



# **Hermes**

**Single Photon Counting Camera**

**Version 1.01A**

## **Software Development Kit Manual**

June, 2023

# Contents

<b>1 Hermes Software Development Kit (Hermes-SDK).</b>	<b>1</b>
<b>2 Module Index</b>	<b>2</b>
2.1 Modules	2
<b>3 File Index</b>	<b>3</b>
3.1 File List	3
<b>4 Module Documentation</b>	<b>4</b>
4.1 Hermes-SDK custom Types	4
4.1.1 Detailed Description	4
4.1.2 Enumeration Type Documentation	4
4.1.2.1 CameraMode	4
4.1.2.2 CorrelationMode	5
4.1.2.3 GateMode	5
4.1.2.4 HermesReturn	6
4.1.2.5 OutFileFormat	6
4.1.2.6 State	7
4.1.2.7 TriggerMode	7
4.2 Constructor, destructor and error handling	8
4.2.1 Detailed Description	8
4.2.2 Function Documentation	8
4.2.2.1 HermesConstr()	8
4.2.2.2 HermesDestr()	9
4.2.2.3 PrintErrorCode()	9
4.3 Set methods	9
4.3.1 Detailed Description	10
4.3.2 Function Documentation	10
4.3.2.1 HermesApplySettings()	10
4.3.2.2 HermesSetAdvancedMode()	11
4.3.2.3 HermesSetBackgroundImg()	11
4.3.2.4 HermesSetBackgroundSubtraction()	12
4.3.2.5 HermesSetCameraPar()	12
4.3.2.6 HermesSetCameraParSubArray()	14
4.3.2.7 HermesSetCoarseGateValues()	15
4.3.2.8 HermesSetDeadTime()	16
4.3.2.9 HermesSetDeadTimeCorrection()	16
4.3.2.10 HermesSetDualGate()	17
4.3.2.11 HermesSetFlimPar()	18
4.3.2.12 HermesSetFlimState()	18
4.3.2.13 HermesSetGateMode()	20
4.3.2.14 HermesSetGateValues()	21
4.3.2.15 HermesSetSyncInState()	21
4.3.2.16 HermesSetTriggerOutState()	22
4.3.2.17 HermesSetTripleGate()	22
4.4 Get methods	23

4.4.1 Detailed Description	23
4.4.2 Function Documentation	24
4.4.2.1 HermesDeviceInfo()	24
4.4.2.2 HermesGetDeadTime()	24
4.4.2.3 HermesGetGateShift()	25
4.4.2.4 HermesGetGateWidth()	26
4.4.2.5 HermesGetSerial()	26
4.4.2.6 HermesGetVersion()	27
4.4.2.7 HermesIs16Bit()	27
4.4.2.8 HermesIsTriggered()	28
4.5 Acquisition methods	28
4.5.1 Detailed Description	28
4.5.2 Function Documentation	29
4.5.2.1 HermesContAcqToFileGetMemory()	29
4.5.2.2 HermesContAcqToFileStart()	30
4.5.2.3 HermesContAcqToFileStop()	30
4.5.2.4 HermesContAcqToMemoryGetBuffer()	31
4.5.2.5 HermesContAcqToMemoryStart()	32
4.5.2.6 HermesContAcqToMemoryStop()	32
4.5.2.7 HermesLiveGetImg()	33
4.5.2.8 HermesLiveSetModeOFF()	34
4.5.2.9 HermesLiveSetModeON()	34
4.5.2.10 HermesSnapAcquire()	35
4.5.2.11 HermesSnapGetImageBuffer()	36
4.5.2.12 HermesSnapGetImgPosition()	36
4.5.2.13 HermesSnapPrepare()	37
4.6 Additional methods	38
4.6.1 Detailed Description	38
4.6.2 Function Documentation	38
4.6.2.1 HermesAverageImg()	38
4.6.2.2 HermesCorrelationImg()	39
4.6.2.3 HermesReadHermesFileFormatImage()	40
4.6.2.4 HermesResetOverilluminationProtection()	40
4.6.2.5 HermesSaveAveragedImgDisk()	41
4.6.2.6 HermesSaveCorrelationImg()	42
4.6.2.7 HermesSaveFlimDisk()	43
4.6.2.8 HermesSaveImgDisk()	43
4.6.2.9 HermesSetCorrelationMode()	46
4.6.2.10 HermesStDevImg()	46
<b>5 File Documentation</b>	<b>48</b>
5.1 Hermes_SDK.h File Reference	48
5.1.1 Detailed Description	50
5.1.2 Macro Definition Documentation	50
5.1.2.1 MAX_DEAD_TIME	50
5.1.2.2 MIN_DEAD_TIME	50
5.2 Hermes_SDK.h	51
<b>6 Example Documentation</b>	<b>55</b>
6.1 SDK_Example.c	55
<b>Index</b>	<b>63</b>

## Chapter 1

# Hermes Software Development Kit (Hermes-SDK).

Hermes 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:

Hermes_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:

Hermes-SDK custom Types . . . . .	4
Constructor, destructor and error handling . . . . .	8
Set methods . . . . .	9
Get methods . . . . .	23
Acquisition methods . . . . .	28
Additional methods . . . . .	38

## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">Hermes_SDK.h</a>	
Hermes software development kit . . . . .	48

## Chapter 4

# Module Documentation

### 4.1 Hermes-SDK custom Types

#### Typedefs

- typedef struct \_Hermes\_H \* **Hermes\_H**
- typedef unsigned char \* **BUFFER\_H**

#### Enumerations

- enum [HermesReturn](#) {  
    [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 ,  
    [HERMES\\_MEMORY\\_FULL](#) = -18 , [PERSISTING\\_TOO\\_MUCH\\_LIGHT](#) = -19 }  
• enum [OutFileFormat](#) { [HERMES\\_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 Enumeration Type Documentation

##### 4.1.2.1 CameraMode

enum [CameraMode](#)

Hermes working mode.

The camera contains for each pixel 8-bit counters. If the exposure time is too long, the counters 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. <b>WARNING:</b> the counters can overflow.

## 4.1.2.2 CorrelationMode

```
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.

## 4.1.2.3 GateMode

```
enum GateMode
```

Gate setting.

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

## Enumerator

Continuous	The gate signal is always "ON".
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.



#### 4.1.2.4 HermesReturn

enum `HermesReturn`

Error table.

Error code returned by the Hermes 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 Hermes 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 current operation mode may be incompatible with the called function.
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 power supply or reset the protection by software.
INVALID_NIMG_CORRELATION	The acquired number of images is not sufficient to calculate the required correlation function.
HERMES_MEMORY_FULL	The Hermes internal memory got full during continuous acquisition. Possible data loss.
PERSISTING_TOO_MUCH_LIGHT	The overillumination protection circuit has been reset by the user 3 times. It is now necessary to disconnect the camera from the power supply, after having carefully checked the illumination level!

#### 4.1.2.5 OutFileFormat

enum `OutFileFormat`

Output file format.

Table of the available output file formats for the saved images

## Enumerator

HERMES_FILEFORMAT	Hermes custom file format. See  See also  <a href="#">HermesSaveImgDisk()</a> for more details.
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: <a href="http://www.openmicroscopy.org/site/support/ome-model/ome-tiff/">http://www.openmicroscopy.org/site/support/ome-model/ome-tiff/</a> <b>WARNING</b> the creation of TIFF files might require longer execution times.

## 4.1.2.6 State

enum [State](#)

Disable or enable a specific function.

## Enumerator

Disabled	The function is disabled.
Enabled	The function is enabled.

## 4.1.2.7 TriggerMode

enum [TriggerMode](#)

Type of synchronization output.

The SYNC 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 Constructor, destructor and error handling

### Functions

- [HermesReturn HermesConstr](#) ([Hermes\\_H](#) \*Hermes\_in, [CameraMode](#) m, char \*Device\_ID)
- [HermesReturn HermesDestr](#) ([Hermes\\_H](#) Hermes)
- void [PrintErrorCode](#) (FILE \*fout, const char \*FunName, [HermesReturn](#) retcode)

### 4.2.1 Detailed Description

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

### 4.2.2 Function Documentation

#### 4.2.2.1 HermesConstr()

```
HermesReturn HermesConstr (
    Hermes_H * Hermes_in,
    CameraMode m,
    char * Device_ID )
```

Constructor.

It allocates a memory block to contain all the information and buffers required by the Hermes. 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>Hermes_in</i>	Pointer to Hermes handle
<i>m</i>	Camera Working mode
<i>Device_ID</i>	Unique ID to identify the connected device

#### Returns

OK

INVALID\_OP The Hermes\_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](#).

#### 4.2.2.2 HermesDestr()

```
HermesReturn HermesDestr (
    Hermes_H Hermes )
```

Destructor.

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

##### Parameters

<i>Hermes</i>	Hermes handle
---------------	---------------

##### Returns

OK

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

##### Examples

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

#### 4.2.2.3 PrintErrorCode()

```
void PrintErrorCode (
    FILE * fout,
    const char * FunName,
    HermesReturn 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

## 4.3 Set methods

### Functions

- [HermesReturn HermesSetCameraPar](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Exposure, [UInt32](#) NFrames, [UInt16](#) NIntegFrames, [UInt16](#) NCounters, [State](#) Force8bit, [State](#) Half\_array, [State](#) Signed\_data)

- [HermesReturn HermesSetCameraParSubArray](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Exposure, [UInt32](#) NFrames, [UInt16](#) NIntegFrames, [State](#) Force8bit, [UInt16](#) Npixels)
- [HermesReturn HermesSetDeadTime](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Val)
- [HermesReturn HermesSetDeadTimeCorrection](#) ([Hermes\\_H](#) Hermes, [State](#) s)
- [HermesReturn HermesSetAdvancedMode](#) ([Hermes\\_H](#) Hermes, [State](#) s)
- [HermesReturn HermesSetBackgroundImg](#) ([Hermes\\_H](#) Hermes, [UInt16](#) \*Img)
- [HermesReturn HermesSetBackgroundSubtraction](#) ([Hermes\\_H](#) Hermes, [State](#) s)
- [HermesReturn HermesSetGateMode](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter, [GateMode](#) Mode)
- [HermesReturn HermesSetGateValues](#) ([Hermes\\_H](#) Hermes, [Int16](#) Shift, [Int16](#) Length)
- [HermesReturn HermesSetDualGate](#) ([Hermes\\_H](#) Hermes, [State](#) DualGate\_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int Gap)
- [HermesReturn HermesSetTripleGate](#) ([Hermes\\_H](#) Hermes, [State](#) TripleGate\_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int ThirdGateWidth, int Gap1, int Gap2)
- [HermesReturn HermesSetCoarseGateValues](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Counter, [UInt16](#) Start, [UInt16](#) Stop)
- [HermesReturn HermesSetTriggerOutState](#) ([Hermes\\_H](#) Hermes, [TriggerMode](#) Mode)
- [HermesReturn HermesSetSyncInState](#) ([Hermes\\_H](#) Hermes, [State](#) s, int frames)
- [HermesReturn HermesSetFlimPar](#) ([Hermes\\_H](#) Hermes, [UInt16](#) FLIM\_steps, [UInt16](#) FLIM\_shift, [Int16](#) FLIM\_start, [UInt16](#) Length, int \*FLIM\_frame\_time)
- [HermesReturn HermesSetFlimState](#) ([Hermes\\_H](#) Hermes, [State](#) FLIM\_State)
- [HermesReturn HermesApplySettings](#) ([Hermes\\_H](#) Hermes)

### 4.3.1 Detailed Description

Functions to set parameters of the Hermes camera.

### 4.3.2 Function Documentation

#### 4.3.2.1 HermesApplySettings()

```
HermesReturn HermesApplySettings (
    Hermes_H Hermes )
```

Apply settings to the camera.

