



# Single Photon Counting Camera

## SPC<sup>3</sup>

Version 2.2.0B

Software Development Kit  
Manual

October, 2018



# Contents

<b>1</b>	<b>Single Photon Counting Camera Software Development Kit (SPC3-SDK).</b>	<b>5</b>
<b>2</b>	<b>Module Index</b>	<b>7</b>
2.1	Modules . . . . .	7
<b>3</b>	<b>File Index</b>	<b>9</b>
3.1	File List . . . . .	9
<b>4</b>	<b>Module Documentation</b>	<b>11</b>
4.1	SPC3-SDK custom Types . . . . .	11
4.1.1	Detailed Description . . . . .	11
4.1.2	Typedef Documentation . . . . .	11
4.1.2.1	BUFFER_H . . . . .	11
4.1.2.2	SPC3_H . . . . .	11
4.1.3	Enumeration Type Documentation . . . . .	12
4.1.3.1	CameraMode . . . . .	12
4.1.3.2	CorrelationMode . . . . .	12
4.1.3.3	GateMode . . . . .	12
4.1.3.4	OutFileFormat . . . . .	12
4.1.3.5	SPC3Return . . . . .	13
4.1.3.6	TriggerMode . . . . .	13
4.2	Constructr, destructor and error handling . . . . .	14
4.2.1	Detailed Description . . . . .	14
4.2.2	Function Documentation . . . . .	14
4.2.2.1	PrintErrorCode(FILE *fout, const char *FunName, SPC3Return retcode) . . . . .	14
4.2.2.2	SPC3_Constr(SPC3_H *spc3_in, CameraMode m, char *Device_ID) . . . . .	14
4.2.2.3	SPC3_Destr(SPC3_H spc3) . . . . .	15
4.3	Set methods . . . . .	16
4.3.1	Detailed Description . . . . .	16
4.3.2	Function Documentation . . . . .	16
4.3.2.1	SPC3_Apply_settings(SPC3_H spc3) . . . . .	16
4.3.2.2	SPC3_Set_Advanced_Mode(SPC3_H spc3, State s) . . . . .	17

4.3.2.3	SPC3_Set_Background_Img(SPC3_H spc3, UInt16 *Img) . . . . .	17
4.3.2.4	SPC3_Set_Background_Subtraction(SPC3_H spc3, State s) . . . . .	17
4.3.2.5	SPC3_Set_Camera_Par(SPC3_H spc3, UInt16 Exposure, UInt32 NFrames, UInt16 NIntegFrames, UInt16 NCounters, State Force8bit, State Half_array, State Signed_data) . . . . .	18
4.3.2.6	SPC3_Set_Camera_Par_SubArray(SPC3_H spc3, UInt16 Exposure, UInt32 NFrames, UInt16 NIntegFrames, State Force8bit, UInt16 Npixels) . . . . .	18
4.3.2.7	SPC3_Set_Coarse_Gate_Values(SPC3_H spc3, int Counter, UInt16 Start, UInt16 Stop) . . . . .	19
4.3.2.8	SPC3_Set_DeadTime(SPC3_H spc3, UInt16 Val) . . . . .	19
4.3.2.9	SPC3_Set_DeadTime_Correction(SPC3_H spc3, State s) . . . . .	20
4.3.2.10	SPC3_Set_DualGate(SPC3_H spc3, State DualGate_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int Gap, double *CalibratedBinWidth) . . . . .	20
4.3.2.11	SPC3_Set_FLIM_Par(SPC3_H spc3, UInt16 FLIM_steps, UInt16 FLIM_shift, UInt16 FLIM_start, UInt16 Length, int *FLIM_frame_time, double *FLIM_bin_width) . . . . .	21
4.3.2.12	SPC3_Set_FLIM_State(SPC3_H spc3, State FLIM_State) . . . . .	21
4.3.2.13	SPC3_Set_Gate_Mode(SPC3_H spc3, int counter, GateMode Mode) . . . . .	22
4.3.2.14	SPC3_Set_Gate_Values(SPC3_H spc3, Int16 Shift, Int16 Length) . . . . .	22
4.3.2.15	SPC3_Set_Live_Mode_OFF(SPC3_H spc3) . . . . .	23
4.3.2.16	SPC3_Set_Live_Mode_ON(SPC3_H spc3) . . . . .	23
4.3.2.17	SPC3_Set_Sync_In_State(SPC3_H spc3, State s, int frames) . . . . .	23
4.3.2.18	SPC3_Set_Trigger_Out_State(SPC3_H spc3, TriggerMode Mode) . . . . .	24
4.3.2.19	SPC3_Set_TripleGate(SPC3_H spc3, State TripleGate_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int ThirdGateWidth, int Gap1, int Gap2, double *CalibratedBinWidth) . . . . .	24
4.4	Get methods . . . . .	26
4.4.1	Detailed Description . . . . .	26
4.4.2	Function Documentation . . . . .	26
4.4.2.1	SPC3_DeviceInfo(char *Device_ID, char *Camera_serial, double *Firmware_Version, double *Software_Version, char *Firmware_Custom_Version, char *Software_Custom_Version) . . . . .	26
4.4.2.2	SPC3_Get_DeadTime(SPC3_H spc3, UInt16 Val, UInt16 *RetVal) . . . . .	27
4.4.2.3	SPC3_Get_GateShift(SPC3_H spc3, int counter, Int16 Val, Int16 *RetVal) . . . . .	27
4.4.2.4	SPC3_Get_GateWidth(SPC3_H spc3, int counter, Int16 Val, double *RetVal) . . . . .	28
4.4.2.5	SPC3_Get_Image_Buffer(SPC3_H spc3, BUFFER_H *buffer) . . . . .	28
4.4.2.6	SPC3_Get_Img_Position(SPC3_H spc3, UInt16 *Img, UInt32 Position, UInt16 counter) . . . . .	28
4.4.2.7	SPC3_Get_Live_Img(SPC3_H spc3, UInt16 *Img) . . . . .	29
4.4.2.8	SPC3_Get_Memory(SPC3_H spc3, double *total_bytes) . . . . .	29
4.4.2.9	SPC3_Get_Memory_Buffer(SPC3_H spc3, double *total_bytes, BUFFER_H *buffer) . . . . .	30
4.4.2.10	SPC3_Get_Snap(SPC3_H spc3) . . . . .	31
4.4.2.11	SPC3_GetSerial(SPC3_H spc3, char *Camera_ID, char *Camera_serial) . . . . .	31
4.4.2.12	SPC3_GetVersion(SPC3_H spc3, double *Firmware_Version, double *Software_Version, char *Custom_version) . . . . .	31

4.4.2.13	SPC3_Is16Bit(SPC3_H spc3, short *is16bit)	32
4.4.2.14	SPC3_IsTriggered(SPC3_H spc3, short *isTriggered)	32
4.4.2.15	SPC3_Prepare_Snap(SPC3_H spc3)	33
4.4.2.16	SPC3_Start_ContAcq(SPC3_H spc3, char filename[256])	33
4.4.2.17	SPC3_Start_ContAcq_in_Memory(SPC3_H spc3)	34
4.4.2.18	SPC3_Stop_ContAcq(SPC3_H spc3)	34
4.4.2.19	SPC3_Stop_ContAcq_in_Memory(SPC3_H spc3)	34
4.5	Additional methods	36
4.5.1	Detailed Description	36
4.5.2	Function Documentation	36
4.5.2.1	SPC3_Average_Img(SPC3_H spc3, double *Img, int counter)	36
4.5.2.2	SPC3_Correlation_Img(SPC3_H spc3, int counter)	36
4.5.2.3	SPC3_ReadSPC3FileFormatImage(char *filename, UInt32 ImgIdx, UInt16 counter, UInt16 *Img, char header[1024])	37
4.5.2.4	SPC3_Save_Averaged_Img_Disk(SPC3_H spc3, int counter, char *filename, OutFileFormat mode)	37
4.5.2.5	SPC3_Save_Correlation_Img(SPC3_H spc3, char *filename)	38
4.5.2.6	SPC3_Save_FLIM_Disk(SPC3_H spc3, char *filename, OutFileFormat mode)	39
4.5.2.7	SPC3_Save_Img_Disk(SPC3_H spc3, UInt32 Start_Img, UInt32 End_Img, char *filename, OutFileFormat mode)	40
4.5.2.8	SPC3_Set_Correlation_Mode(SPC3_H spc3, CorrelationMode CM, int NCorChannels, State s)	42
4.5.2.9	SPC3_StDev_Img(SPC3_H spc3, double *Img, int counter)	43
4.6	MPD only - Calibration functions	44
4.6.1	Detailed Description	44
4.6.2	Function Documentation	44
4.6.2.1	SPC3_Calibrate_DeadTime(SPC3_H spc3)	44
4.6.2.2	SPC3_Calibrate_Gate(SPC3_H spc3)	44
4.6.2.3	SPC3_Calibrate_Gate_Skew(SPC3_H spc3, int Start)	44
<b>5</b>	<b>File Documentation</b>	<b>47</b>
5.1	SPC3_SDK.h File Reference	47
5.1.1	Detailed Description	49
5.1.2	Macro Definition Documentation	49
5.1.2.1	MAX_DEAD_TIME	49
5.1.2.2	MIN_DEAD_TIME	49
<b>6</b>	<b>Example Documentation</b>	<b>51</b>
6.1	SDK_Example.c	51

<a href="#">Index</a>	60
<a href="#">Index</a>	60

## 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:

SPC3-SDK custom Types . . . . .	11
Constructr, destructor and error handling . . . . .	14
Set methods . . . . .	16
Get methods . . . . .	26
Additional methods . . . . .	36
MPD only - Calibration functions . . . . .	44



# Chapter 3

## File Index

### 3.1 File List

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

<a href="#">SPC3_SDK.h</a>	SPC3 software development kit	47
----------------------------	-------------------------------	----



# 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\\_NO\\_COMPRESSION](#) = 2 }
- enum [GateMode](#) { [Continuous](#) = 0, [Pulsed](#) = 1, [Coarse](#) = 2 }
- 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 10.40 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 Pulsed, 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 square wave at 50MHz. The photons, which are detected when the gate signal is "ON", are counted, otherwise they are discarded.

