**EinScan SDK User Manual**

**Version 1.0**

**Shining3d**

**Contents**

[Data Structures and Macro Definitions 4](#_Toc22045)

[Data Structures 4](#_Toc12424)

[Data Initialization 4](#_Toc17774)

[Scan Parameters 4](#_Toc16889)

[Image Data Structure 5](#_Toc25489)

[Video Frame Data Structure 6](#_Toc7762)

[The Calibration Status Data Structure 6](#_Toc8164)

[3D Point Data Structure 8](#_Toc5218)

[Point Cloud Data Structure 8](#_Toc24378)

[Transformation Matrix RT 9](#_Toc2537)

[Texture Index Coordinate 10](#_Toc7082)

[Tri patch Data Structure 10](#_Toc22190)

[Mesh Data Structure 11](#_Toc9149)

[Mesh Data Process Parameters 12](#_Toc5485)

[Macro Definitions 13](#_Toc31792)

[The Definitions of Initialization Type 13](#_Toc285)

[The Notification Events of Calibration 13](#_Toc16550)

[The Constants of Calibration Type 13](#_Toc26960)

[The Constants of Calibration Capture Status 13](#_Toc758)

[The Constants of Calibration Calculation Status 13](#_Toc24261)

[The Macro Definitions of Scan Notification Events 14](#_Toc26119)

[Alignment Mode 15](#_Toc357)

[Mesh Mode 15](#_Toc22604)

[Mesh Resulotion 15](#_Toc25407)

[The Definitions of Return Values 15](#_Toc13153)

[Handle Scan Interface Definitions 16](#_Toc20538)

[The Open Device Handle 16](#_Toc29021)

[Close Device Handle 17](#_Toc2208)

[Parameters Configuration 18](#_Toc28104)

[Get the Camera Brightness Value Range 18](#_Toc11432)

[Set Camera Brightness 18](#_Toc5995)

[Get Current Camera Brightness Level 19](#_Toc18858)

[Import Global Reference Point 20](#_Toc24899)

[Set Scan Parameters 20](#_Toc15784)

[Get Scan Parameters 21](#_Toc29788)

[Pro Calibration 21](#_Toc3140)

[Register Callback Function of Calibration Events Notification 22](#_Toc29042)

[Callback Function of Calibration Events 22](#_Toc22632)

[Capture Calibration Images 24](#_Toc18550)

[Skip the Current Calibration Phase 25](#_Toc4136)

[Handle Rapid Scan 26](#_Toc7539)

[Register Callback Function of Scan Events Notification 26](#_Toc1183)

[Callback Function of Scan Events 27](#_Toc11623)

[Start to Scan 29](#_Toc13788)

[Stop Scanning 30](#_Toc31493)

[Pause Scanning 30](#_Toc6941)

[Delete Scan Data 31](#_Toc2696)

[Get Current Point Cloud Data 31](#_Toc12771)

[Get Merged Whole Point Cloud Data 32](#_Toc27886)

[Get Current Reference Points 33](#_Toc12966)

[Get ALL of Reference Points 33](#_Toc7778)

[Delete Part of The Scan Data 34](#_Toc52)

[Handle HD Scan 36](#_Toc6299)

[Mesh Process Interface Definitions 36](#_Toc20224)

[Get The Current Mesh Data 36](#_Toc16754)

[Use Procedure 37](#_Toc11243)

[Calibrate Procedure 37](#_Toc24327)

[HD and Rapid Scan Procedure 38](#_Toc1611)

[Fix Scan Interface Definitions 39](#_Toc17961)

[The Open Device Handle 39](#_Toc16191)

[Close Device Handle 40](#_Toc23774)

[Register Callback Function of Display video image 40](#_Toc4241)

[Callback Function of Calibration Events 41](#_Toc6711)

[Get the Camera Brightness Max Value 42](#_Toc28596)

[Set Camera Brightness 42](#_Toc13793)

[Set Texture Scan Parameters 43](#_Toc30110)

[Import Global Mark Point 43](#_Toc3337)

[The Code Scan of turntable 44](#_Toc23185)

[The Mark Scan of turntable 45](#_Toc12172)

[The Free Scan 46](#_Toc19730)

[The Open Turntable Device 47](#_Toc24464)

[Close Turntable Device 48](#_Toc26005)

[Run A Specific Angle 48](#_Toc24310)

[Check The State Of Turntable 49](#_Toc17588)

[Use Procedure 50](#_Toc325)

# Data Structures and Macro Definitions

## **Data Structures**

### **Data Initialization**

**SN3D\_INIT\_DATA**

|  |
| --- |
| **typedef struct tag SN3D\_INIT\_DATA**  **{**  char\*  device\_type;  wchar\_t config\_path[SN3D\_MAX\_PATH];  unsigned char reserved[256];  **} SN3D\_INIT\_DATA, \*LPSN3D\_INIT\_DATA;** |

**Members**

device\_type

The type of device to control different devices. Reserved word.

config\_path

The path of configuration file. Reserved word.

reserved

Reserved word.

R**emarks**

### **Scan Parameters**

**SN3D\_SCAN\_PARAM**

|  |
| --- |
| **typedef struct tag SCAN\_PARAM**  **{**  double resolution;  int flag\_texture;  int align\_mode;  **} SN3D\_SCAN\_PARAM, \*LPSN3D\_CAN\_PARAM;** |

**Members**

resolution

Space resolution. Value: 0.2-3.0 for HD scan;0.5-3.0 for Rapid Scan .

flag\_texture

Scan with color setting, 0 : scan without color ,1 : scan with color.

align\_mode

Alignment mode. Reference: [Alignment mode macro definition](#_Alignment_Mode_1) .

R**emarks**

### **Image Data Structure**

**SN3D\_IMAGE\_DATA**

|  |
| --- |
| typedef struct tag **SN3D\_**IMAGE\_DATA  {  int width;  int height ;  int channel ;  int length ;  unsigned char\* data;  } **SN3D\_IMAGE\_DATA**, **\*LPSN3D\_IMAGE\_DATA**; |

**Members**

width

The width of image.

height

The height of image.

channel

The channels of image.

length

width\*height；The length of image, the size is width\*height\*channel .

data

The data of image.

R**emarks**

### **Video Frame Data Structure**

**SN3D\_VIDEO\_FRAME**

|  |
| --- |
| typedef struct tag **SN3D\_VIDEO\_FRAME**  {  int id ;  int fps ;  unsigned long long stamp;  IMAGE\_DATA video\_data;  } **SN3D\_VIDEO\_FRAME**, **\*LPSN3D\_VIDEO\_FRAME**; |

**Members**

id

The ID of cameras. 0 : Left camera , 1: Right camera , 2: Color camera.

fps

Frame rate. Reserved word.

stamp

The time stamp of image. Reserved word.

video\_data

The data of image.。

**Remarks**

### **The Calibration Status Data Structure**

**SN3D\_STATE\_CALIBRATE**

|  |
| --- |
| typedef struct tag**SN3D\_STATE\_CALIBRATE**  {  int current\_calibrate;  int distance\_indicate;  int current\_group ;  int snap\_state;  int compute\_state;  } **SN3D\_STATE\_CALIBRATE, \*LPSN3D\_TATE\_CALIBRATE;** |

**Members**

current\_calibrate

Reference:[Calibration type macro definition](#_The_Constants_of).

