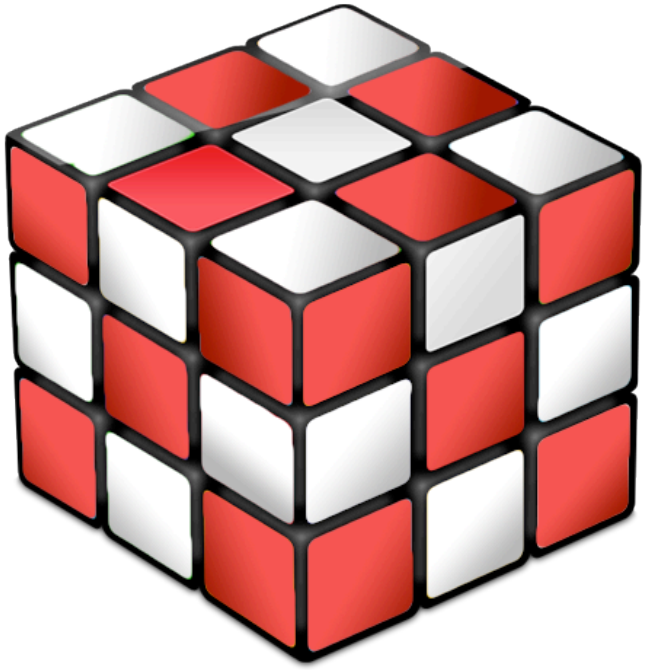
ADVACAM



PIXET

Developer Documentation

Table of Contents

1. Introduction

The **PIXet** is a multi-platform software developed in ADVACAM company. It is a basic software that allows measurement control and saving of measured data with Medipix detectors. It supports Medipix2, Medipix3, Timepix and Timepix3 detectors and all the readout-devices sold by ADVACAM company such as FitPIX, AdvaPIX, WidePIX, etc. It is written in C++ language and uses multi-platform Qt libraries.

This document describes a developer interface of the **PIXet** software. This developer interface consists of dynamic linked library **pxcore.dll** (Windows) or **libpxcore.so** (Mac or Linux), the corresponding header file for the library **pxcapi.h** and few other supporting libraries (fitpix.dll, Visual Studio runtime libraries, etc.).

2. API Functions

2.1 pxcInitialize

Summary

This function initializes the Pixet software and all connected devices. This function has to be called first before any other function except **pxcGetLastError**.

Definition

PXCAPI int **pxcInitialize**(int argc = 0, char const* argv[] = NULL)

Parameters

argc – number of program command line arguments (optional parameter)
argv – command line program arguments (optional parameter)

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code.

2.2 pxcExit

Summary

This function deinitializes Pixet software and all the connected devices. This function has to be called as last function before unloading the pxcore library.

Definition

PXCAPI int **pxcExit**()

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.3 pxcGetDevicesCount

Summary

This function returns number of connected and initialized devices.

Definition

PXCAPI int **pxcGetDevicesCount**()

Return Value

Number of devices, otherwise the return value is a PXCERR_XXX code

2.4 pxcRefreshDevices

Summary

This function looks for newly connected devices and removed disconnected devices from the device list.

Definition

PXCAPI int **pxcRefreshDevices**()

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code.

2.5 pxcReconnectDevice

Summary

If the device was disconnected or experienced communication problems, this function will try to reconnect the device and reinitialize it.

Definition

PXCAPI int **pxcReconnectDevice**(unsigned deviceIndex)

Parameters

deviceIndex - index of the device, starting from zero

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code.

2.6 pxcGetDeviceName

Summary

This function returns the full name of the selected device.

Definition

PXCAPI int **pxcGetDeviceName**(unsigned deviceIndex, char* nameBuffer, unsigned size)

Parameters

deviceIndex - index of the device, starting from zero
nameBuffer - buffer where the name of the device will be saved. Cannot be NULL
size - size of the supplied name buffer

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code.