**Coarse** The gate signal is a square wave, with period equal to the integration time, and width and position adjustable in 10ns-steps. 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\_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 continuous 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

- DIISDKExport [SPC3Return SPC3\\_Constr](#) ([SPC3\\_H](#) \*spc3\_in, [CameraMode](#) m, char \*Device\_ID)
- DIISDKExport [SPC3Return SPC3\\_Destr](#) ([SPC3\\_H](#) spc3)
- DIISDKExport 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 DIISDKExport 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

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

#### 4.2.2.2 DIISDKExport [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 camera 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 a list of connected device is printed on the screen upon calling this function. An empty string is accepted too. In this case, the first device on the list will be connected.

#### Parameters

<i>spc3_in</i>	Pointer to SPC3 handle
<i>m</i>	Camera Working mode
<i>Device_ID</i>	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 DIISDKExport 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**

<i>spc3</i>	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

- DIISDKExport [SPC3Return SPC3\\_Set\\_Camera\\_Par](#) (SPC3\_H spc3, UInt16 Exposure, UInt32 NFrames, UInt16 NIntegFrames, UInt16 NCounters, State Force8bit, State Half\_array, State Signed\_data)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Camera\\_Par\\_SubArray](#) (SPC3\_H spc3, UInt16 Exposure, UInt32 NFrames, UInt16 NIntegFrames, State Force8bit, UInt16 Npixels)
- DIISDKExport [SPC3Return SPC3\\_Set\\_DeadTime](#) (SPC3\_H spc3, UInt16 Val)
- DIISDKExport [SPC3Return SPC3\\_Set\\_DeadTime\\_Correction](#) (SPC3\_H spc3, State s)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Advanced\\_Mode](#) (SPC3\_H spc3, State s)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Background\\_Img](#) (SPC3\_H spc3, UInt16 \*Img)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Background\\_Subtraction](#) (SPC3\_H spc3, State s)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Gate\\_Values](#) (SPC3\_H spc3, Int16 Shift, Int16 Length)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Coarse\\_Gate\\_Values](#) (SPC3\_H spc3, int Counter, UInt16 Start, UInt16 Stop)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Gate\\_Mode](#) (SPC3\_H spc3, int counter, GateMode Mode)
- DIISDKExport [SPC3Return SPC3\\_Set\\_DualGate](#) (SPC3\_H spc3, State DualGate\_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int Gap, double \*CalibratedBinWidth)
- DIISDKExport [SPC3Return SPC3\\_Set\\_TripleGate](#) (SPC3\_H spc3, State TripleGate\_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int ThirdGateWidth, int Gap1, int Gap2, double \*CalibratedBinWidth)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Trigger\\_Out\\_State](#) (SPC3\_H spc3, TriggerMode Mode)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Sync\\_In\\_State](#) (SPC3\_H spc3, State s, int frames)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Live\\_Mode\\_ON](#) (SPC3\_H spc3)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Live\\_Mode\\_OFF](#) (SPC3\_H spc3)
- DIISDKExport [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)
- DIISDKExport [SPC3Return SPC3\\_Set\\_FLIM\\_State](#) (SPC3\_H spc3, State FLIM\_State)
- DIISDKExport [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 DIISDKExport 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. If several Set functions need to be called, there is no need to call this function after each Set function. A single call to this function at the end is enough to apply all the settings.

#### Parameters

<i>spc3</i>	SPC3 handle
-------------	-------------

#### Returns

- OK
- NULL\_POINTER The provided SPC3\_H points to an empty memory location

#### Examples:

[SDK\\_Example.c](#).

4.3.2.2 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>s</i>	Enable or disable the advanced mode

## Returns

OK

NULL\_POINTER The provided SPC3\_H points to an empty memory location

4.3.2.3 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>Img</i>	Pointer to a 2048 UInt16 array containing the background image. <b>WARNING</b> 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 DIISDKExport SPC3Return SPC3\_Set\_Background\_Subtraction ( SPC3\_H *spc3*, State *s* )

Enable or disable the hardware background subtraction.

## Parameters

<i>spc3</i>	SPC3 handle
<i>s</i>	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 DIISDKExport 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

<i>spc3</i>	SPC3 handle.
<i>Exposure</i>	Exposure time for a single frame. The time unit is 10 ns. Meaningful only for Advanced mode. Accepted values: 1 ... 65534
<i>NFrames</i>	Number of frames per acquisition. Meaningful only for Snap acquisition. Accepted values: 1 ... 65534
<i>NIntegFrames</i>	Number of integrated frames. Each output frame is the result of the sum of NIntegFrames. Accepted values: 1 ... 65534
<i>NCounters</i>	Number of counters per pixels to be used. Accepted values: 1 ... 3
<i>Force8bit</i>	Force 8 bit per pixel acquisition. Counts are trunked. Meaningful only for Advanced mode.
<i>Half_array</i>	Acquire only a 32x32 array.
<i>Signed_data</i>	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 DIISDKExport SPC3Return SPC3\_Set\_Camera\_Par\_SubArray ( SPC3\_H *spc3*, UInt16 *Exposure*, UInt32 *NFrames*, UInt16 *NIntegFrames*, State *Force8bit*, UInt16 *Npixels* )

Set the acquisition parameters for the camera when a subarray is used.

This function has to be used in alternative to the function [SPC3\\_Set\\_Camera\\_Par\(\)](#) when the acquisition of a subarray is needed. Subarray acquisition has few limitations:

- the pixels will be readout starting from the upper-left one and moving by lines of 32 pixels toward the center of the array
- in live mode it is possible to acquire only 1, 2, 4, 8 full lines. If Live mode is started when the camera is set for a different number of pixels, an INVALID\_OP error will be returned.
- in snap and continuous mode any number of pixels ranging from 1 to 256 in the upper semiarray can be acquired (in any case starting from the corner and then reading by lines, i.e. if 67 pixel are required 2 lines of 32 pixels + 3 pixels from the third line will be acquired), PROVIDED that the total data acquired is an integer multiple of 1024 bytes. For the example above and assuming you camera is set to 8bit/pixel, this means that 1024 frames (or multiples) must be acquired. Each frame will be Npixel\*10ns + 160ns long. Acquisitions with a smaller number of frames are not possible. If an acquisition is triggered with invalid values of the parameter, an INVALID\_OP error will be returned.
- the subarray acquisition is only available when using 1 counter.

This function behaves differently depending on the operating mode setting. In case of Normal working mode, the exposure time is fixed to  $N_{\text{pixel}} \times 10\text{ns} + 160\text{ns}$ . 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

<i>spc3</i>	SPC3 handle.
<i>Exposure</i>	Exposure time for a single frame. The time unit is 10 ns. Meaningful only for Advanced mode. Accepted values: 1 ... 65534
<i>NFrames</i>	Number of frames per acquisition. Meaningful only for Snap acquisition. Accepted values: 1 ... 65534
<i>NIntegFrames</i>	Number of integrated frames. Each output frame is the result of the sum of NIntegFrames. Accepted values: 1 ... 65534
<i>Force8bit</i>	Force 8 bit per pixel acquisition. Counts are trunked. Meaningful only for Advanced mode.
<i>Npixels</i>	Number of pixels to be acquired. Accepted values 1 ... 256 (see limitations above for Live mode)

#### Returns

OK  
 NULL\_POINTER The provided SPC3\_H points to an empty memory location  
 OUT\_OF\_BOUND Exposure, NFrames, NIntegFrames or NPixels out of bound.

#### 4.3.2.7 DIISDKExport SPC3Return SPC3\_Set\_Coarse\_Gate\_Values ( SPC3\_H *spc3*, int *Counter*, UInt16 *Start*, UInt16 *Stop* )

Change the coarse 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 has a period equal to the hardware integration time, and the start and stop time of the ON period can be adjusted with 10ns steps. Different gate settings can be applied to the 3 counters.

#### Parameters

<i>spc3</i>	SPC3 handle
<i>Counter</i>	Counter to which settings refer
<i>Start</i>	Starting position of the ON period. Can range from 0 to (HIT - 6), where units is 10ns and HIT is the Hardware Integration Time set with <a href="#">SPC3_Set_Camera_Par()</a> .
<i>Stop</i>	Stop position of the ON period. Can range from (Start+1) to (HIT - 5), where units is 10ns and HIT is the Hardware Integration Time set with <a href="#">SPC3_Set_Camera_Par()</a> .

#### Returns

OK  
 NULL\_POINTER The provided SPC3\_H points to an empty memory location  
 OUT\_OF\_BOUND Start or Stop are outside the valid values

#### 4.3.2.8 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>Val</i>	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](#).

4.3.2.9 DIISDKExport SPC3Return SPC3\_Set\_DeadTime\_Correction ( SPC3\_H *spc3*, State *s* )

Enable or disable the dead-time correction.

The default setting is disabled.

## Parameters

<i>spc3</i>	SPC3 handle
<i>s</i>	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](#).

4.3.2.10 DIISDKExport SPC3Return SPC3\_Set\_DualGate ( SPC3\_H *spc3*, State *DualGate\_State*, int *StartShift*, int *FirstGateWidth*, int *SecondGateWidth*, int *Gap*, double \* *CalibratedBinWidth* )

Set parameters for DualGate mode.