0：  Scanner calibration. The work flow is as follows.

1. Flatten the calibration board.
2. Project the cross image into the white rectangle of calibration board .
3. Capture five images from different distances (reference: distance\_indicate) by moving the scanner from bottom to top.
4. Change calibration board position, and repeat four more times according to B and C .
5. Capture finished and start to calibrate.

1 ： HD calibration.

Capture 12 images from different distances by moving scanner from bottom to top.

2 ： White Balance.

Capture the image which is the sixth position by moving scanner from bottom to top to run the white balance.

3 ： Calibration finished.

distance\_indicate

The work distance between scanner and calibration board, bigger value, further distance.

The ranges are different depends on calibration type.

Scanner calibration: ranges:0-4. Each group must appear once for each value, for a total of 5 groups.

HD calibration: ranges:0-11.Each value must appear once.

White balance:ranges :0-11.Calibration is successful when value is 6.

-1: Invalid data。

current\_group

-1: Invalid data；1-5 means which step is running in scanner calibration phase.

snap\_state

Reference :[The Constants of Calibration Capture Status](#_The_Constants_of_1).

-1 : Invalid data, 0: Capturing, 1 : Capture finished.

compute\_state

Reference :[The Constants of Calibration Calculation Status](#_The_Constants_of_2).

-1: Invalid data；0: Calculating, 1: Calculation success, 2: Calculation failure.

**Remarks**

### **3D Point Data Structure**

**SN3D\_POINT\_DATA**

|  |
| --- |
| typedef struct tag **SN3D\_POINT\_DATA**  {  int id;  float x;  float y;  float z;  } **SN3D\_POINT\_DATA**, **\*LPSN3D\_POINT\_DATA**; |

**Members**

id

The ID of data object, Reserved word.

x

X coordinate value.

y

Y coordinate value.

z

Z coordinate value.

**Remarks**

### **Point Cloud Data Structure**

**SN3D\_CLOUD\_POINT**

|  |
| --- |
| typedef struct tag **SN3D\_CLOUD\_POINT**  {  int id;  int vertex\_count ;  [POINT\_DATA](#_3D_Point_Data) \*vertex\_data;  int norma\_count ;  [POINT\_DATA](#_3D_Point_Data) \*norma\_data;  int vertex\_color\_count ;  [POINT\_DATA](#_3D_Point_Data) \* vertex\_color \_data;  } **SN3D\_CLOUD\_POINT**, **\*LPSN3D\_CLOUD\_POINT**; |

**Members**

id

The ID of data object, Reserved word.

vertex\_count

The number of vertex .

vertex\_data

The data of vertex.

norma\_count

The number of vertex normal.

norma\_data

The data of vertex normal.

vertex\_ color \_count

The number of vertex color.

vertex\_color\_data

The data of vertex color.

**Remarks**

### **Transformation Matrix RT**

**SN3D\_SCANNER\_RT**

|  |
| --- |
| typedef struct tag **SN3D\_SCANNER\_RT**  {  float rotate[9];  float trans[3];  } **SN3D\_SCANNER\_RT**, **\*LPSN3D\_SCANNER\_RT**; |

**Members**

rotate

Rotation Matrix.

trans

Translation Matrix.

**Remarks**

### **Texture Index Coordinate**

**SN3D\_UVCOORD**

|  |
| --- |
| typedef struct tag **SN3D\_UVCOORD**  {  int uu;  int vv;  } **SN3D\_UVCOORD**, **\*SN3D\_UVCOORD**; |

**Members**

uu

The index of the texture coordinates:X-axis

vv

The index of the texture coordinates:Y-axis

**Remarks**

### **Tri patch Data Structure**

**SN3D\_TRI\_FACE**

|  |
| --- |
| typedef struct tag **SN3D\_TRI\_FACE**  {  int vid[3];  } **SN3D\_TRI\_FACE**, **\*LSN3D\_TRI\_FACE**; |

**Members**

vid[3]

Three vertex indic。

**Remarks**

### **Mesh Data Structure**

**SN3D\_TRI\_MESH**

|  |
| --- |
| typedef struct tag **SN3D\_TRI\_MESH**  {  int id;  int vertex\_count ;  [POINT\_DATA](#_点数据) \*vertex\_data; [POINT\_DATA](#_点数据) \*norma\_data;  [POINT\_DATA](#_点数据) \*vertex\_color \_data;  int face\_count ;  SN3D\_TRI\_FACE \*face\_ids;  SN3D\_UVCOORDtex\_uuvv;  IMAGE\_DATA tex\_data;  }**SN3D\_TRI\_MESH**, **\*LSN3D\_TRI\_MESH**; |

**Members**

id

The ID of data object, Reserved word.

vertex\_count

The number of vertex .

vertex\_data

The data of vertex.

norma\_data

The data of vertex normal, Reserved word.

vertex\_color\_data

The data of vertex color, Reserved word.

face\_count

The number of tri.。

face\_ids

the vertex index of triangle

tex\_uuvv

The index of the texture coordinates, Reserved word.

tex\_data

Texture image data, Reserved word.

### **Mesh Data Process Parameters**

**SN3D\_MESH\_RPOCESS\_PARAM**

|  |
| --- |
| typedef struct tag **SN3D\_MESH\_PROCESS\_PARAM**  {  int mesh\_type;  int mesh\_resolution;  double simplification\_ratio;  int smooth\_flag;  int sharpen\_flag;  int mark\_point\_fill\_hole\_flag;  } **SN3D\_MESH\_PROCESS\_PARAM**, **\*LPSN3D\_MESH\_PROCESS\_PARAM**; |

**Members**

mesh\_type

Mesh type

mesh\_resolution

mesh resolution, affect the scan data’s details

simplification\_ratio

simplification ratio，simplify the scan data

smooth\_flag

Smooth: 0 forbidden, 1enable

sharpen\_flag

Sharpen: 0 forbidden, 1enable

mark\_point\_fill\_hole\_flag

Mark point fill hole: 0 forbidden, 1enable

plaint\_fill\_hole\_flag

fill hole: 0 forbidden, 1 enable

plaint\_fill\_hole\_perimeter

fill hole perimeter, suggest perimeter is 10--100 , unit is mm.

**Remarks**

The fill hole parameters are only significant when choose the unwatertight, if choose the watertight the parameters will be ignored.

