image is **1280 width by 1024 heigth**. Pixel storage is unsigned short, with 16bit effective, LSB aligned.

Along IR Image, some meta data are provides.

7.4.2 Function Documentation

7.4.2.1 Proxy1280_12USB_GetImage()

```
eDALProxy1280_12USBErr Proxy1280_12USB_GetImage (
            HANDLE paHandle,
            unsigned short * paImage,
            int * paMeta,
            int paiTimeout )
```

Query image from Crius1280.

Parameters

in	*paHandle*	Crius1280 handle.
out	*paImage*	Image placeholder for new image. Must be at least *1280 x 1024 x 2= 2000KB*.

Parameters

out	*paMeta*	Meta-Data placeholder. Must be at least 135 32bit values : • [0] fpa temperature in celsius (cast float to get it). • [1] period from previous image (in microsecond). • [2] frame counter (16bit effective). • [3-4] microseconds since epoch (Jan 1, 1970), on 64 bits (use 2 values). • [5-6] Reserved. • [7-134] Histogram.
in	*paiTimeout*	Operation timeout in millisecond.

7.5 Crius1280 Storage

Store and retrieve processing settings into Crius1280.

Functions

- eDALProxy1280_12USBErr Proxy1280_12USB_StartupDefault (HANDLE paHandle, unsigned char ∗paiIdxGains, unsigned char ∗paiIdxOffsets, unsigned char ∗paiIdxBank)
- eDALProxy1280_12USBErr Proxy1280_12USB_SlotType (HANDLE paHandle, unsigned char paiIndex, unsigned char ∗paeType, void ∗paData)
- eDALProxy1280_12USBErr Proxy1280_12USB_LoadTableGain (HANDLE paHandle, unsigned char paiIndex, float ∗paTableGain, void ∗paData)
- eDALProxy1280_12USBErr Proxy1280_12USB_LoadTableOffset (HANDLE paHandle, unsigned char paiIndex, short ∗paTableOffset, void ∗paData)
- eDALProxy1280_12USBErr Proxy1280_12USB_LoadBadPixels (HANDLE paHandle, unsigned short ∗paTableX, unsigned short ∗paTableY, unsigned short ∗paCount)
- eDALProxy1280_12USBErr Proxy1280_12USB_SaveTableGain (HANDLE paHandle, unsigned char paiIndex, const float ∗paTableGain, void ∗paData)
- eDALProxy1280_12USBErr Proxy1280_12USB_SaveTableOffset (HANDLE paHandle, unsigned char paiIndex, const short ∗paTableOffset, void ∗paData)
- eDALProxy1280_12USBErr Proxy1280_12USB_SaveBadPixels (HANDLE paHandle, const unsigned short ∗paTableX, const unsigned short ∗paTableY, unsigned short paCount)
- eDALProxy1280_12USBErr Proxy1280_12USB_LoadCurrentTableGain (HANDLE paHandle, unsigned char paiIndex)
- eDALProxy1280_12USBErr Proxy1280_12USB_LoadCurrentTableOffset (HANDLE paHandle, unsigned char paiIndex)
- eDALProxy1280_12USBErr Proxy1280_12USB_LoadCurrentBadPixels (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_SaveCurrentTableGain (HANDLE paHandle, unsigned char paiIndex, const void ∗paData)
- eDALProxy1280_12USBErr Proxy1280_12USB_SaveCurrentTableOffset (HANDLE paHandle, unsigned char paiIndex, const void ∗paData)
- eDALProxy1280_12USBErr Proxy1280_12USB_SaveCurrentBadPixels (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_SaveCurrentShutterlessTables (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_LoadCurrentShutterlessTables (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_ClearSavedCalibrationData (HANDLE paHandle, unsigned char paiType)

7.5.1 Detailed Description

Store and retrieve processing settings into Crius1280.

Crius1280 provides **8** slots to store Gain or Offset value. Slot are not dedicated to a kind of data.

Attention

Storage space is limited into Crius1280. Hence, data (Gain or Offset) are rounded to fit into slot. This may involve difference if your store data, read it, and compare to your initial values. For coherence, this data reduction is also apply when update NUC processing data (see Crius1280 Processing).