In this mode two counters are used, both in gated mode. Position and width of Gate 1 and width of Gate 2 can be set by user, whereas position of Gate 2 is automatically set at the end of Gate 1 plus a gap selected by the user, but not smaller than 2ns. Total duration of Gate 1 and Gate 2 plus gap can not exceed 90% of the gate period of 20ns.

## Parameters

<i>spc3</i>	SPC3 handle
<i>DualGate_State</i>	Enable or disable dualgate mode
<i>StartShift</i>	Start delay for for the first gate in thousandths of gate period (20ns). Accepted values: -400 ... +400
<i>FirstGateWidth</i>	Duration of the ON gate 1 signal. The unit is percentage. Accepted values: 0 ... 100
<i>SecondGateWidth</i>	Duration of the ON gate 2 signal. The unit is percentage. Accepted values: 0 ... 100
<i>Gap</i>	Gap between the two gates in thousandths of nominal gate period (20ns). Accepted values: >= 100
<i>CalibratedBinWidth</i>	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

INVALID\_OP This mode is not compatible with FLIM mode.

OUT\_OF\_RANGE 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.

4.3.2.11 DIISDKExport 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 employing 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\\_Camera\\_Par\(\)](#). The total time required to perform each FLIM acquisition is passed back to the caller through the referenced *FLIM\_frame\_time* variable.

## Parameters

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

## See also

[SPC3\\_Set\\_Camera\\_Par\(\)](#)4.3.2.12 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>FLIM_State</i>	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\\_Camera\\_Par\(\)](#)[SPC3\\_Set\\_FLIM\\_Par\(\)](#)4.3.2.13 DIISDKExport SPC3Return SPC3\_Set\_Gate\_Mode ( SPC3\_H *spc3*, int *counter*, GateMode *Mode* )

Set the gate mode to continuous, coarse or pulsed (only counter 1)

## Parameters

<i>spc3</i>	SPC3 handle
<i>counter</i>	Counter to which settings refer
<i>Mode</i>	New gate mode

## Returns

OK

NULL\_POINTER The provided SPC3\_H points to an empty memory location

INVALID\_OP Only counter 1 can be set to Pulsed mode, for fast gating also counter 2 and 3 refers to [SPC3\\_Set\\_DualGate\(\)](#) and [SPC3\\_Set\\_TripleGate\(\)](#) functions.

## Examples:

[SDK\\_Example.c](#).4.3.2.14 DIISDKExport SPC3Return SPC3\_Set\_Gate\_Values ( SPC3\_H *spc3*, Int16 *Shift*, Int16 *Length* )

Change the fast Gate settings for counter 1.

A gate signal is generated within the control electronics to select valid photons for counter 1, 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

<i>spc3</i>	SPC3 handle
<i>Shift</i>	Phase shift of the gate signal in the ON state. The unit is thousandths, 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
<i>Length</i>	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](#).



4.3.2.15 DIISDKExport SPC3Return SPC3\_Set\_Live\_Mode\_OFF ( SPC3\_H *spc3* )

Turn off the Live mode.

## Parameters

<i>spc3</i>	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](#).

4.3.2.16 DIISDKExport 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

<i>spc3</i>	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](#).

4.3.2.17 DIISDKExport SPC3Return SPC3\_Set\_Sync\_In\_State ( SPC3\_H *spc3*, State *s*, int *frames* )

Set the sync-in state.

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

## Parameters

<i>spc3</i>	SPC3 handle
<i>s</i>	Enable or disable the synchronization input
<i>frames</i>	If the synchronization input is enabled, this is the number of frames that are acquired for each pulse (0 means that the acquisition will wait only the first pulse and then continue to the end with no further pauses).

## Returns

OK  
 NULL\_POINTER The provided SPC3\_H points to an empty memory location

## Examples:

[SDK\\_Example.c](#).

4.3.2.18 DIISDKExport SPC3Return SPC3\_Set\_Trigger\_Out\_State ( SPC3\_H *spc3*, TriggerMode *Mode* )

Select the output signal.

## Parameters

<i>spc3</i>	Pointer to the SPC3 handle
<i>Mode</i>	New trigger mode

## Returns

OK  
 NULL\_POINTER The provided SPC3\_H points to an empty memory location

## Examples:

[SDK\\_Example.c](#).

4.3.2.19 DIISDKExport SPC3Return SPC3\_Set\_TripleGate ( SPC3\_H *spc3*, State *TripleGate\_State*, int *StartShift*, int *FirstGateWidth*, int *SecondGateWidth*, int *ThirdGateWidth*, int *Gap1*, int *Gap2*, double \* *CalibratedBinWidth* )

Set parameters for TripleGate mode.

In this mode three counters are used, all in gated mode. The three gates cannot overlap (even over different periods), and they have to follow the order: Gate1, Gate 3, Gate2. Position and width of Gate 1, and width of Gate 3 and 2 can be set by user, whereas position of Gate 3 and Gate 2 are automatically set by the function depending on the Gap1 and Gap2 values specified by the user. Gap1 between Gate1 and Gate3 can be as low as 0, Gap2 between Gate3 and Gate2 must be higher than 2ns. Total duration of Gate1, Gate3, Gate2 plus Gap1 and Gap2 can not exceed 90% of the gate period of 20ns.

## Parameters

<i>spc3</i>	SPC3 handle
<i>TripleGate_State</i>	Enable or disable triple-gate mode
<i>StartShift</i>	Start delay for for the first gate in thousandths of gate period (20ns). Accepted values: -400 ... +400
<i>FirstGateWidth</i>	Duration of the ON gate 1 signal. The unit is percentage. Accepted values: 0 ... 100
<i>SecondGateWidth</i>	Duration of the ON gate 3 signal. The unit is percentage. Accepted values: 0 ... 100
<i>ThirdGateWidth</i>	Duration of the ON gate 2 signal. The unit is percentage. Accepted values: 0 ... 100

<i>Gap1</i>	Gap between the gate1 and gate3 in thousandths of nominal gate period (20ns). Accepted values: $\geq 0$
<i>Gap2</i>	Gap between the gate3 and gate2 in thousandths of nominal gate period (20ns). Accepted values: $\geq 100$
<i>CalibratedBin-Width</i>	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

INVALID\_OP This mode is not compatible with FLIM mode.

OUT\_OF\_RANGE 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.

**Examples:**

[SDK\\_Example.c](#).

## 4.4 Get methods

### Functions

- DIISDKExport [SPC3Return SPC3\\_Get\\_Live\\_Img](#) (SPC3\_H spc3, UInt16 \*Img)
- DIISDKExport [SPC3Return SPC3\\_Prepare\\_Snap](#) (SPC3\_H spc3)
- DIISDKExport [SPC3Return SPC3\\_Get\\_Snap](#) (SPC3\_H spc3)
- DIISDKExport [SPC3Return SPC3\\_Get\\_Image\\_Buffer](#) (SPC3\_H spc3, BUFFER\_H \*buffer)
- DIISDKExport [SPC3Return SPC3\\_Get\\_Img\\_Position](#) (SPC3\_H spc3, UInt16 \*Img, UInt32 Position, UInt16 counter)
- DIISDKExport [SPC3Return SPC3\\_Start\\_ContAcq](#) (SPC3\_H spc3, char filename[256])
- DIISDKExport [SPC3Return SPC3\\_Get\\_Memory](#) (SPC3\_H spc3, double \*total\_bytes)
- DIISDKExport [SPC3Return SPC3\\_Stop\\_ContAcq](#) (SPC3\_H spc3)
- DIISDKExport [SPC3Return SPC3\\_Start\\_ContAcq\\_in\\_Memory](#) (SPC3\_H spc3)
- DIISDKExport [SPC3Return SPC3\\_Get\\_Memory\\_Buffer](#) (SPC3\_H spc3, double \*total\_bytes, BUFFER\_H \*buffer)
- DIISDKExport [SPC3Return SPC3\\_Stop\\_ContAcq\\_in\\_Memory](#) (SPC3\_H spc3)
- DIISDKExport [SPC3Return SPC3\\_Get\\_DeadTime](#) (SPC3\_H spc3, UInt16 Val, UInt16 \*RetVal)
- DIISDKExport [SPC3Return SPC3\\_Get\\_GateWidth](#) (SPC3\_H spc3, int counter, Int16 Val, double \*RetVal)
- DIISDKExport [SPC3Return SPC3\\_Get\\_GateShift](#) (SPC3\_H spc3, int counter, Int16 Val, Int16 \*RetVal)
- DIISDKExport [SPC3Return SPC3\\_Is16Bit](#) (SPC3\_H spc3, short \*is16bit)
- DIISDKExport [SPC3Return SPC3\\_IsTriggered](#) (SPC3\_H spc3, short \*isTriggered)
- DIISDKExport [SPC3Return SPC3\\_GetVersion](#) (SPC3\_H spc3, double \*Firmware\_Version, double \*Software\_Version, char \*Custom\_version)
- DIISDKExport [SPC3Return SPC3\\_GetSerial](#) (SPC3\_H spc3, char \*Camera\_ID, char \*Camera\_serial)
- DIISDKExport [SPC3Return SPC3\\_DeviceInfo](#) (char \*Device\_ID, char \*Camera\_serial, double \*Firmware\_Version, double \*Software\_Version, char \*Firmware\_Custom\_Version, char \*Software\_Custom\_Version)

#### 4.4.1 Detailed Description

Functions to get status or data from SPC3 camera.

