EinScan SDK User Manual

Version 2.5.0.7

Shining3d

Version history		
Version number	Time	Modify record
V2.5.0.7	2017-09-28	1 Hand Scan SN3D_INIT_DATA add plus device type 2 The distance bars of HD and White Balance calibration reduce to 7 3 Hand scan add preview

Contents

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.The value is EinScan-Pro or EinScan-Plus.

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 .

Remarks

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.

Remarks

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 tagSN3D_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.

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

A、Flatten the calibration board.

B、Project the cross image into the white rectangle of calibration board .

C、Capture five images from different distances (reference: distance_indicate) by moving the scanner from bottom to top.

D、Change calibration board position, and repeat four more times according to B and C .

E、Capture finished and start to calibrate.

1： HD calibration.

Capture 7 images from different distances by moving scanner from bottom to top.

2： White Balance.

Capture the image which is the 2-4 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:0-6.Each value must appear once.

White balance:ranges :0-6.Calibration is successful when value is 2-4.

-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.

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

compute_state

Reference :The Constants of Calibration Calculation Status.

-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   *vertex_data;
    int          norma_count ;
    POINT_DATA   *norma_data;
    int          vertex_color_count ;
    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          *vertex_data;
    POINT_DATA          *norma_data;
    POINT_DATA          *vertex_color _data;
    int                 face_count ;
    SN3D_TRI_FACE       *face_ids;
    SN3D_UVCOORD        tex_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
SN3D_PREVIEW_SCAN_POINT_CLOUD_READY	29	The Notification of current preview scan data
SN3D_PREVIEW_SCAN_MARKPOINT_DATA_READY	30	The Notification of preview reference points 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

init_data

[in] Initialize device parameters. Reference : SN3D_INIT_DATA

handle

[out] Device handle.

Return Values

Reference :The Definitions of Return Values.

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** 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.

Return Values

Reference : The Definitions of Return Values.

Remarks

1. Function: Release 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 release the current device handle first.
3. You can not invoke this function when calibration computing .

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
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.

Remarks

Set Camera Brightness

sn3d_set_ brightness

```
int sn3d_set_brightness (
void* handle ,
int    brightness
)
```

Parameters

handle

[in] Device handle from sn3d_initialize.

brightness

[in] The level of brightness , Value ranges reference **sn3d_get_brightness_range** .

Return Values

Reference : The Definitions of Return Values.

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.

brightness

[out] The level of brightness , Value ranges reference **sn3d_get_brightness_range**.

Return Values

Reference : The Definitions of Return Values.

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.

mark_point

[in] Global reference point, at least four points. Reference: **SN3D_CLOUD_POINT**

Return Values

Reference : The Definitions of Return Values.

Remarks

1. It only can be called before calling **sn3d_set_scan_param** .

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.

param

[in] Scan configuration parameters. Reference: **SN3D_SCAN_PARAM**

Return Values

Reference : The Definitions of Return Values.

Remarks

1. It is only valid in SN3D_INIT_RAPIDSCAN or SN3D_INIT_HD_SCAN is set by sn3d_initialize .
2. It only can be called before calling sn3d_start_scan 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.

param

[out] Scan configuration parameters. Reference: **SN3D_SCAN_PARAM**

Return Values

Reference : The Definitions of Return Values.

Remarks

It is only valid in SN3D_INIT_RAPIDSCAN or SN3D_INIT_HD_SCAN is set by sn3d_initialize .

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.

call_back

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

void _sn3d_callback(int,int,void*,int,void*)

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.

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.

event_type

[out] Events Type. Reference : The Notification Events of Calibration.