This function must be called after any Set function, except [HermesLiveSetModeON\(\)](#) and [HermesLiveSetModeOFF\(\)](#), 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>Hermes</i>	Hermes handle
---------------	---------------

#### Returns

OK

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

## Examples

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

## 4.3.2.2 HermesSetAdvancedMode()

```
HermesReturn HermesSetAdvancedMode (
    Hermes_H Hermes,
    State s )
```

Change the operating mode.

Set the operating mode to Normal or Advanced. Normal mode is the default setting. Before starting a new acquisition check that the relevant parameters for the selected modality are correctly set.

## Parameters

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

## Returns

OK

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

## Examples

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

## 4.3.2.3 HermesSetBackgroundImg()

```
HermesReturn HermesSetBackgroundImg (
    Hermes_H Hermes,
    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>Hermes</i>	Hermes 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 Hermes\_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 HermesSetBackgroundSubtraction()

```
HermesReturn HermesSetBackgroundSubtraction (
    Hermes_H Hermes,
    State s )
```

Enable or disable the hardware background subtraction.

## Parameters

<i>Hermes</i>	Hermes handle
<i>s</i>	Enable or disable the background subtraction

## Returns

OK

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

## Examples

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

#### 4.3.2.5 HermesSetCameraPar()

```
HermesReturn HermesSetCameraPar (
    Hermes_H Hermes,
    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 can be 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 period. For example, the value of 10 means 100 ns exposure.



## Parameters

<i>Hermes</i>	Hermes handle.
<i>Exposure</i>	Exposure time for a single frame (Hardware Integration Time - HIT). 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 truncated. 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 8-bit integer part and 1 bit sign.

## Returns

OK

NULL\_POINTER The provided Hermes\_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 HermesSetCameraParSubArray()

```

HermesReturn HermesSetCameraParSubArray (
    Hermes_H Hermes,
    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 [HermesSetCameraPar\(\)](#) 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 rows of 32 pixels toward the center of the array
- in live mode it is possible to acquire only 1, 2, 4, 8 full rows. 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 semi-array can be acquired (in any case starting from the corner and then reading by rows, i.e. if 67 pixel are required 2 rows of 32 pixels + 3 pixels from the third row will be acquired), PROVIDED that the total data acquired is an integer multiple of 1024 bytes. For the example above and assuming the camera is set to 8-bit/pixel, it 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.

- only counter number 1 is available.

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 can be 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 period. For example, the value of 10 means 100 ns exposure.

#### Parameters

<i>Hermes</i>	Hermes handle.
<i>Exposure</i>	Exposure time for a single frame (Hardware Integration Time - HIT). 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 truncated. 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 Hermes\_H points to an empty memory location

OUT\_OF\_BOUND Exposure, NFrames, NIntegFrames or NPixels out of bound.

#### 4.3.2.7 HermesSetCoarseGateValues()

```
HermesReturn HermesSetCoarseGateValues (
    Hermes_H Hermes,
    UInt16 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>Hermes</i>	Hermes handle
<i>Counter</i>	Counter to which settings refer. Accepted values: 1..3
<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="#">HermesSetCameraPar()</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="#">HermesSetCameraPar()</a> .

**Returns**

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

**See also**

[HermesSetCameraPar\(\)](#)  
[HermesSetCameraParSubArray\(\)](#)

**4.3.2.8 HermesSetDeadTime()**

```
HermesReturn HermesSetDeadTime (
    Hermes_H Hermes,
    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>Hermes</i>	Hermes handle
<i>Val</i>	New dead-time value in nanoseconds

**Returns**

OK  
NULL\_POINTER The provided Hermes\_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 HermesSetDeadTimeCorrection()**

```
HermesReturn HermesSetDeadTimeCorrection (
    Hermes_H Hermes,
    State s )
```

Enable or disable the dead-time correction.

The default setting is disabled.

## Parameters

<i>Hermes</i>	Hermes handle
<i>s</i>	New state for the dead-time corrector

## Returns

OK

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

## Examples

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

## 4.3.2.10 HermesSetDualGate()

```

HermesReturn HermesSetDualGate (
    Hermes_H Hermes,
    State DualGate_State,
    int StartShift,
    int FirstGateWidth,
    int SecondGateWidth,
    int Gap )

```

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>Hermes</i>	Hermes handle
<i>DualGate_State</i>	Enable or disable dual-gate mode
<i>StartShift</i>	Start delay for for the first gate in thousandths of gate period (20ns). Accepted values: -500 ... +500
<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 ... 1000

## Returns

OK

NULL\_POINTER The provided Hermes\_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 HermesSetFlimPar()

```
HermesReturn HermesSetFlimPar (
    Hermes_H Hermes,
    UInt16 FLIM_steps,
    UInt16 FLIM_shift,
    Int16 FLIM_start,
    UInt16 Length,
    int * FLIM_frame_time )
```

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 [HermesSetCameraPar\(\)](#). The total time required to perform each FLIM acquisition is passed back to the caller through the referenced FLIM\_↔ frame\_time variable.

##### Parameters

<i>Hermes</i>	Hermes handle
<i>FLIM_steps</i>	Number of gate delay steps to be performed. Accepted values: 1 ... 1000
<i>FLIM_shift</i>	Delay shift between steps in thousandths of gate period (20ns). Accepted values: 1 ... 1000
<i>FLIM_start</i>	Start delay for FLIM sequence in thousandths of gate period (20ns). Accepted values: -500 ... +500
<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.

##### Returns

OK

NULL\_POINTER The provided Hermes\_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

[HermesSetCameraPar\(\)](#)

#### 4.3.2.12 HermesSetFlimState()

```
HermesReturn HermesSetFlimState (
    Hermes_H Hermes,
    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>Hermes</i>	Hermes handle
<i>FLIM_State</i>	Enable or disable the FLIM mode

## Returns

OK

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

INVALID\_OP Exposure time is lower than 1040.

## See also

[HermesSetCameraPar\(\)](#)

[HermesSetFlimPar\(\)](#)

**4.3.2.13 HermesSetGateMode()**

```
HermesReturn HermesSetGateMode (
    Hermes_H Hermes,
    UInt16 counter,
    GateMode Mode )
```

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

## Parameters

<i>Hermes</i>	Hermes handle
<i>counter</i>	Counter to which settings refer. Accepted values: 1..3
<i>Mode</i>	New gate mode

## Returns

OK

NULL\_POINTER The provided Hermes\_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 [HermesSetDualGate\(\)](#) and [HermesSetTripleGate\(\)](#) functions.

## Examples

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

**4.3.2.14 HermesSetGateValues()**

```
HermesReturn HermesSetGateValues (
    Hermes_H Hermes,
    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>Hermes</i>	Hermes 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: -500 ... +500
<i>Length</i>	Duration of the ON gate signal. The unit is percentage. Accepted values: 0 ... 100

**Returns**

OK

NULL\_POINTER The provided Hermes\_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 HermesSetSyncInState()**

```
HermesReturn HermesSetSyncInState (
    Hermes_H Hermes,
    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>Hermes</i>	Hermes 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). Accepted values: 0 ... 100.



**Returns**

OK

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

**Examples**[SDK\\_Example.c](#).**4.3.2.16 HermesSetTriggerOutState()**

```
HermesReturn HermesSetTriggerOutState (
    Hermes_H Hermes,
    TriggerMode Mode )
```

Select the output signal.

**Parameters**

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

**Returns**

OK

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

**Examples**[SDK\\_Example.c](#).**4.3.2.17 HermesSetTripleGate()**

```
HermesReturn HermesSetTripleGate (
    Hermes_H Hermes,
    State TripleGate_State,
    int StartShift,
    int FirstGateWidth,
    int SecondGateWidth,
    int ThirdGateWidth,
    int Gap1,
    int Gap2 )
```

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>Hermes</i>	Hermes 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: -500 ... +500
<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: 0 ... 1000
<i>Gap2</i>	Gap between the gate3 and gate2 in thousandths of nominal gate period (20ns). Accepted values: 100 ... 1000

## Returns

OK

NULL\_POINTER The provided Hermes\_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

- [HermesReturn HermesGetDeadTime](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Val, [UInt16](#) \*ReturnVal)
- [HermesReturn HermesGetGateWidth](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter, [Int16](#) Val, double \*ReturnVal)
- [HermesReturn HermesGetGateShift](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter, [Int16](#) Val, [Int16](#) \*ReturnVal)
- [HermesReturn HermesIs16Bit](#) ([Hermes\\_H](#) Hermes, short \*is16bit)
- [HermesReturn HermesIsTriggered](#) ([Hermes\\_H](#) Hermes, short \*isTriggered)
- [HermesReturn HermesGetVersion](#) ([Hermes\\_H](#) Hermes, double \*Firmware\_Version, double \*Software\_Version, char \*Custom\_version)
- [HermesReturn HermesGetSerial](#) ([Hermes\\_H](#) Hermes, char \*Camera\_ID, char \*Camera\_serial)
- [HermesReturn HermesDeviceInfo](#) (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 settings from Hermes camera.

## 4.4.2 Function Documentation

### 4.4.2.1 HermesDeviceInfo()

```
HermesReturn HermesDeviceInfo (
    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 Hermes object. Useful when constructor fails due to incompatible firmware and SDK versions or for powering issues.

#### Parameters

<i>Device_ID</i>	Hermes 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.
<i>Camera_serial</i>	Hermes 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 "A" is returned. This parameter is referenced.
<i>Software_Custom_Version</i>	Customization version of the software. For standard model "A" is returned. This parameter is referenced.

#### Returns

OK

### 4.4.2.2 HermesGetDeadTime()

```
HermesReturn HermesGetDeadTime (
    Hermes_H Hermes,
    UInt16 Val,
    UInt16 * ReturnVal )
```

Get the calibrated dead-time value.

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

## Parameters

<i>Hermes</i>	Hermes 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 Hermes\_H or ReturnVal point to an empty memory location

## See also

[HermesSetDeadTime\(\)](#)

## Examples

[SDK\\_Example.c.](#)**4.4.2.3 HermesGetGateShift()**

```

HermesReturn HermesGetGateShift (
    Hermes_H Hermes,
    UInt16 counter,
    Int16 Val,
    Int16 * ReturnVal )

```

Get the calibrated gate shift value.

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

## Parameters

<i>Hermes</i>	Hermes handle
<i>counter</i>	Counter for which the gate shift is requested. Accepted values: 1..3
<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 ReturnVal.
<i>ReturnVal</i>	Closest gate-shift value possible. This parameter is referenced.

## Returns

OK

NULL\_POINTER The provided Hermes\_H or ReturnVal point to an empty memory location

#### 4.4.2.4 HermesGetGateWidth()

```
HermesReturn HermesGetGateWidth (
    Hermes_H Hermes,
    UInt16 counter,
    Int16 Val,
    double * ReturnVal )
```

Get the calibrated gate width value.

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

##### Parameters

<i>Hermes</i>	Hermes handle
<i>counter</i>	Counter for which the gate width is requested. Accepted values: 1..3
<i>Val</i>	Desired gate-width value in percentage of 20ns. No error is generated when the value is out of range, instead the real boundaries are forced on ReturnVal.
<i>ReturnVal</i>	Closest gate-width value possible. This parameter is referenced.

##### Returns

OK

NULL\_POINTER The provided Hermes\_H or ReturnVal point to an empty memory location

#### 4.4.2.5 HermesGetSerial()

```
HermesReturn HermesGetSerial (
    Hermes_H Hermes,
    char * Camera_ID,
    char * Camera_serial )
```

Get the camera serial number and ID.

##### Parameters

<i>Hermes</i>	Hermes 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>	Hermes camera serial number. A string of at least 33 character is required as parameter. This parameter is referenced.

##### Returns

OK

NULL\_POINTER The provided provided handle or pointers point to an empty memory location.

#### 4.4.2.6 HermesGetVersion()

