Single Photon Counting Camera

SPC^3

Version 1.0

Software Development Kit

Manual

June, 2015

Contents

Chapter 1

Single Photon Counting Camera Software Development Kit (SPC3-SDK).

SPC3 is a 2D imaging chip based on a 64 x 32 array of smart pixels. Each pixel comprises a single-photon avalanche diode detector, an analog front-end and a digital processing electronics. This on-chip integrated device provides single-photon sensitivity, high electronic noise immunity, and fast readout speed. The imager can be operated at a maximum of about 100.000 frame per second with negligible dead-time between frames. It features high photon-detection efficiency in the visible spectral region, and low dark-counting rates, even at room temperature. The imager is easily integrated into different applications thanks to the input optical adapter and a high-speed USB 3.0 computer interface.The camera differs from conventional CCD or CMOS sensors because it performs a fully digital acquisition of the light signal. Each pixel effectively counts the number of photons which are detected by the sensor during the acquisition time.

IMPORTANT In order to execute a program which links to the SDK libraries, a set of DLL should be placed in the same directory as the executable. The list of the required files is:

```
SPC3_SDK.dll       Software development kit interface
okFrontPanel.dll           Low-level interface
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Module Documentation

4.1 SPC3-SDK custom Types

Typedefs

- typedef struct _SPC3_H * SPC3_H
- typedef unsigned char * BUFFER_H

Enumerations

- enum SPC3Return {
  OK = 0, USB_DEVICE_NOT_RECOGNIZED = -1, CAMERA_NOT_POWERING_UP =-3, COMMUNICATION_ERROR =-5,
  OUT_OF_BOUND = -6, MISSING_DLL = -7, EMPTY_BUFFER = -8, NOT_EN_MEMORY = -9,
  NULL_POINTER = -10, INVALID_OP = -11, UNABLE_CREATE_FILE = -12, UNABLE_READ_FILE = -13,
  FIRMWARE_NOT_COMPATIBLE =-14, POWER_SUPPLY_ERROR = -15, TOO_MUCH_LIGHT = -16, INVALID_NIMG_CORRELATION = -17,
  SPC3_MEMORY_FULL = -18 }
- enum OutFileFormat { SPC3_FILEFORMAT = 0, TIFF_LZW_COMPRESSION = 1, TIFF_NO_COMPRESSION = 2 }
- enum GateMode { Continuous = 0, Pulsed = 1 }
- enum CameraMode { Normal = 0, Advanced = 1 }
- enum TriggerMode { None = 0, Gate_Clk = 1, Frame = 2 }
- enum State { **Disabled** = 0, **Enabled** = 1 }
- enum CorrelationMode { Linear = 0, MultiTau = 1 }

4.1.1 Detailed Description

Custom types used by the SDK.

4.1.2 Typedef Documentation

4.1.2.1 typedef unsigned char* BUFFER_H

Handle to the SPC3 buffer.

4.1.2.2 typedef struct _SPC3_H* SPC3_H

Handle to the SPC3 structure.

4.1.3 Enumeration Type Documentation

4.1.3.1 enum CameraMode

SPC3 working mode.

The camera contains for each pixel an 8-bit binary counter. If the exposure time is too long, the counter can overflow and generate a distorted image. Therefore, two operating modes have been implemented: a "normal" one which prevents the overflow of the counters, and an advanced one which gives full control of the camera to the user.

Enumerator

Normal The camera settings are tuned by the software to avoid the overflow of the counters. In this working mode, the exposure time of each image is fixed to a multiple of 20.74 microseconds. Longer exposures are obtained by integrating more frames.

Advanced The user has full control of the camera settings. **WARNING:** the counters can overflow.

4.1.3.2 enum CorrelationMode

Type of correlation function.

The SDK implements two autocorrelation algorithms which can be applied to the acquired sequence of images. The multi-tau autocorrelation has been implemented according to Culbertson and Burden "A distributed algorithm for multi-tau autocorrelation.", Rev Sci Instrum 78, 044102 (2007) (standard version) and the linear one similar to Press, Teukolsky, Vetterling and Flannery,"Numerical Recipes 3rd Edition: The Art of Scientific Computing.", (2007) "autocor.cpp".

Enumerator

Linear Selects the linear correlation algorithm.

MultiTau Selects the linear multi-tau algorithm.

4.1.3.3 enum GateMode

Gate setting.

Enable and disable the software gating. When the setting is Enabled, the SPC3 discards the detected photons by the SPAD matrix if measured outside a valid gate signal.

Enumerator

Continuous The gate signal is always valid.

Pulsed The gate signal is a a square wave. The photons, which are detected when the gate signal is "ON", are counted. Otherwise they are discarded.

4.1.3.4 enum OutFileFormat

Output file format.

Table of the available output file formats for the saved images

Enumerator

SPC3_FILEFORMAT SPC3 custom file format: the first byte contains the value 8 or 16 to define whether the image has 8 or 16 bit per pixel. Then the pixel values follow in row-major order. The byte order is little-endian for the 16 bit images.

TIFF_LZW_COMPRESSION Multipage TIFF files LZW compressed. The file follows the OME-TIFF specification. It may be read with any reader able to open TIFF file, but OME-TIFF compatible reader will also show embedded metadata on acquisition parameters. For more information, see the OME-TIFF web site: http://www.openmicroscopy.org/site/support/ome-model/ome-tiff/ **WARNING** the creation of TIFF files might require long execution times.

TIFF_NO_COMPRESSION Multipage TIFF without compression. The file follows the OME-TIFF specification. It may be read with any reader able to open TIFF file, but OME-TIFF compatible reader will also show embedded metadata on acquisition parameters. For more information, see the OME-TIFF web site: http://www.openmicroscopy.org/site/support/ome-model/ome-tiff/ **WARNING** the creation of TIFF files might require long execution times.

4.1.3.5 enum SPC3Return

Error table.

Error code returned by the SPC3 functions.

Enumerator

OK The function returned successfully.

USB_DEVICE_NOT_RECOGNIZED The USB device driver has not been initialized. Is there any device connected?

CAMERA_NOT_POWERING_UP Internal power supply is not powering up. Check connections and restart the camera. If problems persists contact MPD.

COMMUNICATION_ERROR Communication error during readout. Check USB connection.

OUT_OF_BOUND One or more parameters passed to the function are outside the valid boundaries.

MISSING_DLL One or more SPC3 libraries are missing.

EMPTY_BUFFER An empty buffer image has been provided to the function.

NOT_EN_MEMORY Not enough memory is available to operate the camera.

NULL_POINTER A null pointer has been provided to the function.

INVALID_OP The required function can not be executed. The device could be still in "Live mode".

UNABLE_CREATE_FILE An output file can not be created.

UNABLE_READ_FILE The provided file can not be accessed.

FIRMWARE_NOT_COMPATIBLE The camera firmware is not compatible with the current software.

POWER_SUPPLY_ERROR Voltage drop on internal power supply. Check connections and restart the camera. If problems persists contact MPD.

TOO_MUCH_LIGHT Too much light was detected by the camera. The protection mechanism has been enabled. Decrease the amount of light on the sensor. Then, disconnect and reconnect the camera to the USB port.

INVALID_NIMG_CORRELATION The acquired number of images is not sufficient to calculate the required correlation function.

SPC3_MEMORY_FULL The SPC3 internal memory got full during continous acquisition. Possible data loss.

4.1.3.6 enum TriggerMode

Type of synchronization output.

The Synch-out SMA port can output different signals

Enumerator

None No output signal.

Gate_Clk A square wave of 50 MHz and 50% duty cycle synchronized with the software gate signal and the camera clock.

Frame A 60 ns pulse every time a new frame is acquired.

4.2 Constructr, destructor and error handling

Functions

- DllSDKExport SPC3Return SPC3_Constr (SPC3_H *spc3_in, CameraMode m, char *Device_ID)
- DllSDKExport SPC3Return SPC3_Destr (SPC3_H spc3)
- DllSDKExport void PrintErrorCode (FILE *fout, const char *FunName, SPC3Return retcode)

4.2.1 Detailed Description

Functions to construct and destruct SPC3 objects, and for error handling.

4.2.2 Function Documentation

4.2.2.1 DllSDKExport void PrintErrorCode (FILE * *fout*, const char * *FunName*, SPC3Return *retcode*)

Print an error message.

All the SDK functions return an error code to inform the user whether the issued command was successfully executed or not. The result of the execution of a function can be redirect to a text file by providing a valid file pointer.

Parameters

fout	Output text file
FunName	Additional text to define the warning/error. Usually the name of the calling function is provided.
retcode	Error code returned by a SDK command

4.2.2.2 DllSDKExport SPC3Return SPC3_Constr (SPC3_H * *spc3_in*, CameraMode *m*, char * *Device_ID*)

Constructor.

It allocates a memory block to contain all the information and buffers required by the SPC3. If multiple devices are connected to the computer, a unique Device ID should be provided to correctly identify the camera. The camera ID can be found in the camera documentation (9 numbers and a letter) and it is printed on the screen during initialization. An empty string is accepted too. In this case, the devices will be connected in the order which is printed on the screen.

Parameters

spc3_in	Pointer to SPC3 handle
