Single Photon Counting Camera SDK

SPC²

Version 3.2.1

Software Development Kit
Manual

Feb 15, 2014
4.3.2.3 SPC2_Set_Background_Img .................................................. 18
4.3.2.4 SPC2_Set_Background_Subtraction ........................................ 18
4.3.2.5 SPC2_Set_Camera_Par .......................................................... 18
4.3.2.6 SPC2_Set_Camera_SubArray .................................................. 19
4.3.2.7 SPC2_Set_DeadTime ............................................................ 19
4.3.2.8 SPC2_Set_DeadTime_Correction ............................................. 20
4.3.2.9 SPC2_Set_Gate_Mode ........................................................... 20
4.3.2.10 SPC2_Set_Gate_Values ....................................................... 20
4.3.2.11 SPC2_Set_Live_Mode_OFF ................................................... 21
4.3.2.12 SPC2_Set_Live_Mode_ON .................................................... 21
4.3.2.13 SPC2_Set_Sync_In_State ........................................................ 22
4.3.2.14 SPC2_Set_Trigger_Out_State ................................................. 22

4.4 Get methods ................................................................. 23
4.4.1 Detailed Description .......................................................... 23
4.4.2 Function Documentation ....................................................... 23
  4.4.2.1 SPC2_Get_DeadTime ......................................................... 23
  4.4.2.2 SPC2_Get_GateShift .......................................................... 23
  4.4.2.3 SPC2_Get_GateWidth ......................................................... 24
  4.4.2.4 SPC2_Get_Image_Buffer ..................................................... 24
  4.4.2.5 SPC2_Get_Img_Position ..................................................... 25
  4.4.2.6 SPC2_Get_Live_Img ........................................................... 25
  4.4.2.7 SPC2_Get_Memory ............................................................. 26
  4.4.2.8 SPC2_Get_Snap ............................................................... 26
  4.4.2.9 SPC2_GetVersion .............................................................. 27
  4.4.2.10 SPC2_Is16Bit ................................................................. 27
  4.4.2.11 SPC2_IsTriggered ............................................................ 27
  4.4.2.12 SPC2_Prepare_Snap .......................................................... 28
  4.4.2.13 SPC2_Start_ContAcq ........................................................ 28
  4.4.2.14 SPC2_Stop_ContAcq .......................................................... 29

4.5 Additional methods .......................................................... 30
4.5.1 Detailed Description .......................................................... 30
4.5.2 Function Documentation ....................................................... 30
  4.5.2.1 SPC2_Average_Img ........................................................... 30
  4.5.2.2 SPC2_Correlation_Img ....................................................... 30
  4.5.2.3 SPC2_ReadSPC2FileFormatImage ......................................... 31
  4.5.2.4 SPC2_Save_Correlation_Img ............................................... 31
  4.5.2.5 SPC2_Save_Img_Disk .......................................................... 32
  4.5.2.6 SPC2_Set_Correlation_Mode ................................................. 34
  4.5.2.7 SPC2_StDev_Img .............................................................. 34

4.6 MPD only - Calibration functions .......................................... 36
Chapter 1

Single Photon Counting Camera Software Development Kit (SPC2-SDK).

SPC2 is a 2-D imaging chip based on a 32 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 50 000 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 c-mount mechanical adapter and a high-speed USB 2.0 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:

- SPC2_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:

- SPC2-SDK custom Types .......................................................... 11
- Constructor, destructor and error handling ..................................... 15
- Set methods .............................................................................. 17
- Get methods ............................................................................. 23
- Additional methods .................................................................. 30
- MPD only - Calibration functions .............................................. 36
Chapter 3

File Index

3.1 File List

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

<table>
<thead>
<tr>
<th>File Path</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPC2_SDK.h</td>
<td>SPC2 software development kit</td>
<td>37</td>
</tr>
</tbody>
</table>
Chapter 4

Module Documentation

4.1 SPC2-SDK custom Types

Typedefs

- typedef struct _SPC2_H * SPC2_H
- typedef unsigned char * BUFFER_H

Enumerations

- enum SPC2Return {
  OK = 0, USB_DEVICE_NOT_RECOGNIZED = -1, ELECTRONIC_INTERFACE_NOT_RECOGNIZED = -2, FAILED_FPGA_CONFIGURATION = -3, FPGA_USB_DRIVER_FAILURE = -4, COMMUNICATION_ERROR = -5, OUT_OF_BOUND = -6, MISSING_DLL = -7, EMPTY_BUFFER = -8, NOT_EN_MEMORY = -9, NULL_POINTER = -10, INVALID_OP = -11, UNABLE_CREATE_FILE = -12, UNABLE_READ_FILE = -13, FIRMWARE_NOT_COMPATIBLE = -14, USB_PORT_NOT_EN_POWER = -15, TOO_MUCH_LIGHT = -16, INVALID_NIMG_CORRELATION = -17, SPC2_MEMORY_FULL = -18 }
- enum OutFileFormat { SPC2_FILEFORMAT = 0, TIFF_LZW_COMPRESSION = 1, TIFF_NO_COMPRESSION = 2 }
- enum GateMode { Continuous = 0, Pulsed = 1 }
- enum CameraMode { Normal = 0, Advanced = 1 }
- enum TriggerMode { None = 0, Gate_Clk = 1, Frame = 2 }
- enum State { Disabled = 0, Enabled = 1 }
- enum CorrelationMode { Linear = 0, MultiTau = 1 }

4.1.1 Detailed Description

Custom types used by the SDK.

4.1.2 Typedef Documentation

4.1.2.1 typedef unsigned char * BUFFER_H

Handle to the SPC2 buffer.
4.1.2.2 typedef struct SPC2_H* SPC2_H

Handle to the SPC2 structure.

4.1.3 Enumeration Type Documentation

4.1.3.1 enum CameraMode

SPC2 working mode.

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

Enumerator

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

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

4.1.3.2 enum CorrelationMode

Type of correlation function.

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

Enumerator

Linear Selects the linear correlation algorithm.

MultiTau Selects the linear multi-tau algorithm.

4.1.3.3 enum GateMode

Gate setting.

Enable and disable the software gating. When the setting is Enabled, the SPC2 discards the detected photons by the SPAD matrix if measured outside a valid gate signal.

Enumerator

Continuous The gate signal is always valid.

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

4.1.3.4 enum OutFileFormat

Output file format.

Table of the available output file formats for the saved images
4.1 SPC2-SDK custom Types

4.1.3.5 enum SPC2Return

Error table.

Error code returned by the SPC2 functions.

**Enum**

SPC2_FILEFORMAT SPC2 custom file format: the first byte contains the value 8 or 16 to define whether the image has 8 or 16 bit per pixel. Then the pixel values follow in row-major order. The byte order is little-endian for the 16 bit images.

TIFF_LZW_COMPRESSION Multipage TIFF files LZW compressed. **WARNING** the creation of TIFF files might require long execution times.

TIFF_NO_COMPRESSION Multipage TIFF without compression. **WARNING** the creation of TIFF files might require long execution times.

4.1.3.6 enum TriggerMode

Type of synchronization output.

The Synch-out SMA port can output different signals.
Enumerator

- **None**  No output signal.
- **Gate_Clk**  A square wave of 50 MHz and 50% duty cycle synchronized with the software gate signal and the camera clock.
- **Frame**  A 60 ns pulse every time a new frame is acquired.
4.2 Constructr, destructor and error handling

Functions

- DllSDKExport SPC2Return SPC2_Constr (SPC2_H *spc2_in, UInt16 minRow, UInt16 maxRow, UInt16 minCol, UInt16 maxCol, CameraMode m, char *Device_ID)
- DllSDKExport SPC2Return SPC2_Destr (SPC2_H spc2)
- DllSDKExport void PrintErrorCode (FILE *fout, const char *FunName, SPC2Return retcode)

4.2.1 Detailed Description

Functions to construct and destruct SPC2 objects, and for error handling.

4.2.2 Function Documentation

4.2.2.1 DllSDKExport void PrintErrorCode ( FILE *fout, const char *FunName, SPC2Return 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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>fout</td>
<td>Output text file</td>
</tr>
<tr>
<td>FunName</td>
<td>Additional text to define the warning/error. Usually the name of the calling function is provided.</td>
</tr>
<tr>
<td>retcode</td>
<td>Error code returned by a SDK command</td>
</tr>
</tbody>
</table>

4.2.2.2 DllSDKExport SPC2Return SPC2_Constr ( SPC2_H *spc2_in, UInt16 minRow, UInt16 maxRow, UInt16 minCol, UInt16 maxCol, CameraMode m, char *Device_ID )

Constructor.