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**	The notification of image data.
SN3D_CALIBRATION_STETE__READY	SN3D_STATE_CALIBRATE	The Calibration status

data_len

[out] The length of data.

user_data

[out] The parameter needs to be imported into **sn3d_regist_callback.**

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_start_calibrate (
    void* handle,
);
```

Parameters

handle

[in] Device handle from sn3d_initialize

Return Values

Reference : The Definitions of Return Values.

Remarks

1. SDK includes camera calibration, HD calibration, white balance.
   Reference : The Constants of Calibration Type
2. Camera Calibration.
   A. SDK enters scanner calibration status waiting for running this function to capture image after initializing.
   B. Capture images from 5 different positions, and capture 5 images from different distances at every position.
   C. Send capture status by callback while capturing.(Reference:The Calibration Status Data Structure)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.
   D. Calculate calibration result after finishing 5 group images capture. Users can call this function or exit calibration status until the calibration is done.
   E. If calibration calculation is successful, SDK will enter HD calibration status automatically.

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

3. 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 7 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 2-3 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_skip_calibrate (
    void* handle,
);
```

Parameters

handle

[in] Device handle from sn3d_initialize

Return Values

Reference : The Definitions of Return Values.

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

call_back

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

```
void _sn3d_callback(int,int,void*,int,void*)
```

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.

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.

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

event_type

[out] Event types.Reference :The Macro Definitions of Scan Notification Events.

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	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	The notification of transformation matrix that relatives with the first frame data.

SN3D_PREVIEW_SCAN_MARKPOINT_DATA_READY	0	The notification of preview reference points: Call sn3d_get_current_mark_point to get point cloud data in the callback function while received The notification.
SN3D_PREVIEW_SCAN_POINT_CLOUD_READY	0	The notification of preview current point cloud data: Call sn3d_get_current_scan_point_cloud to get point cloud data in the callback function while received The notification.
SN3D_CURRENT_MARKPOINT_DATA_READY	0	The notification of reference points: Call sn3d_get_current_mark_point 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: Callsn3d_get_current_scan_point_cloud 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: Callsn3d_get_current_scan_whole_point_cloud 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 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**

Return Values

Reference : The Definitions of 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 except **sn3d_get_current_scan_point_cloud**, **sn3d_get_current_scan_whole_point_cloud**,**sn3d_get_current_mark_point**,**sn3d_get_whole_mark_point**.

Start to Scan

sn3d_start_scan

```
int sn3d_start_scan(
    void* handle
);
```

Parameters

handle

[in] Device handle from sn3d_initialize

Return Values

Reference : The Definitions of Return Values

Remarks

1. This function needs to be called after being registered.
2. The first call **sn3d_start_scan** after each call set_scan_param to start preview scan,then you can call **sn3d_start_scan** again to start scan.
3. 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 and sn3d_get_current_scan_point_cloud is only for 3D display.

Get reference points information from sn3d_get_whole_mark_point and sn3d_get_current_mark_point. Reference :sn3d_callback.

4. Call sn3d_finish_scan to stop scanning.
5. Call sn3d_pause_scan to pause scanning.
6. Call sn3d_abandon_scan to delete scan data.
7. Calling sn3d_set_scan_param is forbidden.

Stop Scanning

sn3d_finish_scan

```
int sn3d_finish_scan(
    void*    handle
);
```

Parameters

handle

[in] Device handle from sn3d_initialize.

Return Values

Reference : The Definitions of Return Values

Remarks

1. Get merged data from sn3d_get_current_scan_whole_point_cloud.
2. Call sn3d_start_scan to continue to scan.
3. Call sn3d_abandon_scan to delete scan data.
4. Calling sn3d_set_scan_param is forbidden.

Pause Scanning

sn3d_pause_scan

```
int sn3d_pause_scan(
    void*    handle
);
```

Parameters

handle

[in] Device handle from sn3d_initialize

Return Values

Reference : The Definitions of Return Values

Remarks

1. Call sn3d_start_scan to continue to scan.
2. Call sn3d_abandon_scan to cancel scan data.
3. Call sn3d_finish_scan to finish scan.
4. Calling sn3d_set_scan_param is forbidden.

Delete Scan Data

sn3d_abandon_scan

```
int sn3d_abandon_scan(
    void*    handle
);
```

Parameters

handle

[in] Device handle from sn3d_initialize

Return Values

Reference : The Definitions of Return Values

Remarks

1. Clear all scan data.
2. Call sn3d_set_scan_param to reset scan parameters.
3. Call sn3d_start_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
);
```

Parameters

handle

[in] Device handle from sn3d_initialize

point_cloud

[in,out] Point cloud data.

Return Values

Reference : The Definitions of Return Values

Remarks

1. Calling this function is valid before getting the return value from callback function and receiving the SN3D_CURRENT_SCAN_POINT_CLOUD_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.
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
);
```

Parameters

handle
[in] Device handle from sn3d_initialize.
point_cloud
[in,out] Point cloud data.