Example

```
char nameBuffer[512];
int rc = pxcGetDeviceName(0, nameBuffer, 512);
printf("Device Name is: %s (Error code: %d)\n", nameBuffer, rc);
```

2.7 pxcGetDeviceChipCount

Summary

This function returns number of chips in the device.

Definition

PXCAPI int **pxcGetDeviceChipCount**(unsigned deviceIndex)

Parameters

deviceIndex - index of the device, starting from zero

Return Value

Number of chips if successful, otherwise the return value is a PXCERR_XXX code

2.8 pxcGetDeviceChipID

Summary

This function returns the ID of chip of the detector connected to the readout device.

Definition

PXCAPI int **pxcGetDeviceChipID**(unsigned deviceIndex, unsigned chipIndex, char* chipIDBuffer, unsigned size)

Parameters

deviceIndex - index of the device, starting from zero
chipIndex – index of the chip in the device, starting from zero
chipIDBuffer - buffer where the chipID of the detector will be saved. Cannot be NULL
size - size of the supplied chipID buffer

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.9 pxcGetBias

Summary

This function gets the high voltage (bias voltage) of the sensor on Medipix/Timepix chip.

Definition

PXCAPI int **pxcGetBias**(unsigned deviceIndex, double* bias)

Parameters

deviceIndex - index of the device, starting from zero
bias – pointer to double variable where current bias will be returned

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.10 pxcGetBiasRange

Summary

This function gets the range of the allowed minimal and maximal bias values.

Definition

PXCAPI int **pxcGetBiasRange**(unsigned deviceIndex, double* minBias, double* maxBias)

Parameters

deviceIndex - index of the device, starting from zero
minBias – pointer to double variable where minimum allowed bias will be returned
maxBias – pointer to double variable where maximum allowed bias will be returned

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.11pxcSetBias

Summary

This function sets the high voltage (bias) of the detector.

Definition

PXCAPI int **pxcSetBias**(unsigned deviceIndex, double bias)

Parameters

deviceIndex - index of the device, starting from zero
bias – high voltage in volts (0 to 100 V)

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.12 pxcGetThreshold

Summary

This function gets the threshold of the Medipix/Timepix detector in detector DAC values.

Definition

PXCAPI int **pxcGetThreshold**(unsigned deviceIndex, unsigned thresholdIndex,
double* threshold)

Parameters

deviceIndex - index of the device, starting from zero
thresholdIndex – for Timepix and Timepix3 always 0, for Medipix3 index of corresponding threshold starting from zero
threshold – pointer to double variable where threshold will be saved.
detector threshold (0 to 1024). The sense is reversed, for higher threshold lower number and the other way around

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.13 pxcGetThresholdRange

Summary

This function gets the allowed range of values for threshold

Definition

PXCAPI int **pxcGetThresholdRange**(unsigned deviceIndex, int thresholdIndex, double* minThreshold, double* maxThreshold)

Parameters

deviceIndex - index of the device, starting from zero
thresholdIndex – for Timepix and Timepix3 always 0, for Medipix3 index of corresponding threshold starting from zero
minThreshold – pointer to double variable where the minimal allowed threshold will be returned
maxThreshold – pointer to double variable where the maximal allowed threshold will be returned

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.14 pxcSetThreshold

Summary

This function sets the threshold of the detector in KeV.

Definition

PXCAPI int **pxcSetThreshold**(unsigned deviceIndex, unsigned thresholdIndex, double threshold)

Parameters

deviceIndex - index of the device, starting from zero
thresholdIndex - for Timepix and Timepix3 always 0, for Medipix3 index of corresponding threshold starting from zero
threshold – detector threshold in keV.

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.15 pxcGetDAC

Summary

This function gets a single DAC value of the detector.