#### 4.4.2 Function Documentation

4.4.2.1 DIISDKExport [SPC3Return SPC3\\_DeviceInfo](#) ( char \* *Device\_ID*, char \* *Camera\_serial*, double \* *Firmware\_Version*, double \* *Software\_Version*, char \* *Firmware\_Custom\_Version*, char \* *Software\_Custom\_Version* )

Get device info.

It gets device serial number, Unique ID, version and SDK version, without constructing an SPC3 object. Useful when constructor fails due to incompatible firmware and SDK versions or for powering issues.

#### Parameters

<i>Device_ID</i>	SPC3 camera Unique ID. A string of at least 11 character is required as parameter. If multiple devices are connected to the computer, a unique camera ID should be provided to correctly identify the camera. The camera ID can be found in the camera documentation. An empty string is accepted too. In this case, the first available device will be connected and the function will write in this variable the camera ID. This parameter is referenced. A string of at least 11 character is required as parameter. This parameter is referenced.
<i>Camera_serial</i>	SPC3 camera serial number. A string of at least 33 character is required as parameter. This parameter is referenced.
<i>Firmware_Version</i>	Version of the camera firmware in the format x.xx. This parameter is referenced.
<i>Software_Version</i>	Version of the SDK in the format x.xx. This parameter is referenced.

<i>Firmware_Custom_Version</i>	Customization version of the firmware. For standard model " " is returned. This parameter is referenced.
<i>Software_Custom_Version</i>	Customization version of the software. For standard model " " is returned. This parameter is referenced.

## Returns

OK

4.4.2.2 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>Val</i>	Desired dead-time value in ns. No error is generated when the value is above MAX_DEAD_TIME.
<i>ReturnVal</i>	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](#).4.4.2.3 DIISDKExport SPC3Return SPC3.Get\_GateShift ( SPC3\_H *spc3*, int *counter*, Int16 *Val*, Int16 \* *ReturnVal* )

Get the calibrated gate shift value.

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

## Parameters

<i>spc3</i>	SPC3 handle
<i>counter</i>	Counter for which the gate shift is requested
<i>Val</i>	Desired gate shift value in thousandths of 20ns. No error is generated when the value out of range, instead the real boundaries are forced on <i>ReturnVal</i> .
<i>ReturnVal</i>	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

#### 4.4.2.4 DIISDKExport SPC3Return SPC3\_Get\_GateWidth ( SPC3\_H *spc3*, int *counter*, Int16 *Val*, double \* *RetVal* )

Get the calibrated gate width value.

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

##### Parameters

<i>spc3</i>	SPC3 handle
<i>counter</i>	Counter for which the gate width is requested
<i>Val</i>	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 <i>RetVal</i> .
<i>RetVal</i>	Closest gate-width value possible. This parameter is referenced.

##### Returns

OK

NULL\_POINTER The provided SPC3\_H or *RetVal* point to an empty memory location

#### 4.4.2.5 DIISDKExport SPC3Return SPC3\_Get\_Image\_Buffer ( SPC3\_H *spc3*, BUFFER\_H \* *buffer* )

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

The first byte indicates if data is 8 or 16 bit. **WARNING** User must pay attention not to exceed the dimension of the buffer ( $2 \times 1024 \times 65534 + 1$  bytes) when accessing it.

##### Parameters

<i>spc3</i>	SPC3 handle
<i>buffer</i>	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\(\)](#)

#### 4.4.2.6 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>Img</i>	Pointer to the output image array. The size of the array must be at least 2 KB.
<i>Position</i>	Index of the image to save. Accepted values: 1 ... Number of acquired images
<i>counter</i>	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\(\)](#)4.4.2.7 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>Img</i>	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.](#)4.4.2.8 DIISDKExport 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 in SPC3 file format (for details on the format see function [SPC3\\_Save\\_Img\\_Disk](#)).

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

<i>spc3</i>	SPC3 handle
<i>total_bytes</i>	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 Continuous acquisition 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\(\)](#)[SPC3\\_Save\\_Img\\_Disk\(\);](#)**Examples:**[SDK\\_Example.c.](#)**4.4.2.9 DIISDKExport SPC3Return SPC3\_Get\_Memory\_Buffer ( SPC3\_H *spc3*, double \* *total\_bytes*, BUFFER\_H \* *buffer* )**

Dump the camera memory to the PC and pass the pointer to the image buffer in which acquisition is stored.

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** User must pay attention not to exceed the dimension of the buffer when accessing it. Refer to the value saved into the parameter *total\_bytes*.

**Parameters**

<i>spc3</i>	SPC3 handle
<i>total_bytes</i>	Total number of bytes read. Value is referenced
<i>buffer</i>	Pointer to the buffer Handle in which the function will save reference to the camera buffer. Value is referenced.

**Returns**

OK

NULL\_POINTER The provided SPC3\_H or BUFFER\_H point to an empty memory location

INVALID\_OP Continuous acquisition 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\\_in\\_Memory\(\)](#)[SPC3\\_SPC3\\_Stop\\_ContAcq\\_in\\_Memory\(\)](#)**Examples:**[SDK\\_Example.c.](#)



4.4.2.10 DIISDKExport 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

<i>spc3</i>	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](#).

4.4.2.11 DIISDKExport SPC3Return SPC3\_GetSerial ( SPC3\_H *spc3*, char \* *Camera\_ID*, char \* *Camera\_serial* )

Get the camera serial number and ID.

## Parameters

<i>spc3</i>	SPC3 handle
<i>Camera_ID</i>	Unique camera ID. A string of at least 11 character is required as parameter. This parameter is referenced.
<i>Camera_serial</i>	SPCX3 camera serial number. A string of at least 33 character is required as parameter. This parameter is referenced.

## Returns

OK

NULL\_POINTER The provided SPC3\_H points to an empty memory location

4.4.2.12 DIISDKExport SPC3Return SPC3\_GetVersion ( SPC3\_H *spc3*, double \* *Firmware\_Version*, double \* *Software\_Version*, char \* *Custom\_version* )

Get the SDK and camera firmware version.

## Parameters

<i>spc3</i>	SPC3 handle
<i>Firmware_Version</i>	Version of the camera firmare in the format x.xx. This parameter is referenced.
<i>Software_Version</i>	Version of the SDK in the format x.xx. This parameter is referenced.
<i>Custom_version</i>	Customization version of the firmware and SDK. For standard model " " is returned. This parameter is referenced.

## Returns

OK  
 NULL\_POINTER The provided SPC3\_H points to an empty memory location

4.4.2.13 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>is16bit</i>	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

4.4.2.14 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>isTriggered</i>	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](#).

4.4.2.15 DIISDKExport SPC3Return SPC3.Prepare\_Snap ( SPC3\_H *spc3* )

Prepare the camera to the acquisition 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

<i>spc3</i>	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](#).

4.4.2.16 DIISDKExport SPC3Return SPC3.Start\_ContAcq ( SPC3\_H *spc3*, char *filename*[256] )

Put the camera in "continuous acquisition" mode.

Compatible with FLIM mode. If the camera was set to wait for an external sync, the acquisition 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 function. 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

<i>spc3</i>	SPC3 handle
<i>filename</i>	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](#).

#### 4.4.2.17 DIISDKExport SPC3Return SPC3\_Start\_ContAcq\_in\_Memory ( SPC3\_H *spc3* )

Put the camera in "continuous acquisition" mode.

Compatible with FLIM mode. If the camera was set to wait for an external sync, the acquisition will start as soon as a pulse is detected on the Sync input, otherwise it will start immediately. Data are stored in the camera internal memory and must be downloaded calling the [SPC3\\_Get\\_Memory\\_Buffer\(\)](#) function as soon as possible, in order to avoid data loss.

##### Parameters

<i>spc3</i>	SPC3 handle
-------------	-------------

##### Returns

OK  
 NULL\_POINTER The provided SPC3\_H or BUFFER\_H point to an empty memory location

##### See also

[SPC3\\_Get\\_Memory\\_Buffer\(\)](#)  
[SPC3\\_SPC3\\_Stop\\_ContAcq\\_in\\_Memory\(\)](#)

##### Examples:

[SDK\\_Example.c](#).

#### 4.4.2.18 DIISDKExport SPC3Return SPC3\_Stop\_ContAcq ( SPC3\_H *spc3* )

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

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

##### Parameters

<i>spc3</i>	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](#).

#### 4.4.2.19 DIISDKExport SPC3Return SPC3\_Stop\_ContAcq\_in\_Memory ( SPC3\_H *spc3* )

Stop the continuous acquisition of data.

This function must be called at the end of the continuous acquisition. **WARNING** If not called, camera may have unexpected behaviour if other functions are called.

**Parameters**

<i>spc3</i>	SPC3 handle
-------------	-------------

**Returns**

OK  
NULL\_POINTER The provided SPC3\_H point to an empty memory location

**See also**

[SPC3\\_SPC3\\_Start\\_ContAcq\\_in\\_Memory\(\)](#)  
[SPC3\\_Get\\_Memory\\_Buffer\(\)](#)

**Examples:**

[SDK\\_Example.c](#).