m	Camera Working mode
Device_ID	Unique ID to identify the connected device

Returns

OK
INVALID_OP The SPC3_H points to an occupied memory location
FIRMWARE_NOT_COMPATIBLE The SDK and Firmware versions are not compatible
NOT_EN_MEMORY There is not enough memory to run the camera

Examples:

SDK_Example.c.

4.2.2.3 DllSDKExport SPC3Return SPC3_Destr (SPC3_H *spc3*)

Destructor.

It deallocates the memory block which contains all the information and buffers required by the SPC3. **WARNING** the user must call the destructor before the end of the program to avoid memory leakages.

Parameters

spc3	SPC3 handle

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

Examples:

SDK_Example.c.

4.3 Set methods

Functions

- DllSDKExport SPC3Return SPC3_Set_Camera_Par (SPC3_H spc3, UInt16 Exposure, UInt32 NFrames, UInt16 NIntegFrames, UInt16 NCounters, State Force8bit, State Half_array, State Signed_data)
- DllSDKExport SPC3Return SPC3_Set_DeadTime (SPC3_H spc3, UInt16 Val)
- DllSDKExport SPC3Return SPC3_Set_DeadTime_Correction (SPC3_H spc3, State s)
- DllSDKExport SPC3Return SPC3_Set_Advanced_Mode (SPC3_H spc3, State s)
- DllSDKExport SPC3Return SPC3_Set_Background_Img (SPC3_H spc3, UInt16 *Img)
- DllSDKExport SPC3Return SPC3_Set_Background_Subtraction (SPC3_H spc3, State s)
- DllSDKExport SPC3Return SPC3_Set_Gate_Values (SPC3_H spc3, Int16 Shift, Int16 Length)
- DllSDKExport SPC3Return SPC3_Set_Gate_Mode (SPC3_H spc3, GateMode Mode)
- DllSDKExport SPC3Return SPC3_Set_Trigger_Out_State (SPC3_H spc3, TriggerMode Mode)
- DllSDKExport SPC3Return SPC3_Set_Sync_In_State (SPC3_H spc3, State s)
- DllSDKExport SPC3Return SPC3_Set_Live_Mode_ON (SPC3_H spc3)
- DllSDKExport SPC3Return SPC3_Set_Live_Mode_OFF (SPC3_H spc3)
- DllSDKExport SPC3Return SPC3_Set_FLIM_Par (SPC3_H spc3, UInt16 FLIM_steps, UInt16 FLIM_shift, Int16 FLIM_start, UInt16 Length, int *FLIM_frame_time, double *FLIM_bin_width)
- DllSDKExport SPC3Return SPC3_Set_FLIM_State (SPC3_H spc3, State FLIM_State)
- DllSDKExport SPC3Return SPC3_Apply_settings (SPC3_H spc3)

4.3.1 Detailed Description

Functions to set parameters of the SPC3 camera.

4.3.2 Function Documentation

4.3.2.1 DllSDKExport SPC3Return SPC3_Apply_settings (SPC3_H *spc3*)

Apply settings to the camera.

This function must be called after any Set function, except SPC3_Set_Live_Mode_ON() and SPC3_Set_Live_Mode_OFF(), in order to apply the settings to the camera.

Parameters

spc3	SPC3 handle

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

Examples:

SDK_Example.c.

4.3.2.2 DllSDKExport SPC3Return SPC3_Set_Advanced_Mode (SPC3_H *spc3,* State *s*)

Change the operating mode.

Set the operating mode to Normal or Advanced. Normal mode is the default setting.

Parameters

spc3	SPC3 handle
s	Enable or disable the advanced mode

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

4.3.2.3 DllSDKExport SPC3Return SPC3_Set_Background_Img (SPC3_H *spc3*, UInt16 * *Img*)

Load a background image to perform hardware background subtraction.

The control electronics is capable of performing real-time background subtraction. A background image is loaded into the internal camera memory.

Parameters

spc3	SPC3 handle
Img	Pointer to a 2048 UInt16 array containing the background image. **WARNING** The user should check the array size to avoid the corruption of the memory heap.

Returns

OK
NULL_POINTER The provided SPC3_H or Img point to an empty memory location
INVALID_OP Unable to set the background image when the live-mode is ON

Examples:

SDK_Example.c.

4.3.2.4 DllSDKExport SPC3Return SPC3_Set_Background_Subtraction (SPC3_H *spc3*, State *s*)

Enable or disable the hardware background subtraction.

Parameters

spc3	SPC3 handle
s	Enable or disable the background subtraction

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

Examples:

SDK_Example.c.

4.3.2.5 DllSDKExport SPC3Return SPC3_Set_Camera_Par (SPC3_H *spc3*, UInt16 *Exposure*, UInt32 *NFrames*, UInt16 *NIntegFrames*, UInt16 *NCounters*, State *Force8bit*, State *Half_array*, State *Signed_data*)

Set the acquisition parameters for the camera.

This function behaves differently depending on the operating mode setting. In case of Normal working mode, the exposure time is fixed to 10.40 microseconds. Therefore, the parameter Exposure is not considered. Longer exposures are obtained by summing multiple frames (i.e. by setting NIntegFrames). This operating mode does not degrade the signal to noise ratio. In fact, the camera does not have any read-out noise. In case of Advanced mode, all the parameters are controlled by the user which can set very long exposure times. The time unit of the Exposure parameter is clock cycles i.e. the exposure time is an integer number of internal clock cycles of 10 ns periode. For example, the value of 10 means 100 ns exposure.

Parameters

spc3	SPC3 handle.
Exposure	Exposure time for a single frame. The time unit is 10 ns. Meaningful only for Advanced mode. Accepted values: 1 ... 65534
NFrames	Number of frames per acquisition. Meaningful only for Snap acquistion. Accepted values: 1 ... 65534
NIntegFrames	Number of integrated frames. Each output frame is the result of the sum of NIntegFrames. Accepted values: 1 ... 65534
NCounters	Number of counters per pixels to be used. Accepted values: 1 ... 3
Force8bit	Force 8 bit per pixel acquisition. Counts are trunked. Meaningful only for Advanced mode.
Half_array	Acquire only a 32x32 array.
Signed_data	If enabled, data from counters 2 and 3 are signed data with 8bit integer part and 1 bit sign.

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
OUT_OF_BOUND Exposure, NFrames and NIntegFrames must be all greater than zero and smaller than 65535

Examples:

SDK_Example.c.

4.3.2.6 DllSDKExport SPC3Return SPC3_Set_DeadTime (SPC3_H *spc3,* UInt16 *Val*)

Update the dead-time setting.

Every time a photon is detected in a pixel, that pixel remains blind for a fix amount of time which is called dead-time. This setting is user-defined and it ranges from MIN_DEAD_TIME and MAX_DEAD_TIME. Only a sub-set of this range is practically selectable: a dead-time calibration is performed during the production of the device. This function will set the dead-time to the closest calibrated value to Val. The default dead-time value is 50 ns.

Parameters

spc3	SPC3 handle
Val	New dead-time value in nanoseconds

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP Unable to change the dead-time when the live-mode is ON

Examples:

SDK_Example.c.

4.3.2.7 DllSDKExport SPC3Return SPC3_Set_DeadTime_Correction (SPC3_H *spc3*, State *s*)

Enable or disable the dead-time correction.

The default setting is disabled.

Parameters

spc3	SPC3 handle
s	New state for the dead-time corrector

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

Examples:

SDK_Example.c.

4.3.2.8 DllSDKExport SPC3Return SPC3_Set_FLIM_Par (SPC3_H *spc3*, UInt16 *FLIM_steps*, UInt16 *FLIM_shift*, Int16 *FLIM_start*, UInt16 *Length*, int * *FLIM_frame_time*, double * *FLIM_bin_width*)

Set FLIM parameters.

The camera can perform automatic time-gated FLIM measurements emplyoing the embedded gate generator. Call this function to setup the FLIM acquisition parameters. Each "FLIM acquisition" is composed by FLIM_steps frames, each one consisting of an acquisition with Exposure and NIntegFrames as set with SPC3_Set_Par(). The total time required to perform each FLIM acquisition is passed back to the caller through the referenced FLIM_frame_time variable.

Parameters

spc3	SPC3 handle
FLIM_steps	Number of gate delay steps to be performed. Accepted values: 1 ... 800
FLIM_shift	Delay shift between steps in thousenths of gate period (20ns). Accepted values: 1 ... 800
FLIM_start	Start delay for FLIM sequence in thousenths of gate period (20ns). Accepted values: -400 ... +400
Length	Duration of the ON gate signal. The unit is percentage. Accepted values: 0 ... 100
FLIM_frame_time	Total time required to perform each FLIM acquisition in multiples of 10ns. Value is referenced.
FLIM_bin_width	Calibrated bin-width in ps for the specific camera. Nominal value is 20ps, i.e. 1/1000 of gate period. Value is referenced.

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
OUT_OF_BOUND Parameters are out of bound. Please note that the function not only checks if the single parameters are acceptable, but also checks if the combination of parameters would result in an invalid gate setting. E.g. FLIM_steps=100, FLIM_start=0, FLIM_shift=15 are not allowed, since they would result in a final gate shift of +1500 thousenths of period

See also

SPC3_Set_Par()

4.3.2.9 DllSDKExport SPC3Return SPC3_Set_FLIM_State (SPC3_H *spc3,* State *FLIM_State*)

Enable or disable FLIM mode.

FLIM mode automatically set the number of used counters to 1. FLIM mode cannot be enabled if Exposure time is set to a value lower than 1040.

Parameters

spc3	SPC3 handle
FLIM_State	Enable or disable the FLIM mode

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location.
INVALID_OP Exposure time is lower than 1040.

See also

SPC3_Set_Par()
SPC3_Set_FLIM_Par()

4.3.2.10 DllSDKExport SPC3Return SPC3_Set_Gate_Mode (SPC3_H *spc3,* GateMode *Mode*)

Set the gate mode either continuous or pulsed.

Parameters

spc3	SPC3 handle
Mode	New gate mode

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