```
HermesReturn HermesGetVersion (
    Hermes_H Hermes,
    double * Firmware_Version,
    double * Software_Version,
    char * Custom_version )
```

Get the SDK and camera firmware version.

##### Parameters

<i>Hermes</i>	Hermes 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 "A" is returned. This parameter is referenced.

##### Returns

OK

NULL\_POINTER The provided handle or pointers point to an empty memory location

#### 4.4.2.7 HermesIs16Bit()

```
HermesReturn HermesIs16Bit (
    Hermes_H Hermes,
    short * is16bit )
```

Get the actual bit depth of acquired data.

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

##### Parameters

<i>Hermes</i>	Hermes handle
<i>is16bit</i>	Actual status. The value is 0 if bit depth is 8-bit and 1 if bit depth is 16-bit. This parameter is referenced.

##### Returns

OK

NULL\_POINTER The provided Hermes\_H or is16bit pointers point to an empty memory location

#### 4.4.2.8 HermesIsTriggered()

```
HermesReturn HermesIsTriggered (
    Hermes_H Hermes,
    short * isTriggered )
```

Poll the camera for external trigger status.

Poll 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.

##### Parameters

<i>Hermes</i>	Hermes 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 Hermes\_H or is Triggered pointers point to an empty memory location

##### Examples

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

## 4.5 Acquisition methods

### Functions

- [HermesReturn HermesLiveSetModeON](#) (Hermes\_H Hermes)
- [HermesReturn HermesLiveSetModeOFF](#) (Hermes\_H Hermes)
- [HermesReturn HermesLiveGetImg](#) (Hermes\_H Hermes, UInt16 \*Img)
- [HermesReturn HermesSnapPrepare](#) (Hermes\_H Hermes)
- [HermesReturn HermesSnapAcquire](#) (Hermes\_H Hermes)
- [HermesReturn HermesSnapGetImageBuffer](#) (Hermes\_H Hermes, BUFFER\_H \*buffer, int \*DataDepth)
- [HermesReturn HermesSnapGetImgPosition](#) (Hermes\_H Hermes, UInt16 \*Img, UInt32 Position, UInt16 counter)
- [HermesReturn HermesContAcqToFileStart](#) (Hermes\_H Hermes, char \*filename)
- [HermesReturn HermesContAcqToFileGetMemory](#) (Hermes\_H Hermes, double \*total\_bytes)
- [HermesReturn HermesContAcqToFileStop](#) (Hermes\_H Hermes)
- [HermesReturn HermesContAcqToMemoryStart](#) (Hermes\_H Hermes)
- [HermesReturn HermesContAcqToMemoryGetBuffer](#) (Hermes\_H Hermes, double \*total\_bytes, BUFFER\_H \*buffer)
- [HermesReturn HermesContAcqToMemoryStop](#) (Hermes\_H Hermes)

#### 4.5.1 Detailed Description

Functions for acquiring data from the Hermes camera.

## 4.5.2 Function Documentation

### 4.5.2.1 HermesContAcqToFileGetMemory()

```
HermesReturn HermesContAcqToFileGetMemory (
    Hermes_H Hermes,
    double * total_bytes )
```

Dump the camera memory to the PC and save data to the file specified with the [HermesContAcqToFileStart\(\)](#) function in Hermes file format (for details on the format see function [HermesSaveImgDisk\(\)](#)).

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>Hermes</i>	Hermes handle
<i>total_bytes</i>	Total number of bytes read. Value is referenced.

#### Returns

OK

NULL\_POINTER The provided Hermes\_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 [Hermes\\_HermesContAcqToFileStart\(\)](#) before calling this function.

COMMUNICATION\_ERROR Communication error during data download.

Hermes\_MEMORY\_FULL Camera internal memory got full during data download. Data loss occurred. Reduce frame-rate or optimize your software to reduce dead-time between subsequent calling of the function.

#### See also

[HermesContAcqToFileStart\(\)](#)

[HermesContAcqToFileStop\(\)](#)

[HermesSaveImgDisk\(\)](#);

#### Examples

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



#### 4.5.2.2 HermesContAcqToFileStart()

```
HermesReturn HermesContAcqToFileStart (
    Hermes_H Hermes,
    char * filename )
```

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 [HermesContAcqToFileGetMemory\(\)](#) function as soon as possible, in order to avoid data loss.

##### Parameters

<i>Hermes</i>	Hermes handle
<i>filename</i>	Full path of the output file. The string length must not exceed 1024 characters. Value is referenced.

##### Returns

OK

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

UNABLE\_CREATE\_FILE It was not possible to create the output file.

##### See also

[HermesContAcqToFileGetMemory\(\)](#)

[HermesContAcqToFileStop\(\)](#)

##### Examples

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

#### 4.5.2.3 HermesContAcqToFileStop()

```
HermesReturn HermesContAcqToFileStop (
    Hermes_H Hermes )
```

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 behavior if other functions are called beforehand.

##### Parameters

<i>Hermes</i>	Hermes handle
---------------	---------------

## Returns

OK

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

UNABLE\_CREATE\_FILE It was not possible to access the output file.

## See also

[HermesContAcqToFileStart\(\)](#)[HermesContAcqToFileGetMemory\(\)](#)

## Examples

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

## 4.5.2.4 HermesContAcqToMemoryGetBuffer()

```

HermesReturn HermesContAcqToMemoryGetBuffer (
    Hermes_H Hermes,
    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. Refer to Figure 7 of the User Manual for pixels readout order and position in the image. **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>Hermes</i>	Hermes 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 Hermes\_H or BUFFER\_H point to an empty memory location

INVALID\_OP Continues acquisition was not yet started. Use [HermesContAcqToMemoryStart\(\)](#) before calling this function.

COMMUNICATION\_ERROR Communication error during data download.

Hermes\_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

[HermesContAcqToMemoryStart\(\)](#)[HermesContAcqToMemoryStop\(\)](#)

## Examples

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

#### 4.5.2.5 HermesContAcqToMemoryStart()

```
HermesReturn HermesContAcqToMemoryStart (
    Hermes_H Hermes )
```

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 [HermesContAcqToMemoryGetBuffer\(\)](#) function as soon as possible, in order to avoid data loss.

## Parameters

<i>Hermes</i>	Hermes handle
---------------	---------------

## Returns

OK

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

## See also

[HermesContAcqToMemoryGetBuffer\(\)](#)[HermesContAcqToMemoryStop\(\)](#)

## Examples

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

#### 4.5.2.6 HermesContAcqToMemoryStop()

```
HermesReturn HermesContAcqToMemoryStop (
    Hermes_H Hermes )
```

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 behavior if other functions are called.

## Parameters

<i>Hermes</i>	Hermes handle
---------------	---------------

## Returns

OK

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

## See also

[HermesContAcqToMemoryStart\(\)](#)[HermesContAcqToMemoryGetBuffer\(\)](#)

## Examples

[SDK\\_Example.c.](#)**4.5.2.7 HermesLiveGetImg()**

```
HermesReturn HermesLiveGetImg (
    Hermes_H Hermes,
    UInt16 * Img )
```

Get a Live image.

Acquire a live image and store the data into the user-allocated Img array. This command is working only when the Live mode is turned on by the [HermesLiveSetModeON\(\)](#) function.

## Parameters

<i>Hermes</i>	Hermes handle
<i>Img</i>	Pointer to the output image array. The size of the array must be at least 4 KB.

## Returns

OK

NULL\_POINTER The provided Hermes\_H or Img point to an empty memory location

INVALID\_OP The live-mode has not been started yet

## See also

[HermesLiveSetModeON\(\)](#)

## Examples

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

#### 4.5.2.8 HermesLiveSetModeOFF()

```
HermesReturn HermesLiveSetModeOFF (
    Hermes_H Hermes )
```

Turn off the Live mode.

##### Parameters

<i>Hermes</i>	Hermes handle
---------------	---------------

##### Returns

OK

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

INVALID\_OP The live mode is already inactive

##### See also

[HermesLiveSetModeON\(\)](#)

##### Examples

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

#### 4.5.2.9 HermesLiveSetModeON()

```
HermesReturn HermesLiveSetModeON (
    Hermes_H Hermes )
```

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 [HermesSnapAcquire\(\)](#) or [HermesContAcqToFileGetMemory\(\)](#) can be performed.

##### Parameters

<i>Hermes</i>	Hermes handle
---------------	---------------

##### Returns

OK

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

INVALID\_OP The live mode has been already started

## See also

[HermesLiveSetModeOFF\(\)](#)  
[HermesContAcqToFileGetMemory\(\)](#)  
[HermesSnapAcquire\(\)](#)

## Examples

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

#### 4.5.2.10 HermesSnapAcquire()

```
HermesReturn HermesSnapAcquire (
    Hermes_H Hermes )
```

Get a selected number of images.

Acquire a set of images according to the parameters defined by [HermesSetCameraPar\(\)](#). In FLIM mode NFrames "FLIM acquisitions" will be acquired. This command works only when [HermesSnapPrepare\(\)](#) 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 poll the camera for the trigger state, using the [HermesIsTriggered\(\)](#) function.

## Parameters

<i>Hermes</i>	Hermes handle
---------------	---------------

## Returns

OK

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

INVALID\_OP Unable to acquire images when the live mode is ON. Use instead [HermesLiveGetImg\(\)](#).

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

[HermesSetCameraPar\(\)](#)  
[HermesSnapPrepare\(\)](#)  
[HermesSetSyncInState\(\)](#)  
[HermesIsTriggered\(\)](#)

## Examples

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

#### 4.5.2.11 HermesSnapGetImageBuffer()

```
HermesReturn HermesSnapGetImageBuffer (
    Hermes_H Hermes,
    BUFFER_H * buffer,
    int * DataDepth )
```

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

Refer to Figure 7 of the User Manual for pixels readout order and position in the image. **WARNING** User must pay attention not to exceed the dimension of the buffer (2\*1024\*65534 bytes) when accessing it.

##### Parameters

<i>Hermes</i>	Hermes handle
<i>buffer</i>	Pointer to the buffer Handle in which the function will save reference to the camera image buffer
<i>DataDepth</i>	Return the number of bits per pixel (8 or 16) in the image. The value is referenced.