## 4.5 Additional methods

### Functions

- DIISDKExport [SPC3Return SPC3\\_Save\\_Img\\_Disk](#) (SPC3\_H spc3, UInt32 Start\_Img, UInt32 End\_Img, char \*filename, [OutFileFormat](#) mode)
- DIISDKExport [SPC3Return SPC3\\_Save\\_Averaged\\_Img\\_Disk](#) (SPC3\_H spc3, int counter, char \*filename, [OutFileFormat](#) mode)
- DIISDKExport [SPC3Return SPC3\\_Save\\_FLIM\\_Disk](#) (SPC3\_H spc3, char \*filename, [OutFileFormat](#) mode)
- DIISDKExport [SPC3Return SPC3\\_ReadSPC3FileFormatImage](#) (char \*filename, UInt32 ImgIdx, UInt16 counter, UInt16 \*Img, char header[1024])
- DIISDKExport [SPC3Return SPC3\\_Average\\_Img](#) (SPC3\_H spc3, double \*Img, int counter)
- DIISDKExport [SPC3Return SPC3\\_StDev\\_Img](#) (SPC3\_H spc3, double \*Img, int counter)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Correlation\\_Mode](#) (SPC3\_H spc3, [CorrelationMode](#) CM, int NCorrChannels, [State](#) s)
- DIISDKExport [SPC3Return SPC3\\_Correlation\\_Img](#) (SPC3\_H spc3, int counter)
- DIISDKExport [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 DIISDKExport 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

<i>spc3</i>	SPC3 handle.
<i>Img</i>	Pointer to the output double image array. The size of the array must be at least 8 kB.
<i>counter</i>	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 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>counter</i>	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 DIISDKExport SPC3Return SPC3\_ReadSPC3FileFormatImage ( char \* filename, UInt32 *ImgIdx*, UInt16 *counter*, UInt16 \* *Img*, char *header*[1024] )

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

<i>filename</i>	Name of the output file
<i>ImgIdx</i>	Image index in the file. Accepted values: 1 ... 65534
<i>counter</i>	Desired counter. Accepted values: 1 ... 3
<i>Img</i>	Pointer to the output image array. The size of the array must be at least 2 kB.
<i>header</i>	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 DIISDKExport SPC3Return SPC3\_Save\_Averaged\_Img\_Disk ( SPC3\_H *spc3*, int *counter*, char \* *filename*, OutFileFormat *mode* )

Save the selected images on the hard disk.

This function saves the average of the images acquired by a specified counter on the hard disk. File format can be proprietary SPC3 or TIFF, as explained in [SPC3\\_Save\\_Img\\_Disk\(\)](#) function.

## Parameters

<i>spc3</i>	SPC3 handle
<i>counter</i>	Number of the counter to be saved
<i>filename</i>	Name of the output file. Value is referenced.
<i>mode</i>	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](#).

## 4.5.2.5 DIISDKExport 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 (N Pix)
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 <sup>th</sup> pixel
12 + 8 * (N Pix - 1) * NLag	double	8 * NLag	Autocorrelation values of the last pixel
12 + 8 * N Pix * 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

<i>spc3</i>	SPC3 handle
<i>filename</i>	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](#).4.5.2.6 DIISDKExport 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, containing 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 (0x4d5044ff03000001, starting with 4d on byte 0), and a metadata section of 1024 byte, as follows (multibyte fields are little-endian):

Byte offset	Number of bytes	Description
0	10	Unique camera ID (string)
10	32	SPC3 serial number (string)
42	2	Firmware version (1.11 saved as 111)
44	1	Firmware custom version (standard = 0)
45	20	Acquisition data&time (string)
65	35	Unused
100	1	Number of rows
101	1	Number of columns
102	1	Bit per pixel
103	1	Counters in use
104	2	Hardware integration time (multiples of 10ns)
106	2	Summed frames
108	1	Dead time correction enabled
109	1	Internal gate duty-cycle for counter1 (0-100%)
110	2	Holdoff time (ns)
112	1	Background subtraction enabled
113	1	Data for counters 1 and 2 are signed
114	4	Number of frames in the file

Byte offset	Number of bytes	Description
118	1	Image is averaged
119	1	Counter which is averaged
120	2	Number of averaged images
122	1	Internal gate duty-cycle for counter2 (0-100%)
123	1	Internal gate duty-cycle for counter3 (0-100%)
124	2	Frames per sync-in pulse
126	2	Number of pixels
128	72	Unused
200	1	FLIM enabled
201	2	FLIM shift %
203	2	FLIM steps
205	4	FLIM frame length (multiples of 10ns)
209	2	FLIM bin width (fs)
211	89	Unused
300	1	PDE measurement
301	2	Start wavelength (nm)
303	2	Stop wavelength (nm)
305	2	Step (nm)
307	717	unused

SPC3 file can be read using the provided ImageJ/Fiji plugin v2.20B. File generated with SDK v1.01 has a different header format and file signature, which is incompatible with plugin v1.11. A command line tool to convert SPC3 file from 1.01 format to 1.11 format is provided.

#### Parameters

<i>spc3</i>	SPC3 handle
<i>filename</i>	Name of the output file. Value is referenced.
<i>mode</i>	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

#### 4.5.2.7 DIISDKExport 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/>. If subarray acquisition is enabled and the number of pixels is not an integer multiple of 32, the TIFF files will have has much rows of 32 pixels needed to accomodate all pixels, and the missing pixels will be put to 0, e.g. if 67 pixels are acquired, the TIFF image will be 32x3, with the last 29 pixels of the 3rd row set to 0. SPC3 file are binary files composed by a

header with acquisition metadata followed by raw image data, containing 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 (0x4d5044ff03000001, starting with 4d on byte 0), and a metadata section of 1024 byte, as follows (multibyte fields are little-endian):

Byte offset	Number of bytes	Description
0	10	Unique camera ID (string)
10	32	SPC3 serial number (string)
42	2	Firmware version (1.11 saved as 111)
44	1	Firmware custom version (standard = 0)
45	20	Acquisition data&time (string)
65	35	Unused
100	1	Number of rows
101	1	Number of columns
102	1	Bit per pixel
103	1	Counters in use
104	2	Hardware integration time (multiples of 10ns)
106	2	Summed frames
108	1	Dead time correction enabled
109	1	Internal gate duty-cycle for counter1 (0-100%)
110	2	Holdoff time (ns)
112	1	Background subtraction enabled
113	1	Data for counters 1 and 2 are signed
114	4	Number of frames in the file
118	1	Image is averaged
119	1	Counter which is averaged
120	2	Number of averaged images
122	1	Internal gate duty-cycle for counter2 (0-100%)
123	1	Internal gate duty-cycle for counter3 (0-100%)
124	2	Frames per sync-in pulse
126	2	Number of pixels
128	72	Unused
200	1	FLIM enabled
201	2	FLIM shift %
203	2	FLIM steps
205	4	FLIM frame length (multiples of 10ns)
209	2	FLIM bin width (fs)
211	9	Unused
220	1	Multi gate mode: 2 = dual, 3 = triple
221	2	Multi gate mode: start position (-400 - +400)
223	1	Multi gate mode: first gate width (0-100%)
224	1	Multi gate mode: second gate width (0-100%)
225	1	Multi gate mode: third gate width (0-100%)
226	2	Multi gate mode: gap1 (0-800)
228	2	Multi gate mode: gap2 (0-800)
230	2	Multi gate mode: calibrated binwidth in fs
232	1	Coarse gate 1 enabled
233	2	Coarse gate 1 start
235	2	Coarse gate 1 stop
237	1	Coarse gate 2 enabled

Byte offset	Number of bytes	Description
238	2	Coarse gate 2 start
240	2	Coarse gate 2 stop
242	1	Coarse gate 3 enabled
243	2	Coarse gate 3 start
245	2	Coarse gate 3 stop
247	53	Unused
300	1	PDE measurement
301	2	Start wavelength (nm)
303	2	Stop wavelength (nm)
305	2	Step (nm)
307	717	unused

SPC3 file can be read using the provided ImageJ/Fiji plugin v2.01B. File generated with SDK v1.01 has a different header format and file signature, which is incompatible with plugin v1.11. A command line tool to convert SPC3 file from 1.01 format to 1.11 format is provided.

#### Parameters

<i>spc3</i>	SPC3 handle
<i>Start_Img</i>	Index of the first image to save. Accepted values: 1 ... Number of acquired images
<i>End_Img</i>	Index of the last image to save. Accepted values: Start_Img ... Number of acquired images
<i>filename</i>	Name of the output file. Value is referenced.
<i>mode</i>	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](#).

#### 4.5.2.8 DIISDKExport 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

<i>spc3</i>	SPC3 handle
<i>CM</i>	Selected autocorrelation algorithm
<i>NCorrChannels</i>	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 $2^i * \text{Exposure time}$ .
<i>s</i>	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.](#)4.5.2.9 DIISDKExport 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

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

## Returns

OK

NULL\_POINTER The provided SPC3\_H points to an empty memory location

INVALID\_OP No images were acquired

## 4.6 MPD only - Calibration functions

### Functions

- DIISDKExport [SPC3Return SPC3\\_Calibrate\\_DeadTime \(SPC3\\_H spc3\)](#)
- DIISDKExport [SPC3Return SPC3\\_Calibrate\\_Gate \(SPC3\\_H spc3\)](#)
- DIISDKExport [SPC3Return SPC3\\_Calibrate\\_Gate\\_Skew \(SPC3\\_H spc3, int Start\)](#)

#### 4.6.1 Detailed Description

Functions for internal MPD use.

#### 4.6.2 Function Documentation

##### 4.6.2.1 DIISDKExport 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

<i>spc3</i>	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 DIISDKExport 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

<i>spc3</i>	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

##### 4.6.2.3 DIISDKExport SPC3Return SPC3\_Calibrate\_Gate\_Skew ( SPC3\_H spc3, int Start )

Calibrate the gate-width.

Only for MPD use Calibrate the relative skew among gates. The function reads and updates the calibration file "GateCalib.dat" previously created by [SPC3\\_Calibrate\\_Gate\(\)](#).