Examples:

SDK_Example.c.

4.3.2.11 DllSDKExport SPC3Return SPC3_Set_Gate_Values (SPC3_H *spc3,* Int16 *Shift,* Int16 *Length*)

Change the Gate settings.

A gate signal is generated within the control electronics to select valid photons, i.e. only photons which arrives when the Gate is ON are counted. The gate signal is a 50 MHz square wave: shift and length define the phase and duty-cycle of the signal.

Parameters

spc3	SPC3 handle
Shift	Phase shift of the gate signal in the ON state. The unit is thousenths, i.e. 10 means a delay time of 0.01 times a 20 ns periodic signal, which is equal to 200ps. Accepted values: -400 ... +400
Length	Duration of the ON gate signal. The unit is percentage. Accepted values: 0 ... 100

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
OUT_OF_BOUND Shift or length are outside the valid values

Examples:

SDK_Example.c.

4.3.2.12 DllSDKExport SPC3Return SPC3_Set_Live_Mode_OFF (SPC3_H *spc3*)

Turn off the Live mode.

Parameters

spc3	SPC3 handle

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP The live mode is already inactive

See also

SPC3_Set_Live_Mode_ON()

Examples:

SDK_Example.c.

4.3.2.13 DllSDKExport SPC3Return SPC3_Set_Live_Mode_ON (SPC3_H *spc3*)

Turn on the Live mode.

The camera is set in the Live mode, i.e. it continuously acquires images (free-running mode). The frames which are not transferred to the computer are discarded. Therefore, the time-laps between two frames is not constant and it will depend on the transfer speed between the host computer and the camera. This mode is very useful to adjust optical components or to align the camera position. When the camera is in Live mode, no acquisition of images by SPC3_Get_Snap() or SPC3_Get_Memory() can be performed.

Parameters

spc3	SPC3 handle

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP The live mode has been already started

See also

SPC3_Set_Live_Mode_OFF()
SPC3_Get_Memory()
SPC3_Get_Snap()

Examples:

SDK_Example.c.

4.3.2.14 DllSDKExport SPC3Return SPC3_Set_Sync_In_State (SPC3_H *spc3,* State *s*)

Set the sync-in state.

Set the camera to wait for an input trigger signal before starting an acquisition.

Parameters

spc3	SPC3 handle
s	Enable or disable the synchronization input

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

Examples:

SDK_Example.c.

4.3.2.15 DllSDKExport SPC3Return SPC3_Set_Trigger_Out_State (SPC3_H *spc3,* TriggerMode *Mode*)

Select the output signal.

Parameters

spc3	Pointer to the SPC3 handle
Mode	New trigger mode

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

Examples:

SDK_Example.c.

4.4 Get methods

Functions

- DllSDKExport SPC3Return SPC3_Get_Live_Img (SPC3_H spc3, UInt16 *Img)
- DllSDKExport SPC3Return SPC3_Prepare_Snap (SPC3_H spc3)
- DllSDKExport SPC3Return SPC3_Get_Snap (SPC3_H spc3)
- DllSDKExport SPC3Return SPC3_Get_Image_Buffer (SPC3_H spc3, BUFFER_H *buffer)
- DllSDKExport SPC3Return SPC3_Get_Img_Position (SPC3_H spc3, UInt16 *Img, UInt32 Position, UInt16 counter)
- DllSDKExport SPC3Return SPC3_Start_ContAcq (SPC3_H spc3, char filename[256])
- DllSDKExport SPC3Return SPC3_Get_Memory (SPC3_H spc3, double *total_bytes)
- DllSDKExport SPC3Return SPC3_Stop_ContAcq (SPC3_H spc3)
- DllSDKExport SPC3Return SPC3_Get_DeadTime (SPC3_H spc3, UInt16 Val, UInt16 *ReturnVal)
- DllSDKExport SPC3Return SPC3_Get_GateWidth (SPC3_H spc3, Int16 Val, double *ReturnVal)
- DllSDKExport SPC3Return SPC3_Get_GateShift (SPC3_H spc3, Int16 Val, Int16 *ReturnVal)
- DllSDKExport SPC3Return SPC3_Is16Bit (SPC3_H spc3, short *is16bit)
- DllSDKExport SPC3Return SPC3_IsTriggered (SPC3_H spc3, short *isTriggered)
- DllSDKExport SPC3Return SPC3_GetVersion (SPC3_H spc3, double *Firmware_Version, double *Software_Version)

4.4.1 Detailed Description

Functions to get status or data from SPC3 camera.

4.4.2 Function Documentation

4.4.2.1 DllSDKExport SPC3Return SPC3_Get_DeadTime (SPC3_H *spc3*, UInt16 *Val*, UInt16 * *ReturnVal*)

Get the calibrated dead-time value.

This function provides the closest calibrated dead-time value to Val.

Parameters

spc3	SPC3 handle
Val	Desired dead-time value in ns. No error is generated when the value is above MAX_DEAD_TIME.
ReturnVal	Closest dead-time value possible. This parameter is referenced.

Returns

OK
NULL_POINTER The provided SPC3_H or ReturnVal point to an empty memory location

See also

SPC3_Set_DeadTime()

Examples:

SDK_Example.c.

4.4.2.2 DllSDKExport SPC3Return SPC3_Get_GateShift (SPC3_H *spc3*, Int16 *Val*, Int16 * *ReturnVal*)

Get the calibrated gate shift value.

This function provides the closest calibrated gate shift value to Val.

Parameters

spc3	SPC3 handle
Val	Desired gate shift value in thousenths of 20ns. No error is generated when the value out of range, instead the real boundaries are forced on ReturnVal.
ReturnVal	Closest gate-shift value possible. This parameter is referenced.

Returns

OK
NULL_POINTER The provided SPC3_H or ReturnVal point to an empty memory location

4.4.2.3 DllSDKExport SPC3Return SPC3_Get_GateWidth (SPC3_H *spc3*, Int16 *Val*, double * *ReturnVal*)

Get the calibrated gate width value.

This function provides the closest calibrated gate-width value to Val.

Parameters

spc3	SPC3 handle
Val	Desired gate-width value in percentage of 20ns. No error is generated when the value out of range, instead the real boundaries are forced on ReturnVal.
ReturnVal	Closest gate-width value possible. This parameter is referenced.

Returns

OK
NULL_POINTER The provided SPC3_H or ReturnVal point to an empty memory location

4.4.2.4 DllSDKExport SPC3Return SPC3_Get_Image_Buffer (SPC3_H *spc3*, BUFFER_H * *buffer*)

Get the pointer to the image buffer in which snap acquisition is stored.

WARNING User must pay attention not to exceed the dimension of the buffer (2∗1024∗65534 + 1 bytes) when accessing it.

Parameters

spc3	SPC3 handle
buffer	Pointer to the buffer Handle in which the function will save reference to the camera image buffer

Returns

OK
NULL_POINTER The provided SPC3_H or BUFFER_H point to an empty memory location

See also

SPC3_Get_Snap()

4.4.2.5 DllSDKExport SPC3Return SPC3_Get_Img_Position (SPC3_H *spc3,* UInt16 * *Img,* UInt32 *Position,* UInt16 *counter*)

Export an acquired image to an user allocated memory array.

Once a set of images have been acquired by SPC3_Get_Snap(), a single image can be exported from the SDK image buffer and saved in the memory (Img array).

Parameters

spc3	SPC3 handle
Img	Pointer to the output image array. The size of the array must be at least 2 KB.
Position	Index of the image to save. Accepted values: 1 ... Number of acquired images
counter	Number of the desired counter. Accepted values: 1 ... Number of used counters

Returns

OK
NULL_POINTER The provided SPC3_H or Img point to an empty memory location
OUT_OF_BOUND Parameters are out of bound.

See also

SPC3_Get_Snap()

4.4.2.6 DllSDKExport SPC3Return SPC3_Get_Live_Img (SPC3_H *spc3,* UInt16 * *Img*)

Get a Live image.

Acquire a live image and store the data into the Img array. This command is working only when the Live mode is turned on by the SPC3_Set_Live_Mode_ON() function.

Parameters

spc3	SPC3 handle
Img	Pointer to the output image array. The size of the array must be at least 2 KB.

Returns

OK
NULL_POINTER The provided SPC3_H or Img point to an empty memory location
INVALID_OP The live-mode has not been started yet

See also

SPC3_Set_Live_Mode_ON()

Examples:

SDK_Example.c.

4.4.2.7 DllSDKExport SPC3Return SPC3_Get_Memory (SPC3_H *spc3,* double * *total_bytes*)

Dump the camera memory to the PC and save data to the file specified with the SPC3_Start_ContAcq() function.

This function must be repeatedly called, as fast as possible, in order to free the camera internal memory and keep the acquisition going. If the internal camera memory get full during acquisition an error is generated. **WARNING**

The camera can generate data with very high throughput, up to about 205MB/s. Be sure to have enough disk space for your measurement.

Parameters

spc3	SPC3 handle
total_bytes	Total number of bytes read. Value is referenced.

Returns

OK
NULL_POINTER The provided SPC3_H or BUFFER_H point to an empty memory location
UNABLE_CREATE_FILE It was not possible to access the output file.
INVALID_OP Continuos acquistion was not yet started. Use SPC3_SPC3_Start_ContAcq() before calling this function.
COMMUNICATION_ERROR Communication error during data download.
SPC3_MEMORY_FULL Camera internal memory got full during data download. Datta loss occurred. Reduce frame-rate or optimize your software to reduce deadtime between subsequent calling of the function.

See also

SPC3_SPC3_Start_ContAcq()
SPC3_SPC3_Stop_ContAcq()

Examples:

SDK_Example.c.

4.4.2.8 DllSDKExport SPC3Return SPC3_Get_Snap (SPC3_H *spc3*)

Get a selected number of images.

Acquire a set of images according to the parameters defined by SPC3_Set_Camera_Par(). In FLIM mode NFrames "FLIM acquisitions" will be acquired. This command works only when SPC3_Prepare_Snap() has already been called. This function will not exit until the required number of images has been downloaded. For this reason, if the camera is configured for waiting and External Sync, before calling this function it could be useful to pool the camera for the trigger state, using the SPC3_IsTriggered() function.