It allocates a memory block to contain all the information and buffers required by the SPC2. A contiguous subset of columns and rows can be selected instead of the whole chip. If multiple devices are connected to the computer, a unique Device ID should be provided to correctly identify the camera. The camera ID can be found in the camera documentation (9 numbers and a letter) and it is printed on the screen during initialization. An empty string is accepted too. In this case, the devices will be connected in the order which is printed on the screen.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spc2_in</td>
<td>Pointer to SPC2 handle</td>
</tr>
<tr>
<td>minRow</td>
<td>First selected row. Accepted values: 1 ... 32</td>
</tr>
<tr>
<td>maxRow</td>
<td>Last selected row. Accepted values: minRow + 1 ... 32</td>
</tr>
<tr>
<td>minCol</td>
<td>First selected column. Accepted values: 1 ... 32</td>
</tr>
<tr>
<td>maxCol</td>
<td>Last selected column. Accepted values: minCol + 1 ... 32</td>
</tr>
<tr>
<td>m</td>
<td>Camera Working mode</td>
</tr>
<tr>
<td>Device_ID</td>
<td>Unique ID to identify the connected device</td>
</tr>
</tbody>
</table>

Returns

OK
- INVALID_OP The SPC2_H points to an occupied memory location
- OUT_OF_BOUND maxCol or maxRow is above 32, minCol or min Row is below 1 or min > max
- FIRMWARE_NOT_COMPATIBLE The SDK and Firmware versions are not compatible
- NOT_EN_MEMORY There is not enough memory to run the camera
Examples:

SDK_Example.c.

4.2.2.3 DllSDKExport SPC2Return SPC2_Destr ( SPC2_H spc2 )

Destructor.
It deallocates the memory block which contains all the information and buffers required by the SPC2. **WARNING** the user must call the destructor before the end of the program to avoid memory leakages.

Parameters

| spc2 | SPC2 handle |

Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location

Examples:

SDK_Example.c.
4.3 Set methods

Functions

- DllSDKExport SPC2Return SPC2_Set_Camera_Par (SPC2_H spc2, UInt16 Exposure, UInt32 NFrames, UInt16 NIntegFrames)
- DllSDKExport SPC2Return SPC2_Set_Camera_SubArray (SPC2_H spc2, UInt16 minRow, UInt16 maxRow, UInt16 minCol, UInt16 maxCol)
- DllSDKExport SPC2Return SPC2_Set_DeadTime (SPC2_H spc2, UInt16 Val)
- DllSDKExport SPC2Return SPC2_Set_DeadTime_Correction (SPC2_H spc2, State s)
- DllSDKExport SPC2Return SPC2_Set_Advanced_Mode (SPC2_H spc2, State s)
- DllSDKExport SPC2Return SPC2_Set_Background_Img (SPC2_H spc2, UInt16 *Img)
- DllSDKExport SPC2Return SPC2_Set_Background_Subtraction (SPC2_H spc2, State s)
- DllSDKExport SPC2Return SPC2_Set_Gate_Values (SPC2_H spc2, Int16 Shift, Int16 Length)
- DllSDKExport SPC2Return SPC2_Set_Gate_Mode (SPC2_H spc2, GateMode Mode)
- DllSDKExport SPC2Return SPC2_Set_Trigger_Out_State (SPC2_H spc2, TriggerMode Mode)
- DllSDKExport SPC2Return SPC2_Set_Sync_In_State (SPC2_H spc2, State s)
- DllSDKExport SPC2Return SPC2_Set_Live_Mode_ON (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Set_Live_Mode_OFF (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Apply_settings (SPC2_H spc2)

4.3.1 Detailed Description

Functions to set parameters of the SPC2 camera.

4.3.2 Function Documentation

4.3.2.1 DllSDKExport SPC2Return SPC2_Apply_settings ( SPC2_H spc2 )

Apply settings to the camera.
This function must be called after any Set function, except SPC2_Set_Live_Mode_ON() and SPC2_Set_Live_Mode_OFF(), in order to apply the settings to the camera.

Parameters

| spc2 | SPC2 handle |

Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location

Examples:

SDK_Example.c.

4.3.2.2 DllSDKExport SPC2Return SPC2_Set_Advanced_Mode ( SPC2_H spc2, State s )

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

Parameters

| spc2 | SPC2 handle |
| s | Enable or disable the advanced mode |
Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location

4.3.2.3 DiISDKExport SPC2Return SPC2_Set_Background_Img ( SPC2_H spc2, 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

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Img</td>
<td>Pointer to a 1024 UInt16 array containing the background image. <strong>WARNING</strong> The user should check the array size to avoid the corruption of the memory heap.</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_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 SPC2Return SPC2_Set_Background_Subtraction ( SPC2_H spc2, State s )

Enable or disable the hardware background subtraction.

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>Enable or disable the background subtraction</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location

Examples:

SDK_Example.c.

4.3.2.5 DiISDKExport SPC2Return SPC2_Set_Camera_Par ( SPC2_H spc2, UInt16 Exposure, UInt32 NFrames, UInt16 NIntegFrames )

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 20.74 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
4.3 Set methods

Parameter is clock cycles i.e. the exposure time is an integer number of internal clock cycles of 20 ns period. For example, the value of 10 means 200 ns exposure.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spc2</td>
<td>SPC2 handle.</td>
</tr>
<tr>
<td>Exposure</td>
<td>Exposure time for a single frame. The time unit is 20 ns. Meaningful only for Advanced mode. Accepted values: 1 ... 65534</td>
</tr>
<tr>
<td>NFrames</td>
<td>Number of frames per acquisition. Meaningful only for Snap acquisition. Accepted values: 1 ... 65534</td>
</tr>
<tr>
<td>NIntegFrames</td>
<td>Number of integrated frames. Each output frame is the result of the sum of NIntegFrames. Accepted values: 1 ... 65534</td>
</tr>
</tbody>
</table>

Returns

OK
NULL POINTER The provided SPC2_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 DiIISDKExport SPC2Return SPC2_SetCamera_SubArray ( SPC2_H spc2, UInt16 minRow, UInt16 maxRow, UInt16 minCol, UInt16 maxCol )

Set the subarray to be acquired.

This function allows the setting of a continuous subarray in any position of the array, down to a 2x2 subarray. Setting a subarray smaller than 32x32, allows a faster readout, thus an higher frame-rate.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spc2</td>
<td>SPC2 handle.</td>
</tr>
<tr>
<td>minRow</td>
<td>First row of the subarray (1-31)</td>
</tr>
<tr>
<td>maxRow</td>
<td>Last row of the subarray (2-32)</td>
</tr>
<tr>
<td>minCol</td>
<td>First column of the subarray (1-31)</td>
</tr>
<tr>
<td>maxCol</td>
<td>Last column of the subarray (2-32)</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location
OUT_OF_BOUND Rows or columns parameters are out of allowed values

Examples:

SDK_Example.c.

4.3.2.7 DiIISDKExport SPC2Return SPC2_SetDeadTime ( SPC2_H spc2, 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 250 ns.
4.3.2.8  DllSDKExport SPC2Return SPC2_Set_DeadTime_Correction ( SPC2_H spc2, State s )

Enable or disable the dead-time correction.
The default setting is disabled.

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>New state for the dead-time corrector</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_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  DllSDKExport SPC2Return SPC2_Set_Gate_Mode ( SPC2_H spc2, GateMode Mode )

Set the gate mode either continuous or pulsed.

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>New gate mode</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location

Examples:

SDK_Example.c.

4.3.2.10  DllSDKExport SPC2Return SPC2_Set_Gate_Values ( SPC2_H spc2, Int16 Shift, Int16 Length )

Change the Gate settings.
A gate signal is generated within the control electronics to select valid photons, i.e., only photons which arrive 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

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shift</strong></td>
<td>Phase shift of the gate signal in the ON state. The unit is percentage, i.e. 10 means 10% time-delay of a 20 ns periodic signal, which is equal to 2 ns. Accepted values: -50 ... +50</td>
</tr>
<tr>
<td><strong>Length</strong></td>
<td>Duration of the ON gate signal. The unit is percentage. Accepted values: 0 ... 100</td>
</tr>
</tbody>
</table>

### Returns

- **OK**
- **NULL_POINTER** The provided SPC2_H points to an empty memory location
- **OUT_OF_BOUND** Shift or length are outside the valid values

### Examples:

```
SDK_Example.c.
```

4.3.2.11 DllSDKExport SPC2Return SPC2_Set_Live_Mode_OFF ( SPC2_H spc2 )

Turn off the Live mode.

### Parameters

| spc2 | SPC2 handle |

### Returns

- **OK**
- **NULL_POINTER** The provided SPC2_H points to an empty memory location
- **INVALID_OP** The live mode is already inactive

### See Also

```
SPC2_Set_Live_Mode_ON()
```

### Examples:

```
SDK_Example.c.
```

4.3.2.12 DllSDKExport SPC2Return SPC2_Set_Live_Mode_ON ( SPC2_H spc2 )

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-lapses 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 `SPC2_Get_Snap()` or `SPC2_Get_Memory()` can be performed.

### Parameters

| spc2 | SPC2 handle |
Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location
INVALID_OP The live mode has been already started

See Also

SPC2_Set_Live_Mode_OFF()
SPC2_Get_Memory()
SPC2_Get_Snap()

Examples:

SDK_Example.c.

4.3.2.13 DllSDKExport SPC2Return SPC2_Set_Sync_In_State ( SPC2_H spc2, State s )

Set the sync-in state.
Set the camera to wait for an input trigger signal before starting an acquisition.

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>Enable or disable the synchronization input</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location

Examples:

SDK_Example.c.

4.3.2.14 DllSDKExport SPC2Return SPC2_Set_Trigger_Out_State ( SPC2_H spc2, TriggerMode Mode )

Select the output signal.