## Parameters

<i>spc3</i>	SPC3 handle
<i>Start</i>	Gate start position

## Returns

OK

NULL\_POINTER The provided SPC3\_H points to an empty memory location

UNABLE\_CREATE\_FILE Missing "GateCalib.dat" file or editing impossible





# Chapter 5

## File Documentation

### 5.1 SPC3\_SDK.h File Reference

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

#### Macros

- #define `MIN_DEAD_TIME` 30
- #define `MAX_DEAD_TIME` 150
- #define `MAX_GATE_WIDTH` 100
- #define `MIN_GATE_WIDTH` 0
- #define `MAX_GATE_SHIFT` +400
- #define `MIN_GATE_SHIFT` -400

#### Typedefs

- typedef unsigned short `UInt16`
- typedef short `Int16`
- typedef unsigned `UInt32`
- 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_NO_COMPRESSION` = 2 }
- enum `GateMode` { `Continuous` = 0, `Pulsed` = 1, `Coarse` = 2 }
- 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

- `DIISDKExport SPC3Return SPC3_Constr` (`SPC3_H` \*spc3\_in, `CameraMode` m, char \*Device\_ID)
- `DIISDKExport SPC3Return SPC3_Destr` (`SPC3_H` spc3)
- `DIISDKExport void PrintErrorCode` (FILE \*fout, const char \*FunName, `SPC3Return` retcode)
- `DIISDKExport SPC3Return SPC3_Set_Camera_Par` (`SPC3_H` spc3, `UInt16` Exposure, `UInt32` NFrames, `UInt16` NIntegFrames, `UInt16` NCounters, `State` Force8bit, `State` Half\_array, `State` Signed\_data)
- `DIISDKExport SPC3Return SPC3_Set_Camera_Par_SubArray` (`SPC3_H` spc3, `UInt16` Exposure, `UInt32` NFrames, `UInt16` NIntegFrames, `State` Force8bit, `UInt16` Npixels)
- `DIISDKExport SPC3Return SPC3_Set_DeadTime` (`SPC3_H` spc3, `UInt16` Val)
- `DIISDKExport SPC3Return SPC3_Set_DeadTime_Correction` (`SPC3_H` spc3, `State` s)
- `DIISDKExport SPC3Return SPC3_Set_Advanced_Mode` (`SPC3_H` spc3, `State` s)
- `DIISDKExport SPC3Return SPC3_Set_Background_Img` (`SPC3_H` spc3, `UInt16` \*Img)
- `DIISDKExport SPC3Return SPC3_Set_Background_Subtraction` (`SPC3_H` spc3, `State` s)
- `DIISDKExport SPC3Return SPC3_Set_Gate_Values` (`SPC3_H` spc3, `Int16` Shift, `Int16` Length)
- `DIISDKExport SPC3Return SPC3_Set_Coarse_Gate_Values` (`SPC3_H` spc3, int Counter, `UInt16` Start, `UInt16` Stop)
- `DIISDKExport SPC3Return SPC3_Set_Gate_Mode` (`SPC3_H` spc3, int counter, `GateMode` Mode)
- `DIISDKExport SPC3Return SPC3_Set_DualGate` (`SPC3_H` spc3, `State` DualGate\_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int Gap, double \*CalibratedBinWidth)
- `DIISDKExport SPC3Return SPC3_Set_TripleGate` (`SPC3_H` spc3, `State` TripleGate\_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int ThirdGateWidth, int Gap1, int Gap2, double \*CalibratedBinWidth)
- `DIISDKExport SPC3Return SPC3_Set_Trigger_Out_State` (`SPC3_H` spc3, `TriggerMode` Mode)
- `DIISDKExport SPC3Return SPC3_Set_Sync_In_State` (`SPC3_H` spc3, `State` s, int frames)
- `DIISDKExport SPC3Return SPC3_Set_Live_Mode_ON` (`SPC3_H` spc3)
- `DIISDKExport SPC3Return SPC3_Set_Live_Mode_OFF` (`SPC3_H` spc3)
- `DIISDKExport 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)
- `DIISDKExport SPC3Return SPC3_Set_FLIM_State` (`SPC3_H` spc3, `State` FLIM\_State)
- `DIISDKExport SPC3Return SPC3_Apply_settings` (`SPC3_H` spc3)
- `DIISDKExport SPC3Return SPC3_Get_Live_Img` (`SPC3_H` spc3, `UInt16` \*Img)
- `DIISDKExport SPC3Return SPC3_Prepare_Snap` (`SPC3_H` spc3)
- `DIISDKExport SPC3Return SPC3_Get_Snap` (`SPC3_H` spc3)
- `DIISDKExport SPC3Return SPC3_Get_Image_Buffer` (`SPC3_H` spc3, `BUFFER_H` \*buffer)
- `DIISDKExport SPC3Return SPC3_Get_Img_Position` (`SPC3_H` spc3, `UInt16` \*Img, `UInt32` Position, `UInt16` counter)
- `DIISDKExport SPC3Return SPC3_Start_ContAcq` (`SPC3_H` spc3, char filename[256])
- `DIISDKExport SPC3Return SPC3_Get_Memory` (`SPC3_H` spc3, double \*total\_bytes)
- `DIISDKExport SPC3Return SPC3_Stop_ContAcq` (`SPC3_H` spc3)
- `DIISDKExport SPC3Return SPC3_Start_ContAcq_in_Memory` (`SPC3_H` spc3)
- `DIISDKExport SPC3Return SPC3_Get_Memory_Buffer` (`SPC3_H` spc3, double \*total\_bytes, `BUFFER_H` \*buffer)
- `DIISDKExport SPC3Return SPC3_Stop_ContAcq_in_Memory` (`SPC3_H` spc3)
- `DIISDKExport SPC3Return SPC3_Get_DeadTime` (`SPC3_H` spc3, `UInt16` Val, `UInt16` \*RetVal)
- `DIISDKExport SPC3Return SPC3_Get_GateWidth` (`SPC3_H` spc3, int counter, `Int16` Val, double \*RetVal)
- `DIISDKExport SPC3Return SPC3_Get_GateShift` (`SPC3_H` spc3, int counter, `Int16` Val, `Int16` \*RetVal)
- `DIISDKExport SPC3Return SPC3_Is16Bit` (`SPC3_H` spc3, short \*is16bit)
- `DIISDKExport SPC3Return SPC3_IsTriggered` (`SPC3_H` spc3, short \*isTriggered)
- `DIISDKExport SPC3Return SPC3_GetVersion` (`SPC3_H` spc3, double \*Firmware\_Version, double \*Software\_Version, char \*Custom\_version)
- `DIISDKExport SPC3Return SPC3_GetSerial` (`SPC3_H` spc3, char \*Camera\_ID, char \*Camera\_serial)

- DIISDKExport [SPC3Return SPC3\\_DeviceInfo](#) (char \*Device\_ID, char \*Camera\_serial, double \*Firmware\_Version, double \*Software\_Version, char \*Firmware\_Custom\_Version, char \*Software\_Custom\_Version)
- DIISDKExport [SPC3Return SPC3\\_Save\\_Img\\_Disk](#) (SPC3\_H spc3, [UInt32](#) Start\_Img, [UInt32](#) End\_Img, char \*filename, [OutFileFormat](#) mode)
- DIISDKExport [SPC3Return SPC3\\_Save\\_Averaged\\_Img\\_Disk](#) (SPC3\_H spc3, int counter, char \*filename, [OutFileFormat](#) mode)
- DIISDKExport [SPC3Return SPC3\\_Save\\_FLIM\\_Disk](#) (SPC3\_H spc3, char \*filename, [OutFileFormat](#) mode)
- DIISDKExport [SPC3Return SPC3\\_ReadSPC3FileFormatImage](#) (char \*filename, [UInt32](#) ImgIdx, [UInt16](#) counter, [UInt16](#) \*Img, char header[1024])
- DIISDKExport [SPC3Return SPC3\\_Average\\_Img](#) (SPC3\_H spc3, double \*Img, int counter)
- DIISDKExport [SPC3Return SPC3\\_StDev\\_Img](#) (SPC3\_H spc3, double \*Img, int counter)
- DIISDKExport [SPC3Return SPC3\\_Set\\_Correlation\\_Mode](#) (SPC3\_H spc3, [CorrelationMode](#) CM, int NCorrChannels, [State](#) s)
- DIISDKExport [SPC3Return SPC3\\_Correlation\\_Img](#) (SPC3\_H spc3, int counter)
- DIISDKExport [SPC3Return SPC3\\_Save\\_Correlation\\_Img](#) (SPC3\_H spc3, char \*filename)
- DIISDKExport [SPC3Return SPC3\\_Calibrate\\_DeadTime](#) (SPC3\_H spc3)
- DIISDKExport [SPC3Return SPC3\\_Calibrate\\_Gate](#) (SPC3\_H spc3)
- DIISDKExport [SPC3Return SPC3\\_Calibrate\\_Gate\\_Skew](#) (SPC3\_H spc3, int Start)

### 5.1.1 Detailed Description

SPC3 software development kit.

This C header contains all the functions to operate the SPC3 camera in user defined applications.

### 5.1.2 Macro Definition Documentation

#### 5.1.2.1 #define MAX\_DEAD\_TIME 150

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 30

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. 50 ns.

Examples:

[SDK\\_Example.c](#).