Parameters

spc3	PC2 handle

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP Unable to acquire images when the live mode is ON. Use instead SPC3_Get_Live_Img().
INVALID_OP When the background subtraction, dead-time correction or normal acquisition mode are enabled, a maximum of 65536 images can be acquired

See also

SPC3_Set_Camera_Par()
SPC3_Prepare_Snap()
SPC3_Set_Sync_In_State()
SPC3_IsTriggered()

Examples:

SDK_Example.c.

4.4.2.9 DllSDKExport SPC3Return SPC3_GetVersion (SPC3_H *spc3*, double * *Firmware_Version*, double * *Software_Version*)

Get the SDK and camera firmware version.

Parameters

spc3	SPC3 handle
Firmware_Version	Version of the camera firmare in the format x.xx. This parameter is referenced.
Software_Version	Version of the SDK in the format x.xx. This parameter is referenced.

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

4.4.2.10 DllSDKExport SPC3Return SPC3_Is16Bit (SPC3_H *spc3*, short * *is16bit*)

Get the actual bitdepth of acquired data.

Data from the camera will be 16bit per pixel, if NFramesInteg $>$ 1, or DTC is enabled, or background subtraction is enabled, or 8bit per pixel otherwise. This function provides actual bitdepth with the current settings.

Parameters

spc3	SPC3 handle
is16bit	Actual status. The value is 0 if bitdepth is 8bit and 1 if bitdepth is 16bit. This parameter is referenced.

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

4.4.2.11 DllSDKExport SPC3Return SPC3_IsTriggered (SPC3_H *spc3*, short * *isTriggered*)

Pool the camera for external trigger staus.

Pool the camera in order to know if an external sync pulse was detected. The result is meaningfull only if the camera was previously set to wait for an external sync.

Parameters

spc3	SPC3 handle
isTriggered	Actual status. The value is 0 if no sync pulse was detected so far, 1 otherwise. This parameter is referenced.

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location

Examples:

SDK_Example.c.

4.4.2.12 DllSDKExport SPC3Return SPC3_Prepare_Snap (SPC3_H *spc3*)

Prepare the camera to the acquistion of a snap.

This command configures the camera to acquire a snap of NFrames images, as set by the SPC3_Set_Camera_Par() function. In FLIM mode NFrames "FLIM acquisitions" of a FLIM sequence will be acquired. If an External Sync is required, the camera will wait for a pulse on the Sync input before acquiring the images and saving them to the internal memory, otherwise they are acquired and saved immediately. Once acquired, snap must then be transferred to the PC using the SPC3_Get_Snap() function.

Parameters

spc3	SPC3 handle

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP Unable to acquire images when the live mode is ON. Use instead SPC3_Get_Live_Img().
INVALID_OP When the background subtraction, dead-time correction or normal acquisition mode are enabled, a maximum of 65536 images can be acquired

See also

SPC3_Set_Camera_Par()
SPC3_Get_Snap()
SPC3_Set_Sync_In_State()

Examples:

SDK_Example.c.

4.4.2.13 DllSDKExport SPC3Return SPC3_Start_ContAcq (SPC3_H *spc3,* char *filename[256]*)

Put the camera in "continuos acquisition" mode.

Compatible with FLIM mode. If the camera was set to wait for an external sync, the acquistion will start as soon as a pulse is detected on the Sync input, otherwise it will start immediately. The output file name must be provided when calling this fuction. Data are stored in the camera internal memory and must be downloaded calling the SPC3_Get_Memory() function as soon as possible, in order to avoid data loss.

Parameters

spc3	SPC3 handle
filename	Name of output file.

Returns

OK
NULL_POINTER The provided SPC3_H or BUFFER_H point to an empty memory location
UNABLE_CREATE_FILE It was not possible to create the output file.

See also

SPC3_Get_Memory()
SPC3_SPC3_Stop_ContAcq()

Examples:

SDK_Example.c.

4.4.2.14 DllSDKExport SPC3Return SPC3_Stop_ContAcq (SPC3_H *spc3*)

Stop the continuos acquisition of data and close the output file.

This function must be called at the end of the continuos acquisition, in order to properly close the file. **WARNING** If not called, the output file may be unreadable, and camera may have unexepected behaeviour if other functions are called.

Parameters

spc3	SPC3 handle

Returns

OK
NULL_POINTER The provided SPC3_H or BUFFER_H point to an empty memory location
UNABLE_CREATE_FILE It was not possible to access the output file.

See also

SPC3_SPC3_Start_ContAcq()
SPC3_Get_Memory()

Examples:

SDK_Example.c.

4.5 Additional methods

Functions

- DllSDKExport SPC3Return SPC3_Save_Img_Disk (SPC3_H spc3, UInt32 Start_Img, UInt32 End_Img, char *filename, OutFileFormat mode)
- DllSDKExport SPC3Return SPC3_Save_FLIM_Disk (SPC3_H spc3, char *filename, OutFileFormat mode)
- DllSDKExport SPC3Return SPC3_ReadSPC3FileFormatImage (char *filename, UInt32 ImgIdx, UInt16 counter, UInt16 *Img, char header[32])
- DllSDKExport SPC3Return SPC3_Average_Img (SPC3_H spc3, double *Img, int counter)
- DllSDKExport SPC3Return SPC3_StDev_Img (SPC3_H spc3, double *Img, int counter)
- DllSDKExport SPC3Return SPC3_Set_Correlation_Mode (SPC3_H spc3, CorrelationMode CM, int NCorrChannels, State s)
- DllSDKExport SPC3Return SPC3_Correlation_Img (SPC3_H spc3, int counter)
- DllSDKExport SPC3Return SPC3_Save_Correlation_Img (SPC3_H spc3, char *filename)

4.5.1 Detailed Description

Additional utility functions.

4.5.2 Function Documentation

4.5.2.1 DllSDKExport SPC3Return SPC3_Average_Img (SPC3_H *spc3,* double * *Img,* int *counter*)

Calculate the average image.

Once a set of images have been acquired by SPC3_Get_Snap(), an image which contains for each pixel the average value over all the acquired images is calculated. This is stored in the Img array.

Parameters

spc3	SPC3 handle.
Img	Pointer to the output double image array. The size of the array must be at least 8 kB.
counter	Desired counter.

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP No images were acquired

Examples:

SDK_Example.c.

4.5.2.2 DllSDKExport SPC3Return SPC3_Correlation_Img (SPC3_H *spc3,* int *counter*)

Calculate the autocorrelation function.

The autocorrelation function is estimated for each pixel. This function requires that a set of images have been previously acquired by SPC3_Get_Snap() and that the correlation mode is set to Enabled. Depending on the selected algorithm and the total number of collected images, this function can take several tens of seconds.

Parameters

spc3	SPC3 handle
counter	Desired counter.

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP No images were acquired or the correlation mode was not enabled
NOT_EN_MEMORY Not enough memory to calculate the correlation function
INVALID_NIMG_CORRELATION The required number of time lags of the correlation function can not be calculated from the available number of images

See also

SPC3_Set_Correlation_Mode()
SPC3_Get_Snap()

Examples:

SDK_Example.c.

4.5.2.3 DllSDKExport SPC3Return SPC3_ReadSPC3FileFormatImage (char * *filename*, UInt32 *ImgIdx*, UInt16 *counter*, UInt16 * *Img*, char *header[32]*)

Read a spc3 image from file.

Read the image at the ImgIdx position and for desired counter in the given spc3 file from the hard disk.

Parameters

filename	Name of the output file
ImgIdx	Image index in the file. Accepted values: 1 ... 65534
counter	Desired counter. Accepted values: 1 ... 3
Img	Pointer to the output image array. The size of the array must be at least 2 kB.
header	Array in which the header of SPC3 file is saved.

Returns

OK
UNABLE_READ_FILE Unable to read the input file. Is it a SPC3 file?
OUT_OF_BOUND The desired counter or image exceeds the file size.
NOT_EN_MEMORY Not enough memory to store the data contained in the file

Examples:

SDK_Example.c.

4.5.2.4 DllSDKExport SPC3Return SPC3_Save_Correlation_Img (SPC3_H *spc3*, char * *filename*)

Save the autocorrelation functions on the hard disk.

This function requires that SPC3_Set_Correlation_Mode() has been previously called. The autocorrelation data are stored in a .spcc binary file. The spcc binary file is organized as follows:

Byte offset	Type	Number of bytes	Description
0	int	4	Number of lag-times (NLag)

4	int	4	Number of pixels. This value must be 1024 (NPix)
8	int	4	Selected algorithm: 0 Linear, 1 Multi-tau
12	double	8 ∗ NLag	Autocorrelation values of the first pixel
12 + 8 ∗ NLag	double	8 ∗ NLag	Autocorrelation values of the second pixel
...	double	8 ∗ NLag	Autocorrelation values of the N^{th} pixel
12 + 8 ∗ (NPix-1) ∗ NLag	double	8 ∗ NLag	Autocorrelation values of the last pixel
12 + 8 ∗ NPix ∗ NLag	double	8 ∗ NLag	Lag times

A simple Matlab script can be used to read the data for further processing or visualization.

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 MPD .SPCC file reader
3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4
5 function data = Read_SPCC(fname)
6 f = fopen(fname,'rb');
7 buf = fread(f,3,'int32');
8 data.NChannel = buf(1);
9 data.NPixel = buf(2);
10 data.IsMultiTau = (buf(3) == 1);
11
12 data.CorrelationImage = reshape(fread(f,data.NPixel*data.NChannel,'float64'), ...
13             data.NChannel,32,32);
14 data.CorrelationImage = permute(data.CorrelationImage,[2 3 1]);
15 data.t=fread(f, data.NChannel,'float64');
16 fclose(f);
```