Save functions provides a *MakeDefault* parameter. When set to 1 (enable), this will mark slot as default. When application connect to Crius1280, Proxy1280_12USB_ConnectToModule() function will look for default slot, and load into processing data from slot.

7.5.2 Function Documentation

7.5.2.1 Proxy1280_12USB_StartupDefault()

```
eDALProxy1280_12USBErr Proxy1280_12USB_StartupDefault (
            HANDLE paHandle,
            unsigned char * paiIdxGains,
            unsigned char * paiIdxOffsets,
            unsigned char * paiIdxBank )
```

Default slot index for Gain values and Offset values, last setting's bank used.

Parameters

in	*paHandle*	Crius1280 handle.
out	*paiIdxGains*	Gain slot index, of 255 if no default Gain slot index.
out	*paiIdxOffsets*	Offset slot index, of 255 if no default Offset slot index.
out	*paiIdxBank*	Settings bank index, of 255 if no default settings index.

Note

No need to call and use this function (already done at Crius1280 connection)

7.5.2.2 Proxy1280_12USB_SlotType()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SlotType (
            HANDLE paHandle,
            unsigned char paiIndex,
            unsigned char * paeType,
            void * paData )
```

Query slot data type.

Parameters

in	*paHandle*	Crius1280 handle.

Parameters

in	*paiIndex*	Slot index to query.
out	*paeType*	Slot type.
out	*paData*	Table associate data. NULL, or 60 bytes placeholder. paData is additional data associated to Gain or Offset array, which can be used freely by application for instance to keep a trace of Gain or Offset table calibration conditions, either sensitivity, either focal plane array temperature. Slot type value are : • 0 :Empty slot. • 1 :Gain values. • 2 :Offset values.

7.5.2.3 Proxy1280_12USB_LoadTableGain()

```
eDALProxy1280_12USBErr Proxy1280_12USB_LoadTableGain (
            HANDLE paHandle,
            unsigned char paiIndex,
            float * paTableGain,
            void * paData )
```

Retrieve Crius1280 slot data as Gain values.
This function may failed if slot is empty, or slot data are not Gain values.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paiIndex*	Slot index as data.
out	*paTableGain*	Gain values from Crius1280 slot.
out	*paData*	Table associate data. NULL, or 60 bytes placeholder. paData is additional data associated to Gain or Offset array, which can be used freely by application for instance to keep a trace of Gain or Offset table calibration conditions, either sensitivity, either focal plane array temperature.

7.5.2.4 Proxy1280_12USB_LoadTableOffset()

```
eDALProxy1280_12USBErr Proxy1280_12USB_LoadTableOffset (
            HANDLE paHandle,
            unsigned char paiIndex,
            short * paTableOffset,
            void * paData )
```

Retrieve Crius1280 slot data as Offset values.
This function may failed if slot is empty, or slot data are not Offset values.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paiIndex*	Slot index as data.

Parameters

out	*paTableOffset*	Offset values from Crius1280 slot.
out	*paData*	Table associate data. NULL, or 60 bytes placeholder. paData is additional data associated to Gain or Offset array, which can be used freely by application for instance to keep a trace of Gain or Offset table calibration conditions, either sensitivity, either focal plane array temperature.

7.5.2.5 Proxy1280_12USB_LoadBadPixels()

```
eDALProxy1280_12USBErr Proxy1280_12USB_LoadBadPixels (
            HANDLE paHandle,
            unsigned short * paTableX,
            unsigned short * paTableY,
            unsigned short * paCount )
```

Retrieve bad pixel position in image from Crius1280.

Parameters

in	*paHandle*	Crius1280 handle.
out	*paTableX,paTableY*	Bad pixels position in image.
	paCount	Initial bad pixels array size, on return, bad pixel count.

Note

paCount must be init with paTableX / paTableY placeholder size (to avoid overflow), and will be modified by function with current bad pixel count.

7.5.2.6 Proxy1280_12USB_SaveTableGain()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SaveTableGain (
            HANDLE paHandle,
            unsigned char paiIndex,
            const float * paTableGain,
            void * paData )
```

Save Gain values into Crius1280 slot data.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paiIndex*	Slot index.
in	*paTableGain*	Gain values to store into Crius1280 slot.
in	*paData*	Table associate data. NULL, or 60 bytes placeholder. paData is additional data associated to Gain or Offset array, which can be used freely by application for instance to keep a trace of Gain or Offset table calibration conditions, either sensitivity, either focal plane array temperature.