##### Returns

OK

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

##### See also

[HermesSnapAcquire\(\)](#)

#### 4.5.2.12 HermesSnapGetImgPosition()

```
HermesReturn HermesSnapGetImgPosition (
    Hermes_H Hermes,
    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 [HermesSnapAcquire\(\)](#), a single image can be exported from the SDK image buffer and saved in the memory (Img array).

##### Parameters

<i>Hermes</i>	Hermes handle
<i>Img</i>	Pointer to the output image array. The size of the array must be at least 4kB.
<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 Hermes\_H or Img point to an empty memory location

OUT\_OF\_BOUND Parameters are out of bound.

**See also**[HermesSnapAcquire\(\)](#)**4.5.2.13 HermesSnapPrepare()**

```
HermesReturn HermesSnapPrepare (
    Hermes_H Hermes )
```

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 [HermesSetCameraPar\(\)](#) 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 [HermesSnapAcquire\(\)](#) function.

**Parameters**

<i>Hermes</i>	Hermes handle
---------------	---------------

**Returns**

OK

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

INVALID\_OP Unable to acquire images when the live mode is ON. Use instead [HermesLiveGetImg\(\)](#).

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**[HermesSetCameraPar\(\)](#)[HermesSnapAcquire\(\)](#)[HermesSetSyncInState\(\)](#)**Examples**[SDK\\_Example.c](#).



## 4.6 Additional methods

### Functions

- [HermesReturn HermesSaveImgDisk](#) ([Hermes\\_H](#) Hermes, [UInt32](#) Start\_Img, [UInt32](#) End\_Img, char \*filename, [OutFileFormat](#) mode)
- [HermesReturn HermesSaveAveragedImgDisk](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter, char \*filename, [OutFileFormat](#) mode, short isDouble)
- [HermesReturn HermesSaveFlimDisk](#) ([Hermes\\_H](#) Hermes, char \*filename, [OutFileFormat](#) mode)
- [HermesReturn HermesReadHermesFileFormatImage](#) (char \*filename, [UInt32](#) ImgIdx, [UInt16](#) counter, [UInt16](#) \*Img, char header[1024])
- [HermesReturn HermesAveragelmg](#) ([Hermes\\_H](#) Hermes, double \*Img, [UInt16](#) counter)
- [HermesReturn HermesStDevImg](#) ([Hermes\\_H](#) Hermes, double \*Img, [UInt16](#) counter)
- [HermesReturn HermesSetCorrelationMode](#) ([Hermes\\_H](#) Hermes, [CorrelationMode](#) CM, int NCorrChannels, State s)
- [HermesReturn HermesCorrelationImg](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter)
- [HermesReturn HermesSaveCorrelationImg](#) ([Hermes\\_H](#) Hermes, char \*filename)
- [HermesReturn HermesResetOverilluminationProtection](#) ([Hermes\\_H](#) Hermes)

### 4.6.1 Detailed Description

Additional utility functions.

### 4.6.2 Function Documentation

#### 4.6.2.1 HermesAveragelmg()

```
HermesReturn HermesAverageImg (
    Hermes_H Hermes,
    double * Img,
    UInt16 counter )
```

Calculate the average image.

Once a set of images have been acquired by [HermesSnapAcquire\(\)](#), 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>Hermes</i>	Hermes handle.
<i>Img</i>	Pointer to the output double image array. The size of the array must be at least 16 kiB, i.e. 2048 double elements.
<i>counter</i>	Desired counter. Accepted values: 1..3

**Returns**

OK  
NULL\_POINTER The provided Hermes\_H points to an empty memory location  
INVALID\_OP No images were acquired

**See also**

[HermesSnapAcquire\(\)](#)

**Examples**

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

**4.6.2.2 HermesCorrelationImg()**

```
HermesReturn HermesCorrelationImg (
    Hermes_H Hermes,
    UInt16 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 [HermesSnapAcquire\(\)](#) 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>Hermes</i>	Hermes handle
<i>counter</i>	Desired counter. Accepted values: 1..3

**Returns**

OK  
NULL\_POINTER The provided Hermes\_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**

[HermesSetCorrelationMode\(\)](#)  
[HermesSnapAcquire\(\)](#)

#### 4.6.2.3 HermesReadHermesFileFormatImage()

```
HermesReturn HermesReadHermesFileFormatImage (
    char * filename,
    UInt32 ImgIdx,
    UInt16 counter,
    UInt16 * Img,
    char header[1024] )
```

Read an integer (8 - 16 bit) Hermes image from file.

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

##### Parameters

<i>filename</i>	Full path of the output file. Value is referenced.
<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 4 kiB.
<i>header</i>	Array in which the header of Hermes file is saved.

##### Returns

OK

UNABLE\_READ\_FILE Unable to read the input file. Is it a Hermes 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

NULL\_POINTER The provided provided handle or pointers point to an empty memory location.

##### Examples

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

#### 4.6.2.4 HermesResetOverilluminationProtection()

```
HermesReturn HermesResetOverilluminationProtection (
    Hermes_H Hermes )
```

Reset the internal overillumination protection circuit.

This function resets the Hermes internal protection triggered by excessive illumination. It is mandatory to check that the overillumination condition is removed before calling this function. In any case, in order to avoid damage to the Hermes, up to 3 reset cycles are allowed before having to disconnect the camera from the power supply.

##### Parameters

<i>Hermes</i>	Hermes handle
---------------	---------------

**Returns**

OK

TOO\_MUCH\_LIGHT The protection was reset, but the illumination is still too much!

PERSISTING\_TOO\_MUCH\_LIGHT The number of allowed reset cycles has been exceeded!

**Examples**[SDK\\_Example.c.](#)**4.6.2.5 HermesSaveAveragedImgDisk()**

```
HermesReturn HermesSaveAveragedImgDisk (
    Hermes_H Hermes,
    UInt16 counter,
    char * filename,
    OutFileFormat mode,
    short isDouble )
```

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 Hermes or TIFF, as explained in [HermesSaveImgDisk\(\)](#) function.

**Parameters**

<i>Hermes</i>	Hermes handle
<i>counter</i>	Number of the counter to be saved. Accepted values: 1..3
<i>filename</i>	Full path of the output file. Value is referenced.
<i>mode</i>	File format of the output images
<i>isDouble</i>	Number format. 0 for UInt16, 1 for Double

**Returns**

OK

NULL\_POINTER The provided Hermes\_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

**See also**[HermesSaveImgDisk\(\)](#)**Examples**[SDK\\_Example.c.](#)

#### 4.6.2.6 HermesSaveCorrelationImg()

```
HermesReturn HermesSaveCorrelationImg (
    Hermes_H Hermes,
    char * filename )
```

Save the autocorrelation functions on the hard disk.

This function requires that [HermesSetCorrelationMode\(\)](#) and [HermesCorrelationImg\(\)](#) have been previously called. The autocorrelation data are stored in a .hrmc binary file. The hrmc binary file is organized as follows:

Byte offset	Type	Number of bytes	Description
0	int	4	Number of lag-times (NLag)
4	int	4	Number of pixels. This value must be 1024 (NPix)
8	int	4	Selected algorithm: 0 Linear, 1 Multi-tau
12	double	8 * NLag	Autocorrelation values of the first pixel
12 + 8 * NLag	double	8 * NLag	Autocorrelation values of the second pixel
...	double	8 * NLag	Autocorrelation values of the N <sup>th</sup> pixel
12 + 8 * (NPix-1) * NLag	double	8 * NLag	Autocorrelation values of the last pixel
12 + 8 * NPix * NLag	double	8 * NLag	Lag times

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

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MPD .HRMC file reader
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function data = Read_HRMC(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

<i>Hermes</i>	Hermes handle
<i>filename</i>	Full path of the output file

#### Returns

OK

NULL\_POINTER The provided Hermes\_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

[HermesSetCorrelationMode\(\)](#)

#### 4.6.2.7 HermesSaveFlimDisk()

```
HermesReturn HermesSaveFlimDisk (
    Hermes_H Hermes,
    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 Hermes format. For standard measurements, use the `Hermes_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 a 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>). Hermes 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 (refer to Figure 7 of the User Manual for pixels position and 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 described in function `HermesSaveImgDisk()`. Hermes file can be read using the provided ImageJ/Fiji plugin.

##### Parameters

<i>Hermes</i>	Hermes handle
<i>filename</i>	Full path of the output file. Value is referenced.
<i>mode</i>	File format of the output images

##### Returns

OK

NULL\_POINTER The provided Hermes\_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

##### See also

[HermesSaveImgDisk\(\)](#)

#### 4.6.2.8 HermesSaveImgDisk()

```
HermesReturn HermesSaveImgDisk (
    Hermes_H Hermes,
    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 Hermes format. For FLIM measurements, use the [HermesSaveFlimDisk\(\)](#) function. If TIFF format is selected, the desired images will be saved in a file for each enabled counter. If Hermes format is selected a single Hermes 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 a 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 as much rows of 32 pixels as needed to accommodate 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. Hermes file are binary files composed by a header with acquisition metadata followed by raw image data, containing the 8/16 bit (integer) or 64 bit (double precision) pixel values in row-major order (refer to Figure 7 of the User Manual for pixels position and order). The byte order is little-endian for the 16 or 64 bit images. In case more counters are used, data are interleaved, 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 (0x4d5044ff04000000, 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	Hermes serial number (string)
42	2	Firmware version (x.xx saved as xxx)
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	Hold-off 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 (thousandths of gate period)
203	2	FLIM steps

Byte offset	Number of bytes	Description
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 (-500 - +500)
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 bin-width 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
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

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

#### Parameters

<i>Hermes</i>	Hermes 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>	Full path of the output file. Value is referenced.
<i>mode</i>	File format of the output images

#### Returns

OK

NULL\_POINTER The provided Hermes\_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](#).



#### 4.6.2.9 HermesSetCorrelationMode()

```
HermesReturn HermesSetCorrelationMode (
    Hermes_H Hermes,
    CorrelationMode CM,
    int NCorrChannels,
    State s )
```

Enable the correlation mode.

This function must be called before invoking [HermesCorrelationImg\(\)](#). 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 `Hermes_destr()` function is called or by setting the State `s` equal to Disabled.

##### Parameters

<i>Hermes</i>	Hermes 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 Hermes\_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

[HermesCorrelationImg\(\)](#)

#### 4.6.2.10 HermesStDevImg()

```
HermesReturn HermesStDevImg (
    Hermes_H Hermes,
    double * Img,
    UInt16 counter )
```

Calculate the standard deviation image.

Once a set of images have been acquired by [HermesSnapAcquire\(\)](#), 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>Hermes</i>	Pointer to the Hermes handle
<i>Img</i>	Pointer to the output double image array. The size of the array must be at least 16 kiB, i.e. 2048 double elements.
<i>counter</i>	Desired counter. Accepted values: 1..3

## Returns

OK

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

INVALID\_OP No images were acquired

## See also

[HermesSnapAcquire\(\)](#)

## Chapter 5

# File Documentation

### 5.1 Hermes\_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` +500
- #define `MIN_GATE_SHIFT` -500

#### Typedefs

- typedef unsigned short `UInt16`
- typedef short `Int16`
- typedef unsigned `UInt32`
- typedef struct \_Hermes\_H \* `Hermes_H`
- typedef unsigned char \* `BUFFER_H`

#### Enumerations

- enum `HermesReturn` {  
    `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 ,  
    `HERMES_MEMORY_FULL` = -18 , `PERSISTING_TOO_MUCH_LIGHT` = -19 }
- enum `OutFileFormat` { `HERMES_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