Return Values

Reference : The Definitions of Return Values

Remarks

1. Calling this function is valid before getting the return value from callback function and receiving the SN3D_WHOLE_SCAN_POINT_CLOUD_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.
3. After calling sn3d_finish_scan, 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
);
```

Parameters

handle
[in] Device handle from sn3d_initialize
point_cloud
[in,out] Point cloud data.

Return Values

Reference : The Definitions of Return Values

Remarks

1. Calling this function is valid before getting the return value from callback function and receiving the SN3D_CURRENT_MARKPOINT_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.

Get ALL of Reference Points

sn3d_get_whole_mark_point

```
int sn3d_get_whole_mark_point(
    void* handle,
    SN3D_CLOUD_POINT& 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

Remarks

1. Calling this function is valid before getting the return value from callback function and receiving the SN3D_WHOLE_MARKPOINT_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.

Delete Part of The Scan Data

sn3d_update_cloud_point

```
int sn3d_update_cloud_point(
    void*      handle
    SN3D_CLOUD_POINT& 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

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).

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**

Return Values

Reference : The Definitions of Return Values

Remarks

1 It only can be called after calling **sn3d_finish_scan** 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 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

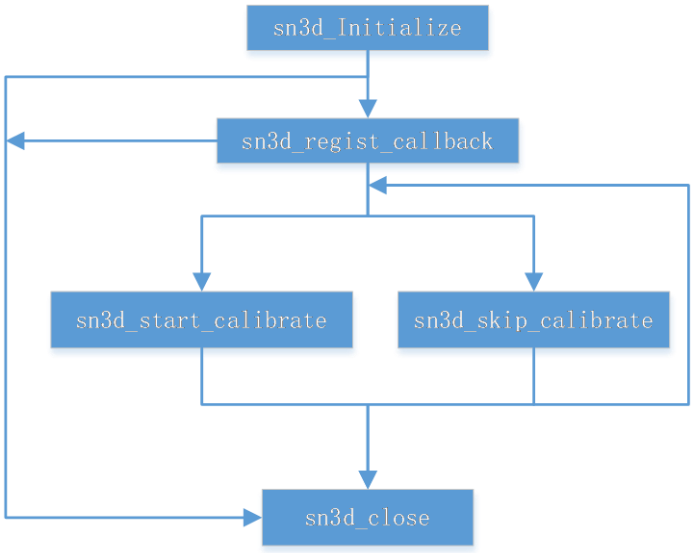
Remarks

1 Calling this function is valid before getting the return value from callback function and receiving theSN3D_MESH_DATA _READY

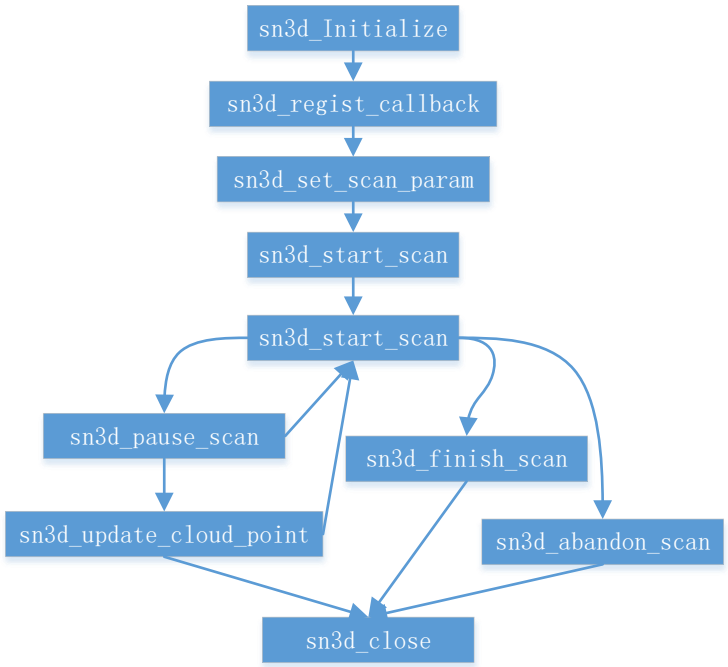
2 This function needs to be called twice. The first time, keep point_cloud pointer member is NULL, the function will return the point number. Allocate the memory according to the point number from outside; when the pointer is not NULL , user needs to call this function again ,and it will return point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

Use Procedure

Calibrate Procedure



HD and Rapid Scan Procedure



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

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. to initialize a device handle , you need to call **sn3d_Destroy_Scan_Session** 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..

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. 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.

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.

cameraID

[out] Image id. Reference : SN3D_VIDEO_FRAME

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.

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.

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.

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.

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.

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 or the free scan.
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..

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.

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.

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 Shot Calibration Image

Shot Calibration Image