7.5.2.7 Proxy1280_12USB_SaveTableOffset()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SaveTableOffset (
            HANDLE paHandle,
            unsigned char paiIndex,
            const short * paTableOffset,
            void * paData )
```

Save Offset values into Crius1280 slot data.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paiIndex*	Slot index.
in	*paTableOffset*	Offset values to store into Crius1280 slot.
in	*paData*	Table associate data. NULL, or 60 bytes placeholder. paData is additional data associated to Gain or Offset array, which can be used freely by application for instance to keep a trace of Gain or Offset table calibration conditions, either sensitivity, either focal plane array temperature.

7.5.2.8 Proxy1280_12USB_SaveBadPixels()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SaveBadPixels (
            HANDLE paHandle,
            const unsigned short * paTableX,
            const unsigned short * paTableY,
            unsigned short paCount )
```

Save bad pixel position into Crius1280 slot data.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paTableX,paTableY*	Bad pixels position in image.
in	*paCount*	bad pixels count.

7.5.2.9 Proxy1280_12USB_LoadCurrentTableGain()

```
eDALProxy1280_12USBErr Proxy1280_12USB_LoadCurrentTableGain (
            HANDLE paHandle,
            unsigned char paiIndex )
```

Use Crius1280 slot data as Gain values for NUC processing, i.e. retrieve it from Crius1280, and set it to NUC processing.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paiIndex*	Slot index as data.

7.5.2.10 Proxy1280_12USB_LoadCurrentTableOffset()

```
eDALProxy1280_12USBErr Proxy1280_12USB_LoadCurrentTableOffset (
            HANDLE paHandle,
            unsigned char paiIndex )
```

Use Crius1280 slot data as Offset values for NUC processing, i.e. retrieve it from Crius1280, and set it to NUC processing.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paiIndex*	Slot index as data.

7.5.2.11 Proxy1280_12USB_LoadCurrentBadPixels()

```
eDALProxy1280_12USBErr Proxy1280_12USB_LoadCurrentBadPixels (
            HANDLE paHandle )
```

Use Crius1280 stored bad pixel for bad pixel correction.

Parameters

in	*paHandle*	Crius1280 handle.

7.5.2.12 Proxy1280_12USB_SaveCurrentTableGain()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SaveCurrentTableGain (
            HANDLE paHandle,
            unsigned char paiIndex,
            const void * paData )
```

Save current Gain values into Crius1280 slot data.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paiIndex*	Slot index.
in	*paData*	Table associate data. NULL, or 60 bytes placeholder. paData is additional data associated to Gain or Offset array, which can be used freely by application for instance to keep a trace of Gain or Offset table calibration conditions, either sensitivity, either focal plane array temperature.

7.5.2.13 Proxy1280_12USB_SaveCurrentTableOffset()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SaveCurrentTableOffset (
            HANDLE paHandle,
            unsigned char paiIndex,
            const void * paData )
```

Save Offset values into Crius1280 slot data.

Parameters

in	*paHandle*	Crius1280 handle.
in	*paiIndex*	Slot index.
in	*paData*	Table associate data. NULL, or 60 bytes placeholder. paData is additional data associated to Gain or Offset array, which can be used freely by application for instance to keep a trace of Gain or Offset table calibration conditions, either sensitivity, either focal plane array temperature.