Definition

PXCAPI int **pxcGetDAC**(unsigned deviceIndex, unsigned chipIndex, unsigned dacIndex, unsigned short* value);

Parameters

deviceIndex - index of the device, starting from zero
chipIndex – index of the chip, starting from zero
value – returned value

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.16 pxcSetDAC

Summary

This function sets a single DAC value of the detector.

Definition

PXCAPI int **pxcSetDAC**(unsigned deviceIndex, unsigned chipIndex, unsigned dacIndex, unsigned short value);

Parameters

deviceIndex - index of the device, starting from zero
chipIndex – index of the chip, starting from zero
value – new DAC value

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.17 pxcGetTimepixClock

Summary

This function gets the current value of measurement clock for Timepix detector (in MHz).

Definition

PXCAPI int **pxcGetTimepixClock**(unsigned deviceIndex, double* clock)

Parameters

deviceIndex - index of the device, starting from zero
clock – pointer to double variable where the clock will be saved

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.18 pxcSetTimepixClock

Summary

This function sets the value of measurement clock for Timepix detector (in MHz).

Definition

PXCAPI int **pxcSetTimepixClock**(unsigned deviceIndex, double clock)

Parameters

deviceIndex - index of the device, starting from zero
clock – new value of the measurement clock for Timepix detector

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.19 pxcGetTimepixMode

Summary

This function gets the current value of the Timepix mode (Counting, Energy,…)

Definition

PXCAPI int **pxcGetTimepixMode**(unsigned deviceIndex)

Parameters

deviceIndex - index of the device, starting from zero

Return Value

Timepix mode if successful, otherwise the return value is a PXCERR_XXX code.
Timepix mode can be:
PXC_TPX_MODE_MEDIPIX – counting mode
PXC_TPX_MODE_TOT – energy mode
PXC_TPX_MODE_TIMEPIX – timepix mode

2.20 pxcSetTimepixMode

Summary

This function sets the value of Timepix mode

Definition

PXCAPI int **pxcSetTimepixMode**(unsigned deviceIndex, int mode)

Parameters

deviceIndex - index of the device, starting from zero
mode – new value of the Timepix mode. One of the values:
PXC_TPX_MODE_MEDIPIX – counting mode
PXC_TPX_MODE_TOT – energy mode
PXC_TPX_MODE_TIMEPIX – timepix mode

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.21 pxcSetTimepixCalibrationEnabled

Summary

This function enables or disables the calibration of Timepix ToT counts to energy in keV

Definition

PXCAPI int **pxcSetTimepixCalibrationEnabled**(unsigned deviceIndex, bool enabled)

Parameters

deviceIndex - index of the device, starting from zero
enabled – if the calibration is enabled or disable

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.22 pxcIsTimepixCalibrationEnabled

Summary

This function returns if the calibration of Timepix ToT counts to energy in keV is enabled

Definition

PXCAPI int **pxcIsTimepixCalibrationEnabled**(unsigned deviceIndex)

Parameters

deviceIndex - index of the device, starting from zero

Return Value

0 if disabled, greater than 0 enabled, negative value a PXCERR_XXX code

2.23 pxcLoadDeviceConfiguration

Summary

This function loads device configuration from xml file

Definition

PXCAPI int **pxcLoadDeviceConfiguration**(unsigned deviceIndex, const char* filePath)

Parameters

deviceIndex - index of the device, starting from zero
filePath – path to xml configuration file

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.24 pxcSaveDeviceConfiguration

Summary

This function saves device configuration to xml file

Definition

PXCAPI int **pxcSaveDeviceConfiguration**(unsigned deviceIndex, const char* filePath)

Parameters

deviceIndex - index of the device, starting from zero
filePath – path to xml configuration file

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.25 pxcSetupTestPulseMeasurement

Summary

Enables / Disables and setups parameters of the test pulse measurements

Definition

PXCAPI int **pxcSetupTestPulseMeasurement**(unsigned deviceIndex, bool tpEnabled,
double height, double period,
unsigned count, unsigned spacing);