- [HermesReturn HermesConstr](#) ([Hermes\\_H](#) \*Hermes\_in, [CameraMode](#) m, char \*Device\_ID)
- [HermesReturn HermesDestr](#) ([Hermes\\_H](#) Hermes)
- void [PrintErrorCode](#) (FILE \*fout, const char \*FunName, [HermesReturn](#) retcode)
- [HermesReturn HermesSetCameraPar](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Exposure, [UInt32](#) NFrames, [UInt16](#) NIntegFrames, [UInt16](#) NCounters, [State](#) Force8bit, [State](#) Half\_array, [State](#) Signed\_data)
- [HermesReturn HermesSetCameraParSubArray](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Exposure, [UInt32](#) NFrames, [UInt16](#) NIntegFrames, [State](#) Force8bit, [UInt16](#) Npixels)
- [HermesReturn HermesSetDeadTime](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Val)
- [HermesReturn HermesSetDeadTimeCorrection](#) ([Hermes\\_H](#) Hermes, [State](#) s)
- [HermesReturn HermesSetAdvancedMode](#) ([Hermes\\_H](#) Hermes, [State](#) s)
- [HermesReturn HermesSetBackgroundImg](#) ([Hermes\\_H](#) Hermes, [UInt16](#) \*Img)
- [HermesReturn HermesSetBackgroundSubtraction](#) ([Hermes\\_H](#) Hermes, [State](#) s)
- [HermesReturn HermesSetGateMode](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter, [GateMode](#) Mode)
- [HermesReturn HermesSetGateValues](#) ([Hermes\\_H](#) Hermes, [Int16](#) Shift, [Int16](#) Length)
- [HermesReturn HermesSetDualGate](#) ([Hermes\\_H](#) Hermes, [State](#) DualGate\_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int Gap)
- [HermesReturn HermesSetTripleGate](#) ([Hermes\\_H](#) Hermes, [State](#) TripleGate\_State, int StartShift, int FirstGateWidth, int SecondGateWidth, int ThirdGateWidth, int Gap1, int Gap2)
- [HermesReturn HermesSetCoarseGateValues](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Counter, [UInt16](#) Start, [UInt16](#) Stop)
- [HermesReturn HermesSetTriggerOutState](#) ([Hermes\\_H](#) Hermes, [TriggerMode](#) Mode)
- [HermesReturn HermesSetSyncInState](#) ([Hermes\\_H](#) Hermes, [State](#) s, int frames)
- [HermesReturn HermesSetFliMPar](#) ([Hermes\\_H](#) Hermes, [UInt16](#) FLIM\_steps, [UInt16](#) FLIM\_shift, [Int16](#) FLIM\_start, [UInt16](#) Length, int \*FLIM\_frame\_time)
- [HermesReturn HermesSetFliMState](#) ([Hermes\\_H](#) Hermes, [State](#) FLIM\_State)
- [HermesReturn HermesApplySettings](#) ([Hermes\\_H](#) Hermes)
- [HermesReturn HermesGetDeadTime](#) ([Hermes\\_H](#) Hermes, [UInt16](#) Val, [UInt16](#) \*ReturnVal)
- [HermesReturn HermesGetGateWidth](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter, [Int16](#) Val, double \*ReturnVal)
- [HermesReturn HermesGetGateShift](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter, [Int16](#) Val, [Int16](#) \*ReturnVal)
- [HermesReturn HermesIs16Bit](#) ([Hermes\\_H](#) Hermes, short \*is16bit)
- [HermesReturn HermesIsTriggered](#) ([Hermes\\_H](#) Hermes, short \*isTriggered)
- [HermesReturn HermesGetVersion](#) ([Hermes\\_H](#) Hermes, double \*Firmware\_Version, double \*Software\_Version, char \*Custom\_version)
- [HermesReturn HermesGetSerial](#) ([Hermes\\_H](#) Hermes, char \*Camera\_ID, char \*Camera\_serial)
- [HermesReturn HermesDeviceInfo](#) (char \*Device\_ID, char \*Camera\_serial, double \*Firmware\_Version, double \*Software\_Version, char \*Firmware\_Custom\_Version, char \*Software\_Custom\_Version)
- [HermesReturn HermesLiveSetModeON](#) ([Hermes\\_H](#) Hermes)
- [HermesReturn HermesLiveSetModeOFF](#) ([Hermes\\_H](#) Hermes)
- [HermesReturn HermesLiveGetImg](#) ([Hermes\\_H](#) Hermes, [UInt16](#) \*Img)
- [HermesReturn HermesSnapPrepare](#) ([Hermes\\_H](#) Hermes)
- [HermesReturn HermesSnapAcquire](#) ([Hermes\\_H](#) Hermes)
- [HermesReturn HermesSnapGetImageBuffer](#) ([Hermes\\_H](#) Hermes, [BUFFER\\_H](#) \*buffer, int \*DataDepth)
- [HermesReturn HermesSnapGetImgPosition](#) ([Hermes\\_H](#) Hermes, [UInt16](#) \*Img, [UInt32](#) Position, [UInt16](#) counter)
- [HermesReturn HermesContAcqToFileStart](#) ([Hermes\\_H](#) Hermes, char \*filename)
- [HermesReturn HermesContAcqToFileGetMemory](#) ([Hermes\\_H](#) Hermes, double \*total\_bytes)
- [HermesReturn HermesContAcqToFileStop](#) ([Hermes\\_H](#) Hermes)
- [HermesReturn HermesContAcqToMemoryStart](#) ([Hermes\\_H](#) Hermes)
- [HermesReturn HermesContAcqToMemoryGetBuffer](#) ([Hermes\\_H](#) Hermes, double \*total\_bytes, [BUFFER\\_H](#) \*buffer)
- [HermesReturn HermesContAcqToMemoryStop](#) ([Hermes\\_H](#) Hermes)
- [HermesReturn HermesSaveImgDisk](#) ([Hermes\\_H](#) Hermes, [UInt32](#) Start\_Img, [UInt32](#) End\_Img, char \*filename, [OutFileFormat](#) mode)

- [HermesReturn HermesSaveAveragedImgDisk](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter, char \*filename, [OutFileFormat](#) mode, short isDouble)
- [HermesReturn HermesSaveFlimDisk](#) ([Hermes\\_H](#) Hermes, char \*filename, [OutFileFormat](#) mode)
- [HermesReturn HermesReadHermesFileFormatImage](#) (char \*filename, [UInt32](#) ImgIdx, [UInt16](#) counter, [UInt16](#) \*Img, char header[1024])
- [HermesReturn HermesAveragedImg](#) ([Hermes\\_H](#) Hermes, double \*Img, [UInt16](#) counter)
- [HermesReturn HermesStdDevImg](#) ([Hermes\\_H](#) Hermes, double \*Img, [UInt16](#) counter)
- [HermesReturn HermesSetCorrelationMode](#) ([Hermes\\_H](#) Hermes, [CorrelationMode](#) CM, int NCorrChannels, [State](#) s)
- [HermesReturn HermesCorrelationImg](#) ([Hermes\\_H](#) Hermes, [UInt16](#) counter)
- [HermesReturn HermesSaveCorrelationImg](#) ([Hermes\\_H](#) Hermes, char \*filename)
- [HermesReturn HermesResetOverilluminationProtection](#) ([Hermes\\_H](#) Hermes)

### 5.1.1 Detailed Description

Hermes software development kit.

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

### 5.1.2 Macro Definition Documentation

#### 5.1.2.1 MAX\_DEAD\_TIME

```
#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 MIN\_DEAD\_TIME

```
#define MIN_DEAD_TIME 30
```

Minimum allowed dead-time in nanoseconds.

The dark-counts rate and after-pulsing probability depend on the used dead-time setting: the lower the dead-time, the higher the dark-counts rate and after-pulsing probability. However, a long dead-time limits the maximum number of photons per second detected by the matrix of SPADs. It is recommended to set this parameter to an intermediate value, e.g. 50 ns.

#### Examples

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

## 5.2 Hermes\_SDK.h

[Go to the documentation of this file.](#)

```

1  /*
2  #####
3
4  Copyright 2023 Micro-Photon-Devices s.r.l.
5
6  SOFTWARE PRODUCT: Hermes_SDK 1.0.1A
7
8  Micro-Photon-Devices (MPD) expressly disclaims any warranty for the SOFTWARE PRODUCT.
9  The SOFTWARE PRODUCT is provided 'As Is' without any express or implied warranty of any kind,
10 including but not limited to any warranties of merchantability, non-infringement, or
11 fitness of a particular purpose. MPD does not warrant or assume responsibility for the
12 accuracy or completeness of any information, text, graphics, links or other items contained
13 within the SOFTWARE PRODUCT. MPD further expressly disclaims any warranty or representation
14 to Authorized Users or to any third party.
15 In no event shall MPD be liable for any damages (including, without limitation, lost profits,
16 business interruption, or lost information) rising out of 'Authorized Users' use of or inability
17 to use the SOFTWARE PRODUCT, even if MPD has been advised of the possibility of such damages.
18 In no event will MPD be liable for loss of data or for indirect, special, incidental,
19 consequential (including lost profit), or other damages based in contract, tort
20 or otherwise. MPD shall have no liability with respect to the content of the
21 SOFTWARE PRODUCT or any part thereof, including but not limited to errors or omissions contained
22 therein, libel, infringements of rights of publicity, privacy, trademark rights, business
23 interruption, personal injury, loss of privacy, moral rights or the disclosure of confidential
24 information.
25
26 #####
27 */
28
29 #ifndef __Hermes_SDK_h__
30 #define __Hermes_SDK_h__
31
32 #ifdef __cplusplus
33 extern "C" {
34 #endif
35
36 #include <stdio.h>
37 #include <math.h>
38 #include <string.h>
39
40 #ifndef DLLSDKExport
41 #if defined(_WIN32)
42 #define DLLSDKExport __declspec(dllexport)
43 #else
44 #define DLLSDKExport __attribute__((visibility("default")))
45 #endif
46 #endif
47
48 typedef unsigned short UInt16;
49
50 typedef short Int16;
51
52 typedef unsigned UInt32;
53
54 #define MIN_DEAD_TIME 30
55 #define MAX_DEAD_TIME 150
56
57 #define MAX_GATE_WIDTH 100
58
59 #define MIN_GATE_WIDTH 0
60
61 #define MAX_GATE_SHIFT +500
62
63 #define MIN_GATE_SHIFT -500
64
65 typedef enum{
66     OK = 0,
67     USB_DEVICE_NOT_RECOGNIZED= -1,
68     CAMERA_NOT_POWERING_UP=-3,
69     COMMUNICATION_ERROR=-5,
70     OUT_OF_BOUND = -6,
71     MISSING_DLL = -7,
72     EMPTY_BUFFER = -8,
73     NOT_EN_MEMORY = -9,
74     NULL_POINTER = -10,
75     INVALID_OP = -11,
76     UNABLE_CREATE_FILE = -12,
77     UNABLE_READ_FILE = -13,
78     FIRMWARE_NOT_COMPATIBLE=-14,
79     POWER_SUPPLY_ERROR = -15,
80     TOO_MUCH_LIGHT = -16,
81     INVALID_NIMG_CORRELATION = -17,
82     HERMES_MEMORY_FULL = -18,
83     PERSISTING_TOO_MUCH_LIGHT = -19
84 }

```