7.5.2.14 Proxy1280_12USB_SaveCurrentBadPixels()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SaveCurrentBadPixels (
            HANDLE paHandle )
```

Save bad pixel position into Crius1280 slot data.

Parameters

in	*paHandle*	Crius1280 handle.

7.5.2.15 Proxy1280_12USB_SaveCurrentShutterlessTables()

```
eDALProxy1280_12USBErr Proxy1280_12USB_SaveCurrentShutterlessTables (
            HANDLE paHandle )
```

Save Shutterless Tables into Crius1280.

Parameters

in	*paHandle*	Crius1280 handle.

Returns

eProxy1280_12USBFeatureNotAvailable error if Shutterless is unavailable on the module.

7.5.2.16 Proxy1280_12USB_LoadCurrentShutterlessTables()

```
eDALProxy1280_12USBErr Proxy1280_12USB_LoadCurrentShutterlessTables (
            HANDLE paHandle )
```

Load Shutterless Tables from Crius1280 in order to use it with shutterless processing

Parameters

in	*paHandle*	Crius1280 handle.

Returns

eProxy1280_12USBFeatureNotAvailable error if Shutterless is unavailable on the module.

7.5.2.17 Proxy1280_12USB_ClearSavedCalibrationData()

```
eDALProxy1280_12USBErr Proxy1280_12USB_ClearSavedCalibrationData (
            HANDLE paHandle,
            unsigned char paiType )
```

Clear Saved Calibration data format from Crius1280 in order to be able to save the other format (to avoid format mismatch error).

Parameters

in	*paHandle*	Crius1280 handle.
in	*paiType*	Calibration type. Calibration type values are : • 1 :Standard calibration. • 2 :Shutterless calibration.

7.6 Crius1280 Calibration

Crius1280 NUC, bad pixel and Shutterless correction calibration. Refer to calibration example provided with the SDK for detailed usage of the following function.

Functions

- eDALProxy1280_12USBErr Proxy1280_12USB_AbortCalibration (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_InitShutter2PtsCalibration (HANDLE paHandle, unsigned int iStage)
- eDALProxy1280_12USBErr Proxy1280_12USB_StepShutter2PtsCalibration (HANDLE paHandle, unsigned int iStage)
- eDALProxy1280_12USBErr Proxy1280_12USB_FinishShutter2PtsCalibration (HANDLE paHandle, unsigned int iStage)
- eDALProxy1280_12USBErr Proxy1280_12USB_InitShutterCalibration (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_StepShutterCalibration (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_FinishShutterCalibration (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_InitSLCalibrationT0 (HANDLE paHandle, unsigned int iStage)
- eDALProxy1280_12USBErr Proxy1280_12USB_StepSLCalibrationT0 (HANDLE paHandle, unsigned int iStage)
- eDALProxy1280_12USBErr Proxy1280_12USB_FinishSLCalibrationT0 (HANDLE paHandle, unsigned int iStage)
- eDALProxy1280_12USBErr Proxy1280_12USB_InitSLCalibrationT1 (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_StepSLCalibrationT1 (HANDLE paHandle)
- eDALProxy1280_12USBErr Proxy1280_12USB_FinishSLCalibrationT1 (HANDLE paHandle)

7.6.1 Detailed Description

Crius1280 NUC, bad pixel and Shutterless correction calibration. Refer to calibration example provided with the SDK for detailed usage of the following function.

NUC can be a two points calibration, or a one point calibration. Shutterless can be a T0 calibration only or T0 and T1 calibration.

This set of function provide 2 kind of NUC calibrations :

- Full Calibration.
- Fast Calibration.

And Shutterless Calibration compose of :

- T0 Calibration.
- T1 Calibration.

See also

User's Guide or Calibration Process for details.

7.6.2 Function Documentation

7.6.2.1 Proxy1280_12USB_AbortCalibration()

```
eDALProxy1280_12USBErr Proxy1280_12USB_AbortCalibration (
            HANDLE paHandle )
```

Abort a Calibration process and reset the sequencing. None of the corrections table will be change by the abort calibration.

Parameters

in	*paHandle*	Crius1280 Handle.

7.6.2.2 Proxy1280_12USB_InitShutter2PtsCalibration()