## **Macro Definitions**

### **The Definitions of Initialization Type**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_INIT\_CALIBRATE | 0x00000000 | Calibration initialization |
| SN3D\_INIT\_RAPIDSCAN | 0x00000001 | Rapid scan initialization |
| SN3D\_INIT\_HD\_SCAN | 0x00000002 | HD scan initialization |
| SN3D\_INIT\_FIX\_SCAN | 0x00000003 | Fix scan initialization |

### **The Notification Events of Calibration**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_VIDEO\_IMAGE\_DATA\_READY | 6 | The notification of image data |
| SN3D\_CALIBRATION\_STATE\_DATA | 13 | The notification of Calibration status |

### **The Constants of Calibration Type**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_CALIBRATE\_STATE\_CAMERA | 0 | Scanner calibration |
| SN3D\_CALIBRATE\_STATE\_HD | 1 | HD calibration |
| SN3D\_CALIBRATE\_STATE\_WB | 2 | White Balance |
| SN3D\_CALIBRATE\_STATE\_EXIT | 3 | Calibration done |

### **The Constants of Calibration Capture Status**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_CALIBRATE\_INVALID | -1 | Invalid data , ignore it. |
| SN3D\_CALIBRATE\_SNAP\_STATE\_ON | 0 | Capture calibration image. |
| SN3D\_CALIBRATE\_SNAP\_STATE\_OFF | 1 | Stop capturing calibration image |

### **The Constants of Calibration Calculation Status**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_CALIBRATE\_INVALID | -1 | Invalid data , ignore it. |
| SN3D\_CALIBRATE\_COMPUTING | 0 | Calculating |
| SN3D\_CALIBRATE\_COMPUTE\_SUCCESS | 1 | Calibration success |
| SN3D\_CALIBRATE\_COMPUTE\_FAILED | 2 | Calibration failure |

### **The Macro Definitions of Scan Notification Events**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_VIDEO\_IMAGE\_DATA\_READY | 6 | The Notification of image data |
| SN3D\_DISTANCE\_INDECATE | 7 | The Notification of scan work distance |
| SN3D\_SCANNER\_RT\_READY | 8 | The Notification of scan transformation matrix |
| SN3D\_CURRENT\_MARKPOINT\_DATA\_READY | 9 | The Notification of reference points data |
| SN3D\_CURRENT\_SCAN\_POINT\_CLOUD\_READY | 10 | The Notification of current scan data |
| SN3D\_WHOLE\_SCAN\_POINT\_CLOUD\_READY | 11 | The Notification of the whole point cloud data |
| SN3D\_WHOLE\_MARKPOINT\_DATA\_READY | 12 | The Notification of the whole of reference points data |
| SN3D\_SCANNER\_DOUBLECLICK | 13 | The Notification of ‘Scan’ button is double-clicked |
| SN3D\_SCANNER\_CLICK | 14 | The Notification of ‘Scan’ button is single-clicked |
| SN3D\_SCANNER\_PLUS | 15 | The Notification of “+” button is clicked |
| SN3D\_SCANNER\_SUB | 16 | The Notification of “-” button is clicked |
| SN3D\_START\_REQUESTED | 20 | Start scan request |
| SN3D\_START\_COMPLETED | 21 | Start scan finished |
| SN3D\_PAUSE\_REQUESTED | 22 | Pause scan request |
| SN3D\_PAUSE\_COMPLETED | 23 | Pause scan finished |
| SN3D\_STOP\_REQUESTED | 24 | Stop scan request |
| SN3D\_STOP\_COMPLETED | 25 | Stop scan finished |
| SN3D\_RESET\_REQUESTED | 26 | Reset scan request |
| SN3D\_RESET\_COMPLETED | 27 | Reset scan finished |
| SN3D\_MESH\_DATA \_READY | 28 | The Notification of mesh data |

### **Alignment Mode**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_ALIGN\_MARK\_FEATURE | 2 | Alignment based on features |
| SN3D\_ALIGN\_MODE\_MARK\_POINT | 1 | Alignment based on reference points |
| SN3D\_ALIGN\_MODE\_GLOABL\_MARK\_POINT | 3 | Alignment based on global reference points |
| SN3D\_ALIGN\_TURTABLE\_CODE\_POINT | 4 | Alignment based on turntable reference points |

### **Mesh Mode**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_MESH\_WATERTIGHT | 1 | Wartertight Model |
| SN3D\_MESH\_UNWATERTIGHT | 2 | Unwartertight Model |

### **Mesh Resulotion**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_MESH\_HIGHT | 1 | High detail |
| SN3D\_MESH\_MIDDLE | 2 | Medium detail |
| SN3D\_MESH\_LOW | 3 | Low detail |

### **The Definitions of Return Values**

|  |  |  |
| --- | --- | --- |
| Macro Definitions | The Value of Macro Definitions | Meanings |
| SN3D\_RET\_NOERROR | 0 | No errors happened |
| SN3D\_RET\_PARAM\_ERROR | -1 | Parameters Error |
| SN3D\_RET\_ORDER\_ERROR | -2 | Call order error |
| SN3D\_RET\_TIME\_OUT\_ERROR | -3 | Call timeout |
| SN3D\_RET\_NOT\_SUPPORT\_ERROR | -4 | Unsupported |
| SN3D\_RET\_NO\_DEVICE\_ERROR | -6 | No device is connected |
| SN3D\_RET\_DEVICE\_LICENSE\_ERROR | -7 | The device license error |
| SN3D\_RET\_GPU\_ERROR | -8 | The display card is not compatible |
| SN3D\_RET\_INNER\_ERROR | -9 | The SDK error |
| SN3D\_RET\_NOT\_CALIBRATE\_ERROR | -10 | No Calibration data |
| SN3D\_RET\_LOST\_CONFIG\_FILE\_ERROR | -11 | Configuration file lost |
| SN3D\_RET\_NO\_DATA\_ERROR | -12 | No Point cloud data |
| SN3D\_RET\_LOST\_CALIBRATE\_FILE\_ERROR | -13 | Calibration file lost |
| SN3D\_RET\_NO\_GLOBAL\_MARK\_POINT\_PARAM\_ERROR | -14 | No global mark point data |

# Handle Scan Interface Definitions

## **The Open Device Handle**

sn3d\_initialize

Initialize device handle.

|  |
| --- |
| **void\* sn3d\_Initialize**(  int type,  LPSN3D\_INIT\_DATA init\_data,  void\* & handle  ); |

**Parameters**

type

[in] Initialization type. Reference : [The Definitions of Initialization Type](#_The_Definitions_of)

init\_data

[in] Initialize device parameters. Reference : [SN3D\_INIT\_DATA](#_Data_Initialization_1)

handle

[out] Device handle.

**Return Values**

Reference :[The Definitions of Return Values](#_The_Definitions_of_1).

**Remarks**

1 . Function: Initialize device handle.

2 . Only One device handle can be used for one time.

3. Before calling **sn3d\_initialize** to initialize a device handle, you need to call **[sn3d\_close](#_Close_Device_Handle)** to release current device handle.