```

134 } HermesReturn;
135
136 typedef enum{
137     HERMES_FILEFORMAT = 0,
138     TIFF_NO_COMPRESSION = 2
139 } OutFileFormat;
140
141 typedef enum{
142     Continuous = 0,
143     Pulsed = 1,
144     Coarse = 2
145 } GateMode;
146
147 typedef enum{
148     Normal = 0,
149     Advanced = 1
150 } CameraMode;
151
152 typedef enum{
153     None = 0,
154     Gate_Clk = 1,
155     Frame = 2
156 } TriggerMode;
157
158 typedef enum{
159     Disabled = 0,
160     Enabled = 1
161 } State;
162
163 typedef enum{
164     Linear = 0,
165     MultiTau = 1
166 } CorrelationMode;
167
168 typedef struct _Hermes_H * Hermes_H;
169
170 typedef unsigned char * BUFFER_H;
171
172 //----- Constructor -----
173
174 DllSDKEExport HermesReturn HermesConstr(Hermes_H* Hermes_in, CameraMode m, char* Device_ID);
175
176 DllSDKEExport HermesReturn HermesDestr(Hermes_H Hermes);
177
178 //----- Error handling -----
179
180 DllSDKEExport void PrintErrorCode(FILE* fout, const char* FunName, HermesReturn retcode);
181
182 //----- Set methods -----
183
184 DllSDKEExport HermesReturn HermesSetCameraPar(Hermes_H Hermes, UInt16 Exposure, UInt32 NFrames, UInt16
    NIntegFrames, UInt16 NCounters, State Force8bit, State Half_array, State Signed_data);
185
186 DllSDKEExport HermesReturn HermesSetCameraParSubArray(Hermes_H Hermes, UInt16 Exposure, UInt32 NFrames,
    UInt16 NIntegFrames, State Force8bit, UInt16 Npixels);
187
188 DllSDKEExport HermesReturn HermesSetDeadTime(Hermes_H Hermes, UInt16 Val);
189
190 DllSDKEExport HermesReturn HermesSetDeadTimeCorrection(Hermes_H Hermes, State s);
191
192 DllSDKEExport HermesReturn HermesSetAdvancedMode(Hermes_H Hermes, State s);
193
194 DllSDKEExport HermesReturn HermesSetBackgroundImg(Hermes_H Hermes, UInt16* Img);
195
196 DllSDKEExport HermesReturn HermesSetBackgroundSubtraction(Hermes_H Hermes, State s);
197
198 DllSDKEExport HermesReturn HermesSetGateMode(Hermes_H Hermes, UInt16 counter, GateMode Mode);
199
200 DllSDKEExport HermesReturn HermesSetGateValues(Hermes_H Hermes, Int16 Shift, Int16 Length);
201
202 DllSDKEExport HermesReturn HermesSetDualGate(Hermes_H Hermes, State DualGate_State, int StartShift, int
    FirstGateWidth, int SecondGateWidth, int Gap);
203
204 DllSDKEExport HermesReturn HermesSetTripleGate(Hermes_H Hermes, State TripleGate_State, int StartShift,
    int FirstGateWidth, int SecondGateWidth, int ThirdGateWidth, int Gap1, int Gap2);
205
206 DllSDKEExport HermesReturn HermesSetCoarseGateValues(Hermes_H Hermes, UInt16 Counter, UInt16 Start,
    UInt16 Stop);
207
208 DllSDKEExport HermesReturn HermesSetTriggerOutState(Hermes_H Hermes, TriggerMode Mode);
209
210 DllSDKEExport HermesReturn HermesSetSyncInState(Hermes_H Hermes, State s, int frames);
211
212 DllSDKEExport HermesReturn HermesSetFlimPar(Hermes_H Hermes, UInt16 FLIM_steps, UInt16 FLIM_shift, Int16
    FLIM_start, UInt16 Length, int* FLIM_frame_time);
213
214 DllSDKEExport HermesReturn HermesSetFlimState(Hermes_H Hermes, State FLIM_State);
215
216 DllSDKEExport HermesReturn HermesApplySettings(Hermes_H Hermes);

```

```

470
472 //----- Get methods -----
487 DllSDKEExport HermesReturn HermesGetDeadTime(Hermes_H Hermes, UInt16 Val, UInt16* ReturnVal);
488
498 DllSDKEExport HermesReturn HermesGetGateWidth(Hermes_H Hermes, UInt16 counter, Int16 Val, double*
    ReturnVal);
499
509 DllSDKEExport HermesReturn HermesGetGateShift(Hermes_H Hermes, UInt16 counter, Int16 Val, Int16*
    ReturnVal);
510
519 DllSDKEExport HermesReturn HermesIs16Bit(Hermes_H Hermes, short* is16bit);
520
528 DllSDKEExport HermesReturn HermesIsTriggered(Hermes_H Hermes, short* isTriggered);
529
538 DllSDKEExport HermesReturn HermesGetVersion(Hermes_H Hermes, double* Firmware_Version, double*
    Software_Version, char* Custom_version);
539
547 DllSDKEExport HermesReturn HermesGetSerial(Hermes_H Hermes, char* Camera_ID, char* Camera_serial);
548
560 DllSDKEExport HermesReturn HermesDeviceInfo(char* Device_ID, char* Camera_serial, double*
    Firmware_Version, double* Software_Version, char* Firmware_Custom_Version, char*
    Software_Custom_Version);
561
563 //----- Acquisition methods -----
581 DllSDKEExport HermesReturn HermesLiveSetModeON(Hermes_H Hermes);
582
590 DllSDKEExport HermesReturn HermesLiveSetModeOFF(Hermes_H Hermes);
591
601 DllSDKEExport HermesReturn HermesLiveGetImg(Hermes_H Hermes, UInt16* Img);
602
617 DllSDKEExport HermesReturn HermesSnapPrepare(Hermes_H Hermes);
618
636 DllSDKEExport HermesReturn HermesSnapAcquire(Hermes_H Hermes);
637
646 DllSDKEExport HermesReturn HermesSnapGetImageBuffer(Hermes_H Hermes, BUFFER_H* buffer, int* DataDepth);
647
660 DllSDKEExport HermesReturn HermesSnapGetImgPosition(Hermes_H Hermes, UInt16* Img, UInt32 Position, UInt16
    counter);
661
662
674 DllSDKEExport HermesReturn HermesContAcqToFileStart(Hermes_H Hermes, char* filename);
675
692 DllSDKEExport HermesReturn HermesContAcqToFileGetMemory(Hermes_H Hermes, double* total_bytes);
693
703 DllSDKEExport HermesReturn HermesContAcqToFileStop(Hermes_H Hermes);
704
713 DllSDKEExport HermesReturn HermesContAcqToMemoryStart(Hermes_H Hermes);
714
728 DllSDKEExport HermesReturn HermesContAcqToMemoryGetBuffer(Hermes_H Hermes, double* total_bytes, BUFFER_H*
    buffer);
729
737 DllSDKEExport HermesReturn HermesContAcqToMemoryStop(Hermes_H Hermes);
738
740 //----- Utilities -----
828 DllSDKEExport HermesReturn HermesSaveImgDisk(Hermes_H Hermes, UInt32 Start_Img, UInt32 End_Img, char*
    filename, OutFileFormat mode);
829
843 DllSDKEExport HermesReturn HermesSaveAveragedImgDisk(Hermes_H Hermes, UInt16 counter, char* filename,
    OutFileFormat mode, short isDouble);
844
864 DllSDKEExport HermesReturn HermesSaveFlinDisk(Hermes_H Hermes, char* filename, OutFileFormat mode);
865
866
880 DllSDKEExport HermesReturn HermesReadHermesFileFormatImage(char* filename, UInt32 ImgIdx, UInt16 counter,
    UInt16* Img, char header[1024]);
881
893 DllSDKEExport HermesReturn HermesAverageImg(Hermes_H Hermes, double* Img, UInt16 counter);
894
906 DllSDKEExport HermesReturn HermesStDevImg(Hermes_H Hermes, double* Img, UInt16 counter);
907
925 DllSDKEExport HermesReturn HermesSetCorrelationMode(Hermes_H Hermes, CorrelationMode CM, int
    NCorrChannels, State s);
926
941 DllSDKEExport HermesReturn HermesCorrelationImg(Hermes_H Hermes, UInt16 counter);
942
988 DllSDKEExport HermesReturn HermesSaveCorrelationImg(Hermes_H Hermes, char* filename);
989
998 DllSDKEExport HermesReturn HermesResetOverilluminationProtection(Hermes_H Hermes);
999
1002 #ifdef _MPD_DEV_
1003 #include "MPD_util.h"
1004 #endif
1005
1007 #ifdef __cplusplus
1008 }
1009 #endif
1010

```



```
1011 #endif //__Hermes_SDK_h__
```

## Chapter 6

# Example Documentation

### 6.1 SDK\_Example.c

```
/*
#####

Copyright 2022 Micro-Photon-Devices s.r.l.

SOFTWARE PRODUCT: Hermes_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 "Hermes_SDK.h"
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>
#include <string.h>
#ifdef __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 //
    Hermes_H Hermes= NULL;
    UInt16* Img= NULL,hist[65535],AppliedDT=0;
    int loop = 1;
    int i_c;
    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;
    HermesReturn error = OK;
    FILE* f= NULL;
    time_t start,stop,leap;
    char *menu_strings[] = { "Live mode: write on stdout 10 images", //a
                             "Holdoff: mean number of photons at different holdoff values", //b
                             "Dead-time corrector improves the image quality", //c
                             "Background subtraction: 2 output files, with and without BG", //d
                             "Gate: 1000 images normal, 1000 images with gate 3 ns shift 5 ns",
                             //e
                             "Synchronization output", //f
                             "Background statistics", //g
                             "Gate: calibrate the length of the gate signal", //h
                             "Triple gate mode", //i
                             "Read and write test images", //j
                             "Continuous acquisition on file", //k
                             "Continuous acquisition in memory", //l
                             "Reset overcurrent protection" //m
                           };

    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 //

    HermesConstr(&Hermes, Normal, "");
    while (loop) {
        // ----- MENU -----

        printf("\n*****\n");
        printf(" Hermes Test program\n");
        printf("*****\n");
        for (i = 0; i < sizeof(menu_strings) / sizeof(char*); i++) {
            printf("\t%c) %s\n", i + 'a', menu_strings[i]);
        }
        printf("\tq) Quit\n> ");
        c = getchar();
        getchar();
        if (c >= 'a' && c <= 'a' + sizeof(menu_strings) / sizeof(char*) - 1) {

            printf("*****\n");
            printf("%s\n", menu_strings[c - 'a']);
        }
    }
}

```

```

printf("*****\n");
}
switch (c)
{
case 'a': //Test live mode
    //Hermes parameter setting
    HermesSetCameraPar(Hermes, 100, 30000, 300, 1, Disabled, Disabled, Disabled);
    HermesSetTriggerOutState(Hermes, Frame);
    HermesApplySettings(Hermes);
    HermesLiveSetModeON(Hermes);
    //Acquisition of 10 live images
    for (i = 0; i < 10; i++)
    {
        printf("Image %d:\n", i);
        error = HermesLiveGetImg(Hermes, Img);
        if (error == OK)
            for (j = 0; j < 32; j++)
            {
                for (k = 0; k < 32; k++)
                    printf("%d ", Img[32 * j + k]);
                printf("\n");
            }
        else
            break;
    }
    //Live mode off
    HermesLiveSetModeOFF(Hermes);
    break;
case 'b': //Test dead-time
    //Hermes parameter setting
    HermesSetAdvancedMode(Hermes, Enabled);
    HermesSetCameraPar(Hermes, 1040, 10, 1, 1, Disabled, Disabled, Disabled);
    HermesApplySettings(Hermes);
    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;
        HermesSetDeadTime(Hermes, i);
        HermesApplySettings(Hermes);
        HermesGetDeadTime(Hermes, i, &AppliedDT);
        x[k] = (double)AppliedDT;
        HermesSnapPrepare(Hermes);
        HermesSnapAcquire(Hermes);
        HermesAverageImg(Hermes, Img, 1);
        data[k] = mean_double(Img, 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 'c': //Dead-time corrector effect
    //Hermes parameter setting
    HermesSetCameraPar(Hermes, 4096, 1000, 100, 1, Disabled, Disabled, Disabled);
    HermesSetDeadTime(Hermes, 100);
    //acquisition with DTC on
    printf("Acquire the image using the dead-time correction\n");
    HermesSetDeadTimeCorrection(Hermes, Enabled);
    HermesApplySettings(Hermes);
    //Acquire the BG image first
    printf("\n\nClose the camera shutter and press ENTER...\n");
    getchar();
    HermesSnapPrepare(Hermes);
    HermesSnapAcquire(Hermes);
    //Calculate the average image
    HermesAverageImg(Hermes, 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
    HermesSetBackgroundImg(Hermes, Img);

```