Parameters

deviceIndex - index of the device, starting from zero
tpEnabled - enables/disables test pulse measurement (in functions Measure..Frame(s))
height – test pulse height (0 – 1.5 V)
period – single test pulse period (1 – 256 us)
count – number of test pulses (1 – 10000)
spacing – spacing that is used during measurement (sub acquisition), good value is 4

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.26 pxcMeasureSingleFrame

Summary

Performs a measurement of single frame and returns its data

Definition

PXCAPI int **pxcMeasureSingleFrame**(unsigned deviceIndex, double frameTime,
unsigned short* frameData, unsigned* size,
unsigned trgstg)

Parameters

deviceIndex - index of the device, starting from zero
frameTime - time of the measurment in seconds
frameData - pointer to buffer where data will be saved. For single detector size is 65536
size - pointer to varible with the size of the buffer. The actual size will be output to this variable
trgStg – settings of external trigger - one of the PXC_TRG_XXX values. Default PXC_TRG_NO.

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.27 pxcMeasureSingleFrameMpx3

Summary

Performs a measurement of single frame and returns its data. This is only for Medipix3 chips

Definition

PXCAPI int **pxcMeasureSingleFrame**(unsigned deviceIndex, double frameTime,
unsigned* frameData1, unsigned* frameData2,
unsigned* size, unsigned trgstg)

Parameters

deviceIndex - index of the device, starting from zero
frameTime - time of the measurment in seconds
frameData1 - pointer to buffer where data from first counter will be saved. For single detector size is 65536
frameData2 - pointer to buffer where data from second counter will be saved. For single detector size is 65536
size - pointer to varible with the size of the buffer. The actual size will be output to this variable
trgStg – settings of external trigger - one of the PXC_TRG_XXX values. Default PXC_TRG_NO.

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.28 pxcMeasureSingleFrameTpx3

Summary

Performs a measurement of single frame and returns its data. This is only for Timepix3 detector.

Definition

PXCAPI int **pxcMeasureSingleFrame**(unsigned deviceIndex, double frameTime,
double* frameToaITot, unsigned short* frameTotEvent,
unsigned* size, unsigned trgstg)

Parameters

deviceIndex - index of the device, starting from zero
frameTime - time of the measurment in seconds
frameToaITot - pointer to buffer where data from ToA or iToT counter (based on set operation mode) will be saved. For single detector size is 65536
frameTotEvent - pointer to buffer where data from ToT or Event counter (based on set operation mode) will be saved. For single detector size is 65536
size - pointer to varible with the size of the buffer. The actual size will be output to this variable
trgStg – settings of external trigger - one of the PXC_TRG_XXX values. Default PXC_TRG_NO.

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.29 pxcMeasureMultipleFrames

Summary

Performs a measurement of several frames to memory

Definition

PXCAPI int **pxcMeasureMultipleFrames**(unsigned deviceIndex, unsigned fameCount,
double frameTime, unsigned trgStg)

Parameters

deviceIndex - index of the device, starting from zero
frameCount - numeber of frames to measure
frameTime - time of the measurment in seconds
trgStg – settings of external trigger - one of the PXC_TRG_XXX values. Default PXC_TRG_NO.

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.30 pxcMeasureMultipleFramesWithCallback

Summary

Performs a measurement of several frames to memory. When each frame is measured, the supplied callback function is called and the userData parameter is passed as argument.

Definition

PXCAPI int **pxcMeasureMultipleFramesWithCallback**(unsigned deviceIndex, unsigned fameCount, double frameTime, unsigned trgStg, FrameMeasuredCallback callback, intptr_t userData)

Parameters

deviceIndex - index of the device, starting from zero
frameCount - numeber of frames to measure
frameTime - time of the measurment in seconds
trgStg – settings of external trigger - one of the PXC_TRG_XXX values. Default PXC_TRG_NO.
callback – pointer to function of FrameMeasuredCallback type
userData – pointer to some user object/memory that is passed in callback function

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.31 pxcMeasureContinuous

Summary

Performs an "endless" measurement of several frames to memory. The measurement is run until it's aborted by pxcAbortMeasurement function. When each frame is measured, the supplied callback function is called and the userData parameter is passed as argument.