```
eDALProxy1280_12USBErr Proxy1280_12USB_InitShutter2PtsCalibration (
            HANDLE paHandle,
            unsigned int iStage )
```

Prepare NUC Calibration engine.

Parameters

in	*paHandle*	Crius1280 Handle.
in	*iStage*	Stage of the calibration (1 or 2).

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.3 Proxy1280_12USB_StepShutter2PtsCalibration()

```
eDALProxy1280_12USBErr Proxy1280_12USB_StepShutter2PtsCalibration (
            HANDLE paHandle,
            unsigned int iStage )
```

Add image for Shutter 2pts calibration.

Low temperature image for iStage = 1, High temperature image for iStage = 2.

Parameters

in	*paHandle*	Crius1280 Handle.
in	*iStage*	Stage of calibration (1 (low) or 2 (high)).

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.4 Proxy1280_12USB_FinishShutter2PtsCalibration()

```
eDALProxy1280_12USBErr Proxy1280_12USB_FinishShutter2PtsCalibration (
            HANDLE paHandle,
            unsigned int iStage )
```

Perform two points calibration using low and high temperature images.

Once calibration is done, new Gain, Offset and bad pixel are set to current NUC and BPC processing.

Parameters

in	*paHandle*	Crius1280 handle.
in	*iStage*	Stage of the calibration. If stage = 2, perform the final step of calibration. Once is done, new Gain, Offset and bad pixel are set to current NUC and BPC processing.

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.5 Proxy1280_12USB_InitShutterCalibration()

```
eDALProxy1280_12USBErr Proxy1280_12USB_InitShutterCalibration (
            HANDLE paHandle )
```

Prepare Shutter Calibration engine, also called one point calibration.

This calibration will only produce new Offset values.

Parameters

in	*paHandle*	Crius1280 handle.

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.6 Proxy1280_12USB_StepShutterCalibration()

```
eDALProxy1280_12USBErr Proxy1280_12USB_StepShutterCalibration (
            HANDLE paHandle )
```

Add image to prepare Shutter Calibration

Parameters

in	*paHandle*	Crius1280 handle.

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.7 Proxy1280_12USB_FinishShutterCalibration()

```
eDALProxy1280_12USBErr Proxy1280_12USB_FinishShutterCalibration (
            HANDLE paHandle )
```

Perform Shutter calibration.

Once calibration is done, Offset values are set to current NUC processing.

Parameters

in	*paHandle*	Crius1280 handle.

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.8 Proxy1280_12USB_InitSLCalibrationT0()

```
eDALProxy1280_12USBErr Proxy1280_12USB_InitSLCalibrationT0 (
            HANDLE paHandle,
            unsigned int iStage )
```

Initialise Stage for Shutterless Calibration T0

Must be use on correct sequencing with Crius1280 shutterless module.

Parameters

in	*paHandle*	Crius1280 handle.
in	*iStage*	Stage number of calibration.

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.9 Proxy1280_12USB_StepSLCalibrationT0()

```
eDALProxy1280_12USBErr Proxy1280_12USB_StepSLCalibrationT0 (
            HANDLE paHandle,
            unsigned int iStage )
```

Add image for Shutterless Calibration T0

Must be used on correct sequencing with Crius1280 shutterless module.

Parameters

in	*paHandle*	Crius1280 handle.
in	*iStage*	Stage number of calibration (1 = low temperature, 2 = high temperature).

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.10 Proxy1280_12USB_FinishSLCalibrationT0()

```
eDALProxy1280_12USBErr Proxy1280_12USB_FinishSLCalibrationT0 (
            HANDLE paHandle,
            unsigned int iStage )
```

Perform Shutterless T0 calibration

Must be used on correct sequencing with Crius1280 shutterless module. Once calibration is done, new Shutterless tables are set to current Shutterless processing.

Parameters

in	*paHandle*	Crius1280 handle.
in	*iStage*	Stage number of calibration (1 = low temperature, 2 = high temperature).

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.11 Proxy1280_12USB_InitSLCalibrationT1()