```
sn3d_shot_calibrate_image(
    HSCAN hScan,
     int* imageID);
```

Parameters

hScan

[in] Device handle from sn3d_Initialize_Scan_Session.

imageID

[out] Image id

Return Values

Reference:0 success ,other failed

Remarks

1 the function is used before sn3d_calibrate_camera and must be used more than 3 groups.

Calibrate the Desk Camera

Calculate the camera

```
sn3d_calibrate_camera(
    HSCAN hScan,
    int   shot_step);
```

Parameters

hScan

[in] Device handle from sn3d_Initialize_Scan_Session.

shot_step

[in] the group of snap image(3 group)

Return Values

Reference:0 success ,other failed

Remarks

The white balance test function

```
sn3d_white_balance_test(
HSCAN hScan,
double & LrScale,
double &LgScale,
 double &LbScale);
```

Parameters

hScan

[in] Device handle from sn3d_Initialize_Scan_Session.

LrScale

[out] R Parameters

LgScale

[out] G Parameters

LbScale

[out] B Parameters

eturn Values

Reference:0 success ,other failed

Remarks

Set the white balance Parameters

```
sn3d_set_white_balance(
HSCAN hScan,
const double LrScale,
const double LgScale,
const double LbScale)
```

Parameters

hScan

[in] Device handle from sn3d_Initialize_Scan_Session.

LrScale

[out] R Parameters

LgScale

[out] G Parameters

LbScale

[out] B Parameters

eturn Values

Reference:0 success ,other failed

Remarks

The auto scan

```
sn3d_auto_scan(
HSCAN hScan,
sn3DCore::sn3DRangeData*& data,
double Angle,
bool IsFirstScan);
```

Parameters

hScan

[in] Device handle from sn3d_Initialize_Scan_Session.

data

[out] the scan data

Angle

[int] the run angle of the turntable

IsFirstScan

[int] the first model of one group is true ,or false.

Return Values

Reference:0 success ,other failed

Remarks

The Group Align Process Function

```
sn3d_group_align_process(HSCAN hScan,
std::vector<sn3DCore::sn3DRangeData*>&group_datas0,
std::vector<sn3DCore::sn3DRangeData*>& group_datas1)
```

Parameters

hScan

[in] Device handle from sn3d_Initialize_Scan_Session.

group_datas0

[in/out] the fix models

group_datas0

[in/out] the float models

Return Values

Reference:0 success ,other failed

Remarks

The function can be used after the auto scan.

Get Mesh Data Function

```
sn3d_get_current_mesh_data(
HSCAN hScan,
const std::vector<sn3DCore::sn3DRangeData*> group_datas,
SN3D_DESK_TRI_MESH& mesh_cloud,
const SN3D_DESK_MESH_PROCESS_PARAM param);
```

Parameters

hScan

[in] Device handle from sn3d_Initialize_Scan_Session.

group_datas

[in] the scan models

mesh_cloud

[out] the mesh_cloud

param

[out] the mesh process

Return Values

Reference:0 success ,other failed

Remarks

The function can be used after the auto scan.

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