Definition

PXCAPI int **pxcMeasureContinuous**(unsigned deviceIndex, unsigned fameBufferSize, double frameTime, unsigned trgStg, FrameMeasuredCallback callback, intptr_t userData)

Parameters

deviceIndex - index of the device, starting from zero
frameTime - time of the measurment in seconds
frameBufferSize - numeber of frames in circular buffer
trgStg – settings of external trigger - one of the PXC_TRG_XXX values. Default PXC_TRG_NO.
callback – pointer to function of FrameMeasuredCallback type
userData – pointer to some user object/memory that is passed in callback function

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.32 pxcMeasureTpx3DataDrivenMode

Summary

Performs a measurement with Timepix3 detector in Data Driven Mode (event by event mode, when stream of pixels is sent).

Definition

PXCAPI int **pxcMeasureTpx3DataDrivenMode**(unsigned deviceIndex, unsigned measTime, const char* filename, unsigned trgStg, AcqEventFunc callback, intptr_t userData)

Parameters

deviceIndex - index of the device, starting from zero
measTImeTime – the total time of the measurement in seconds
filename – output file name and path (extensions must end *.t3pa, *.trp, *.t3r)
trgStg – settings of external trigger - one of the PXC_TRG_XXX values. Default PXC_TRG_NO.
callback – pointer to function of acqEventFunc type
userData – pointer to some user object/memory that is passed in callback function

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.33 pxcAbortMeasurement

Summary

Stopts the currently running measurement.

Definition

PXCAPI int **pxcAbortMeasurement**(unsigned deviceIndex)

Parameters

deviceIndex - index of the device, starting from zero

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.34 pxcGetMeasuredFrameCount

Summary

Returns number of measured frames in memory

Definition

PXCAPI int **pxcGetMeasuredFrameCount**(unsigned deviceIndex)

Return Value

Number of measured frames, otherwise the return value is a PXCERR_XXX code

2.35 pxcSaveMeasuredFrame

Summary

Saves the measured frame to a file on the harddrive

Definition

PXCAPI int **pxcSaveMeasuredFrame**(unsigned deviceIndex, unsigned frameIndex, const char* filePath)

Parameters

deviceIndex - index of the device, starting from zero
frameIndex – index of the frame, starting from zero
filePath – path to the file where frame will be saved

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.36 pxcGetMeasuredFrame

Summary

Gets data of specified measured frame from memory

Definition

PXCAPI int **pxcGetMeasuredFrame**(unsigned deviceIndex, unsigned fameIndex, unsigned short* frameData, unsigned* size)

Parameters

deviceIndex - index of the device, starting from zero
frameIndex – index of the frame, starting from zero
frameData - pointer to buffer where data will be saved. For single detector size is 65536
size - pointer to varible with the size of the buffer. The actual size will be output to this variable

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.37 pxcGetMeasuredFrameMpx3

Summary

Gets data of specified measured frame from memory. For Medipix3 chip

Definition

PXCAPI int **pxcGetMeasuredFrameMpx3**(unsigned deviceIndex, unsigned fameIndex,
unsigned* frameData1, unsigned* frameData2,
unsigned* size)

Parameters

deviceIndex - index of the device, starting from zero
frameIndex – index of the frame, starting from zero
frameData1 - pointer to buffer where data from first counter will be saved. For single detector size is 65536
frameData2 - pointer to buffer where data from second counter will be saved. For single detector size is 65536
size - pointer to varible with the size of the buffer. The actual size will be output to this variable