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>Pointer to the SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>New trigger mode</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location

Examples:

SDK_Example.c.
4.4 Get methods

Functions

- DllSDKExport SPC2Return SPC2_Get_Live_Img (SPC2_H spc2, UInt16 *Img)
- DllSDKExport SPC2Return SPC2_Prepare_Snap (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Get_Snap (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Get_Image_Buffer (SPC2_H spc2, BUFFER_H *buffer)
- DllSDKExport SPC2Return SPC2_Get_Img_Position (SPC2_H spc2, UInt16 *Img, UInt32 Position)
- DllSDKExport SPC2Return SPC2_Start_ContAcq (SPC2_H spc2, char filename[256])
- DllSDKExport SPC2Return SPC2_Get_Memory (SPC2_H spc2, double *total_bytes)
- DllSDKExport SPC2Return SPC2_Stop_ContAcq (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Get_DeadTime (SPC2_H spc2, UInt16 Val, UInt16 *ReturnVal)
- DllSDKExport SPC2Return SPC2_Get_GateWidth (SPC2_H spc2, Int16 Val, double *ReturnVal)
- DllSDKExport SPC2Return SPC2_Is16Bit (SPC2_H spc2, short *is16bit)
- DllSDKExport SPC2Return SPC2_IsTriggered (SPC2_H spc2, short *isTriggered)
- DllSDKExport SPC2Return SPC2_GetVersion (SPC2_H spc2, double *Firmware_Version, double *Software_Version)

4.4.1 Detailed Description

Functions to get status or data from SPC2 camera.

4.4.2 Function Documentation

4.4.2.1 DllSDKExport SPC2Return SPC2_Get_DeadTime (SPC2_H spc2, UInt16 Val, UInt16 *ReturnVal)

Get the calibrated dead-time value.

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

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Val</td>
<td>Desired dead-time value in ns. No error is generated when the value is above MAX_DEAD_TIME.</td>
</tr>
<tr>
<td>ReturnVal</td>
<td>Closest dead-time value possible. This parameter is referenced.</td>
</tr>
</tbody>
</table>

Returns

- OK
- NULL_POINTER The provided SPC2_H or ReturnVal point to an empty memory location

See Also

SPC2_Set_DeadTime()

Examples:

SDK_Example.c

4.4.2.2 DllSDKExport SPC2Return SPC2_Get_GateShift (SPC2_H spc2, Int16 Val, Int16 *ReturnVal)

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

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Val</td>
<td>Desired gate shift value in percentage of 20ns. No error is generated when the value out of range, instead the real boundaries are forced on ReturnVal.</td>
</tr>
<tr>
<td>ReturnVal</td>
<td>Closest gate-shift value possible. This parameter is referenced.</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H or ReturnVal point to an empty memory location

See Also

SPC2_Get_Gate_Values()

4.4.2.3 DllSDKExport SPC2Return SPC2_Get_GateWidth ( SPC2_H spc2, Int16 Val, double *ReturnVal )

Get the calibrated gate width value.

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

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Val</td>
<td>Desired gate-width value in percentage of 20ns. No error is generated when the value out of range, instead the real boundaries are forced on ReturnVal.</td>
</tr>
<tr>
<td>ReturnVal</td>
<td>Closest gate-width value possible. This parameter is referenced.</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H or ReturnVal point to an empty memory location

See Also

SPC2_Get_Gate_Values()

4.4.2.4 DllSDKExport SPC2Return SPC2_Get_Image_Buffer ( SPC2_H spc2, BUFFER_H *buffer )

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

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

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>buffer</td>
<td>Pointer to the buffer Handle in which the function will save reference to the camera image buffer</td>
</tr>
</tbody>
</table>
4.4 Get methods

Returns

OK
NULL_POINTER The provided SPC2_H or BUFFER_H point to an empty memory location

See Also

SPC2_Get_Snap()

4.4.2.5 DllSDKExport SPC2Return SPC2_Get_Img_Position ( SPC2_H spc2, UInt16 * Img, UInt32 Position )

Export an acquired image to an user allocated memory array.
Once a set of images have been acquired by SPC2_Get_Snap(), a single image can be exported from the SDK image buffer and saved in the memory (Img array).

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Img</td>
<td>Pointer to the output image array. The size of the array must be at least 2 KB.</td>
</tr>
<tr>
<td>Position</td>
<td>Index of the image to save. Accepted values: 1 ... Number of acquired images</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H or Img point to an empty memory location
INVALID_OP A smaller number of images were acquired. The index of the first image is the number 1.

See Also

SPC2_Get_Snap()

4.4.2.6 DllSDKExport SPC2Return SPC2_Get_Live_Img ( SPC2_H spc2, 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 SPC2_Set_Live_Mode_ON() function.

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Img</td>
<td>Pointer to the output image array. The size of the array must be at least 2 KB.</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H or Img point to an empty memory location
INVALID_OP The live-mode has not been started yet

See Also

SPC2_Set_Live_Mode_ON()

Examples:

SDK_Example.c.

Micro Photon Devices s.r.l.
4.4.2.7 DllSDKExport SPC2Return SPC2_Get_Memory ( SPC2_H spc2, double * total_bytes )

Dump the camera memory to the PC and save data to the file specified with the SPC2_Start_ContAcq() function. The acquisition is split into several progressively numbered 1GiB files if necessary. 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 30MB/s. Be sure to have enough disk space for your measurement.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spc2</td>
<td>SPC2 handle</td>
</tr>
<tr>
<td>total_bytes</td>
<td>Total number of bytes read. Value is referenced.</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_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 SPC2_SPC2_Start_ContAcq() before calling this function.
COMMUNICATION_ERROR Communication error during data download.
SPC2_MEMORY_FULL Camera internal memory got full during data download. Data loss occurred. Reduce frame-rate or optimize your software to reduce deadtime between subsequent calling of the function.

See Also

SPC2_SPC2_Start_ContAcq()
SPC2_SPC2_Stop_ContAcq()

Examples:

SDK_Example.c.

4.4.2.8 DllSDKExport SPC2Return SPC2_Get_Snap ( SPC2_H spc2 )

Get a selected number of images.

Acquire a set of images according to the parameters defined by SPC2_Set_Camera_Par(). This command works only when SPC2_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 SPC2_IsTriggered() function.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spc2</td>
<td>PC2 handle</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location
INVALID_OP Unable to acquire images when the live mode is ON. Use instead SPC2_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.
4.4 Get methods

See Also
- SPC2_Set_Camera_Par()
- SPC2_Prepare_Snap()
- SPC2_Set_Sync_In_State()
- SPC2_IsTriggered()

Examples:
- SDK_Example.c.

4.4.2.9 DllSDKExport SPC2Return SPC2_GetVersion ( SP2C_H spc2, double * Firmware_Version, double * Software_Version )

Get the SDK and camera firmware version.

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Firmware_Version</th>
<th>Version of the camera firmware in the format x.xx. This parameter is referenced.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software_Version</td>
<td>Version of the SDK in the format x.xx. This parameter is referenced.</td>
</tr>
</tbody>
</table>

Returns
- OK
- NULL_POINTER The provided SPC2_H points to an empty memory location

4.4.2.10 DllSDKExport SPC2Return SPC2_Is16Bit ( SP2C_H spc2, 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

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
</table>

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

Returns
- OK
- NULL_POINTER The provided SPC2_H points to an empty memory location

4.4.2.11 DllSDKExport SPC2Return SPC2_IsTriggered ( SP2C_H spc2, short * isTriggered )

Pool the camera for external trigger status.

Pool the camera in order to know if an external sync pulse was detected. The result is meaningful only if the camera was previously set to wait for an external sync.
### 4.4.2.12 DllSDKExport SPC2Return SPC2Prepare_Snap (SPC2_H spc2)

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 `SPC2_Set_Camera_Par()` function. 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 `SPC2_Get_Snap()` function.

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spc2</td>
<td>SPC2 handle</td>
</tr>
</tbody>
</table>

#### Returns

- **OK**
- **NULL_POINTER** The provided SPC2_H points to an empty memory location
- **INVALID_OP** Unable to acquire images when the live mode is ON. Use instead `SPC2_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

- `SPC2_Set_Camera_Par()`
- `SPC2_Get_Snap()`
- `SPC2_Set_Sync_In_State()`

#### Examples:

- `SDK_Example.c`

### 4.4.2.13 DllSDKExport SPC2Return SPC2Start_ContAcq (SPC2_H spc2, char filename[256])

Put the camera in "continuous acquisition" 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 `SPC2_Get_Memory()` function as soon as possible, in order to avoid data loss.

#### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>spc2</td>
<td>SPC2 handle</td>
</tr>
<tr>
<td>filename</td>
<td>Name of output file</td>
</tr>
</tbody>
</table>

---

Micro Photon Devices s.r.l.
Returns

OK
NULL_POINTER  The provided SPC2_H or BUFFER_H point to an empty memory location
UNABLE_CREATE_FILE  It was not possible to create the output file.

See Also

SPC2_Get_Memory()
SPC2_SPC2_Stop_ContAcq()

Examples:

SDK_Example.c.

4.4.2.14  DllSDKExport SPC2_Return SPC2_Stop_ContAcq ( SPC2_H spc2 )

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 function are
called.

Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
</table>

Returns

OK
NULL_POINTER  The provided SPC2_H or BUFFER_H point to an empty memory location
UNABLE_CREATE_FILE  It was not possible to access the output file.