## **Close Device Handle**

**sn3d\_close**

Close device handle.

|  |
| --- |
| **int sn3d\_close**(  void\* handle  ); |

**Parameters**

handle

[in] Device handle which is returned by [sn3d\_initialize](#_The_Open_Device).

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1).

**Remarks**

1. Function: Release device handle.

2. Only One device handle can be used for one time.

3. Before calling **[sn3d\_initialize](#_The_Open_Device)** to initialize a device handle, you need to release the current device handle first.

## **Parameters Configuration**

### **Get the Camera Brightness Value Range**

**sn3d\_get\_brightness\_range**

|  |
| --- |
| **int sn3d\_get\_brightness\_range** (  void\* handle,  int& min，  int& max  ) |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device)

min

[out] The minimum value of camera brightness , default is 0. Reserved word.

max

[out] The maximum value of camera brightness.

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1).

**Remarks**

### **Set Camera Brightness**

**sn3d\_set\_** **brightness**

|  |
| --- |
| **int sn3d\_set\_brightness** (  void\* handle ,  int brightness  ) |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize.](#_The_Open_Device)

brightness

[in] The level of brightness , Value ranges reference **[sn3d\_get\_brightness\_range](#_Get_the_Camera)** .

**Return Values**

Reference : [The Definitions of Return Values.](#_The_Definitions_of_1)

**Remarks**

### **Get Current Camera Brightness Level**

**sn3d\_get\_** **brightness**

|  |
| --- |
| **int sn3d\_get\_** **brightness** (  void\* handle,  int& brightness  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize.](#_The_Open_Device)

brightness

[out] The level of brightness , Value ranges reference **[sn3d\_get\_brightness\_range](#_Get_the_Camera)**.

**Return Values**

Reference : [The Definitions of Return Values.](#_The_Definitions_of_1)

**Remarks**

### **Import Global Reference Point**

**sn3d\_import\_global\_mark\_point**

|  |
| --- |
| **int sn3d\_import\_global\_mark\_point** (  **void\* handle,**  **const SN3D\_CLOUD\_POINT& mark\_point,**  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize.](#_The_Open_Device)

mark\_point

[in] Global reference point, at least four points. Reference: **[SN3D\_CLOUD\_POINT](#_Point_Cloud_Data)**

**Return Values**

Reference : [The Definitions of Return Values.](#_The_Definitions_of_1)

**Remarks**

1. It only can be called before calling **[sn3d\_set\_scan\_param](#_Set_Scan_Parameters)** .

### **Set Scan Parameters**

**sn3d\_set\_scan\_param**

|  |
| --- |
| **int sn3d\_set\_scan\_param** (  **void\* handle,**  **LPSN3D\_SCAN\_PARAM param,**  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize.](#_The_Open_Device)

param

[in] Scan configuration parameters. Reference: **[SN3D\_SCAN\_PARAM](#_Scan_Parameters_1)**

**Return Values**

Reference : [The Definitions of Return Values.](#_The_Definitions_of_1)

**Remarks**

1. It is only valid in [SN3D\_INIT\_RAPIDSCAN or SN3D\_INIT\_HD\_SCAN](#_Toc324947433) is set by [sn3d\_initialize](#_打开设备句柄) .

2. It only can be called before calling [sn3d\_start\_scan](#_开始扫描) or after calling sn3d\_abandon\_scan.

3. Calling this function to set scan parameters will clear scan data that are exist.

### **Get Scan Parameters**

**sn3d\_get\_scan\_param**

|  |
| --- |
| **int sn3d\_get\_ param** (  **void\* handle,**  **SN3D\_SCAN\_PARAM& param,**  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize.](#_The_Open_Device)

param

[out] Scan configuration parameters. Reference: **[SN3D\_SCAN\_PARAM](#_Scan_Parameters_1)**

**Return Values**

Reference : [The Definitions of Return Values.](#_The_Definitions_of_1)

**Remarks**

It is only valid in [SN3D\_INIT\_RAPIDSCAN or SN3D\_INIT\_HD\_SCAN](#_Toc324947433) is set by [sn3d\_initialize](#_The_Open_Device) .

## **Pro Calibration**

### **Register Callback Function of Calibration Events Notification**

**sn3d\_regist\_callback**

|  |
| --- |
| **int sn3d\_regist\_callback(**  **void\*** handle,  **sn3d\_callback** call\_back,  **void\*** user\_data  **)** ; |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize.](#_The_Open_Device)

call\_back

[in] The system sends the capture status events to cameras by calling this function.

|  |
| --- |
| **[void \_sn3d\_callback(int,int,void\*,int,void\*)](#_Callback_Function_of)** |

user\_data

[in] Parameters of call\_back. The system sends the parameters to user when the user status has been changed.

**Return Values**

Reference : [The Definitions of Return Values.](#_The_Definitions_of_1)

**Remarks**

The system sends the right events to the registrant after registering this callback function. This callback function can't be registered with scan callback function at the same time.

### **Callback Function of Calibration Events**

**sn3d\_callback**

|  |
| --- |
| **void sn3d\_callback (**  void\* handle,  int event\_type,  int event\_sub\_type,  void\*data  int data\_len  void\* user\_data  **);** |

**Parameters**

handle

[out] Device handle from [sn3d\_initialize.](#_The_Open_Device)

event\_type

[out] Events Type. Reference : [The Notification Events of Calibration.](#_The_Notification_Events)

event\_sub\_type

[out] Events subtype. Not used yet.

data

[out] The data structure pointer matched with event.

|  |  |  |
| --- | --- | --- |
| event\_type | data | Meaning |
| SN3D\_VIDEO\_IMAGE\_DATA\_READY | **[SN3D\_VIDEO\_FRAME](#_Video_Frame_Data)** | The notification of image data. |
| SN3D\_CALIBRATION\_STETE\_\_READY | [SN3D\_STATE\_CALIBRATE](#_The_Calibration_Status) | The Calibration status |

data\_len

[out] The length of data.

user\_data

[out] The parameter needs to be imported into **[sn3d\_regist\_callback.](#_Register_Callback_Function)**

**Return Values**

**NONE**

**Remarks**

1. This function needs to be called after being registered, and be maintained by users.

2. Pay attention to multiple threads processing in outside due to the callback function runs in the inside thread.

3. Can't call other API functions in the callback function.

### **Capture Calibration Images**

sn3d\_start\_calibrate

Notify SDK to capture calibration image and calculate.

|  |
| --- |
| **int sn3d\_s**tart\_calibrate (  **void\*** handle,  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device)

**Return Values**

Reference : [The Definitions of Return Values.](#_The_Definitions_of_1)

**Remarks**

1. SDK includes camera calibration, HD calibration, white balance.

Reference : [The Constants of Calibration Type](#_The_Constants_of)

1. Camera Calibration.
2. SDK enters scanner calibration status waiting for running this function to capture image after initializing.
3. Capture images from 5 different positions, and capture 5 images from different distances at every position.
4. Send capture status by callback while capturing.(Reference:[The Calibration Status Data Structure](#_The_Calibration_Status))After one group capture is done, SDK will stop to capture image and send capture status by callback function, it will be called to capture next group calibration images until the calibration board position has been adjusted.
5. Calculate calibration result after finishing 5 group images capture. Users can call this function or exit calibration status until the calibration is done.
6. If calibration calculation is successful, SDK will enter HD calibration status automatically.
7. If calibration calculation is failed, SDK will still remain in the scanner calibration state, waiting for calling the function again to capture.
8. HD Calibration.