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.38 pxcGetMeasuredFrameTpx3

Summary

Gets data of specified measured frame from memory. For Timepix3 chip

Definition

PXCAPI int **pxcGetMeasuredFrameTpx3**(unsigned deviceIndex, unsigned fameIndex,
double* frameToaITot,
unsigned short* frameToTEvent, unsigned* size)

Parameters

deviceIndex - index of the device, starting from zero
frameIndex – index of the frame, starting from zero
frameToaITot - pointer to buffer where data from ToA or iToT counter (based on set operation mode) will be saved. For single detector size is 65536
frameTotEvent - pointer to buffer where data from ToT or Event counter (based on set operation mode) will be saved. For single detector size is 65536
size - pointer to varible with the size of the buffer. The actual size will be output to this variable

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.39 pxcGetMeasuredTpx3PixelsCount

Summary

Gets the number of measured Timepix3 pixels in data driven mode

Definition

PXCAPI int **pxcGetMeasuredTpx3PixelsCount**(unsigned deviceIndex, unsigned* pixelCount)

Parameters

deviceIndex - index of the device, starting from zero
pixelCount – pointer to unsigned variable where number of pixel count is set

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.40 pxcGetMeasuredTpx3Pixels

Summary

Gets the measured Timepix3 pixels data

Definition

PXCAPI int **pxcGetMeasuredTpx3Pixels** (unsigned deviceIndex, Tpx3Pixel* pixels, unsigned pixelCount)

Parameters

deviceIndex - index of the device, starting from zero
pixels – pointer to array of Tpx3Pixels that will be filled with measured pixels
pixelCount – size of the supplied array as number of pixels

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.41 pxcGetDeviceParameter

Summary

Returns the value of device parameter (e.g. settings of trigger)

Definition

PXCAPI int **pxcGetDeviceParameter**(unsigned deviceIndex, const char* parameterName)

Parameters

deviceIndex - index of the device, starting from zero
parameterName – name of the device parameter

Return Value

Value of the device or PXCERR_XXX code if error occurs

2.42 pxcSetDeviceParameter

Summary

Sets a value of the device parameter (e.g. settings of trigger)

Definition

PXCAPI int **pxcSetDeviceParameter**(unsigned deviceIndex, const char* parameterName,
Int parameterValue)

Parameters

deviceIndex - index of the device, starting from zero
parameterName – name of the device parameter
parameterValue – new value of the parameter

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.43 pxcGetDeviceParameterDouble

Summary

Returns the value of device double parameter

Definition

PXCAPI int **pxcGetDeviceParameterDouble**(unsigned deviceIndex,
const char* parameterName, double* parameterValue)

Parameters

deviceIndex - index of the device, starting from zero
parameterName – name of the device parameter
parameterValue – pointer to double variable where the parameter value will be saved

Return Value

PXCERR_XXX code if error occurs

2.44 pxcSetDeviceParameterDouble

Summary

Sets a value of the device double parameter

Definition

PXCAPI int **pxcSetDeviceParameterDouble**(unsigned deviceIndex, const char* parameterName,
double parameterValue)

Parameters

deviceIndex - index of the device, starting from zero
parameterName – name of the device parameter
parameterValue – new value of the parameter

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.45 pxcGetDeviceParameterString

Summary

Returns the value of device string parameter

Definition

PXCAPI int **pxcGetDeviceParameterString**(unsigned deviceIndex,
const char* parameterName, const char* parameterValue, unsigned size)

Parameters

deviceIndex - index of the device, starting from zero
parameterName – name of the device parameter
parameterValue – pointer to string buffer where the parameter value will be saved
size – size of the passed buffer

Return Value

PXCERR_XXX code if error occurs

2.46 pxcSetDeviceParameterString

Summary

Sets a value of the device string parameter

Definition

PXCAPI int **pxcSetDeviceParameterString**(unsigned deviceIndex, const char* parameterName,

const char* parameterValue)