See Also

SPC2_SPC2_Start_ContAcq()
SPC2_Get_Memory()

Examples:

SDK_Example.c.
4.5 Additional methods

Functions

- DllSDKExport SPC2Return SPC2_Save_Img_Disk (SPC2_H spc2, UInt32 Start_Img, UInt32 End_Img, char ∗filename, OutFileFormat mode)
- DllSDKExport SPC2Return SPC2_ReadSPC2FileFormatImage (char ∗filename, UInt32 ImgIdx, UInt16 ∗Img, UInt16 dim[])
- DllSDKExport SPC2Return SPC2_Average_Img (SPC2_H spc2, double ∗Img)
- DllSDKExport SPC2Return SPC2_StDev_Img (SPC2_H spc2, double ∗Img)
- DllSDKExport SPC2Return SPC2_Set_Correlation_Mode (SPC2_H spc2, CorrelationMode CM, int NCorrChannels, State s)
- DllSDKExport SPC2Return SPC2_Correlation_Img (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Save_Correlation_Img (SPC2_H spc2, char ∗filename)

4.5.1 Detailed Description

Additional utility functions.

4.5.2 Function Documentation

4.5.2.1 DllSDKExport SPC2Return SPC2_Average_Img (SPC2_H spc2, double ∗Img)

Calculate the average image.

Once a set of images have been acquired by SPC2_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

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Img</td>
<td>Pointer to the output double image array. The size of the array must be at least 8 kB.</td>
</tr>
</tbody>
</table>

Returns

OK
NULL_POINTER The provided SPC2_H points to an empty memory location
INVALID_OP No images were acquired

Examples:

SDK_Example.c.

4.5.2.2 DllSDKExport SPC2Return SPC2_Correlation_Img (SPC2_H spc2)

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 SPC2_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

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
</table>
Returns

OK
NULL_POINTER The provided SPC2_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
SPC2_Set_Correlation_Mode()
SPC2_Get_Snap()

Examples:
SDK_Example.c.

4.5.2.3 DllSDKExport SPC2_Return SPC2_ReadSPC2FileFormatImage ( char * filename, UInt32 ImgIdx, UInt16 * Img, UInt16 dim[] )

Read a spc2 image from file.
Read the image at the ImgIdx position in the given spc2 file from the hard disk.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the output file</td>
</tr>
<tr>
<td>ImgIdx</td>
<td>Image index in the file. 1 ... 65534</td>
</tr>
<tr>
<td>Img</td>
<td>Pointer to the output image array. The size of the array must be at least 2 kB.</td>
</tr>
<tr>
<td>dim</td>
<td>Array in which the dimensions of the image, retrieved from the file, is saved. dim[0]=number of rows, dim[1]=number of columns</td>
</tr>
</tbody>
</table>

Returns

OK
UNABLE_READ_FILE Unable to read the input file
NOT_EN_MEMORY Not enough memory to store the data contained in the file

Examples:
SDK_Example.c.

4.5.2.4 DllSDKExport SPC2_Return SPC2_Save_Correlation_Img ( SPC2_H spc2, char * filename )

Save the autocorrelation functions on the hard disk.
This function requires that SPC2_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:

<table>
<thead>
<tr>
<th>Byte offset</th>
<th>Type</th>
<th>Number of bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>int</td>
<td>4</td>
<td>Number of lag-times (NLag)</td>
</tr>
</tbody>
</table>

Micro Photon Devices s.r.l.
### Table

<table>
<thead>
<tr>
<th>int</th>
<th>4</th>
<th>Number of pixels. This value must be 1024 (NPix)</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>4</td>
<td>Selected algorithm: 0 Linear, 1 Multi-tau</td>
</tr>
<tr>
<td>double</td>
<td>8 + NLag</td>
<td>Autocorrelation values of the first pixel</td>
</tr>
<tr>
<td>double</td>
<td>8 + NLag</td>
<td>Autocorrelation values of the second pixel</td>
</tr>
<tr>
<td>double</td>
<td>8 + NLag</td>
<td>Autocorrelation values of the Nth pixel</td>
</tr>
<tr>
<td>double</td>
<td>8 + NLag</td>
<td>Autocorrelation values of the last pixel</td>
</tr>
<tr>
<td>double</td>
<td>8 + NLag</td>
<td>Lag times</td>
</tr>
</tbody>
</table>

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

```matlab
function data = Read_SPCC(fname)
    f = fopen(fname,'rb');
    buf = fread(f,3,'int32');
    data.NChannel = buf(1);
    data.NPixel = buf(2);
    data.IsMultiTau = (buf(3) == 1);
    data.CorrelationImage = reshape(fread(f,data.NPixel*data.NChannel,'float64'), ... 
                                data.NChannel,32,32);
    data.CorrelationImage = permute(data.CorrelationImage,[2 3 1]);
    data.t=fread(f, data.NChannel,'float64');
    fclose(f);
```

#### Parameters

<table>
<thead>
<tr>
<th>spc2</th>
<th>SPC2 handle</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>Name of the output file</td>
</tr>
</tbody>
</table>

#### Returns

- OK
- NULL_POINTER The provided SPC2_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

SPC2_Set_Correlation_Mode()

#### Examples:

SDK_Example.c.

### 4.5.2.5 DllSDKExport SPC2Return SPC2_Save_Img_Disk

```c
Save the selected images on the hard disk.
```

This function saves the acquired images on the hard disk. The images are organized in a simple binary format: the first byte contains the number of rows, the second byte contains the number of columns, the third byte contains the value 8 or 16 to define whether the array contains 8 bit or 16 bit unsigned int data. The following bytes contain the 8/16 bit pixel values in row-major order. The byte order is little-endian for the 16 bit images. The total number of bytes of the file is defined by:
The spcc binary file is organized as follows, assuming that N images were stored (NImg), each composed by Npixels (rows \* columns):

<table>
<thead>
<tr>
<th>Byte offset</th>
<th>Type</th>
<th>Number of bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>unsigned char</td>
<td>1</td>
<td>Number of rows</td>
</tr>
<tr>
<td>1</td>
<td>unsigned char</td>
<td>1</td>
<td>Number of columns</td>
</tr>
<tr>
<td>2</td>
<td>unsigned char</td>
<td>1</td>
<td>Bit depth of the images: 8 or 16 (BDepth)</td>
</tr>
<tr>
<td>3</td>
<td>(BDepth == 8) unsigned char (BDepth == 16) unsigned short</td>
<td>Npixel/8 + BDepth</td>
<td>First image</td>
</tr>
<tr>
<td>3 + Npixel/8 * BD</td>
<td>(BDepth == 8) unsigned char (BDepth == 16) unsigned short</td>
<td>Npixel/8 + BDepth</td>
<td>Second image</td>
</tr>
<tr>
<td>...</td>
<td>(BDepth == 8) unsigned char (BDepth == 16) unsigned short</td>
<td>Npixel/8 + BDepth</td>
<td>Nth image</td>
</tr>
<tr>
<td>3 + Npixel/8 * BD * (NImg-1)</td>
<td>(BDepth == 8) unsigned char (BDepth == 16) unsigned short</td>
<td>Npixel/8 + BDepth</td>
<td>Last image</td>
</tr>
</tbody>
</table>

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

```matlab
function data = Read_SPC2(fname, NImage)
    f = fopen(fname,'rb');
    rows = fread(f,1,'uint8');
    cols = fread(f,1,'uint8');
    depth = fread(f,1,'uint8');
    for i = 1: NImage
        if depth == 8
            format = 'uint8';
        else
            format = 'uint16';
        end
        data(:,:,i) = reshape(fread(f,1024,format), rows,col);
    end
    fclose(f);
end
```

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>spc2</code></td>
<td>SPC2 handle</td>
</tr>
<tr>
<td><code>Start_Img</code></td>
<td>Index of the first image to save. Accepted values: 1 ... Number of acquired images</td>
</tr>
<tr>
<td><code>End_Img</code></td>
<td>Index of the last image to save. Accepted values: <code>Start_Img</code> ... Number of acquired images</td>
</tr>
<tr>
<td><code>filename</code></td>
<td>Name of the output file.</td>
</tr>
<tr>
<td><code>mode</code></td>
<td>Select the file format of the output images</td>
</tr>
</tbody>
</table>

### Returns

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td></td>
</tr>
<tr>
<td>NULL_POINTER</td>
<td>The provided SPC2_H points to an empty memory location</td>
</tr>
<tr>
<td>INVALID_OP</td>
<td>No images were acquired or the selected range of images is not valid</td>
</tr>
<tr>
<td>UNABLE_CREATE_FILE</td>
<td>Unable to create the output file</td>
</tr>
</tbody>
</table>

### Examples:

`SDK_Example.c`
4.5.2.6 DllSDKExport SPC2Return SPC2_Set_Correlation_Mode ( SPC2_H spc2, CorrelationMode CM, int NCorrChannels, State s )

Enable the correlation mode.

This function must be called before invoking SPC2_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 SPC2_destr() function is called or by setting the State s equal to Disabled.

Parameters

| spc2 | SPC2 handle |
| CM   | Selected autocorrelation algorithm |
| NCorrChannels | Number of global lag channels. When the linear correlation algorithm is selected, the first N-Channel 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 $2^i \times$ Exposure time. |
| s    | Enable or Disable the correlation mode |

Returns

OK
NULL_POINTER The provided SPC2_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

SPC2_Correlation_Img()

Examples:

SDK_Example.c.

4.5.2.7 DllSDKExport SPC2Return SPC2_StDev_Img ( SPC2_H spc2, double * Img )

Calculate the standard deviation image.