Parameters

spc3	SPC3 handle
filename	Name of the output file

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP The autocorrelation, which has been calculated, is not valid
UNABLE_CREATE_FILE Unable to create the output file

See also

SPC3_Set_Correlation_Mode()

Examples:

SDK_Example.c.

4.5.2.5 DllSDKExport SPC3Return SPC3_Save_FLIM_Disk (SPC3_H *spc3*, char ∗ *filename*, OutFileFormat *mode*)

Save the FLIM acquisition on the hard disk.

This function saves the acquired FLIM images on the hard disk. The output file format can be either a multipage TIFF with embedded acquisition metadata according to the OME-TIFF format or the proprietary SPC3 format. For standard measurements, use the SPC3_Save_Img__Disk() function. For both formats, image data is composed by a set of images following a "FLIM first, time second scheme", i.e. with the following frame sequence: 1st gate shift of 1st FLIM measurement, 2nd gate shift of 1st FLIM measurement,...,nth gate shift of 1st FLIM measurement,1st gate

shift of 2nd FLIM measurement, 2nd gate shift of 2nd FLIM measurement,...,nth gate shift of 2nd FLIM measurement,etc. OME-TIFF file could be opened with any image reader compatible with TIFF file, since metadata are saved into the Image Description tag in XML format. In order to decode OME-TIFF metadata, it is possible to use free OME-TIFF reader, such as OMERO or the Bio-Formats plugin for ImageJ. For more details see the OME-TIFF web site: http://www.openmicroscopy.org/site/support/ome-model/ome-tiff/. OME-TIFF metadata include the ModuloAlongT tag, which allows the processing of FLIM data with dedicated FLIM software such as FLIMfit (see http://www.openmicroscopy.org/site/products/partner/flimfit). SPC3 file are binary files composed by a header with acquisition metadata followed by raw image data, containg the 8/16 bit pixel values in row-major order. The byte order is little-endian for the 16 bit images. The header is composed by a signature of 8 byte, and a metadata section of 32 byte, as follows (multibyte fields are little-endian):

Byte offset	Number of bytes	Description
0	8	File signature: 0x4d5044ff03000000
8	1	Number of rows
9	1	Number of colums
10	1	Bit per pixel
11	1	Counters in use
12	2	Hardware integration time (multiples of 10ns)
14	2	Summed frames
16	1	Dead time correction enabled
17	1	Internal gate duty-cycle (0-100%)
18	2	Holdoff time (ns)
20	1	Background subtraction enabled
21	1	Data for counters 1 and 2 are signed
22	1	FLIM enabled
23	1	FLIM shift %
24	1	FLIM steps
25	4	FLIM frame length (multiples of 10ns)
29	2	FLIM bin width (fs)
31	1	PDE measurement
32	2	Start wavelength (nm)
34	2	Stop wavelength (nm)
36	2	Step (nm)
38	2	unused

SPC3 file can be read using the provided ImageJ/Fiji plugin.

Parameters

spc3	SPC3 handle
filename	Name of the output file. Value is referenced.
mode	File format of the output images

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP No images were acquired or the selected range of images is not valid
UNABLE_CREATE_FILE Unable to create the output file

4.5.2.6 DllSDKExport SPC3Return SPC3_Save_Img_Disk (SPC3_H *spc3*, UInt32 *Start_Img*, UInt32 *End_Img*, char * *filename*, OutFileFormat *mode*)

Save the selected images on the hard disk.

This function saves the acquired images on the hard disk. The output file format can be either a multipage TIFF with embedded acquisition metadata according to the OME-TIFF format or the proprietary SPC3 format. For FLIM measurements, use the SPC3_Save_FLIM_Disk() function. If TIFF format is selected, the desired images will be saved in a file for each enabled counter. If SPC3 format is selected a single SPC3 file will be created for all the counters. OME-TIFF file could be opened with any image reader compatible with TIFF file, since metadata are saved into the Image Description tag in XML format. In order to decode OME-TIFF metadata, it is possible to use free OME-TIFF reader, such as OMERO or the Bio-Formats plugin for ImageJ. For more details see the OME-TIFF web site: http://www.openmicroscopy.org/site/support/ome-model/ome-tiff/. SPC3 file are binary files composed by a header with acquisition metadata followed by raw image data, containg the 8/16 bit pixel values in row-major order. The byte order is little-endian for the 16 bit images. In case more counters are used, data are interlaced, i.e. the sequence of frames is the following: 1st frame of 1st counter, 1st frame of 2nd counter, 1st frame of 3rd counter, 2nd frame of 1st counter, etc. The header is composed by a signature of 8 byte, and a metadata section of 32 byte, as follows (multibyte fields are little-endian):

Byte offset	Number of bytes	Description
0	8	File signature: 0x4d5044ff03000000
8	1	Number of rows
9	1	Number of colums
10	1	Bit per pixel
11	1	Counters in use
12	2	Hardware integration time (multiples of 10ns)
14	2	Summed frames
16	1	Dead time correction enabled
17	1	Internal gate duty-cycle (0-100%)
18	2	Holdoff time (ns)
20	1	Background subtraction enabled
21	1	Data for counters 1 and 2 are signed
22	1	FLIM enabled
23	1	FLIM shift %
24	1	FLIM steps
25	4	FLIM frame length (multiples of 10ns)
29	2	FLIM bin width (fs)
31	1	PDE measurement
32	2	Start wavelength (nm)
34	2	Stop wavelength (nm)
36	2	Step (nm)
38	2	unused

SPC3 file can be read using the provided ImageJ/Fiji plugin.

Parameters

spc3	SPC3 handle
Start_Img	Index of the first image to save. Accepted values: 1 ... Number of acquired images
End_Img	Index of the last image to save. Accepted values: Start_Img ... Number of acquired images
filename	Name of the output file. Value is referenced.
mode	File format of the output images

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP No images were acquired or the selected range of images is not valid
UNABLE_CREATE_FILE Unable to create the output file

Examples:

SDK_Example.c.

4.5.2.7 DllSDKExport SPC3Return SPC3_Set_Correlation_Mode (SPC3_H *spc3*, CorrelationMode *CM*, int *NCorrChannels*, State *s*)

Enable the correlation mode.

This function must be called before invoking SPC3_Correlation_Img(). When this function is called, the memory required to save the new data is allocated in the heap and the previously stored data are cancelled. The deallocation of this memory is automatically performed when the SPC3_destr() function is called or by setting the State s equal to Disabled.

Parameters

spc3	SPC3 handle
CM	Selected autocorrelation algorithm
NCorrChannels	Number of global lag channels. When the linear correlation algorithm is selected, the first NChannel lags are calculated, where NChannel must be greater than 2. This algorithm accepts only a number of images which is a power of 2. For example, if 1025 images were acquired, only 1024 images are used to calculate the autocorrelation function. In case of Multi-tau algorithm, it defines the number of channel groups. The first group has 16 lags of duration equal to the exposure time of a frame. The following groups have 8 lags each, spaced at 2^i ∗ Exposure time.
s	Enable or Disable the correlation mode

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
OUT_OF_BOUND NCorrChannels must be greater than zero for the Multi-tau algorithm and greater than 2 for the Linear one
NOT_EN_MEMORY There is not enough memory to enable the correlation mode

See also

SPC3_Correlation_Img()

Examples:

SDK_Example.c.

4.5.2.8 DllSDKExport SPC3Return SPC3_StDev_Img (SPC3_H *spc3*, double ∗ *Img*, int *counter*)

Calculate the standard deviation image.

Once a set of images have been acquired by SPC3_Get_Snap(), an image which contains for each pixel the standard deviation over all the acquired images is calculated. This is stored in the Img array.

Parameters

spc3	Pointer to the SPC3 handle
Img	Pointer to the output double image array. The size of the array must be at least 8 KB.
counter	Desired counter.

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
INVALID_OP No images were acquired

Examples:

SDK_Example.c.

4.6 MPD only - Calibration functions

Functions

- DllSDKExport SPC3Return SPC3_Calibrate_DeadTime (SPC3_H spc3)
- DllSDKExport SPC3Return SPC3_Calibrate_Gate (SPC3_H spc3)

4.6.1 Detailed Description

Functions for internal MPD use.

4.6.2 Function Documentation

4.6.2.1 DllSDKExport SPC3Return SPC3_Calibrate_DeadTime (SPC3_H *spc3*)

Calibrate the dead-time time.

Only for MPD use Calibrate the dead time of the device. The function outputs a calibration file "Out.dat".

Parameters

spc3	SPC3 handle

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
UNABLE_CREATE_FILE Unable to create the output file

4.6.2.2 DllSDKExport SPC3Return SPC3_Calibrate_Gate (SPC3_H *spc3*)

Calibrate the gate-width.

Only for MPD use Calibrate the gate-width of the device. The function outputs a calibration file "GateCalib.dat".

Parameters

spc3	SPC3 handle

Returns

OK
NULL_POINTER The provided SPC3_H points to an empty memory location
UNABLE_CREATE_FILE Unable to create the output file

Chapter 5

File Documentation

5.1 SPC2_SDK.h File Reference

```
#include <stdio.h>
#include <math.h>
#include <string.h>
```

Macros

- #define MIN_DEAD_TIME 200
- #define MAX_DEAD_TIME 600
- #define MAX_GATE_WIDTH 100
- #define MIN_GATE_WIDTH 0
- #define MAX_GATE_SHIFT +50
- #define MIN_GATE_SHIFT -50

Typedefs

- typedef unsigned short UInt16
- typedef short Int16
- typedef unsigned UInt32
- typedef struct _SPC2_H * SPC2_H
- typedef unsigned char * BUFFER_H

Enumerations

- enum SPC2Return {
  OK = 0, USB_DEVICE_NOT_RECOGNIZED = -1, ELECTRONIC_INTERFACE_NOT_RECOGNIZED =-2,
  FAILED_FPGA_CONFIGURATION =-3,
  FPGA_USB_DRIVER_FAILURE =-4, COMMUNICATION_ERROR =-5, OUT_OF_BOUND = -6, MISSING_-
  DLL = -7,
  EMPTY_BUFFER = -8, NOT_EN_MEMORY = -9, NULL_POINTER = -10, INVALID_OP = -11,
  UNABLE_CREATE_FILE = -12, UNABLE_READ_FILE = -13, FIRMWARE_NOT_COMPATIBLE =-14, USB-
  _PORT_NOT_EN_POWER = -15,
  TOO_MUCH_LIGHT = -16, INVALID_NIMG_CORRELATION = -17, SPC2_MEMORY_FULL = -18 }
- enum OutFileFormat { SPC2_FILEFORMAT = 0, TIFF_LZW_COMPRESSION = 1, TIFF_NO_COMPRESS-
  ION = 2 }
- enum GateMode { Continuous = 0, Pulsed = 1 }

- enum CameraMode { Normal = 0, Advanced = 1 }
- enum TriggerMode { None = 0, Gate_Clk = 1, Frame = 2 }
- enum State { **Disabled** = 0, **Enabled** = 1 }
- enum CorrelationMode { Linear = 0, MultiTau = 1 }