# Chapter 6

## Example Documentation

### 6.1 SDK\_Example.c

```
/*
#####

Copyright 2015-2017 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[1024]="";
    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 gateoff_array[3] = { 0 };
    double counts[3] = {0};
    double *x = NULL,*y= NULL, *z = NULL, *Imgd=NULL;
    double* data= NULL;
    char c=0,*fname=NULL;
    BUFFER_H buffer = NULL;
    int counter = 0, aggressor=0;
    SPC3Return error = OK;
    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
                    "Triple gate mode", //8
                    "Read and write test images", //9
                    "Calculate the correlation image", //a
                    "Continuous acquisition on file", //b
                    "Continuous acquisition in memory" //c
                    };
    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));
    z =(double*)calloc(1, 65535 * sizeof(double));

    // Simple menu for test selection //

    printf("*****\n");
    printf("SPC3 Test program\n");
    printf("*****\n\n\n");
    for (i = 0; i<13; 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' && c != 'c');
    getchar();
    if(c>47 && c<59)

```

```

{
    printf("*****\n");
    printf("%s\n", Test[c-48]);
    printf("*****\n\n");
}
if(c=='a')
{
    printf("*****\n");
    printf("%s\n", Test[10]);
    printf("*****\n\n");
}
if(c=='b')
{
    printf("*****\n");
    printf("%s\n", Test[11]);
    printf("*****\n\n");
}
if (c == 'c')
{
    printf("*****\n");
    printf("%s\n", Test[12]);
    printf("*****\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);
    //Acquisition 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\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\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_NO_COMPRESSION);
printf("The the dead-time corrected image was acquired and stored on the hard disk
successfully!\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\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\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_NO_COMPRESSION);
printf("The the dead-time corrected image was acquired and stored on the hard disk
successfully!\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\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_NO_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 subtraction 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_NO_COMPRESSION);
        //acquire image with background subtraction on
        SPC3_Set_Background_Subtraction(spc3, Enabled);
        SPC3_Prepare_Snap(spc3);
        SPC3_Get_Snap(spc3);
        SPC3_Save_Img_Disk(spc3, 1, 1, "BgSubt",
TIFF_NO_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_NO_COMPRESSION);
        //gated image
        SPC3_Set_Gate_Mode(spc3, 1, 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_NO_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, 0);
        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);
SPC3_Save_Img_Disk(spc3,1,100,"DCR_50ns_movie",
SPC3_FILEFORMAT);
SPC3_Save_Averaged_Img_Disk(spc3,1,"DCR_50ns_averaged",
SPC3_FILEFORMAT);

break;

case '7': // Calibrate gate
//SPC3 constructor and parameter setting
out=(int)SPC3_Constr(&spc3, Advanced,"");
SPC3_Set_Camera_Par(spc3, 1040 * 3, 10000, 5, 3, Disabled, Disabled,
Disabled);
SPC3_Set_DeadTime(spc3, 100);
SPC3_Apply_settings(spc3);
printf("Which counter do you want to acquire?\\n");
scanf("%d", &counter);
if ((counter < 1) || (counter > 3))
{
    printf("Insert a value between 1 and 3!\\n");
    break;
}
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();
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, 1, 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, 1, Pulsed);
SPC3_Apply_settings(spc3);
printf("Acquiring:\\n\\nGate\\t\\tMean\\t\\tActual Gate\\t\\t\\n");
//photon counts for gate width 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,"%3.0f %.2f %.2f\\n",x[i],y[i],y[i+101]);
}
fclose(f);
printf("\\n");
break;

```

```

    case '8': // Triple gate mode
        //SPC3 constructor and parameter setting
        fname = (char*) calloc(256, sizeof(char));
        out=(int)SPC3_Constr(&spc3, Advanced, "");
        SPC3_Set_Camera_Par(spc3, 1040 * 3, 1000, 20, 3, Disabled, Disabled,
Disabled);
        SPC3_Set_DeadTime(spc3, 45);

        printf("Expose the SPC3 camera to a luminous signal (pulsed or constant)\nPress ENTER to
        continue...\n");
        getchar();
        printf("Acquiring...\n");

        sprintf(fname, "Triple gate mode.txt");
        if ((f = fopen(fname, "w")) == NULL)
        {
            printf("Unable to open the file %s.\n", fname);
            break;
        }

        SPC3_Set_TripleGate(spc3, Disabled, -400, 10, 10, 10, 0, 100, NULL);
        SPC3_Set_Trigger_Out_State(spc3,
Gate_Clk);
        SPC3_Apply_settings(spc3);
        SPC3_Prepare_Snap(spc3);
        SPC3_Get_Snap(spc3);
        for (counter = 1; counter <= 3; counter++)
        {
            SPC3_Average_Img(spc3, data, counter);
            gateoff_array[counter - 1] = mean_double(data, 2048);

            printf("\nGate OFF counts on counter %d: %.2f\n", counter, gateoff_array[counter -
1]);
            fprintf(f, "Gate OFF counts on counter %d: %.2f\n", counter, gateoff_array[counter
- 1]);
        }

        for (j = -400; j < 400; j++)
        {
            error = SPC3_Set_TripleGate(spc3, Enabled, j, 10, 10, 10, 0, 100
, NULL);

            if (error < 0) break;
            SPC3_Apply_settings(spc3);
            SPC3_Prepare_Snap(spc3);
            SPC3_Get_Snap(spc3);
            for (counter = 1; counter <= 3; counter++)
            {
                SPC3_Average_Img(spc3, data, counter);
                counts[counter - 1] = mean_double(data, 2048);
            }

            printf("%d\t\t%.4f\t\t%.4f\t\t%.4f\n", j, counts[0] / gateoff_array[0] * 100,
counts[2] / gateoff_array[2] * 100, counts[1] / gateoff_array[1] * 100);
            fprintf(f, "%d\t\t%.4f\t\t%.4f\t\t%.4f\n", j, counts[0] / gateoff_array[0] * 100,
counts[2] / gateoff_array[2] * 100, counts[1] / gateoff_array[1] * 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.spc3", 1,1,Img, header
);
        printf("Rows: %d, Columns: %d\n",header[100], header[101]);
        for (i=1;i<=20;i++)
        {
            SPC3_ReadSPC3FileFormatImage("Test20_Im.spc3", 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;
    case 'c': //Continuous acquisition in memory, 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_in_Memory(spc3);
        for (i = 1; i<11; i++)
        {
            if (SPC3_Get_Memory_Buffer(spc3, &read_bytes, &buffer) ==
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_in_Memory(spc3);
            break;
        default:
            break;
    }
// Destructors //
    if(spc3)
        SPC3_Destr(spc3);
    spc3 = NULL;
    free(Img);
    free(Imgd);
    free(data);
    free(y);
    free(x);
    free(z);
    printf("Press ENTER to continue\n");
    getchar();
    return 0;
}
```

# Index

## Additional methods, [36](#)

- [SPC3\\_Average\\_Img, 36](#)
- [SPC3\\_Correlation\\_Img, 36](#)
- [SPC3\\_ReadSPC3FileFormatImage, 37](#)
- [SPC3\\_Save\\_Averaged\\_Img\\_Disk, 37](#)
- [SPC3\\_Save\\_Correlation\\_Img, 38](#)
- [SPC3\\_Save\\_FLIM\\_Disk, 39](#)
- [SPC3\\_Save\\_Img\\_Disk, 40](#)
- [SPC3\\_Set\\_Correlation\\_Mode, 42](#)
- [SPC3\\_StDev\\_Img, 43](#)

## Advanced

- [SPC3-SDK custom Types, 12](#)

## BUFFER\_H

- [SPC3-SDK custom Types, 11](#)

## CAMERA\_NOT\_POWERING\_UP

- [SPC3-SDK custom Types, 13](#)

## COMMUNICATION\_ERROR

- [SPC3-SDK custom Types, 13](#)

## CameraMode

- [SPC3-SDK custom Types, 12](#)

## Coarse

- [SPC3-SDK custom Types, 12](#)

## Constructr, destructor and error handling, [14](#)

- [PrintErrorCode, 14](#)
- [SPC3\\_Constr, 14](#)
- [SPC3\\_Destr, 14](#)

## Continuous

- [SPC3-SDK custom Types, 12](#)

## CorrelationMode

- [SPC3-SDK custom Types, 12](#)

## EMPTY\_BUFFER

- [SPC3-SDK custom Types, 13](#)

## FIRMWARE\_NOT\_COMPATIBLE

- [SPC3-SDK custom Types, 13](#)

## Frame

- [SPC3-SDK custom Types, 13](#)

## Gate\_Clk

- [SPC3-SDK custom Types, 13](#)

## GateMode

- [SPC3-SDK custom Types, 12](#)

## Get methods, [26](#)

- [SPC3\\_DeviceInfo, 26](#)
- [SPC3\\_Get\\_DeadTime, 27](#)
- [SPC3\\_Get\\_GateShift, 27](#)

## SPC3\_Get\_GateWidth, [27](#)

## SPC3\_Get\_Image\_Buffer, [28](#)

## SPC3\_Get\_Img\_Position, [28](#)

## SPC3\_Get\_Live\_Img, [29](#)

## SPC3\_Get\_Memory, [29](#)

## SPC3\_Get\_Memory\_Buffer, [30](#)

## SPC3\_Get\_Snap, [30](#)

## SPC3\_GetSerial, [31](#)

## SPC3\_GetVersion, [31](#)

## SPC3\_Is16Bit, [32](#)

## SPC3\_IsTriggered, [32](#)

## SPC3\_Prepare\_Snap, [32](#)

## SPC3\_Start\_ContAcq, [33](#)

## SPC3\_Start\_ContAcq\_in\_Memory, [34](#)

## SPC3\_Stop\_ContAcq, [34](#)

## SPC3\_Stop\_ContAcq\_in\_Memory, [34](#)

## INVALID\_NIMG\_CORRELATION

- [SPC3-SDK custom Types, 13](#)

## INVALID\_OP

- [SPC3-SDK custom Types, 13](#)

## Linear

- [SPC3-SDK custom Types, 12](#)