Once a set of images have been acquired by SPC2_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

| spc2  | Pointer to the SPC2 handle |
| Img   | Pointer to the output double image array. The size of the array must be at least 8 KB. |
4.5 Additional methods

Returns

- OK
- NULL_POINTER The provided SPC2_H points to an empty memory location
- INVALID_OP No images were acquired

Examples:

SDK_Example.c.
4.6 MPD only - Calibration functions

Functions

- DllSDKExport SPC2Return SPC2_Calibrate_DeadTime (SPC2_H spc2)
- DllSDKExport SPC2Return SPC2_Calibrate_Gate (SPC2_H spc2)

4.6.1 Detailed Description

Functions for internal MPD use.

4.6.2 Function Documentation

4.6.2.1 DllSDKExport SPC2Return SPC2_Calibrate_DeadTime (SPC2_H spc2)

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

| spc2 | SPC2 handle |

Returns

| OK |
| NULL_POINTER The provided SPC2_H points to an empty memory location |
| UNABLE_CREATE_FILE Unable to create the output file |

4.6.2.2 DllSDKExport SPC2Return SPC2_Calibrate_Gate (SPC2_H spc2)

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

Parameters

| spc2 | SPC2 handle |

Returns

| OK |
| NULL_POINTER The provided SPC2_H points to an empty memory location |
| UNABLE_CREATE_FILE Unable to create the output file |
Chapter 5

File Documentation

5.1 SPC2_SDK.h File Reference

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

Macros

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

Typedefs

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

Enumerations

- `enum SPC2Return {
  OK = 0, USB_DEVICE_NOT_RECOGNIZED = -1, ELECTRONIC_INTERFACE_NOT_RECOGNIZED = -2, FAILED_FPGA_CONFIGURATION = -3, FPGA_USB_DRIVER_FAILURE = -4, COMMUNICATION_ERROR = -5, OUT_OF_BOUND = -6, MISSING_DLL = -7, EMPTY_BUFFER = -8, NOT_EN_MEMORY = -9, NULL_POINTER = -10, INVALID_OP = -11, UNABLE_CREATE_FILE = -12, UNABLE_READ_FILE = -13, FIRMWARE_NOT_COMPATIBLE = -14, USB_PORT_NOT_EN_POWER = -15, TOO_MUCH_LIGHT = -16, INVALID_NIMG_CORRELATION = -17, SPC2_MEMORY_FULL = -18
}
- `enum OutFileFormat { SPC2_FILEFORMAT = 0, TIFF_LZW_COMPRESSION = 1, TIFF_NO_COMPRESSION = 2 }
- `enum GateMode { Continuous = 0, Pulsed = 1 }`
• enum CameraMode { Normal = 0, Advanced = 1 }
• enum TriggerMode { None = 0, Gate_Clk = 1, Frame = 2 }
• enum State { Disabled = 0, Enabled = 1 }
• enum CorrelationMode { Linear = 0, MultiTau = 1 }

Functions

• DllSDKExport SPC2Return SPC2_Constr (SPC2_H *spc2_in, UInt16 minRow, UInt16 maxRow, UInt16 minCol, UlInt16 maxCol, CameraMode m, char *Device_ID)
• DllSDKExport SPC2Return SPC2_Destroy (SPC2_H spc2)
• DllSDKExport void PrintErrorCode (FILE *fout, const char *FunName, SPC2Return retcode)
• DllSDKExport SPC2Return SPC2_Set_Camera_Par (SPC2_H spc2, UInt16 Exposure, UInt32 NFrames, UInt16 NIntegFrames)
• DllSDKExport SPC2Return SPC2_Set_Camera_SubArray (SPC2_H spc2, UInt16 minRow, UInt16 maxRow, UInt16 minCol, UInt16 maxCol)
• DllSDKExport SPC2Return SPC2_Set_DeadTime (SPC2_H spc2, UInt16 Val)
• DllSDKExport SPC2Return SPC2_Set_DeadTime_Correction (SPC2_H spc2, State s)
• DllSDKExport SPC2Return SPC2_Set_Advanced_Mode (SPC2_H spc2, State s)
• DllSDKExport SPC2Return SPC2_Set_Background_Img (SPC2_H spc2, UInt16 *Img)
• DllSDKExport SPC2Return SPC2_Set_Background_Subtraction (SPC2_H spc2, State s)
• DllSDKExport SPC2Return SPC2_Set_Gate_Values (SPC2_H spc2, Int16 Shift, Int16 Length)
• DllSDKExport SPC2Return SPC2_Set_Gate_Mode (SPC2_H spc2, GateMode Mode)
• DllSDKExport SPC2Return SPC2_Set_Trigger_Out_State (SPC2_H spc2, TriggerMode Mode)
• DllSDKExport SPC2Return SPC2_Set_Sync_In_State (SPC2_H spc2, State s)
• DllSDKExport SPC2Return SPC2_Set_Live_Mode_ON (SPC2_H spc2)
• DllSDKExport SPC2Return SPC2_Set_Live_Mode_OFF (SPC2_H spc2)
• DllSDKExport SPC2Return SPC2_Apply_settings (SPC2_H spc2)
• DllSDKExport SPC2Return SPC2_Get_Live_Img (SPC2_H spc2, UInt16 *Img)
• DllSDKExport SPC2Return SPC2_Prepare_Snap (SPC2_H spc2)
• DllSDKExport SPC2Return SPC2_Get_Snap (SPC2_H spc2)
• DllSDKExport SPC2Return SPC2_Get_View (SPC2_H spc2)
• DllSDKExport SPC2Return SPC2_Get_Img_Position (SPC2_H spc2, UInt16 *Img, UInt32 Position)
• DllSDKExport SPC2Return SPC2_Start_ContAcq (SPC2_H spc2, char filename[256])
• DllSDKExport SPC2Return SPC2_Stop_ContAcq (SPC2_H spc2)
• DllSDKExport SPC2Return SPC2_Get_DeadTime (SPC2_H spc2, UInt16 Val, UInt16 *ReturnVal)
• DllSDKExport SPC2Return SPC2_Get_GateWidth (SPC2_H spc2, Int16 Val, double *ReturnVal)
• DllSDKExport SPC2Return SPC2_Get_GateShift (SPC2_H spc2, Int16 Val, Int16 *ReturnVal)
• DllSDKExport SPC2Return SPC2_Is16Bit (SPC2_H spc2, short *is16bit)
• DllSDKExport SPC2Return SPC2_IsTriggered (SPC2_H spc2, short *isTriggered)
• DllSDKExport SPC2Return SPC2_GetVersion (SPC2_H spc2, double *Firmware_Version, double *Software_Version)
• DllSDKExport SPC2Return SPC2_Save_Img_Disk (SPC2_H spc2, UInt32 Start_Img, UInt32 End_Img, char *filename, OutFileFormat mode)
• DllSDKExport SPC2Return SPC2_ReadSPC2FileFormatImage (char *filename, UInt32 ImgIdx, UInt16 *Img, UInt16 dim[])
• DllSDKExport SPC2Return SPC2_Average_Img (SPC2_H spc2, double *Img)
• DllSDKExport SPC2Return SPC2_StDev_Img (SPC2_H spc2, double *Img)
• DllSDKExport SPC2Return SPC2_Set_Correlation_Mode (SPC2_H spc2, CorrelationMode CM, int NCorrChannels, State s)
• DllSDKExport SPC2Return SPC2_Correlation_Img (SPC2_H spc2)
• DllSDKExport SPC2Return SPC2_Save_Correlation_Img (SPC2_H spc2, char *filename)
• DllSDKExport SPC2Return SPC2_Calibrate_DeadTime (SPC2_H spc2)
• DllSDKExport SPC2Return SPC2_Calibrate_Gate (SPC2_H spc2)
5.1.1 Detailed Description

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

5.1.2 Macro Definition Documentation

5.1.2.1 #define MAX_DEAD_TIME 600

Maximum allowed dead-time in nanoseconds.
The dead-time is set to MAX_DEAD_TIME, when higher values are requested.

Examples:

    SDK_Example.c.