A. Calling this function to start HD calibration after finishing scanner calibration successfully or calling sn3d\_skip\_calibrate to skip scanner calibration status.

B. Need to capture 12 images from different distances, and send the capture status by calling callback function.

C. Call callback function to send calibration result after capture and calculation are done.

D. If calibration calculation is successful, SDK will enter white balance status automatically, or exit the calibration status.

E. If calibration calculation is failed, SDK will still remain in the HD calibration state, waiting for calling the function again to capture.

4. White balance.

A.If the scanner has a color camera, the SDK can enter white balance status directly by finishing HD calibration or calling sn3d\_skip\_calibrate to skip HD calibration status.

B.Capture is done when the callback function received distance value 6 by moving scanner from bottom to top.

C.If calibration calculation is successful, SDK will exit calibration status automatically.

D.If calibration calculation is failed, SDK will still remain in the white balance calibration state, waiting for calling the function again to capture.

5. The calibration exit status.

A. calibration ends, all calibration types are calibrated successfully.

B. the state can only call sn3d\_close to clean device handle.

### **Skip the Current Calibration Phase**

sn3d\_skip\_calibrate

|  |
| --- |
| **int sn3d\_s**kip\_calibrate (  **void\*** handle,  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device)

**Return Values**

Reference : [The Definitions of Return Values.](#_The_Definitions_of_1)

**Remarks**

1．It can only achieve the skip function among the camera calibration ,HD calibration and white balance.

2．For example, if it is in the camera calibration phase, the user can call this function to enter the HD calibration directly, if it is in the HD calibration phase, the user can call this function to enter the white balance phase directly, if it is in the white balance phase, the user can call this function to enter the camera calibration phase directly.

## **Handle Rapid Scan**

### **Register Callback Function of Scan Events Notification**

**sn3d\_regist\_callback**

|  |
| --- |
| **int sn3d\_regist\_callback(**  **void\*** handle,  **sn3d\_callback** call\_back,  **void\*** user\_data  **)** ; |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device)

call\_back

[in] The system sends the events by calling this function.

|  |
| --- |
| **[void \_sn3d\_callback(int,int,void\*,int,void\*)](#_Callback_Function_of_1)** |

user\_data

[in] Parameters of call\_back. The system sends the parameters to user when the user status has been changed.

**Return Values**

Reference : [The Definitions of Return Values.](#_The_Definitions_of_1)

**Remarks**

1. The system sends calibration events to registrant after calling this function.

2 .For getting the time notification , the user must call this function before calling [start\_scan](#_Start_to_Scan).

### **Callback Function of Scan Events**

**sn3d\_callback**

|  |
| --- |
| void **sn3d\_callback** (  void\* handle,  int event\_type,  int event\_sub\_type,  void\* data  int data\_len  void\* user\_data  ); |

**Parameters**

handle

[out] Device handle from [sn3d\_initialize](#_The_Open_Device)

event\_type

[out] Event types.Reference :[The Macro Definitions of Scan Notification Events.](#_The_Macro_Definitions)

event\_sub\_type

[out] Event sub-types. Not used yet.

data

[out] The data structure pointer that matched with events.

|  |  |  |
| --- | --- | --- |
| event\_type | data | Meanings |
| SN3D\_VIDEO\_IMAGE\_DATA\_READY | [SN3D\_VIDEO\_FRAME](#_Video_Frame_Data) | The notification of image data |
| SN3D\_DISTANCE\_INDECATE | int | The notification of scan distance:  0 : too close  1-10:normal,from close to far  11: too far  12：tracking lost |
| SN3D\_SCANNER\_RT\_READY | [SN3D\_SCANNER\_PARAM](#_Scan_Parameters_1) | The notification of transformation matrix that relatives with the first frame data. |
| SN3D\_CURRENT\_MARKPOINT\_DATA\_READY | 0 | The notification of reference points:  Call [sn3d\_get\_current\_mark\_point](#_Get_Current_Reference) to get point cloud data in the callback function while received The notification. |
| SN3D\_CURRENT\_SCAN\_POINT\_CLOUD\_READY | 0 | The notification of current point cloud data: Call[sn3d\_get\_current\_scan\_point\_cloud](#_Get_Current_Point) to get point cloud data in the callback function while received The notification. |
| SN3D\_WHOLE\_SCAN\_POINT\_CLOUD\_READY | 0 | The notification of whole point cloud data: Call[sn3d\_get\_current\_scan\_whole\_point\_cloud](#_Get_Merged_Whole) to get point cloud data in the callback function while received The notification. |
| SN3D\_WHOLE\_MARKPOINT\_DATA\_READY | 0 | The notification of all of reference points data:Call [sn3d\_get\_whole\_mark \_point](#_Get_ALL_of) to get point cloud data in the callback function while received The notification. |
| SN3D\_SCANNER\_DOUBLECLICK | int | Device double-click event.  0 is unavailable. |
| SN3D\_SCANNER\_CLICK | int | Device single click event.  0:No scanning, call sn3d\_start\_scan to start to scan.  1:Scanning ,call sn3d\_pause\_scan to pause scanning. |
| SN3D\_SCANNER\_BRIGHTNESS\_PLUS | int | The notification of brightness button(+) clicked.Ranges:0-10. |
| SN3D\_SCANNER\_BRIGHTNESS\_SUB | int | The notification of brightness button(-) clicked.Ranges:0-10. |

data\_len

[out] The length of data

user\_data

[out] The parameter needs to be imported into **[sn3d\_regist\_callback](#_Register_Callback_Function_1)**

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. This function needs to be called after being registered, and be maintained by users.

2. Pay attention to multiple threads processing in outside due to the callback function runs in the inside thread.

3. Can't call other API functions in the callback function except **[sn3d\_get\_current\_scan\_point\_cloud](#_Get_Current_Point)**，**[sn3d\_get\_current\_scan\_whole\_point\_cloud](#_Get_Merged_Whole),[sn3d\_get\_current\_mark\_point](#_Get_Current_Reference),[sn3d\_get\_whole\_mark\_point](#_Get_ALL_of)**.

### **Start to Scan**

sn3d\_start\_scan

|  |
| --- |
| **int sn3d\_start\_scan**(  **void\*** handle  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device)

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. This function needs to be called after being registered.

2. Send scan information that includes scan distance info, scan data etc. to users by calling this function while scanning.

The data from [sn3d\_get\_current\_scan\_whole\_point\_cloud](#_Get_Merged_Whole) and [sn3d\_get\_current\_scan\_point\_cloud](#_Get_Current_Point) is only for 3D display.