## MAX\_DEAD\_TIME

- [SPC3\\_SDK.h, 49](#)

## MIN\_DEAD\_TIME

- [SPC3\\_SDK.h, 49](#)

## MISSING\_DLL

- [SPC3-SDK custom Types, 13](#)

## MPD only - Calibration functions, [44](#)

## SPC3\_Calibrate\_DeadTime, [44](#)

## SPC3\_Calibrate\_Gate, [44](#)

## SPC3\_Calibrate\_Gate\_Skew, [44](#)

## MultiTau

- [SPC3-SDK custom Types, 12](#)

## NOT\_EN\_MEMORY

- [SPC3-SDK custom Types, 13](#)

## NULL\_POINTER

- [SPC3-SDK custom Types, 13](#)

## None

- [SPC3-SDK custom Types, 13](#)

## Normal

- [SPC3-SDK custom Types, 12](#)

## OUT\_OF\_BOUND

- [SPC3-SDK custom Types, 13](#)

- OK
  - SPC3-SDK custom Types, [13](#)
- OutFileFormat
  - SPC3-SDK custom Types, [12](#)
- POWER\_SUPPLY\_ERROR
  - SPC3-SDK custom Types, [13](#)
- PrintErrorCode
  - Constructr, destructor and error handling, [14](#)
- Pulsed
  - SPC3-SDK custom Types, [12](#)
- SPC3-SDK custom Types, [11](#)
  - Advanced, [12](#)
  - BUFFER\_H, [11](#)
  - CAMERA\_NOT\_POWERING\_UP, [13](#)
  - COMMUNICATION\_ERROR, [13](#)
  - CameraMode, [12](#)
  - Coarse, [12](#)
  - Continuous, [12](#)
  - CorrelationMode, [12](#)
  - EMPTY\_BUFFER, [13](#)
  - FIRMWARE\_NOT\_COMPATIBLE, [13](#)
  - Frame, [13](#)
  - Gate\_Clk, [13](#)
  - GateMode, [12](#)
  - INVALID\_NIMG\_CORRELATION, [13](#)
  - INVALID\_OP, [13](#)
  - Linear, [12](#)
  - MISSING\_DLL, [13](#)
  - MultiTau, [12](#)
  - NOT\_EN\_MEMORY, [13](#)
  - NULL\_POINTER, [13](#)
  - None, [13](#)
  - Normal, [12](#)
  - OUT\_OF\_BOUND, [13](#)
  - OK, [13](#)
  - OutFileFormat, [12](#)
  - POWER\_SUPPLY\_ERROR, [13](#)
  - Pulsed, [12](#)
  - SPC3\_FILEFORMAT, [12](#)
  - SPC3\_MEMORY\_FULL, [13](#)
  - SPC3\_H, [11](#)
  - SPC3Return, [13](#)
  - TIFF\_NO\_COMPRESSION, [12](#)
  - TOO\_MUCH\_LIGHT, [13](#)
  - TriggerMode, [13](#)
  - UNABLE\_CREATE\_FILE, [13](#)
  - UNABLE\_READ\_FILE, [13](#)
  - USB\_DEVICE\_NOT\_RECOGNIZED, [13](#)
- SPC3\_Apply\_settings
  - Set methods, [16](#)
- SPC3\_Average\_Img
  - Additional methods, [36](#)
- SPC3\_Calibrate\_DeadTime
  - MPD only - Calibration functions, [44](#)
- SPC3\_Calibrate\_Gate
  - MPD only - Calibration functions, [44](#)
- SPC3\_Calibrate\_Gate\_Skew
  - MPD only - Calibration functions, [44](#)
- SPC3\_Constr
  - Constructr, destructor and error handling, [14](#)
- SPC3\_Correlation\_Img
  - Additional methods, [36](#)
- SPC3\_Destr
  - Constructr, destructor and error handling, [14](#)
- SPC3\_DeviceInfo
  - Get methods, [26](#)
- SPC3\_FILEFORMAT
  - SPC3-SDK custom Types, [12](#)
- SPC3\_Get\_DeadTime
  - Get methods, [27](#)
- SPC3\_Get\_GateShift
  - Get methods, [27](#)
- SPC3\_Get\_GateWidth
  - Get methods, [27](#)
- SPC3\_Get\_Image\_Buffer
  - Get methods, [28](#)
- SPC3\_Get\_Img\_Position
  - Get methods, [28](#)
- SPC3\_Get\_Live\_Img
  - Get methods, [29](#)
- SPC3\_Get\_Memory
  - Get methods, [29](#)
- SPC3\_Get\_Memory\_Buffer
  - Get methods, [30](#)
- SPC3\_Get\_Snap
  - Get methods, [30](#)
- SPC3\_GetSerial
  - Get methods, [31](#)
- SPC3\_GetVersion
  - Get methods, [31](#)
- SPC3\_Is16Bit
  - Get methods, [32](#)
- SPC3\_IsTriggered
  - Get methods, [32](#)
- SPC3\_MEMORY\_FULL
  - SPC3-SDK custom Types, [13](#)
- SPC3\_Prepare\_Snap
  - Get methods, [32](#)
- SPC3\_ReadSPC3FileFormatImage
  - Additional methods, [37](#)
- SPC3\_SDK.h, [47](#)
  - MAX\_DEAD\_TIME, [49](#)
  - MIN\_DEAD\_TIME, [49](#)
- SPC3\_Save\_Averaged\_Img\_Disk
  - Additional methods, [37](#)
- SPC3\_Save\_Correlation\_Img
  - Additional methods, [38](#)
- SPC3\_Save\_FLIM\_Disk
  - Additional methods, [39](#)
- SPC3\_Save\_Img\_Disk
  - Additional methods, [40](#)
- SPC3\_Set\_Advanced\_Mode
  - Set methods, [16](#)
- SPC3\_Set\_Background\_Img
  - Set methods, [17](#)

- SPC3\_Set\_Background\_Subtraction
  - Set methods, [17](#)
- SPC3\_Set\_Camera\_Par
  - Set methods, [17](#)
- SPC3\_Set\_Camera\_Par\_SubArray
  - Set methods, [18](#)
- SPC3\_Set\_Coarse\_Gate\_Values
  - Set methods, [19](#)
- SPC3\_Set\_Correlation\_Mode
  - Additional methods, [42](#)
- SPC3\_Set\_DeadTime
  - Set methods, [19](#)
- SPC3\_Set\_DeadTime\_Correction
  - Set methods, [20](#)
- SPC3\_Set\_DualGate
  - Set methods, [20](#)
- SPC3\_Set\_FLIM\_Par
  - Set methods, [21](#)
- SPC3\_Set\_FLIM\_State
  - Set methods, [21](#)
- SPC3\_Set\_Gate\_Mode
  - Set methods, [22](#)
- SPC3\_Set\_Gate\_Values
  - Set methods, [22](#)
- SPC3\_Set\_Live\_Mode\_OFF
  - Set methods, [22](#)
- SPC3\_Set\_Live\_Mode\_ON
  - Set methods, [23](#)
- SPC3\_Set\_Sync\_In\_State
  - Set methods, [23](#)
- SPC3\_Set\_Trigger\_Out\_State
  - Set methods, [24](#)
- SPC3\_Set\_TripleGate
  - Set methods, [24](#)
- SPC3\_StDev\_Img
  - Additional methods, [43](#)
- SPC3\_Start\_ContAcq
  - Get methods, [33](#)
- SPC3\_Start\_ContAcq\_in\_Memory
  - Get methods, [34](#)
- SPC3\_Stop\_ContAcq
  - Get methods, [34](#)
- SPC3\_Stop\_ContAcq\_in\_Memory
  - Get methods, [34](#)
- SPC3\_H
  - SPC3-SDK custom Types, [11](#)
- SPC3Return
  - SPC3-SDK custom Types, [13](#)
- Set methods, [16](#)
  - SPC3\_Apply\_settings, [16](#)
  - SPC3\_Set\_Advanced\_Mode, [16](#)
  - SPC3\_Set\_Background\_Img, [17](#)
  - SPC3\_Set\_Background\_Subtraction, [17](#)
  - SPC3\_Set\_Camera\_Par, [17](#)
  - SPC3\_Set\_Camera\_Par\_SubArray, [18](#)
  - SPC3\_Set\_Coarse\_Gate\_Values, [19](#)
  - SPC3\_Set\_DeadTime, [19](#)
  - SPC3\_Set\_DeadTime\_Correction, [20](#)
  - SPC3\_Set\_DualGate, [20](#)
  - SPC3\_Set\_FLIM\_Par, [21](#)
  - SPC3\_Set\_FLIM\_State, [21](#)
  - SPC3\_Set\_Gate\_Mode, [22](#)
  - SPC3\_Set\_Gate\_Values, [22](#)
  - SPC3\_Set\_Live\_Mode\_OFF, [22](#)
  - SPC3\_Set\_Live\_Mode\_ON, [23](#)
  - SPC3\_Set\_Sync\_In\_State, [23](#)
  - SPC3\_Set\_Trigger\_Out\_State, [24](#)
  - SPC3\_Set\_TripleGate, [24](#)
- TIFF\_NO\_COMPRESSION
  - SPC3-SDK custom Types, [12](#)
- TOO\_MUCH\_LIGHT
  - SPC3-SDK custom Types, [13](#)
- TriggerMode
  - SPC3-SDK custom Types, [13](#)
- UNABLE\_CREATE\_FILE
  - SPC3-SDK custom Types, [13](#)
- UNABLE\_READ\_FILE
  - SPC3-SDK custom Types, [13](#)
- USB\_DEVICE\_NOT\_RECOGNIZED
  - SPC3-SDK custom Types, [13](#)