```

HermesSetBackgroundSubtraction(Hermes, Enabled);
HermesApplySettings(Hermes);
//now acquire the image with shutter open
printf("\n\nOpen the camera shutter and press ENTER...\n");
getchar();
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
HermesSaveImgDisk(Hermes, 1, 100, "Im_DeadTimeCorrected", TIFF_NO_COMPRESSION);
printf("The the dead-time corrected image was acquired and stored on the hard disk
succesfully!\n");
//acquisition with DTC off
printf("\n\nAcquire the reference image without the dead-time correction\n");
HermesSetBackgroundSubtraction(Hermes, Disabled);
HermesSetDeadTimeCorrection(Hermes, Disabled);
HermesApplySettings(Hermes);
//Acquire the BG image first
printf("\n\nClose the camera shutter and press ENTER...\n");
getchar();
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
//Calculate the average image
HermesAverageImg(Hermes, 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
HermesSetBackgroundImg(Hermes, Img);
HermesSetBackgroundSubtraction(Hermes, Enabled);
HermesApplySettings(Hermes);
//now acquire the image with shutter open
printf("\n\nOpen the camera shutter and press ENTER...\n");
getchar();
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
HermesSaveImgDisk(Hermes, 1, 100, "Im_DeadTimeReference", TIFF_NO_COMPRESSION);
printf("The the dead-time corrected image was acquired and stored on the hard disk
succesfully!\n");
break;
case 'd': //Test background subtraction
//Hermes parameter setting
HermesSetCameraPar(Hermes, 4096, 1000, 100, 1, Disabled, Disabled, Disabled);
HermesApplySettings(Hermes);
//acquire background image
printf("\n\nClose the camera shutter and press ENTER...\n");
getchar();
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
HermesSaveImgDisk(Hermes, 1, 1, "Bg", TIFF_NO_COMPRESSION);
HermesAverageImg(Hermes, 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
HermesSetBackgroundImg(Hermes, Img);
//acquire image with background subtraction off
printf("Open the camera shutter and press ENTER ...\n");
getchar();
HermesSetBackgroundSubtraction(Hermes, Disabled);
HermesApplySettings(Hermes);
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
HermesSaveImgDisk(Hermes, 1, 1, "Normal", TIFF_NO_COMPRESSION);
//acquire image with background subtraction on
HermesSetBackgroundSubtraction(Hermes, Enabled);
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
HermesSaveImgDisk(Hermes, 1, 1, "BgSubt", TIFF_NO_COMPRESSION);
break;
case 'e': // Test gate
//Hermes parameter setting
HermesSetAdvancedMode(Hermes, Enabled);
HermesSetCameraPar(Hermes, 4096, 1000, 100, 1, Disabled, Disabled, Disabled);
HermesSetDeadTime(Hermes, 100);
HermesApplySettings(Hermes);
//Normal image
printf("Acquiring the reference image ...\n");
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
printf("Save the reference image ...\n");
HermesSaveImgDisk(Hermes, 1, 1000, "GateNormal", TIFF_NO_COMPRESSION);
//gated image
HermesSetGateMode(Hermes, 1, Pulsed);
HermesSetGateValues(Hermes, 10, 15); //Shift -10% of 20 ns --> 120 ns, Length 15% of
20 ns --> 3ns
HermesApplySettings(Hermes);

```

```

printf("Acquiring the gated image ...\n");
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
printf("Save the gated image ...\n");
HermesSaveImgDisk(Hermes, 1, 1000, "GatePulsed", TIFF_NO_COMPRESSION);
break;
case 'f': // Test synchronization output
//Hermes parameter setting
HermesSetAdvancedMode(Hermes, Enabled);
HermesSetCameraPar(Hermes, 8192, 0xFFFF, 5, 1, Disabled, Disabled, Disabled);
HermesSetDeadTime(Hermes, 100);
//no output
HermesSetTriggerOutState(Hermes, None);
HermesApplySettings(Hermes);
HermesLiveSetModeON(Hermes);
printf("\n\nNo output ...\n");
printf("Press ENTER to continue\n");
getchar();
HermesLiveSetModeOFF(Hermes);
//gate clock output
HermesSetTriggerOutState(Hermes, Gate_Clk);
HermesSetCameraPar(Hermes, 8192, 0xFFFF, 5, 1, Disabled, Disabled, Disabled);
HermesApplySettings(Hermes);
HermesLiveSetModeON(Hermes);
printf("\n\nGate synchronization signal ...\n");
printf("Press ENTER to continue\n");
getchar();
HermesLiveSetModeOFF(Hermes);
//frame sync output
HermesSetTriggerOutState(Hermes, Frame);
HermesSetCameraPar(Hermes, 8192, 0xFFFF, 5, 1, Disabled, Disabled, Disabled);
HermesApplySettings(Hermes);
HermesLiveSetModeON(Hermes);
printf("\n\nFrame synchronization signal ...\n");
printf("Press ENTER to continue\n");
getchar();
HermesLiveSetModeOFF(Hermes);
printf("\n\nWait for the trigger input ...\n");
//trigger in enabled
HermesSetSyncInState(Hermes, Enabled, 0);
HermesSetCameraPar(Hermes, 4096, 10, 2, 1, Disabled, Disabled, Disabled);
HermesApplySettings(Hermes);
HermesSnapPrepare(Hermes);
while (trig != 1)
    HermesIsTriggered(Hermes, &trig);
printf("Trigger signal received!\n");
break;
case 'g': // 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.

//Hermes parameter setting
HermesSetAdvancedMode(Hermes, Enabled);
HermesSetCameraPar(Hermes, 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);
    HermesSetDeadTime(Hermes, i);
    HermesApplySettings(Hermes);
    HermesSnapPrepare(Hermes);
    HermesSnapAcquire(Hermes);
    HermesAverageImg(Hermes, 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);
HermesSaveImgDisk(Hermes, 1, 100, "DCR_50ns_movie", HERMES_FILEFORMAT);
HermesSaveAveragedImgDisk(Hermes, 1, "DCR_50ns_averaged", HERMES_FILEFORMAT, 1);
HermesSaveAveragedImgDisk(Hermes, 1, "DCR_50ns_averaged", TIFF_NO_COMPRESSION, 1);
break;
case 'h': // Calibrate gate
//Hermes parameter setting
HermesSetAdvancedMode(Hermes, Enabled);

```

```

HermesSetCameraPar(Hermes, 1040 * 3, 10000, 5, 3, Disabled, Disabled, Disabled);
HermesSetDeadTime(Hermes, 100);
HermesApplySettings(Hermes);
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 Hermes 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
HermesSetGateMode(Hermes, 1, Continuous);
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
HermesAverageImg(Hermes, data, 1);
gateoff = mean_double(data, 2048);
printf("Gate OFF counts: %.2f\n", gateoff);
fprintf(f, "Gate OFF counts: %.2f\n", gateoff);
HermesSetGateMode(Hermes, 1, Pulsed);
HermesApplySettings(Hermes);
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)
{
    HermesSetGateValues(Hermes, 0, i);
    HermesApplySettings(Hermes);
    HermesSnapPrepare(Hermes);
    HermesSnapAcquire(Hermes);
    HermesAverageImg(Hermes, data, 1);
    y[i] = mean_double(data, 2048);
    y[i + 101] = y[i] / gateoff * 100; //actual gate width calculated from
    photon counts
    x[i] = (double)i;
    printf("%3.0f\t\t%.2f\t\t%.2f\n", x[i], y[i], y[i + 101]);
    fprintf(f, "%.0f %.2f %.2f\n", x[i], y[i], y[i + 101]);
}
fclose(f);
printf("\n");
break;
case 'i': // Triple gate mode
//Hermes parameter setting
fname = (char*)calloc(256, sizeof(char));
HermesSetAdvancedMode(Hermes, Enabled);
HermesSetCameraPar(Hermes, 1040 * 3, 1000, 20, 3, Disabled, Disabled, Disabled);
HermesSetDeadTime(Hermes, 45);
HermesApplySettings(Hermes);
printf("Expose the Hermes camera to a luminuos 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;
}
HermesSetTripleGate(Hermes, Disabled, -400, 10, 10, 10, 0, 100);
HermesSetTriggerOutState(Hermes, Gate_Clk);
HermesApplySettings(Hermes);
HermesSnapPrepare(Hermes);
HermesSnapAcquire(Hermes);
for (counter = 1; counter <= 3; counter++)
{
    HermesAverageImg(Hermes, 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 = HermesSetTripleGate(Hermes, Enabled, j, 10, 10, 10, 0, 100);
    if (error < 0) break;
    HermesApplySettings(Hermes);
    HermesSnapPrepare(Hermes);
    HermesSnapAcquire(Hermes);
    for (counter = 1; counter <= 3; counter++)

```

```

        {
            HermesAverageImg(Hermes, data, counter);
            counts[counter - 1] = mean_double(data, 2048);
        }
        printf("%d\t%.4f\t%.4f\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%.4f\t%.4f\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 'j': // Save and Read images
    //Hermes parameter setting
    HermesSetAdvancedMode(Hermes, Enabled);
    HermesSetCameraPar(Hermes, 1040, 20, 2, 1, Disabled, Disabled, Disabled);
    HermesSetDeadTime(Hermes, 100);
    HermesApplySettings(Hermes);
    //acquiring and saving images
    printf("Acquiring 20 images and save them on the hard drive in the Hermes file
format.\n");
    HermesSnapPrepare(Hermes);
    HermesSnapAcquire(Hermes);
    HermesSaveImgDisk(Hermes, 1, 20, "Test20_Im", HERMES_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();
    HermesReadHermesFileFormatImage("Test20_Im.Hermes", 1, 1, Img, header);
    printf("Rows: %d, Columns: %d\n", header[100], header[101]);
    for (i = 1; i <= 20; i++)
    {
        HermesReadHermesFileFormatImage("Test20_Im.Hermes", i, 1, Img, header);
        printf("Image %d, pixel value = %hu\n", i, Img[0]);
    }
    break;
case 'k': //Continuous acquisition, number of integration frames selection.
    //Hermes parameter setting
    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));
    HermesSetCameraPar(Hermes, 1050, 1, integFrames, 1, Disabled, Disabled, Disabled);
    HermesSetDeadTime(Hermes, 100);
    HermesApplySettings(Hermes);
    //acquire images
    printf("Continuous acquisition will be started and 10 memory dumps performed.\n");
    printf("Press ENTER to start continuous acquisition...\n");
    getchar();
    getchar();
    HermesContAcqToFileStart(Hermes, "contacq");
    for (i = 1; i < 10; i++)
    {
        if (HermesContAcqToFileGetMemory(Hermes, &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;
    }
    HermesContAcqToFileStop(Hermes);
    printf("Acquisition saved to contacq.Hermes.\n");
    break;
case 'l': //Continuous acquisition in memory, number of integration frames selection.
    //Hermes parameter setting
    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));
    HermesSetCameraPar(Hermes, 1050, 1, integFrames, 1, Disabled, Disabled, Disabled);
    HermesSetDeadTime(Hermes, 100);
    HermesApplySettings(Hermes);
    //acquire images
    printf("Continuous acquisition will be started and 10 memory dumps performed.\n");
    printf("Press ENTER to start continuous acquisition...\n");
    getchar();
    getchar();
    HermesContAcqToMemoryStart(Hermes);
    i = 1;
    start = clock();
    leap = start;

```



```

        for (i = 1; i<11; i++)
        {
            if (HermesContAcqToMemoryGetBuffer(Hermes, &read_bytes, &buffer) == OK)
            {
                total_bytes = total_bytes + read_bytes;
                stop = clock();
                printf("Acquired %.2f MB in %d readout operation, last rate
%.2fMB/s, average rate %.2fMB/s\n", total_bytes / 1000000.0, i, (read_bytes / 1000000.0 / ((stop -
leap) / (float)CLOCKS_PER_SEC)), (total_bytes / 1000000.0 / ((stop - start) /
(float)CLOCKS_PER_SEC)));
                leap = clock();
                SLEEP(1 * MILLIS);
            }
            else
            {
                printf("ERROR!");
                break;
            }
        }
        HermesContAcqToMemoryStop(Hermes);
        break;
    case 'm': //Reset overcurrent protection
        HermesResetOverilluminationProtection(Hermes);
        break;
    case 'q':
        loop = 0;
        break;
    }
}