Parameters

deviceIndex - index of the device, starting from zero
parameterName – name of the device parameter
parameterValue – new value of the parameter

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.47 pxcSetTimepix3Mode

Summary

Sets the operation mode of Timepix3 detector

Definition

PXCAPI int **pxcSetTimepix3Mode**(unsigned deviceIndex, int mode)

Parameters

deviceIndex - index of the device, starting from zero
mode – mode of the detector PXC_TPX3_OPM_XXX values

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.48 pxcSetMedipix3OperationMode

Summary

Sets the operation mode of Medipix3 detector

Definition

PXCAPI int **pxcSetMedipix3OperationMode**(unsigned deviceIndex, int opMode)

Parameters

deviceIndex - index of the device, starting from zero
opMode – mode of the detector PXC_MPX3_OPM_XXX values

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.49pxcSetMedipix3GainMode

Summary

Sets the gain mode of Medipix3 detector

Definition

PXCAPI int **pxcSetMedipix3GainMode**(unsigned deviceIndex, int gain)

Parameters

deviceIndex - index of the device, starting from zero
gain – mode of the detector PXC_MPX3_GAIN_MOD_XXX values

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.50 pxcSetMedipix3AcqParams

Summary

Sets acquisition parameters for Medipix3

Definition

PXCAPI int **pxcSetMedipix3AcqParams**(unsigned deviceIndex, bool colorMode,
bool csm, int gain, bool equalize)

Parameters

deviceIndex - index of the device, starting from zero
colorMode – if color mode is enabled
csm – if charge sharing mode is enabled
gain – gain settings (PXC_MPX3_GAIN_XXX values)
equalize – if equalization bit in Medipix3 is enabled

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.51 pxcSetMedipix3MatrixParams

Summary

Sets parameters of the Meidpix3 pixel matrix

Definition

PXCAPI int **pxcSetMedipix3MatrixParams**(unsigned deviceIndex, int depth,
int counter, int colBlock, int rowBlock)

Parameters

deviceIndex - index of the device, starting from zero
depth – depth of the counters PXC_MPX3_CNTD_XXX values
counter – selected counter (PXC_MPX3_CNT_XXX values)
colBlock – region of interest readout (PXC_MPX3_COLB_XXX values)
rowBlock – region of interest readout (PXC_MPX3_ROWB_XXX values)

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.52 pxcSetPixelMatrix

Summary

Sets the pixel matrix configuration. This is low level function for advanced users.

Definition

PXCAPI int **pxcSetPixelMatrix** (unsigned deviceIndex, unsigned char* pixCfgData,
unsigned byteSize)

Parameters

deviceIndex - index of the device, starting from zero
pixCfgData – pixel matrix configuration data
byteSize – size of the data in bytes for size check

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.53 pxcRegisterAcqEvent

Summary

Registers an acquisition event callback that is called when corresponding event occurs

Definition

PXCAPI int **pxcRegisterAcqEvent**(unsigned deviceIndex, const char* event,
AcqFunc func, intptr_t userData)

Parameters

deviceIndex - index of the device, starting from zero
event – event name (PXC_ACQEVENT_XXX values)
func – callback function of type AcqFunc
userData – user data that are passed to callback function

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.54 pxcUnregisterAcqEvent

Summary

Unregisters the acquisition event callback

Definition

PXCAPI int **pxcUnregisterAcqEvent**(unsigned deviceIndex, const char* event,
AcqFunc func, intptr_t userData)

Parameters

deviceIndex - index of the device, starting from zero
event – event name (PXC_ACQEVENT_XXX values)
func – callback function of type AcqFunc
userData – user data that are passed to callback function

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.55 pxcSetSensorRefresh

Summary

Sets the sensor refresh sequence text

Definition

PXCAPI int **pxcSetSensorRefresh**(unsigned deviceIndex, const char* refreshString)