5.1.2.2 #define MIN_DEAD_TIME 200

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

Examples:

    SDK_Example.c.
Chapter 6

Example Documentation

6.1 SDK_Example.c

/*
   ########################################################################
   Copyright 2012-2013 Micro-Photon-Devices s.r.l.
   SOFTWARE PRODUCT: SPC2_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 "SPC2_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
#define NIMG_DEADT 10
#define NIMG_CORR 100

//***************************************************************
// 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<1024;i++)
        hist[Img[i]]++;
}

// Main code
int main(void)
{ // variables definition //
    SPC2_H spc2= NULL;
    BUFFER_H buffer =NULL;
    UInt16* Img= NULL,hist[65535],AppliedDT=0;
    UInt16 dim[2];
    int minC=1;
    int maxC=32;
    int minR=1;
    int maxR=32;
    double read_bytes=0, total_bytes=0;
    int integFrames=10;
    int i=0,j=0,k=0,l=0,out=0;
    short trig=0;
    double gateoff=0;
    double *x = NULL,*y= NULL,*Imgd=NULL;
    double * corim;
    double* data= NULL;
    char c=0,*fname=NULL;
    FILE* f= NULL;
    time_t start,stop;
    char *Test[] = { "Live mode: write on stdout 10 images", //0
                    "Holdoff: mean number of photons at different holdoff values", //1
                    "Dead-time corrector improves the image quality", //2
                    "Background subtraction: 2 output files, with and without BG", //3
                    "Gate: 1000 images normal, 1000 images with gate 3 ns shift 5 ns", //4
                    "Synchronization output", //5
                    "Background statistics", //6
                    "Gate: width of the gate as a function of the offset", //7
                    "Read and write test images", //8
                    "Calculate the correlation image", //a
                    "Continuous acquisition and subarray selection" //b
    };

    Imgd =(double*) calloc(1, 1024* sizeof(double));
    Img =(UInt16*) calloc(1, 1024* sizeof(UInt16));
    data =(double*) calloc(1, 1024* sizeof(double));
    x =(double*) calloc(1, 65535* sizeof(double));
    y =(double*) calloc(1, 65535* sizeof(double));

    // Simple menu for test selection //
    printf("******************************************************\n");
    printf("SPC2 Test program\n");
    printf("******************************************************\n");

    do
    { c=getchar();
        while((c<48 || c>58) && c!='q' && c!='a' && c!='b')
            getchar();
        switch(c)
        { case 'a':

        case 'b':

        case 'q':

        default:
            break;
    } // End do
if(c>47 && c<59) {
    printf("******************************************************\n");    printf("%s\n",Test[c-48]);    printf("******************************************************\n");}
if(c=='a') {
    printf("******************************************************\n");    printf("%s\n",Test[10]);    printf("******************************************************\n");}
if(c=='b') {
    printf("******************************************************\n");    printf("%s\n",Test[11]);    printf("******************************************************\n");}
switch(c) {
case '0': //Test live mode
    //SPC2 constructor and parameter setting
    SPC2_Constr(&spc2,1,32,1,32, Normal,"");
    SPC2_Set_Camera_Par(spc2, 100, 30000,300);
    SPC2_Set_Trigger_Out_State(spc2,Frame);
    SPC2_Apply_settings(spc2);
    SPC2_Set_Live_Mode_ON(spc2);
    //Acquistion of 10 live images
    for(i=0;i<10;i++){
        printf("Image %d:\n",i);
        SPC2_Get_Live_Img(spc2, Img);
        for(j=0;j<32;j++)
            for(k=0;k<32;k++)
                printf("%d ",Img[32*j+k]);
        printf("\n\n");
    }
    //Live mode off
    SPC2_Set_Live_Mode_OFF(spc2);
    break;
case '1': //Test dead-time
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Advanced,"");
    SPC2_Set_Camera_Par(spc2, 512, NIMG_DEADT,1);
    SPC2_Apply_settings(spc2);
    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;
        SPC2_Set_DeadTime(spc2, i);
        SPC2_Apply_settings(spc2);
        SPC2_Get_DeadTime(spc2, i, &AppliedDT);
        x[k]=(double)AppliedDT;
        SPC2_Prepare_Snap(spc2);
        SPC2_Get_Snap(spc2);
        x[k]=mean_double(Imgd,1024);
        data[k]=mean_double(Imgd,1024);
        printf("%d ns, Applied %d ns, %f \n", i, AppliedDT, data[k]);
        fprintf(f,"%d %f \n",i,data[k]);
        k++;
    }
    //print summary
    printf("Dead-time calibration\n");
    for(i=0;i<j;i++)
        printf("%f %f \n",x[i],data[i]);
    fclose(f);
    break;
case '2': //Dead-time corrector effect
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Normal,"");
    SPC2_Set_Camera_Par(spc2, 4096, 1000,100);
    SPC2_Set_DeadTime(spc2, 250);
    //acquisition with DTC on