// Destructors
//

if(Hermes)
    HermesDestr(Hermes);
Hermes = NULL;
free(Img);
free(Imgd);
free(data);
free(y);
free(x);
free(z);
printf("Press ENTER to continue\n");
getchar();
return 0;
}

```

# Index

## Acquisition methods, [28](#)

- [HermesContAcqToFileGetMemory, 29](#)
- [HermesContAcqToFileStart, 29](#)
- [HermesContAcqToFileStop, 30](#)
- [HermesContAcqToMemoryGetBuffer, 31](#)
- [HermesContAcqToMemoryStart, 32](#)
- [HermesContAcqToMemoryStop, 32](#)
- [HermesLiveGetImg, 33](#)
- [HermesLiveSetModeOFF, 33](#)
- [HermesLiveSetModeON, 34](#)
- [HermesSnapAcquire, 35](#)
- [HermesSnapGetImageBuffer, 35](#)
- [HermesSnapGetImgPosition, 36](#)
- [HermesSnapPrepare, 37](#)

## Additional methods, [38](#)

- [HermesAverageImg, 38](#)
- [HermesCorrelationImg, 39](#)
- [HermesReadHermesFileFormatImage, 39](#)
- [HermesResetOverilluminationProtection, 40](#)
- [HermesSaveAveragedImgDisk, 41](#)
- [HermesSaveCorrelationImg, 41](#)
- [HermesSaveFlimDisk, 42](#)
- [HermesSaveImgDisk, 43](#)
- [HermesSetCorrelationMode, 45](#)
- [HermesStDevImg, 46](#)

## Advanced

- [Hermes-SDK custom Types, 5](#)

## CAMERA\_NOT\_POWERING\_UP

- [Hermes-SDK custom Types, 6](#)

## CameraMode

- [Hermes-SDK custom Types, 4](#)

## Coarse

- [Hermes-SDK custom Types, 5](#)

## COMMUNICATION\_ERROR

- [Hermes-SDK custom Types, 6](#)

## Constructor, destructor and error handling, [8](#)

- [HermesConstr, 8](#)
- [HermesDestr, 8](#)
- [PrintErrorCode, 9](#)

## Continuous

- [Hermes-SDK custom Types, 5](#)

## CorrelationMode

- [Hermes-SDK custom Types, 5](#)

## Disabled

- [Hermes-SDK custom Types, 7](#)

## EMPTY\_BUFFER

- [Hermes-SDK custom Types, 6](#)

## Enabled

- [Hermes-SDK custom Types, 7](#)

## FIRMWARE\_NOT\_COMPATIBLE

- [Hermes-SDK custom Types, 6](#)

## Frame

- [Hermes-SDK custom Types, 7](#)

## Gate\_Clk

- [Hermes-SDK custom Types, 7](#)

## GateMode

- [Hermes-SDK custom Types, 5](#)

## Get methods, [23](#)

- [HermesDeviceInfo, 24](#)
- [HermesGetDeadTime, 24](#)
- [HermesGetGateShift, 25](#)
- [HermesGetGateWidth, 25](#)
- [HermesGetSerial, 26](#)
- [HermesGetVersion, 26](#)
- [HermesIs16Bit, 27](#)
- [HermesIsTriggered, 27](#)

## Hermes-SDK custom Types, [4](#)

- [Advanced, 5](#)
- [CAMERA\\_NOT\\_POWERING\\_UP, 6](#)
- [CameraMode, 4](#)
- [Coarse, 5](#)
- [COMMUNICATION\\_ERROR, 6](#)
- [Continuous, 5](#)
- [CorrelationMode, 5](#)
- [Disabled, 7](#)
- [EMPTY\\_BUFFER, 6](#)
- [Enabled, 7](#)
- [FIRMWARE\\_NOT\\_COMPATIBLE, 6](#)
- [Frame, 7](#)
- [Gate\\_Clk, 7](#)
- [GateMode, 5](#)
- [HERMES\\_FILEFORMAT, 7](#)
- [HERMES\\_MEMORY\\_FULL, 6](#)
- [HermesReturn, 5](#)
- [INVALID\\_NIMG\\_CORRELATION, 6](#)
- [INVALID\\_OP, 6](#)
- [Linear, 5](#)
- [MISSING\\_DLL, 6](#)
- [MultiTau, 5](#)
- [None, 7](#)
- [Normal, 5](#)
- [NOT\\_EN\\_MEMORY, 6](#)
- [NULL\\_POINTER, 6](#)
- [OK, 6](#)

- OUT\_OF\_BOUND, [6](#)
- OutFileFormat, [6](#)
- PERSISTING\_TOO\_MUCH\_LIGHT, [6](#)
- POWER\_SUPPLY\_ERROR, [6](#)
- Pulsed, [5](#)
- State, [7](#)
- TIFF\_NO\_COMPRESSION, [7](#)
- TOO\_MUCH\_LIGHT, [6](#)
- TriggerMode, [7](#)
- UNABLE\_CREATE\_FILE, [6](#)
- UNABLE\_READ\_FILE, [6](#)
- USB\_DEVICE\_NOT\_RECOGNIZED, [6](#)
- HERMES\_FILEFORMAT
  - Hermes-SDK custom Types, [7](#)
- HERMES\_MEMORY\_FULL
  - Hermes-SDK custom Types, [6](#)
- Hermes\_SDK.h, [48](#)
  - MAX\_DEAD\_TIME, [50](#)
  - MIN\_DEAD\_TIME, [50](#)
- HermesApplySettings
  - Set methods, [10](#)
- HermesAveragedImg
  - Additional methods, [38](#)
- HermesConstr
  - Constructor, destructor and error handling, [8](#)
- HermesContAcqToFileGetMemory
  - Acquisition methods, [29](#)
- HermesContAcqToFileStart
  - Acquisition methods, [29](#)
- HermesContAcqToFileStop
  - Acquisition methods, [30](#)
- HermesContAcqToMemoryGetBuffer
  - Acquisition methods, [31](#)
- HermesContAcqToMemoryStart
  - Acquisition methods, [32](#)
- HermesContAcqToMemoryStop
  - Acquisition methods, [32](#)
- HermesCorrelationImg
  - Additional methods, [39](#)
- HermesDestr
  - Constructor, destructor and error handling, [8](#)
- HermesDeviceInfo
  - Get methods, [24](#)
- HermesGetDeadTime
  - Get methods, [24](#)
- HermesGetGateShift
  - Get methods, [25](#)
- HermesGetGateWidth
  - Get methods, [25](#)
- HermesGetSerial
  - Get methods, [26](#)
- HermesGetVersion
  - Get methods, [26](#)
- HermesIs16Bit
  - Get methods, [27](#)
- HermesIsTriggered
  - Get methods, [27](#)
- HermesLiveGetImg
  - Acquisition methods, [33](#)
- HermesLiveSetModeOFF
  - Acquisition methods, [33](#)
- HermesLiveSetModeON
  - Acquisition methods, [34](#)
- HermesReadHermesFileFormatImage
  - Additional methods, [39](#)
- HermesResetOverilluminationProtection
  - Additional methods, [40](#)
- HermesReturn
  - Hermes-SDK custom Types, [5](#)
- HermesSaveAveragedImgDisk
  - Additional methods, [41](#)
- HermesSaveCorrelationImg
  - Additional methods, [41](#)
- HermesSaveFlimDisk
  - Additional methods, [42](#)
- HermesSaveImgDisk
  - Additional methods, [43](#)
- HermesSetAdvancedMode
  - Set methods, [11](#)
- HermesSetBackgroundImg
  - Set methods, [11](#)
- HermesSetBackgroundSubtraction
  - Set methods, [12](#)
- HermesSetCameraPar
  - Set methods, [12](#)
- HermesSetCameraParSubArray
  - Set methods, [14](#)
- HermesSetCoarseGateValues
  - Set methods, [15](#)
- HermesSetCorrelationMode
  - Additional methods, [45](#)
- HermesSetDeadTime
  - Set methods, [16](#)
- HermesSetDeadTimeCorrection
  - Set methods, [16](#)
- HermesSetDualGate
  - Set methods, [17](#)
- HermesSetFlimPar
  - Set methods, [18](#)
- HermesSetFlimState
  - Set methods, [18](#)
- HermesSetGateMode
  - Set methods, [20](#)
- HermesSetGateValues
  - Set methods, [20](#)
- HermesSetSyncInState
  - Set methods, [21](#)
- HermesSetTriggerOutState
  - Set methods, [22](#)
- HermesSetTripleGate
  - Set methods, [22](#)
- HermesSnapAcquire
  - Acquisition methods, [35](#)
- HermesSnapGetImageBuffer
  - Acquisition methods, [35](#)
- HermesSnapGetImgPosition

- Acquisition methods, [36](#)
- HermesSnapPrepare
  - Acquisition methods, [37](#)
- HermesStDevImg
  - Additional methods, [46](#)
- INVALID\_NIMG\_CORRELATION
  - Hermes-SDK custom Types, [6](#)
- INVALID\_OP
  - Hermes-SDK custom Types, [6](#)
- Linear
  - Hermes-SDK custom Types, [5](#)
- MAX\_DEAD\_TIME
  - Hermes\_SDK.h, [50](#)
- MIN\_DEAD\_TIME
  - Hermes\_SDK.h, [50](#)
- MISSING\_DLL
  - Hermes-SDK custom Types, [6](#)
- MultiTau
  - Hermes-SDK custom Types, [5](#)
- None
  - Hermes-SDK custom Types, [7](#)
- Normal
  - Hermes-SDK custom Types, [5](#)
- NOT\_EN\_MEMORY
  - Hermes-SDK custom Types, [6](#)
- NULL\_POINTER
  - Hermes-SDK custom Types, [6](#)
- OK
  - Hermes-SDK custom Types, [6](#)
- OUT\_OF\_BOUND
  - Hermes-SDK custom Types, [6](#)
- OutFileFormat
  - Hermes-SDK custom Types, [6](#)
- PERSISTING\_TOO\_MUCH\_LIGHT
  - Hermes-SDK custom Types, [6](#)
- POWER\_SUPPLY\_ERROR
  - Hermes-SDK custom Types, [6](#)
- PrintErrorCode
  - Constructor, destructor and error handling, [9](#)
- Pulsed
  - Hermes-SDK custom Types, [5](#)
- Set methods, [9](#)
  - HermesApplySettings, [10](#)
  - HermesSetAdvancedMode, [11](#)
  - HermesSetBackgroundImg, [11](#)
  - HermesSetBackgroundSubtraction, [12](#)
  - HermesSetCameraPar, [12](#)
  - HermesSetCameraParSubArray, [14](#)
  - HermesSetCoarseGateValues, [15](#)
  - HermesSetDeadTime, [16](#)
  - HermesSetDeadTimeCorrection, [16](#)
  - HermesSetDualGate, [17](#)
  - HermesSetFlimPar, [18](#)
  - HermesSetFlimState, [18](#)
  - HermesSetGateMode, [20](#)
  - HermesSetGateValues, [20](#)
  - HermesSetSyncInState, [21](#)
  - HermesSetTriggerOutState, [22](#)
  - HermesSetTripleGate, [22](#)
- State
  - Hermes-SDK custom Types, [7](#)
- TIFF\_NO\_COMPRESSION
  - Hermes-SDK custom Types, [7](#)
- TOO\_MUCH\_LIGHT
  - Hermes-SDK custom Types, [6](#)
- TriggerMode
  - Hermes-SDK custom Types, [7](#)
- UNABLE\_CREATE\_FILE
  - Hermes-SDK custom Types, [6](#)
- UNABLE\_READ\_FILE
  - Hermes-SDK custom Types, [6](#)
- USB\_DEVICE\_NOT\_RECOGNIZED
  - Hermes-SDK custom Types, [6](#)