Parameters

deviceIndex - index of the device, starting from zero
refreshString– sensor refresh string

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.56 pxcDoSensorRefresh

Summary

Performs the sensor refresh

Definition

PXCAPI int **pxcDoSensorRefresh**(unsigned deviceIndex)

Parameters

deviceIndex - index of the device, starting from zero

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.57 pxcEnableSensorRefresh

Summary

Enables automatic sensor refresh before each acquisition series and at periodic intervals

Definition

PXCAPI int **pxcEnableSensorRefresh**(unsigned deviceIndex, bool enabled, double refreshTime)

Parameters

deviceIndex - index of the device, starting from zero
enabled – if automatic sensor refresh is enabled
refreshTime – sensor refresh is performed repeatedly after this time in seconds. If thime is 0, then the refresh is done only once before the measurement

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.58 pxcEnableTDI

Summary

Enables TDI (Time Delayed Integration) measurement (if device supports it)

Definition

PXCAPI int **pxcEnableTDI**(unsigned deviceIndex, bool enabled)

Parameters

deviceIndex - index of the device, starting from zero
enabled – if TDI is enabled

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.59 pxcAddBHMask

Summary

Adds a new mask (frame) for Beam-Hardening calibration

Definition

PXCAPI int **pxcAddBHMask**(unsigned* data, unsigned size, double frameTime, double thickness)

Parameters

data – data of the frame tat will be used as BH mask
size – size of the data - number of pixels (width * height)
frameTime – acquisition time of the frame in seconds
thickness – thickness of the measured data

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.60pxcBHMaskCount

Summary

Returns number of inserted Beam-Hardening masks (frames)

Definition

PXCAPI int **pxcBHMaskCount**()

Return Value

Number of masks if successful, otherwise the return value is a PXCERR_XXX code

2.61 pxcRemoveBHMask

Summary

Removes Beam-Hardening mask (frame)

Definition

PXCAPI int **pxcRemoveBHMask**(int index)

Parameters

index – index of the mask to remove

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.62 pxcApplyBHCorrection

Summary

Applies the Beam-Hardening correction to supplied frame

Definition

PXCAPI int **pxcApplyBHCorrection**(unsigned* inData, unsigned size, double frameTime, double* outData)

Parameters

inData – data of the frame tat will be corrected
size – size of the data - number of pixels (width * height)
frameTime – acquisition time of the measured frame in seconds
outData – output data buffer where corrected data will be saved

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

2.63 pxcGetLastError

Summary

Returns text of last error. This function can be called even before pxcInitialize()

Definition

PXCAPI int **pxcGetLastError**(char* errorMsgBuffer, unsigned size)

Parameters

errorMsgBuffer - buffer where text will be saved
size - size of supplied buffer

Return Value

0 if successful, otherwise the return value is a PXCERR_XXX code

3. Appendix

3.1 FitPIX device parameters

TriggerStg

settings of the trigger

Values:

- 0 - trigger reacts to logical 0 (0 V)
- 1 - trigger reacts to logical 1 (5 V)
- 2 - trigger reacts to rising edge
- 3 - trigger reacts to falling edge

TriggerWaitForReady

If the FitPIXes are connected in chain, the device waits for ready signal from the preceding device in the chain.

Values:

- 0 - does not wait
- 1 - waits

TriggerMaster

Sets the device to be the first one in the chain. If devices are connected in chain, master device has the external trigger connected.

Values:

- 0 - is not master device
- 1 - is master device

TriggerOutLevel

Sets the active level of Trigger Out pin

Values:

- 0 – logical 0 (0V)
- 1 – logical 1 (5V)

ADVACAM s.r.o.
U Pergamenky 1145/12, CZ 170 00 Praha 7
Czech Republic

Tel: +420-603-444112, 589854;
Email: info@advacam.com
www.advacam.com

For more information visit ADVACAM website at www.advacam.com
For any question send an email to info@advacam.com