printf("Acquire the image using the dead-time correction\n");
SPC2_Set_DeadTime_Correction(spc2, Enabled);
SPC2_Apply_settings(spc2);
//Acquire the BG image first
printf("\n\nClose the camera shutter and press ENTER...\n");
getchar();
SPC2_Prepare_Snap(spc2);
SPC2_Get_Snap(spc2);
//Calculate the average image
SPC2_Average_Img(spc2, data);
for(i=0;i<1024;i++)
    if(data[i]<= 65535)
        Img[i] = (UInt16) floor(data[i]+0.5);
    else
        Img[i] = 65535; // Avoid overflow
SPC2_Set_Background_Img(spc2, Img);
SPC2_Set_Background_Subtraction(spc2, Enabled);
SPC2_Apply_settings(spc2);
//now acquire the image with shutter open
printf("\n\nOpen the camera shutter and press ENTER...\n");
getchar();
SPC2_Prepare_Snap(spc2);
SPC2_Get_Snap(spc2);
SPC2_Save_Img_Disk(spc2, 1, NIMG_CORR, "Im_DeadTimeCorrected.tif",
TIFF_LZW_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");
SPC2_Set_Background_Subtraction(spc2, Disabled);
SPC2_Set_DeadTime_Correction(spc2, Disabled);
SPC2_Apply_settings(spc2);
//Acquire the BG image first
printf("\n\nClose the camera shutter and press ENTER...\n");
getchar();
SPC2_Prepare_Snap(spc2);
SPC2_Get_Snap(spc2);
//Calculate the average image
SPC2_Average_Img(spc2, data);
for(i=0;i<1024;i++)
    if(data[i]<= 65535)
        Img[i] = (UInt16) floor(data[i]+0.5);
    else
        Img[i] = 65535; // Avoid overflow
SPC2_Set_Background_Img(spc2, Img);
//acquire image with background subtration off
printf("\n\nOpen the camera shutter and press ENTER ...\n");
getchar();
SPC2_Prepare_Snap(spc2);
SPC2_Get_Snap(spc2);
SPC2_Save_Img_Disk(spc2, 1, NIMG_CORR, "Im_DeadTimeReference.tif",
TIFF_LZW_COMPRESSION);
printf("The the dead-time corrected image was acquired and stored on the hard disk
successfully!\n");
case '3': //Test background subtraction
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Normal,"");
    SPC2_Set_Camera_Par(spc2, 4096, 1000,100);
    SPC2_Apply_settings(spc2);
    //acquire background image
    printf("\n\nClose the camera shutter and press ENTER...\n");
    getchar();
    SPC2_Prepare_Snap(spc2);
    SPC2_Get_Snap(spc2);
    SPC2_Save_Img_Disk(spc2, 1, 1, "Bg.tif",
TIFF_LZW_COMPRESSION);
    SPC2_Average_Img(spc2, data);
    for(i=0;i<1024;i++)
        if(data[i]<= 65535)
            Img[i] = (UInt16) floor(data[i]+0.5);
        else
            Img[i] = 65535; // Avoid overflow
    SPC2_Set_Background_Img(spc2, Img);
    //acquire image with background subtraction off
    printf("\n\nOpen the camera shutter and press ENTER ...\n");
    getchar();
    SPC2_Prepare_Snap(spc2);
    SPC2_Get_Snap(spc2);
    SPC2_Set_Background_Subtraction(spc2, Disabled);
    SPC2_Apply_settings(spc2);
    SPC2_Prepare_Snap(spc2);
    SPC2_Get_Snap(spc2);
    SPC2_Save_Img_Disk(spc2, 1, 1, "Normal.tif",
TIFF_LZW_COMPRESSION);
    //acquire image with background subtraction on
SPC2_Set_Background_Subtraction(spc2, Enabled);
SPC2_Prepare_Snap(spc2);
SPC2_Get_Snap(spc2);
SPC2_Save_Img_Disk(spc2, 1, 1, "BgSubt.tif", TIFF_LZW_COMPRESSION);
break;
case '4': // Test gate
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Advanced,"" );
    SPC2_Set_Camera_Par(spc2, 4096, 1000,100);
    SPC2_Set_DeadTime(spc2, 500);
    SPC2_Apply_settings(spc2);
    //Normal image
    printf("Acquiring the reference image ...
");
    SPC2_Prepare_Snap(spc2);
    SPC2_Get_Snap(spc2);
    printf("Save the reference image ...
");
    SPC2_Save_Img_Disk(spc2, 1, 1000, "GateNormal.tif",
                      TIFF_LZW_COMPRESSION);
    //gated image
    SPC2_Set_Gate_Mode(spc2, Pulsed);
    SPC2_Set_Gate_Values(spc2, 10, 15); //Shift -10% of 20 ns --> 120 ns,
                      Length 15% of 20 ns --> 3ns
    SPC2_Apply_settings(spc2);
    printf("Acquiring the gated image ...
");
    SPC2_Prepare_Snap(spc2);
    SPC2_Get_Snap(spc2);
    printf("Save the gated image ...
");
    SPC2_Save_Img_Disk(spc2, 1, 1000, "GatePulsed.tif",
                      TIFF_LZW_COMPRESSION);
    break;
case '5': // Test synchronization output
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Advanced,"" );
    SPC2_Set_Camera_Par(spc2, 8192, 0xFFFF,5);
    SPC2_Set_DeadTime(spc2, 500);
    //no output
    SPC2_Set_Trigger_Out_State(spc2,None);
    SPC2_Apply_settings(spc2);
    SPC2_Set_Live_Mode_ON(spc2);
    printf("No output ...
");
    printf("Press ENTER to continue\n");
    getchar();
    SPC2_Set_Trigger_Out_State(spc2,Gate_Clk);
    SPC2_Set_Camera_Par(spc2, 8192, 0xFFFF,5);
    SPC2_Apply_settings(spc2);
    SPC2_Set_Live_Mode_ON(spc2);
    printf("Gate synchronization signal ...
");
    printf("Press ENTER to continue\n");
    getchar();
    SPC2_Set_Trigger_Out_State(spc2,Frame);
    SPC2_Set_Camera_Par(spc2, 8192, 0xFFFF,5);
    SPC2_Apply_settings(spc2);
    SPC2_Set_Live_Mode_ON(spc2);
    printf("Frame synchronization signal ...
");
    printf("Press ENTER to continue\n");
    getchar();
    SPC2_Set_Trigger_Out_State(spc2,); //wait for the trigger input ...
    //trigger in enabled
    SPC2_Set_Sync_In_State(spc2, Enabled);
    SPC2_Set_Camera_Par(spc2, 4096, 10,2);
    SPC2_Apply_settings(spc2);
    SPC2_Prepare_Snap(spc2);
    while (trig!=1)
        SPC2_IsTriggered(spc2, &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.
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Advanced,"" );
    SPC2_Set_Camera_Par(spc2, 10240, 1000,100);
    f=fopen("BgHist.txt","w");
    printf("Close the camera shutter and press ENTER ...
");
    break;
getchar();
printf("Acquiring: ");
for(i=600;i>=300;i-=150) //different hold-off values
{
    SPC2_Set_DeadTime(spc2, i);
    SPC2_Apply_settings(spc2);
    SPC2_Prepare_Snap(spc2);
    SPC2_Average_Img(spc2, data);
    for(k=0;k<1024;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("%d ns ",i);
}
printf("\n");
fclose(f);
break;
case '7': // Calibrate gate
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Advanced,"");
    SPC2_Set_Camera_Par(spc2, 1024*1, 10000,2);
    SPC2_Set_DeadTime(spc2, 400);
    SPC2_Apply_settings(spc2);
    printf("Expose the SPC2 camera to a time-independent luminous signal\n(room light might oscillate at 50 or 60 Hz)\nPress ENTER to continue ...\n");
    getchar();
    if((f = fopen("GateValues.txt","w")) == NULL)
    {
        printf("Unable to open the output file.\n");
        break;
    }
    //photon counts without any gate
    SPC2_Set_Gate_Mode(spc2, Continuous);
    SPC2_Apply_settings(spc2);
    SPC2_Prepare_Snap(spc2);
    SPC2_Get_Snap(spc2);
    SPC2_Average_Img(spc2, data);
    gateoff = mean_double(data,1024);
    printf("Gate OFF counts: %.2f\n",gateoff);
    fprintf(f,"Gate OFF counts: %.2f\n",gateoff);
    SPC2_Set_Gate_Mode(spc2, Pulsed);
    printf("Acquiring:\n\nGate|t|Mean|t|Actual Gate|t|\n");
    //photon counts for gate width ranging from 0% to 100%
    for(i=0;i<=100;i+=1)
    {
        SPC2_Set_Gate_Values(spc2, 0, i);
        SPC2_Apply_settings(spc2);
        SPC2_Prepare_Snap(spc2);
        SPC2_Get_Snap(spc2);
        SPC2_Average_Img(spc2, data);
        y[i] = mean_double(data,1024);
        y[i+101] = y[i]/gateoff*100; //actual gate width calculated from photon counts
        x[i]=(double) i;
        printf("%3.0f %.2f %.2f \n",x[i],y[i],y[i+101]);
        fprintf(f,"%.0f %.2f %.2f \n",x[i],y[i],y[i+101]);
    }
    fclose(f);
    printf("\n");
    break;
case '8': // Constancy of the gate width
    //SPC2 constructor and parameter setting
    fname = (char*) calloc(256,sizeof(char));
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Advanced,"");
    SPC2_Set_Camera_Par(spc2, 1024*1, 1000,2);
    SPC2_Set_DeadTime(spc2, 400);
    SPC2_Set_Gate_Mode(spc2, Pulsed);
    SPC2_Apply_settings(spc2);
    printf("Expose the SPC2 camera to a time-independent luminous signal\n(room light might oscillate at 50 or 60 Hz)\nPress ENTER to continue ...\n");
    getchar();
    //setting different gate width and shift
    for(j=0;j<=100;j+=10)
    {
        sprintf(fname, "GateOffset_%d.txt",j);
        if((f = fopen(fname,"w")) == NULL)
        {
            printf("Unable to open the file %s.\n",fname);
            break;
        }
    }
printf("\n\nGate length:%d \n",j);
printf("Shift t\tMean t\tStDev\t\n");
for(i=-50;i<=+50;i+=10)
{
    SPC2_Set_Gate_Values(spc2, i, j);
    SPC2_Apply_settings(spc2);
    SPC2_Prepare_Snap(spc2);
    SPC2_Get_Snap(spc2);
    SPC2_Average_Img(spc2, data);
y[i+500] = mean_double(data,1024);
    SPC2_StDev_Img(spc2, data);
y[i+500+101] = mean_double(data,1024);
    x[i+500]=(double) i;
    printf("%.3f %.4f %.4f\n",x[i+500],y[i+500],y[i+101+500]);
    fprintf(f,"%.0f %.4f %.4f\n",x[i+500],y[i+500],y[i+101+500]);
}
fclose(f);
}
printf("\n");
free(fname);
break;
case '9': // Save and Read images
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Advanced,"");
    SPC2_Set_Camera_Par(spc2, 1024, 20,2);
    SPC2_Set_DeadTime(spc2, 400);
    SPC2_Apply_settings(spc2);
    //acquiring and saving images
    printf("Acquiring 20 images and save them on the hard drive in the spc2 file format.\n");
    SPC2_Prepare_Snap(spc2);
    SPC2_Get_Snap(spc2);
    SPC2_Save_Img_Disk(spc2, 1,20,"Test20_Im.spc2", SPC2_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\n(Press ENTER to continue)\n");
getchar();
    SPC2_ReadSPC2FileFormatImage("Test20_Im.spc2", 1,Img, dim );
    printf("Rows: %d, Columns: %d\n",dim[0], dim[1]);
    for(i=1;i<=20;i++)
    {
        SPC2_ReadSPC2FileFormatImage("Test20_Im.spc2", i,Img,
        dim);
        printf("Image %hu, pixel value = %hu\n",i, Img[0]);
    }
    break;
case 'a': // Correlation image
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Normal,"");
    SPC2_Set_Camera_Par(spc2, 10500, 1024*64, 1);
    SPC2_Set_DeadTime(spc2, 300);
    SPC2_Apply_settings(spc2);
    //acquire images
    printf("Acquiring the images...\n");
    SPC2_Prepare_Snap(spc2);
    SPC2_Get_Snap(spc2);
    //computer correlation
    SPC2_Set_Correlation_Mode(spc2, MultiTau, 8, Enabled);
    printf("Starting the multi-tau autocorrelation algorithm\n");
    start = clock();
    SPC2_Correlation_Img(spc2);
    stop = clock();
    printf("The correlation has terminated in %.3f s\n", (stop-start)/(float)CLOCKS_PER_SEC);
    SPC2_Save_Correlation_Img(spc2,"Correlation_MultiTau.spcc");
    break;
case 'b': // Continuous acquisition, subarray selection, number of integration frames selection.
    //SPC2 constructor and parameter setting
    out=(int)SPC2_Constr(&spc2,1,32,1,32, Normal,"");
    printf("Input the number of frames of 20.74us to be integrated (suggested > 10 to avoid
data loss):\n");
    scanf("%d",integFrames);
    printf("Total integration time: %.2fus\n", (float)(20.74+integFrames));
    SPC2_Set_Camera_Par(spc2, 1050, 1, integFrames);
    printf("Input minimum row index:\n");
    scanf("%d",&minR);
    printf("Input maximum row index:\n");
    scanf("%d",&maxR);
    printf("Input minimum column index:\n");
    scanf("%d",&minC);
    printf("Input maximum column index:\n");
    scanf("%d",&maxC);
    printf("Actual array size: %d x %d\n", maxR-minR+1,maxC-minC+1);
SPC2_Set_Camera_SubArray(spc2, minR, maxR, minC, maxC);
SPC2_Set_DeadTime(spc2, 300);
SPC2_Apply_settings(spc2);

//acquire images
printf("Continuous acquisition will be started and 10 memory dumps performed.\n");
printf("Press ENTER to start continuous acquisition...\n");
getchar();
getchar();
SPC2_Start_ContAcq(spc2, "contacq");
for(i=1; i<11; i++)
{
    if (SPC2_Get_Memory(spc2, &read_bytes)==OK)
    {
        total_bytes=total_bytes+read_bytes;
        printf("Acquired %d bytes in %d readout operation\n", (int)(total_bytes), i);
    }
    else
        break;
}
SPC2_Stop_ContAcq(spc2);
printf("Acquisition saved to contacq.spc2.\n");
break;
default:
    break;
}

// Destructors

if(spc2)
    SPC2_Destr(spc2);
free(Img);
free(Imgd);
free(data);
free(y);
free(x);
printf("Press ENTER to continue\n");
getchar();
return 0;
Index

Additional methods, 30
  SPC2_Average_Img, 30
  SPC2_Correlation_Img, 30
  SPC2_ReadSPC2FileFormatImage, 31
  SPC2_Save_Correlation_Img, 31
  SPC2_Save_Img_Disk, 32
  SPC2_Set_Correlation_Mode, 33
  SPC2_StDev_Img, 34