```
eDALProxy1280_12USBErr Proxy1280_12USB_InitSLCalibrationT1 (
            HANDLE paHandle )
```

Initialise Shutterless Calibration T1

Must be use on correct sequencing with Crius1280 shutterless module.

Parameters

in	*paHandle*	Crius1280 handle.

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.12 Proxy1280_12USB_StepSLCalibrationT1()

```
eDALProxy1280_12USBErr Proxy1280_12USB_StepSLCalibrationT1 (
            HANDLE paHandle )
```

Add image for Shutterless Calibration T1

Must be use on correct sequencing with Crius1280 shutterless module.

Parameters

in	*paHandle*	Crius1280 handle.

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.6.2.13 Proxy1280_12USB_FinishSLCalibrationT1()

```
eDALProxy1280_12USBErr Proxy1280_12USB_FinishSLCalibrationT1 (
            HANDLE paHandle )
```

Perform Shutterless Calibration T1

Must be use on correct sequencing with Crius1280 shutterless module. Once Calibration is done, new Shutterless T1 tables are set for current Shutterless processing.

Parameters

in	*paHandle*	Crius1280 handle.

Returns

eProxy1280_12USBSequencingError see Calibration Process for details.

7.7 Function return code

Function execution returned code.

Enumerations

- enum eDALProxy1280_12USBErr {
  eProxy1280_12USBSuccess =0 , eProxy1280_12USBParameterError , eProxy1280_12USBHandleError , eProxy1280_12USBInitFailed ,
  eProxy1280_12USBOpenFailed , eProxy1280_12USBCommFailed , eProxy1280_12USBTimeout , eProxy1280_12USBSyncBroken ,
  eProxy1280_12USBSequencingError , eProxy1280_12USBFeatureNotAvailable , eProxy1280_12USBBistInitFailure , eProxy1280_12USBBistFailure ,
  eProxy1280_12USBFormatFailed , eProxy1280_12USBFormatMismatch , **eProxy1280_12USBErrTotal** }

Functions

- const char * Proxy1280_12USB_GetErrorString (eDALProxy1280_12USBErr paeError)

7.7.1 Detailed Description

Function execution returned code.

eDALProxy1280_12USBErr is return by most functions as a result of execution.

See also

eDALProxy1280_12USBErr() to convert code to user friendly string.

7.7.2 Enumeration Type Documentation

7.7.2.1 eDALProxy1280_12USBErr

`enum eDALProxy1280_12USBErr`

Code return by most functions about execution.

Enumerator

eProxy1280_12USBSuccess	Function call success.
eProxy1280_12USBParameterError	Function call with wrong parameter.
eProxy1280_12USBHandleError	Function call with wrong or invalid Crius1280 handle.
eProxy1280_12USBInitFailed	Internal error occur.
eProxy1280_12USBOpenFailed	Open connection to Crius1280 failed. Maybe already connected
eProxy1280_12USBCommFailed	Exchange with Crius1280 failed.
eProxy1280_12USBTimeout	Operation on Crius1280 timeout before completed.
eProxy1280_12USBSyncBroken	GetImage(), Sync with Crius1280 broken.

Enumerator

eProxy1280_12USBSequencingError	Function call outside correct sequencing
eProxy1280_12USBFeatureNotAvailable	Feature not available on this module or can't be use due to present configuration.
eProxy1280_12USBBistInitFailure	Built-In Self Test initialisation failed.
eProxy1280_12USBBistFailure	Crius1280 reported a Built-In Self Test error.
eProxy1280_12USBFormatFailed	Incompatible file format for Crius1280.
eProxy1280_12USBFormatMismatch	Incompatible calibration format for Crius1280.

7.7.3 Function Documentation

7.7.3.1 Proxy1280_12USB_GetErrorString()

```
const char* Proxy1280_12USB_GetErrorString (
            eDALProxy1280_12USBErr paeError )
```

Convert eDALProxy1280_12USBErr to user message.

Parameters

in	*paeError*	Function returns error code.

Returns

User error message from eDALProxy1280_12USBErr.

Note

String is C-Style, i.e. Ascii with null terminate byte.

Index