Get reference points information from [sn3d\_get\_whole\_mark\_point](#_Get_ALL_of) and [sn3d\_get\_current\_mark\_point](#_Get_Current_Reference). Reference :[sn3d\_callback](#_Callback_Function_of).

3. Call [sn3d\_finish\_scan](#_Stop_Scanning) to stop scanning.

4. Call [sn3d\_pause\_scan](#_Pause_Scanning) to pause scanning.

5. Call [sn3d\_abandon\_scan](#_Delete_Scan_Data) to delete scan data.

6. Calling [sn3d\_set\_scan\_param](#_Set_Scan_Parameters) is forbidden.

### **Stop Scanning**

sn3d\_finish\_scan

|  |
| --- |
| **int sn3d\_finish\_scan**(  **void\*** handle  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device).

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. Get merged data from [sn3d\_get\_current\_scan\_whole\_point\_cloud](#_Get_Merged_Whole).

2. Call [sn3d\_start\_scan](#_Start_to_Scan) to continue to scan.

3. Call [sn3d\_abandon\_scan](#_Delete_Scan_Data) to delete scan data.

4. Calling [sn3d\_set\_scan\_param](#_Set_Scan_Parameters) is forbidden.

### **Pause Scanning**

sn3d\_pause\_scan

|  |
| --- |
| **int sn3d\_pause\_scan**(  **void\*** handle  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device_1)

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. Call [sn3d\_start\_scan](#_Start_to_Scan) to continue to scan.

2. Call [sn3d\_abandon\_scan](#_Delete_Scan_Data) to cancel scan data.

3. Call [sn3d\_finish\_scan](#_Stop_Scanning) to finish scan.

4. Calling [sn3d\_set\_scan\_param](#_Set_Scan_Parameters) is forbidden.

### **Delete Scan Data**

sn3d\_abandon\_scan

|  |
| --- |
| **int sn3d\_abandon\_scan**(  **void\*** handle  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device)

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. Clear all scan data.

2. Call [sn3d\_set\_scan\_param](#_Set_Scan_Parameters) to reset scan parameters.

3. Call [sn3d\_start\_scan](#_Start_to_Scan) to continue to scan.

### **Get Current Point Cloud Data**

sn3d\_get\_current\_scan\_point\_cloud

|  |
| --- |
| **int** sn3d\_get\_current\_scan\_point\_cloud (  void\* handle,  [SN3D\_CLOUD\_POINT](#_Point_Cloud_Data)& point\_cloud  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device)

point\_cloud

[in,out] Point cloud data.

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. Calling this function is valid before getting the return value from callback function and receiving the [SN3D\_CURRENT\_SCAN\_POINT\_CLOUD\_READY](#_The_Macro_Definitions).

2. This function needs to be called twice. The first time, keep point\_cloud pointer member is NULL, the function will return the point number. Allocate the memory according to the point number from outside; when the pointer is not NULL , user needs to call this function again ,and it will return point cloud data.

3. The data is only for 3D display.

### **Get Merged Whole Point Cloud Data**

sn3d\_get\_current\_scan\_whole\_point\_cloud

|  |
| --- |
| **int** sn3d\_get\_current\_scan\_whole\_point\_cloud(  void\* handle,  [SN3D\_CLOUD\_POINT](#_Point_Cloud_Data)& point\_cloud  ); |

**Parameters**

handle

[in] Device handle from [sn3d\_initialize](#_The_Open_Device).

point\_cloud

[in,out] Point cloud data.

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. Calling this function is valid before getting the return value from callback function and receiving the [SN3D\_WHOLE\_SCAN\_POINT\_CLOUD\_READY](#_Callback_Function_of_1)

2. This function needs to be called twice. The first time, keep point\_cloud pointer member is NULL, the function will return the point number. Allocate the memory according to the point number from outside; when the pointer is not NULL, user needs to call this function again, and it will return point cloud data.

3. After calling [sn3d\_finish\_scan](#_Stop_Scanning), the data from this function is the final scan data, or the data is only for 3D display.

### **Get Current Reference Points**

sn3d\_get\_current\_mark\_point

|  |
| --- |
| **int** sn3d\_get\_current\_mark\_point(  void\* handle,  [SN3D\_CLOUD\_POINT](#_Point_Cloud_Data)& point\_cloud  ); |

**Parameters**

handle

[in] Device handle from sn3d\_initialize

point\_cloud

[in,out] Point cloud data.

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. Calling this function is valid before getting the return value from callback function and receiving the [SN3D\_CURRENT\_MARKPOINT\_DATA\_READY](#_Callback_Function_of_1).

2．This function needs to be called twice. The first time, keep point\_cloud pointer member is NULL, the function will return the point number. Allocate the memory according to the point number from outside; when the pointer is not NULL, user needs to call this function again, and it will return point cloud data.

### **Get ALL of Reference Points**

sn3d\_get\_whole\_mark\_point

|  |
| --- |
| **int** sn3d\_get\_whole\_mark\_point(  void\* handle,  [SN3D\_CLOUD\_POINT](#_Point_Cloud_Data)& point\_cloud  ); |

**Parameters**

handle

[in] Device handle from sn3d\_initialize

point\_cloud

[in,out] Point cloud data.

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. Calling this function is valid before getting the return value from callback function and receiving the [SN3D\_WHOLE\_MARKPOINT\_DATA\_READY](#_Callback_Function_of_1).

2. This function needs to be called twice. The first time, keep point\_cloud pointer member is NULL, the function will return the point number. Allocate the memory according to the point number from outside; when the pointer is not NULL , user needs to call this function again ,and it will return point cloud data.

### **Delete Part of The Scan Data**

sn3d\_update\_cloud\_point

|  |
| --- |
| **int sn3d\_update\_cloud\_point**(  **void\*** handle  [SN3D\_CLOUD\_POINT](#_Point_Cloud_Data)& point\_cloud  ); |

**Parameters**

handle

[in] Device handle from sn3d\_initialize.

point\_cloud

[in] The rest of data after editing

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1. The data deleted by this function can't be undone.

2. This function only can be called in the pause scanning status([sn3d\_pause\_scan](#_Pause_Scanning)).

## **Handle HD Scan**

Reference Handle Rapid Scan .