Functions

- DllSDKExport SPC2Return SPC2_Constr (SPC2_H *spc2_in, UInt16 minRow, UInt16 maxRow, UInt16 minCol, UInt16 maxCol, CameraMode m, char *Device_ID)
- DllSDKExport SPC2Return SPC2_Destr (SPC2_H spc2)
- DllSDKExport void PrintErrorCode (FILE *fout, const char *FunName, SPC2Return retcode)
- DllSDKExport SPC2Return SPC2_Set_Camera_Par (SPC2_H spc2, UInt16 Exposure, UInt32 NFrames, UInt16 NIntegFrames)
- DllSDKExport SPC2Return SPC2_Set_Camera_SubArray (SPC2_H spc2, UInt16 minRow, UInt16 maxRow, UInt16 minCol, UInt16 maxCol)
- DllSDKExport SPC2Return SPC2_Set_DeadTime (SPC2_H spc2, UInt16 Val)
- DllSDKExport SPC2Return SPC2_Set_DeadTime_Correction (SPC2_H spc2, State s)
- DllSDKExport SPC2Return SPC2_Set_Advanced_Mode (SPC2_H spc2, State s)
- DllSDKExport SPC2Return SPC2_Set_Background_Img (SPC2_H spc2, UInt16 *Img)
- DllSDKExport SPC2Return SPC2_Set_Background_Subtraction (SPC2_H spc2, State s)
- DllSDKExport SPC2Return SPC2_Set_Gate_Values (SPC2_H spc2, Int16 Shift, Int16 Length)
- DllSDKExport SPC2Return SPC2_Set_Gate_Mode (SPC2_H spc2, GateMode Mode)
- DllSDKExport SPC2Return SPC2_Set_Trigger_Out_State (SPC2_H spc2, TriggerMode Mode)
- DllSDKExport SPC2Return SPC2_Set_Sync_In_State (SPC2_H spc2, State s)
- DllSDKExport SPC2Return SPC2_Set_Live_Mode_ON (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Set_Live_Mode_OFF (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Apply_settings (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Get_Live_Img (SPC2_H spc2, UInt16 *Img)
- DllSDKExport SPC2Return SPC2_Prepare_Snap (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Get_Snap (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Get_Image_Buffer (SPC2_H spc2, BUFFER_H *buffer)
- DllSDKExport SPC2Return SPC2_Get_Img_Position (SPC2_H spc2, UInt16 *Img, UInt32 Position)
- DllSDKExport SPC2Return SPC2_Start_ContAcq (SPC2_H spc2, char filename[256])
- DllSDKExport SPC2Return SPC2_Get_Memory (SPC2_H spc2, int *total_bytes)
- DllSDKExport SPC2Return SPC2_Stop_ContAcq (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Get_DeadTime (SPC2_H spc2, UInt16 Val, UInt16 *ReturnVal)
- DllSDKExport SPC2Return SPC2_Get_GateWidth (SPC2_H spc2, Int16 Val, double *ReturnVal)
- DllSDKExport SPC2Return SPC2_Get_GateShift (SPC2_H spc2, Int16 Val, Int16 *ReturnVal)
- DllSDKExport SPC2Return SPC2_Is16Bit (SPC2_H spc2, short *is16bit)
- DllSDKExport SPC2Return SPC2_IsTriggered (SPC2_H spc2, short *isTriggered)
- DllSDKExport SPC2Return SPC2_GetVersion (SPC2_H spc2, double *Firmware_Version, double *Software_Version)
- DllSDKExport SPC2Return SPC2_Save_Img_Disk (SPC2_H spc2, UInt32 Start_Img, UInt32 End_Img, char *filename, OutFileFormat mode)
- DllSDKExport SPC2Return SPC2_ReadSPC2FileFormatImage (char *filename, UInt32 ImgIdx, UInt16 *Img, UInt16 dim[])
- DllSDKExport SPC2Return SPC2_Average_Img (SPC2_H spc2, double *Img)
- DllSDKExport SPC2Return SPC2_StDev_Img (SPC2_H spc2, double *Img)
- DllSDKExport SPC2Return SPC2_Set_Correlation_Mode (SPC2_H spc2, CorrelationMode CM, int NCorrChannels, State s)
- DllSDKExport SPC2Return SPC2_Correlation_Img (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Save_Correlation_Img (SPC2_H spc2, char *filename)
- DllSDKExport SPC2Return SPC2_Calibrate_DeadTime (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Calibrate_Gate (SPC2_H spc2)

5.1.1 Detailed Description

SPC2 software development kit. This C header contains all the functions to operate the SPC2 camera in user defined applications.

5.1.2 Macro Definition Documentation

5.1.2.1 #define MAX_DEAD_TIME 600

Maximum allowed dead-time in nanoseconds.

The dead-time is set to MAX_DEAD_TIME, when higher values are requested.

Examples:

SDK_Example.c.

5.1.2.2 #define MIN_DEAD_TIME 200

Minimum allowed dead-time in nanoseconds.

The darkcounts rate and after-pulsing probability depend on the used dead-time setting: the lower the dead-time, the higher the darkcounts rate and after-pulsing probability. However, a long dead-time limits the maximum number of photons per second detected by the matrix of avalanche photodiodes. It is recommended to set this parameter to short values as e.g. 250 ns.

Examples:

SDK_Example.c.