Advanced
  SPC2-SDK custom Types, 12

BUFFER_H
  SPC2-SDK custom Types, 11

COMMUNICATION_ERROR
  SPC2-SDK custom Types, 13

CameraMode
  SPC2-SDK custom Types, 12

Construct, destructor and error handling, 15
  PrintErrorCode, 15
  SPC2_Constr, 15
  SPC2_Destr, 16

Continuous
  SPC2-SDK custom Types, 12

CorrelationMode
  SPC2-SDK custom Types, 12

ELECTRONIC_INTERFACE_NOT_RECOGNIZED
  SPC2-SDK custom Types, 13

EMPTY_BUFFER
  SPC2-SDK custom Types, 13

FAILED_FPGA_CONFIGURATION
  SPC2-SDK custom Types, 13

FIRMWARE_NOT_COMPATIBLE
  SPC2-SDK custom Types, 13

FPGA_USB_DRIVER_FAILURE
  SPC2-SDK custom Types, 13

Frame
  SPC2-SDK custom Types, 14

Gate_Clk
  SPC2-SDK custom Types, 14

GateMode
  SPC2-SDK custom Types, 12

Get methods, 23
  SPC2_Get_DeadTime, 23
  SPC2_Get_GateShift, 23
  SPC2_Get_GateWidth, 24

SPC2_Get_Image_Buffer, 24
SPC2_Get_Img_Position, 25
SPC2_Get_Live_Img, 25
SPC2_Get_Memory, 25
SPC2_Get_Snap, 25
SPC2_GetVersion, 27
SPC2_Is16Bit, 27
SPC2_IsTriggered, 27
SPC2_Prepare_Snap, 28
SPC2_Start_ContAcq, 28
SPC2_Stop_ContAcq, 29

INVALID_NIMG_CORRELATION
  SPC2-SDK custom Types, 13

INVALID_OP
  SPC2-SDK custom Types, 13

Linear
  SPC2-SDK custom Types, 12

MISSING_DLL
  SPC2-SDK custom Types, 13

MAX_DEAD_TIME
  SPC2_SDK.h, 39

MIN_DEAD_TIME
  SPC2_SDK.h, 39

MPD only - Calibration functions, 36
  SPC2_Calibrate_DeadTime, 36
  SPC2_Calibrate_Gate, 36

MultiTau
  SPC2-SDK custom Types, 12

NOT_EN_MEMORY
  SPC2-SDK custom Types, 13

NULL_POINTER
  SPC2-SDK custom Types, 13

None
  SPC2-SDK custom Types, 14

Normal
  SPC2-SDK custom Types, 12

OK
  SPC2-SDK custom Types, 13

OUT_OF_BOUND
  SPC2-SDK custom Types, 13

OutFileFormat
  SPC2-SDK custom Types, 12

PrintErrorCode
Constructr, destructor and error handling, 15
Pulsed
SPC2-SDK custom Types, 12
SPC2-SDK custom Types
Advanced, 12
COMMUNICATION_ERROR, 13
Continuous, 12
ELECTRONIC_INTERFACE_NOT_RECOGNIZED, 13
EMPTY_BUFFER, 13
FAILED_FPGA_CONFIGURATION, 13
FIRMWARE_NOT_COMPATIBLE, 13
FPGA_USB_DRIVER_FAILURE, 13
Frame, 14
Gate_Clk, 14
INVALID_NIMG_CORRELATION, 13
INVALID_OP, 13
Linear, 12
MISSING_DLL, 13
MultiTau, 12
NOT_EN_MEMORY, 13
NULL_POINTER, 13
None, 14
Normal, 12
OK, 13
OUT_OF_BOUND, 13
Pulsed, 12
SPC2_FILEFORMAT, 13
SPC2_MEMORY_FULL, 13
TIFF_LZW_COMPRESSION, 13
TIFF_NO_COMPRESSION, 13
TOO_MUCH_LIGHT, 13
UNABLE_CREATE_FILE, 13
UNABLE_READ_FILE, 13
USB_DEVICE_NOT_RECOGNIZED, 13
USB_PORT_NOT_EN_POWER, 13
SPC2_FILEFORMAT
SPC2-SDK custom Types, 13
SPC2_MEMORY_FULL
SPC2-SDK custom Types, 13
SPC2-SDK custom Types, 11
BUFFER_H, 11
CameraMode, 12
CorrelationMode, 12
GateMode, 12
OutFileFormat, 12
SPC2_H, 11
SPC2_Return, 13
TriggerMode, 13
SPC2_Apply_settings
Set methods, 17
SPC2_Average_Img
Additional methods, 30
SPC2_Calibrate_DeadTime
MPD only - Calibration functions, 36
SPC2_Calibrate_Gate
MPD only - Calibration functions, 36
SPC2_Construct
Constructr, destructor and error handling, 15
SPC2_Correlation_Img
Additional methods, 30
SPC2_Destr
Constructr, destructor and error handling, 16
SPC2_Get_DeadTime
Get methods, 23
SPC2_Get_GateShift
Get methods, 23
SPC2_Get_GateWidth
Get methods, 24
SPC2_Get_Image_Buffer
Get methods, 24
SPC2_Get_Img_Position
Get methods, 25
SPC2_Get_Live_Img
Get methods, 25
SPC2_Get_Memory
Get methods, 25
SPC2_Get_Next
Get methods, 26
SPC2_GetVersion
Get methods, 27
SPC2_H
SPC2-SDK custom Types, 11
SPC2_Is16Bit
Get methods, 27
SPC2_IsTriggered
Get methods, 27
SPC2_Prepare_Snap
Get methods, 28
SPC2_ReadSPC2FileFormatImage
Additional methods, 31
SPC2_SDK.h, 37
SPC2_Save_Correlation_Img
Additional methods, 31
SPC2_Save_Img_Disk
Additional methods, 32
SPC2_Set_Advanced_Mode
Set methods, 17
SPC2_Set_Background_Img
Set methods, 18
SPC2_Set_Background_Subtraction
Set methods, 18
SPC2_Set_Camera_Par
Set methods, 18
SPC2_Set_Camera_SubArray
Set methods, 19
SPC2_Set_Correlation_Mode
Additional methods, 33
SPC2_Set_DeadTime
Set methods, 19
SPC2_Set_DeadTime_Correction
Set methods, 20
SPC2_Set_Gate_Mode
Set methods, 20
SPC2_Set_Gate_Values
Set methods, 20
Micro Photon Devices s.r.l.
SPC2_Set_Live_Mode_OFF
Set methods, 21

SPC2_Set_Live_Mode_ON
Set methods, 21

SPC2_Set.Sync_In_State
Set methods, 22

SPC2_Set_Trigger_Out_State
Set methods, 22

SPC2_Set_Sync.In_State
Set methods, 22

SPC2_Set_Sync_In_State
Set methods, 22

SPC2_Set_Trigger_Out_State
Set methods, 22

SPC2_StDev_Img
Additional methods, 34

SPC2_Set_Advanced_Mode
Set methods, 17

SPC2_Set_Background_Img
Set methods, 17

SPC2_Set_Background_Subtraction
Set methods, 17

SPC2_Set_Camera_Par
Set methods, 17

SPC2_Set_Camera_SubArray
Set methods, 17

SPC2_Set_DeadTime
Set methods, 17

SPC2_Set_DeadTime_Correction
Set methods, 17

SPC2_Set_Gate_Mode
Set methods, 17

SPC2_Set_Gate_Values
Set methods, 17

SPC2_Set_Live_Mode_OFF
Set methods, 17

SPC2_Set_Live_Mode_ON
Set methods, 17

SPC2_Set_Sync_In_State
Set methods, 17

SPC2_Set_Trigger_Out_State
Set methods, 17

SPC2_Return
SPC2-SDK custom Types, 13

Set methods, 17

SPC2_Apply_settings
Set methods, 17

SPC2_Set_Advanced_Mode
Set methods, 17

SPC2_Set_Background_Img
Set methods, 17

SPC2_Set_Background_Subtraction
Set methods, 17

SPC2_Set_Camera_Par
Set methods, 17

SPC2_Set_Camera_SubArray
Set methods, 17

SPC2_Set_DeadTime
Set methods, 17

SPC2_Set_DeadTime_Correction
Set methods, 17

SPC2_Set_Gate_Mode
Set methods, 17

SPC2_Set_Gate_Values
Set methods, 17

SPC2_Set_Live_Mode_OFF
Set methods, 17

SPC2_Set_Live_Mode_ON
Set methods, 17

SPC2_Set_Sync_In_State
Set methods, 17

SPC2_Set_Trigger_Out_State
Set methods, 17

SPC2-SDK custom Types, 13

TIFF_LZW_COMPRESSION
SPC2-SDK custom Types, 13

TIFF_NO_COMPRESSION
SPC2-SDK custom Types, 13

TOO_MUCH_LIGHT
SPC2-SDK custom Types, 13

TriggerMode
SPC2-SDK custom Types, 13

UNABLE_CREATE_FILE
SPC2-SDK custom Types, 13

UNABLE_READ_FILE
SPC2-SDK custom Types, 13

USB_DEVICE_NOT_RECOGNIZED
SPC2-SDK custom Types, 13

USB_PORT_NOT_EN_POWER
SPC2-SDK custom Types, 13