### **Mesh Process Interface Definitions**

**sn3d\_mesh\_process**

|  |
| --- |
| **int sn3d\_mesh\_process** (  void\* handle,  **SN3D\_MESH\_PROCESS\_PARAM**& param  ) |

**Parameters**

handle

[in] Device handle from sn3d\_initialize.

param

[in] Mesh processing parameter, Reference : **[SN3D\_MESH\_PROCESS\_PARAM](#_数据处理参数)**

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1 It only can be called after calling **[sn3d\_finish\_scan](#_停止当前扫描_1)** to stop the scanning

2 Getting the return value from callback function, receive SN3D\_MESH\_DATA \_READY

3 Outside use the callback function to call [sn3d\_get\_current\_mesh\_data](#_获取当前网格数据) get data.

### **Get The Current Mesh Data**

sn3d\_get\_current\_mesh\_data

|  |
| --- |
| **int** sn3d\_get\_current\_mesh\_data(  void\* handle,  SN3D\_TRI\_MESH& point\_cloud  ); |

**Parameters**

handle

[in] Device handle from sn3d\_initialize.

point\_cloud

[in,out] mesh data

**Return Values**

Reference : [The Definitions of Return Values](#_The_Definitions_of_1)

**Remarks**

1 Calling this function is valid before getting the return value from callback function and receiving theSN3D\_MESH\_DATA \_READY

2 This function needs to be called twice. The first time, keep point\_cloud pointer member is NULL, the function will return the point number. Allocate the memory according to the point number from outside; when the pointer is not NULL , user needs to call this function again ,and it will return point cloud data.

## **Use Procedure**

### **Calibrate Procedure**



### **HD and Rapid Scan Procedure**

标定.wmf

# Fix Scan Interface Definitions

## **The Open Device Handle**

sn3d\_Initialize\_Scan\_Session

Initialize device handle.

|  |
| --- |
| **int sn3d\_Initialize\_Scan\_Session**(  SCANTYPE scantype,  const TCHAR\* szIniPath,  void\*&hScan,  int unitSize = 1); |

**Parameters**

type

[in] Initialization type. Reference : [The Definitions of Initialization Type](#_The_Definitions_of)

szIniPath

[in] The path of configuration file.

hScan

[out] Device handle .

unitSize

[in] Reserved word.

**Return Values**

Reference : 0 success ,other failed.

**Remarks**

1 . Function: Initialize device handle .

2 . Only One device handle can be used for one time.

3. Before calling **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**. to initialize a device handle , you need to call **[sn3d\_Destroy\_Scan\_Session](#_Close_Device_Handle_2)** to release current device handle .

## **Close Device Handle**

**sn3d\_Destroy\_Scan\_Session**

Close device handle.

|  |
| --- |
| **int sn3d\_Destroy\_Scan\_Session**(  void\* hScan  ); |

**Parameters**

hScan

[in] Device handle which is returned by **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**..

**Return Values**

Reference : 0 success ,other failed.

**Remarks**

1. Function: Release device handle.

2. Only One device handle can be used for one time.

3. Before calling **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**. to initialize a device handle, you need to release the current device handle first.

## **Register Callback Function of Display video image**

**Sn3d\_Regist\_Fix\_Scan\_Camera\_Callback**

|  |
| --- |
| int **sn3d\_Set\_Fix\_Scan\_Camera\_Callback**(  HSCAN hScan,  CAMERA\_DISPLAY\_CALLBACK\_FUNCTION fun\_,  void\* pOwner) |

**Parameters**

hScan

[in] Device handle from [sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1).

call\_back

[in] The system sends the video status events to cameras by calling this function.

user\_data

[in] Parameters of call\_back. The system sends the parameters to user when the user status has been changed.

**Return Values**

Reference : 0 success ,other failed.

**Remarks**

The system sends the right events to the registrant after registering this callback function.

## **Callback Function of Calibration Events**

**CAMERA\_DISPLAY\_CALLBACK\_FUNCTION**

|  |
| --- |
| **void(\*CAMERA\_DISPLAY\_CALLBACK\_FUNCTION)**(  const void\* const& pOwner,  const int& cameraID,  const unsigned char\* const& pImage,  const int& width,  const int& height,  const int& channel  ); |

**Parameters**

pOwner

[out] The parameter needs to be imported into [Sn3d\_Regist\_Fix\_Scan\_Camera\_Callback](#_Register_Callback_Function_2).

cameraID

[out] Image id. Reference : [SN3D\_VIDEO\_FRAME](#_Video_Frame_Data)

pImage

[out] The data of image.

width

[out] The width of image.

height

[out] The height of image.

channel

[out] The channels of image.

**Return Values**

**Remarks**

1. This function needs to be called after being registered, and be maintained by users.

2. Pay attention to multiple threads processing in outside due to the callback function runs in the inside thread.

3. Can't call other API functions in the callback function.

## **Get the Camera Brightness Max Value**

**sn3d\_get\_brightness\_max\_range**

|  |
| --- |
| **int sn3d\_get\_brightness\_max\_range** (  void\* hScan,  int& max  ) |

**Parameters**

hScan

[in] Device handle from **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**.

max

[out] The maximum value of camera brightness.

**Return Values**

Reference : 0 success ,other failed.

.

**Remarks**

## **Set Camera Brightness**

**sn3d\_set\_** **brightness**

|  |
| --- |
| **int sn3d\_set\_brightness** (  void\* handle ,  int brightness  ) |

**Parameters**

hScan

[in] Device handle from **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**.

brightness

[in] The level of brightness , Ranges: 0-12.

**Return Values**

Reference : 0 success ,other failed

**Remarks**

## **Set Texture Scan Parameters**

**sn3d\_enable\_textrue**

|  |
| --- |
| **int sn3d\_enable\_textrue** (  HSCAN hScan,  int bEnable  ); |

**Parameters**

hScan

[in] Device handle from **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**.

bEnable

[in] 1 use texture scan 0 use non-texture scan

**Return Values**

Reference : 0 success ,other failed

**Remarks**

1 It is only used after **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**.

## **Import Global Mark Point**

**sn3d\_import\_global\_mark\_point**

|  |
| --- |
| **int sn3d\_import\_ global\_mark\_point** (  HSCAN hScan,  sn3DTargetAlign::RefPoints**,**  ); |

**Parameters**

hScan

[in] Device handle from **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**[.](#_The_Open_Device_1)

RefPoints

[in] at least four points

**Return Values**

Reference :0 success ,other failed.

**Remarks**

1. It only can be called before calling [the mark scan of turntable](#_The_Mark_Scan) or [the free scan](#_The_free_Scan_1).

2 The global mark points will always used after using this function.

3 global\_mark\_point. Reference:**sdk\_demo\_fix**

## **The Code Scan of turntable**

sn3d\_turntable\_code\_Scan

|  |
| --- |
| **sn3d\_turntable\_code\_Scan(**  **HSCAN hScan,**  **sn3DCore::sn3DRangeData\*& data,**  **bool isfirst\_model**  **);** |

**Parameters**

hScan

[in] Device handle from **[sn3d\_Initialize\_Scan\_Session](#_The Open Device Handle_1)**..

data

[out] The scan data

isfirst\_model

[in] The start position of the turntable (a group turntable scan)

**Return Values**

Reference:0 success ,other failed.