Chapter 6

Example Documentation

6.1 SDK_Example.c

```
/*
########################################

Copyright 2012-2015 Micro-Photon-Devices s.r.l.

SOFTWARE PRODUCT: SPC3_SDK

Micro-Photon-Devices (MPD) expressly disclaims any warranty for the SOFTWARE PRODUCT.
The SOFTWARE PRODUCT is provided 'As Is' without any express or implied warranty of any kind,
including but not limited to any warranties of merchantability, noninfringement, or
fitness of a particular purpose. MPD does not warrant or assume responsibility for the
accuracy or completeness of any information, text, graphics, links or other items contained
within the SOFTWARE PRODUCT. MPD further expressly disclaims any warranty or representation
to Authorized Users or to any third party.
In no event shall MPD be liable for any damages (including, without limitation, lost profits,
business interruption, or lost information) rising out of 'Authorized Users' use of or inability
to use the SOFTWARE PRODUCT, even if MPD has been advised of the possibility of such damages.
In no event will MPD be liable for loss of data or for indirect, special, incidental,
consequential (including lost profit), or other damages based in contract, tort
or otherwise. MPD shall have no liability with respect to the content of the
SOFTWARE PRODUCT or any part thereof, including but not limited to errors or omissions contained
therein, libel, infringements of rights of publicity, privacy, trademark rights, business
interruption, personal injury, loss of privacy, moral rights or the disclosure of confidential
information.

########################################
*/

#include "SPC3_SDK.h"
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>

#if defined(__linux__)
    #define SLEEP usleep
    #define MILLIS 1000
#elif defined(__APPLE__)
    #define SLEEP usleep
    #define MILLIS 1000
    #include <unistd.h>
#elif defined(_WIN32)
    #include <Windows.h>
    #define SLEEP Sleep
    #define MILLIS 1
#endif

//************************************//
//                                    //
//              Support functions                         //
//                                    //
//************************************//

// Calculate the mean value of a UInt16 image
double mean(UInt16 * Img, UInt16 NPixel)
{
        int i=0;
```

```
        double res =0.0;
        for(i=0;i<NPixel;i++)
                res+= (double)Img[i];
        return res/(double)NPixel;
}

// Calculate the mean value of a double image
double mean_double(double * Img, UInt16 NPixel)
{
        int i=0;
        double res =0.0;
        for(i=0;i<NPixel;i++)
                res+= Img[i];
        return res/(double)NPixel;
}

// Create an histogram of the distribution of photon counts over the imager
void Hist(UInt16* Img, UInt16* hist)
{
        int i=0;
        memset(hist,'\0',65535*sizeof(UInt16));
        for(i=0;i<2048;i++)
                hist[Img[i]]++;
}

//**********************************//
//                                                                  //
//              Main code                                   //
//                                                                  //
//**********************************//
int main(void)
{
// variables definition //
        SPC3_H spc3= NULL;
        UInt16* Img= NULL,hist[65535],AppliedDT=0;
        char header[32]="";
        int integFrames=10;
        double read_bytes=0,total_bytes=0;
        int i=0,j=0,k=0,out=0;
        short trig=0;
        double gateoff=0;
        double *x = NULL,*y= NULL,*Imgd=NULL;
        double* data= NULL;
        char c=0,*fname=NULL;
        FILE* f= NULL;
        time_t start,stop;
        char *Test[] = { "Live mode: write on stdout 10 images", //0
                                        "Holdoff: mean number of photons at different holdoff values", //1
                                        "Dead-time corrector improves the image quality", //2
                                        "Background subtraction: 2 output files, with and without BG", //3
                                        "Gate: 1000 images normal, 1000 images with gate 3 ns shift 5 ns",
       //4
                                        "Synchronization output", //5
                                        "Background statistics", //6
                                        "Gate: calibrate the length of the gate signal", //7
                                        "Gate: width of the gate as a function of the offset", //8
                                        "Read and write test images", //9
                                        "Calculate the correlation image", //a
                                        "Continuous acquisition and subarray selection" //b
                                       };
        Imgd =(double*) calloc(1, 2048* sizeof(double));
        Img =(UInt16*) calloc(1, 2048* sizeof(UInt16));
        data =(double*) calloc(1, 2048* sizeof(double));
        x =(double*) calloc(1, 65535* sizeof(double));
        y =(double*) calloc(1, 65535* sizeof(double));

// Simple menu for test selection //

        printf("*******************************************************\n");
        printf("SPC3 Test program\n");
        printf("*******************************************************\n\n\n\n");
        for(i=0;i<12;i++)
                printf("\t%x) %s\n",i, Test[i]);
        printf("\tq) Quit\n");
        do
        {
                c=getchar();
        } while((c<48 || c>58) && c!='q' && c!='a' && c!='b');
        getchar();
        if(c>47 && c<59)
        {
                printf("*******************************************************\n");
                printf("%s\n",Test[c-48]);
                printf("*******************************************************\n\n\n");
        }
        if(c=='a')
        {
```

```
        printf("*****************************************************\n");
        printf("%s\n",Test[10]);
        printf("*****************************************************\n\n\n");
}
if(c=='b')
{
        printf("*****************************************************\n");
        printf("%s\n",Test[11]);
        printf("*****************************************************\n\n\n");
}
switch(c)
{
case '0'://Test live mode
        //SPC3 constructor and parameter setting
        SPC3_Constr(&spc3, Normal,"");
        SPC3_Set_Camera_Par(spc3, 100, 30000,300,1,Disabled,Disabled,Disabled);
        SPC3_Set_Trigger_Out_State(spc3,Frame);
        SPC3_Apply_settings(spc3);
        SPC3_Set_Live_Mode_ON(spc3);
        //Acquistion of 10 live images
        for(i=0;i<10;i++)
        {
                printf("Image %d:\n",i);
                SPC3_Get_Live_Img(spc3, Img);
                for(j=0;j<32;j++)
                {
                        for(k=0;k<32;k++)
                                printf("%d ",Img[32*j+k]);
                        printf("\n");
                }
        }
        //Live mode off
        SPC3_Set_Live_Mode_OFF(spc3);
        break;

case '1'://Test dead-time
        //SPC3 constructor and parameter setting
        out=(int)SPC3_Constr(&spc3, Advanced,"");
        SPC3_Set_Camera_Par(spc3, 1040, 10,1,1,Disabled,Disabled,Disabled);
        SPC3_Apply_settings(spc3);
        k=0;
        printf("Acquiring:\n");
        //Open file
        if((f = fopen("DTValues.txt","w")) == NULL)
        {
                printf("Unable to open the output file.\n");
                break;
        }
        //set deadtime, acquire snap, calculate mean photon count value and save results
        for(i=MAX_DEAD_TIME;i>=MIN_DEAD_TIME;i-=30)
        {
                data[k]=0.0;
                SPC3_Set_DeadTime(spc3, i);
                SPC3_Apply_settings(spc3);
                SPC3_Get_DeadTime(spc3, i, &AppliedDT);
                x[k]=(double)AppliedDT;
                SPC3_Prepare_Snap(spc3);
                SPC3_Get_Snap(spc3);
                SPC3_Average_Img(spc3,Imgd,1);
                data[k]=mean_double(Imgd,2048);
                printf("%d ns, Applied %d ns, %f\n", i, AppliedDT, data[k]);
                fprintf(f,"%d %f\n",i,data[k]);
                k++;
        }
        //print summary
        printf("\nDead-time calibration\n");
        for(i=0;i<k;i++)
        {
                printf("%f %f\n",x[i],data[i]);
        }
        fclose(f);
        break;

case '2': //Dead-time corrector effect
        //SPC3 constructor and parameter setting
        out=(int)SPC3_Constr(&spc3, Normal,"");
        SPC3_Set_Camera_Par(spc3, 4096, 1000,100,1,Disabled,Disabled,Disabled);
        SPC3_Set_DeadTime(spc3, 100);
        //acquisition with DTC on
        printf("Acquire the image using the dead-time correction\n");
        SPC3_Set_DeadTime_Correction(spc3, Enabled);
        SPC3_Apply_settings(spc3);
        //Acquire the BG image first
        printf("\n\n\nClose the camera shutter and press ENTER...\n");
        getchar();
        SPC3_Prepare_Snap(spc3);
        SPC3_Get_Snap(spc3);
```

```
            //Calculate the average image
            SPC3_Average_Img(spc3, data,1);
            for(i=0;i<2048;i++)
                    if(data[i]<= 65535)
                            Img[i] = (UInt16) floor(data[i]+0.5);
                    else
                            Img[i] = 65535; // Avoid overflow
            SPC3_Set_Background_Img(spc3, Img);
            SPC3_Set_Background_Subtraction(spc3, Enabled);
            SPC3_Apply_settings(spc3);
            //now acquire the image with shutter open
            printf("\n\n\nOpen the camera shutter and press ENTER...\n");
            getchar();
            SPC3_Prepare_Snap(spc3);
            SPC3_Get_Snap(spc3);
            SPC3_Save_Img_Disk(spc3, 1, 100, "Im_DeadTimeCorrected",
TIFF_LZW_COMPRESSION);
            printf("The the dead-time corrected image was acquired and stored on the hard disk
 succesfully!\n");
            //acquisition with DTC off
            printf("\n\nAcquire the reference image without the dead-time correction\n");
            SPC3_Set_Background_Subtraction(spc3, Disabled);
            SPC3_Set_DeadTime_Correction(spc3, Disabled);
            SPC3_Apply_settings(spc3);
            //Acquire the BG image first
            printf("\n\n\nClose the camera shutter and press ENTER...\n");
            getchar();
            SPC3_Prepare_Snap(spc3);
            SPC3_Get_Snap(spc3);
            //Calculate the average image
            SPC3_Average_Img(spc3, data,1);
            for(i=0;i<2048;i++)
                    if(data[i]<= 65535)
                            Img[i] = (UInt16) floor(data[i]+0.5);
                    else
                            Img[i] = 65535; // Avoid overflow
            SPC3_Set_Background_Img(spc3, Img);
            SPC3_Set_Background_Subtraction(spc3, Enabled);
            SPC3_Apply_settings(spc3);
            //now acquire the image with shutter open
            printf("\n\n\nOpen the camera shutter and press ENTER...\n");
            getchar();
            SPC3_Prepare_Snap(spc3);
            SPC3_Get_Snap(spc3);
            SPC3_Save_Img_Disk(spc3, 1, 100, "Im_DeadTimeReference",
TIFF_LZW_COMPRESSION);
            printf("The the dead-time corrected image was acquired and stored on the hard disk
 succesfully!\n");
            break;

  case '3'://Test background subtraction
            //SPC3 constructor and parameter setting
            out=(int)SPC3_Constr(&spc3, Normal,"");
            SPC3_Set_Camera_Par(spc3, 4096, 1000,100,1,Disabled,Disabled,Disabled);
            SPC3_Apply_settings(spc3);
            //acquire background image
            printf("\n\n\nClose the camera shutter and press ENTER...\n");
            getchar();
            SPC3_Prepare_Snap(spc3);
            SPC3_Get_Snap(spc3);
            SPC3_Save_Img_Disk(spc3, 1, 1, "Bg",
TIFF_LZW_COMPRESSION);
            SPC3_Average_Img(spc3, data,1);
            for(i=0;i<2048;i++)
                    if(data[i]<= 65535)
                            Img[i] = (UInt16) floor(data[i]+0.5);
                    else
                            Img[i] = 65535; // Avoid overflow
            SPC3_Set_Background_Img(spc3, Img);
            //acquire image with background subtration off
            printf("Open the camera shutter and press ENTER ...\n");
            getchar();
            SPC3_Set_Background_Subtraction(spc3, Disabled);
            SPC3_Apply_settings(spc3);
            SPC3_Prepare_Snap(spc3);
            SPC3_Get_Snap(spc3);
            SPC3_Save_Img_Disk(spc3, 1, 1, "Normal",
TIFF_LZW_COMPRESSION);