**Remarks**

1 sn3DCore::sn3DRangeData. Reference:**sdk\_demo\_fix**

**2 Example:**

int turntable\_times\_ = 8;

int turntable\_angle\_ = 360 /8 + 0.5;

//group 1

for (int turn\_i\_ = 0; turn\_i\_ < turntable\_times\_; ++turn\_i\_)

{

if(0= =turn\_i\_ )

sn3d\_turntable\_code\_Scan(HSCAN, sn3DCore::sn3DRangeData\*& ,true);

else

sn3d\_turntable\_code\_Scan(HSCAN, sn3DCore::sn3DRangeData\*& ,false);

if( sn3DTurnTable::GetTurnTableState())

sn3DTurnTable::Run2Coordinate(X\_ROTATE, turntable\_angle\_, RELATIVE\_ANGLE);

}

//group 2

for (int turn\_i\_ = 0; turn\_i\_ < turntable\_times\_; ++turn\_i\_)

{

if(0= =turn\_i\_ )

sn3d\_turntable\_code\_Scan(HSCAN, sn3DCore::sn3DRangeData\*& ,true);

else

sn3d\_turntable\_code\_Scan(HSCAN, sn3DCore::sn3DRangeData\*& ,false);

if( sn3DTurnTable::GetTurnTableState())

sn3DTurnTable::Run2Coordinate(X\_ROTATE, turntable\_angle\_, RELATIVE\_ANGLE);

}

## **The Mark Scan of turntable**

sn3d\_turntable\_mark\_Scan

|  |
| --- |
| **sn3d\_turntable\_mark\_Scan(**  **HSCAN hScan,**  **sn3DCore::sn3DRangeData\*& data,**  **std::vector<sn3DCore::sn3DRangeData\*> datas,**  **bool isfirst\_model);** |

**Parameters**

hScan

[in] Device handle from **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**.

data

[out] The scan data

Datas

[in] from this state (true ==**isfirst\_model)** to the end of the group.

isfirst\_model

[in] The start position of the turntable (a group turntable scan)

**Return Values**

Reference: 0 success ,other failed.

**Remarks**

1 sn3DCore::sn3DRangeData. Reference:**sdk\_demo\_fix**

**2 Example:multiple group scan**

std::vector<sn3DCore::sn3DRangeData\*> all\_datas

std::vector<sn3DCore::sn3DRangeData\*> group\_datas

int turntable\_times\_ = 8;

int turntable\_angle\_ = 360 /8 + 0.5;

//group 1

for (int turn\_i\_ = 0; turn\_i\_ < turntable\_times\_; ++turn\_i\_)

{

sn3DCore::sn3DRangeData\* data;

if(0= =turn\_i\_ )

sn3d\_turntable\_mark\_Scan(HSCAN, sn3DCore::sn3DRangeData\*& ,true);

else

sn3d\_turntable\_code\_Scan(HSCAN, data ,group\_datas, false);

if( sn3DTurnTable::GetTurnTableState())

sn3DTurnTable::Run2Coordinate(X\_ROTATE, turntable\_angle\_, RELATIVE\_ANGLE);

all\_datas.push\_back(data);

group\_datas.push\_back(data);

}

//group 2

group\_datas.clear();

for (int turn\_i\_ = 0; turn\_i\_ < turntable\_times\_; ++turn\_i\_)

{

sn3DCore::sn3DRangeData\* data;

if(0= =turn\_i\_ )

sn3d\_turntable\_mark\_Scan(HSCAN, sn3DCore::sn3DRangeData\*& ,true);

else

sn3d\_turntable\_code\_Scan(HSCAN, data ,group\_datas, false);

if( sn3DTurnTable::GetTurnTableState())

sn3DTurnTable::Run2Coordinate(X\_ROTATE, turntable\_angle\_, RELATIVE\_ANGLE);

all\_datas.push\_back(data);

group\_datas.push\_back(data);

}

## **The Free Scan**

sn3d\_free\_Scan

|  |
| --- |
| **sn3d\_free\_Scan(**  **HSCAN hScan,**  **std::vector<sn3DCore::sn3DRangeData\*> datas,**  **sn3DCore::sn3DRangeData\*& data);** |

**Parameters**

hScan

[in] Device handle from **[sn3d\_Initialize\_Scan\_Session](#_The_Open_Device_1)**[.](#_The_Open_Device_1)

data

[out] the scan data

Datas

[in] all the scan data

**Return Values**

Reference:0 success ,other failed

**Remarks**

1 sn3DCore::sn3DRangeData. Reference:**sdk\_demo\_fix**

## **The Open Turntable Device**

SetType

The first step of Initialize device .

|  |
| --- |
| **int SetType**(  char \*idType); |

**Parameters**

[in] Initialization type. idType = "COM9-115200-1100"

**Return Values**

Reference :0 success ,other failed

**Remarks**

1. Open the turntable device has three steps : **SetType, OpenTurntable and initTurntable.**
2. **Example:** Open Turntable Device

int set\_ = -1;

char \*motortype = "COM9-115200-1100";

set\_ = sn3DTurnTable::SetType(motortype);

if(0==set)

{

if(0==sn3DTurnTable::OpenTurntable())

set\_ = sn3DTurnTable::initTurntable();

}

return set;

**OpenTurntable**

The second step of Initialize device .

|  |
| --- |
| **int OpenTurntable**(); |

**Return Values**

Reference :0 success ,other failed

**Remarks**

1.Open the turntable device has three steps : **SetType, OpenTurntable** and **initTurntable.**

**initTurntable**

The last step of Initialize device .

|  |
| --- |
| **int initTurntable**(); |

**Return Values**

Reference : 0 success ,other failed

**Remarks**

1.Open the turntable device has three steps : **SetType, OpenTurntable and initTurntable.**

## **Close Turntable Device**

**CloseTurntable**

Close the turntable device .

|  |
| --- |
| **int CloseTurntable** (  ); |

**Return Values**

Reference: 0 success ,other failed.

**Remarks**

## **Run A Specific Angle**

**Run2Coordinate**

|  |
| --- |
| **int Run2Coordinate**(  UCHAR degreefreedomID,  float desangle,  bool state=RELATIVE\_ANGLE  ); |

**Parameters**

degreefreedomID

[in] The degree of the freedom.degreefreedomID = **X\_ROTATE**

desangle

[in] The run angle

state

[in] The run type state = RELATIVE\_ANGLE

**Return Values**

Reference : 0 success ,other failed.

**Remark**

**1.Example: The turntable run 45** degrees

int turntable\_angle\_ = 45;

sn3DTurnTable::Run2Coordinate(X\_ROTATE, turntable\_angle\_, RELATIVE\_ANGLE);

## **Check The State Of Turntable**

**GetTurnTableState**

Check the state of turntable.

|  |
| --- |
| **int GetTurnTableState**(); |

**Parameters**

**Return Values**

1 : can continue to run turntable.

0: can not run turntable.

**Remarks**

**1.Example: check the state of turntable and go on run 45** degrees

int turntable\_angle\_ = 45;

if( sn3DTurnTable::GetTurnTableState())

sn3DTurnTable::Run2Coordinate(X\_ROTATE, turntable\_angle\_, RELATIVE\_ANGLE);

## **Use Procedure**