            //acquire image with background subtration on
            SPC3_Set_Background_Subtraction(spc3, Enabled);
            SPC3_Prepare_Snap(spc3);
            SPC3_Get_Snap(spc3);
            SPC3_Save_Img_Disk(spc3, 1, 1, "BgSubt",
TIFF_LZW_COMPRESSION);
            break;

  case '4': // Test gate
```

```
            //SPC3 constructor and parameter setting
            out=(int)SPC3_Constr(&spc3, Advanced,"");
            SPC3_Set_Camera_Par(spc3, 4096, 1000,100,1,Disabled,Disabled,Disabled);
            SPC3_Set_DeadTime(spc3, 100);
            SPC3_Apply_settings(spc3);
            //Normal image
            printf("Acquiring the reference image ...\n");
            SPC3_Prepare_Snap(spc3);
            SPC3_Get_Snap(spc3);
            printf("Save the reference image ...\n");
            SPC3_Save_Img_Disk(spc3, 1, 1000, "GateNormal",
TIFF_LZW_COMPRESSION);
            //gated image
            SPC3_Set_Gate_Mode(spc3, Pulsed);
            SPC3_Set_Gate_Values(spc3, 10, 15); //Shift -10% of 20 ns --> 120 ns,
 Length 15% of 20 ns --> 3ns
            SPC3_Apply_settings(spc3);
            printf("Acquiring the gated image ...\n");
            SPC3_Prepare_Snap(spc3);
            SPC3_Get_Snap(spc3);
            printf("Save the gated image ...\n");
            SPC3_Save_Img_Disk(spc3, 1, 1000, "GatePulsed",
TIFF_LZW_COMPRESSION);
            break;

  case '5': // Test synchronization output
            //SPC3 constructor and parameter setting
            out=(int)SPC3_Constr(&spc3, Advanced,"");
            SPC3_Set_Camera_Par(spc3, 8192, 0xFFFF,5,1,Disabled,Disabled,Disabled);
            SPC3_Set_DeadTime(spc3, 100);
            //no output
            SPC3_Set_Trigger_Out_State(spc3,None);
            SPC3_Apply_settings(spc3);
            SPC3_Set_Live_Mode_ON(spc3);
            printf("\n\nNo output ...\n");
            printf("Press ENTER to continue\n");
            getchar();
            SPC3_Set_Live_Mode_OFF(spc3);
            //gate clock output
            SPC3_Set_Trigger_Out_State(spc3,
Gate_Clk);
            SPC3_Set_Camera_Par(spc3, 8192, 0xFFFF,5,1,Disabled,Disabled,Disabled);
            SPC3_Apply_settings(spc3);
            SPC3_Set_Live_Mode_ON(spc3);
            printf("\n\nGate synchronization signal ...\n");
            printf("Press ENTER to continue\n");
            getchar();
            SPC3_Set_Live_Mode_OFF(spc3);
            //frame sync output
            SPC3_Set_Trigger_Out_State(spc3,Frame);
            SPC3_Set_Camera_Par(spc3, 8192, 0xFFFF,5,1,Disabled,Disabled,Disabled);
            SPC3_Apply_settings(spc3);
            SPC3_Set_Live_Mode_ON(spc3);
            printf("\n\nFrame synchronization signal ...\n");
            printf("Press ENTER to continue\n");
            getchar();
            SPC3_Set_Live_Mode_OFF(spc3);
            printf("\n\nWait for the trigger input ...\n");
            //trigger in enabled
            SPC3_Set_Sync_In_State(spc3, Enabled);
            SPC3_Set_Camera_Par(spc3, 4096, 10,2,1,Disabled,Disabled,Disabled);
            SPC3_Apply_settings(spc3);
            SPC3_Prepare_Snap(spc3);
            while (trig!=1)
                    SPC3_IsTriggered(spc3, &trig);
            printf("Trigger signal received!\n");
            break;

  case '6': // Test background statistics
            //The output file "BgHist.txt" contains the number of pixels which had a total number of
 counts given by the column index for each row.
            //For example, column 3 contains the /number of pixels which had 3 dark-counts.
            //Several rows are present because the histogram is calculated for several dead-time
 values.
            //SPC3 constructor and parameter setting
            out=(int)SPC3_Constr(&spc3, Advanced,"");
            SPC3_Set_Camera_Par(spc3, 1040, 100,10000,1,Disabled,Disabled,Disabled);
            f=fopen("BgHist.txt","w");
            printf("Close the camera shutter and press ENTER ...\n");
            getchar();
            printf("Acquiring: ");
            for(i=150;i>=50;i-=50) //different hold-off values
            {
                    printf("%d ns ",i);
                    SPC3_Set_DeadTime(spc3, i);
                    SPC3_Apply_settings(spc3);
                    SPC3_Prepare_Snap(spc3);
```

```
                SPC3_Get_Snap(spc3);
                SPC3_Average_Img(spc3, data,1);
                for(k=0;k<2048;k++)
                if(data[k]<= 65535)
                        Img[k] = (UInt16) floor(data[k]+0.5);
                else
                        Img[k] = 65535; // Avoid overflow
                Hist(Img,hist);
                for(j=0;j<65535;j++)
                        fprintf(f,"%hd ",hist[j]);
                fprintf(f,"\n");

        }
        printf("\n");
        fclose(f);
        break;

 case '7': // Calibrate gate
        //SPC3 constructor and parameter setting
        out=(int)SPC3_Constr(&spc3, Advanced,"");
        SPC3_Set_Camera_Par(spc3, 1040, 10000,2,1,Disabled,Disabled,Disabled);
        SPC3_Set_DeadTime(spc3, 100);
        SPC3_Apply_settings(spc3);
        printf("Expose the SPC3 camera to a time-independent luminous signal\n(room light might
oscillate at 50 or 60 Hz)\nPress ENTER to continue ...\n");
        getchar();
        if((f = fopen("GateValues.txt","w")) == NULL)
        {
                printf("Unable to open the output file.\n");
                break;
        }
        //photon counts without any gate
        SPC3_Set_Gate_Mode(spc3, Continuous);
        SPC3_Prepare_Snap(spc3);
        SPC3_Get_Snap(spc3);
        SPC3_Average_Img(spc3, data,1);
        gateoff = mean_double(data,2048);
        printf("Gate OFF counts: %.2f\n",gateoff);
        fprintf(f,"Gate OFF counts: %.2f\n",gateoff);
        SPC3_Set_Gate_Mode(spc3, Pulsed);
        SPC3_Apply_settings(spc3);
        printf("Acquiring:\n\nGate\t\tMean\t\tActual Gate\t\t\n");
        //photon counts for gate witdh ranging from 0% to 100%
        for(i=0;i<=100;i+=1)
        {
                SPC3_Set_Gate_Values(spc3, 0, i);
                SPC3_Apply_settings(spc3);
                SPC3_Prepare_Snap(spc3);
                SPC3_Get_Snap(spc3);
                SPC3_Average_Img(spc3, data,1);
                y[i] = mean_double(data,2048);
                y[i+101] = y[i]/gateoff*100; //actual gate width calculated from photon counts
                x[i]=(double) i;
                printf("%3.0f\t\t%.2f\t\t%.2f\n",x[i],y[i],y[i+101]);
                fprintf(f,"%.0f %.2f %.2f\n",x[i],y[i],y[i+101]);
        }
        fclose(f);
        printf("\n");
        break;

 case '8': // Constancy of the gate width
        //SPC3 constructor and parameter setting
        fname = (char*) calloc(256,sizeof(char));
        out=(int)SPC3_Constr(&spc3, Advanced,"");
        SPC3_Set_Camera_Par(spc3, 1040, 1000,2,1,Disabled,Disabled,Disabled);
        SPC3_Set_DeadTime(spc3, 100);
        //photon counts without any gate
        SPC3_Set_Gate_Mode(spc3, Continuous);
        SPC3_Prepare_Snap(spc3);
        SPC3_Get_Snap(spc3);
        SPC3_Average_Img(spc3, data,1);
        gateoff = mean_double(data,2048);
        printf("Gate OFF counts: %.2f\n",gateoff);
        SPC3_Set_Gate_Mode(spc3, Pulsed);
        SPC3_Apply_settings(spc3);
        printf("Expose the SPC3 camera to a time-independent luminous signal\n(room light might
oscillate at 50 or 60 Hz)\nPress ENTER to continue ...\n");
        getchar();
        //setting different gate width and shift
        for(j=0;j<=100;j+=10)
        {
                sprintf(fname, "GateOffset_%d.txt",j);
                if((f = fopen(fname,"w")) == NULL)
                {
                        printf("Unable to open the file %s.\n",fname);
                        break;
                }
```

```
                printf("\nGate length:%d\n",j);
                printf("Shift\t\tMean\t\tStDev\t\t\n");
                for(i=-50;i<=+50;i+=10)
                {
                        SPC3_Set_Gate_Values(spc3, i, j);
                        SPC3_Apply_settings(spc3);
                        SPC3_Prepare_Snap(spc3);
                        SPC3_Get_Snap(spc3);
                        SPC3_Average_Img(spc3, data,1);
                        y[i+500] = mean_double(data,2048);
                        SPC3_StDev_Img(spc3, data,1);
                        y[i+500+101] = mean_double(data,2048);
                        x[i+500]=(double) i;
                        printf("%3.0f\t\t%.4f\t\t%.4f\n",x[i+500],y[i+500]/gateoff*100,y[i+101+500]
/gateoff*100);
                        fprintf(f,"%.0f %.4f %.4f\n",x[i+500],y[i+500]/gateoff*100,y[i+101+500]/
gateoff*100);
                }
                fclose(f);
        }
        printf("\n");
        free(fname);
        break;

  case '9': // Save and Read images
        //SPC3 constructor and parameter setting
        out=(int)SPC3_Constr(&spc3, Advanced,"");
        SPC3_Set_Camera_Par(spc3, 1040, 20,2,1,Disabled,Disabled,Disabled);
        SPC3_Set_DeadTime(spc3, 100);
        SPC3_Apply_settings(spc3);
        //acquiring and saving images
        printf("Acquiring 20 images and save them on the hard drive in the spc3 file format.\n");
        SPC3_Prepare_Snap(spc3);
        SPC3_Get_Snap(spc3);
        SPC3_Save_Img_Disk(spc3, 1,20,"Test20_Im",
SPC3_FILEFORMAT);
        //reading images from file
        printf("Read the images from the disk and print the value of the top-left-corner pixel for
 each frame.\n(Press ENTER to continue)\n");
        getchar();
        SPC3_ReadSPC3FileFormatImage("Test20_Im", 1,1,Img, header );
        printf("Rows: %d, Columns: %d\n",header[0], header[1]);
        for(i=1;i<=20;i++)
        {
                SPC3_ReadSPC3FileFormatImage("Test20_Im", i,1,Img,
header );
                printf("Image %d, pixel value = %hu\n",i, Img[0]);
        }
        break;

  case 'a': //Correlation image
        //SPC3 constructor and parameter setting
        out=(int)SPC3_Constr(&spc3, Normal,"");
        SPC3_Set_Camera_Par(spc3, 10500, 1024*64, 1,1,Disabled,Disabled,Disabled
);
        SPC3_Set_DeadTime(spc3, 100);
        SPC3_Apply_settings(spc3);
        //acquire images
        printf("Acquiring the images...\n");
        SPC3_Prepare_Snap(spc3);
        SPC3_Get_Snap(spc3);
        //computer correlation
        SPC3_Set_Correlation_Mode(spc3,
MultiTau, 8, Enabled);
        printf("Starting the multi-tau autocorrelation algorithm\n");
        start = clock();
        SPC3_Correlation_Img(spc3,1);
        stop = clock();
        printf("The correlation has terminated in %.3f s\n", (stop-start)/(float)CLOCKS_PER_SEC);
        SPC3_Save_Correlation_Img(spc3,"Correlation_MultiTau.spcc");
        break;

  case 'b': //Continuous acquisition, number of integration frames selection.
        //SPC3 constructor and parameter setting
        out=(int)SPC3_Constr(&spc3, Normal,"");
        printf("\n");
        printf("Input the number of frames of 10.40us to be integrated (suggested > 10 to avoid
 data loss):\n");
        scanf("%d",&integFrames);
        printf("Total integration time: %.2fus\n", (float)(10.40*integFrames));
        SPC3_Set_Camera_Par(spc3, 1050, 1, integFrames,1,Disabled,Disabled,
Disabled);
        SPC3_Set_DeadTime(spc3, 100);
        SPC3_Apply_settings(spc3);
        //acquire images
        printf("Continuous acquisition will be started and 10 memory dumps performed.\n");
        printf("Press ENTER to start continuous acquisition...\n");
```

```
                getchar();
                getchar();
                SPC3_Start_ContAcq(spc3, "contacq.spc3");
                for(i=1; i<10; i++)
                {
                        if (SPC3_Get_Memory(spc3,&read_bytes)==OK)
                        {
                                total_bytes=total_bytes+read_bytes;
                                printf("Acquired %f bytes in %d readout operation\n",total_bytes, i);
                                SLEEP(1*MILLIS);
                        }
                        else
                                break;
                }
                SPC3_Stop_ContAcq(spc3);
                printf("Acquisition saved to contacq.spc3.\n");
                break;
        default:
                break;
        }

// Destructors                                          //

        if(spc3)
                SPC3_Destr(spc3);
        free(Img);
        free(Imgd);
        free(data);
        free(y);
        free(x);
        printf("Press ENTER to continue\n");
        getchar();
        return 0;
}
```

Index