



eYS3D
Microelectronics

eYs3D Windows SDK

1.5.8.8

Generated by Doxygen 1.9.1

1 Introduction	3
2 Data Structure Index	5
2.1 Data Structures	5
3 File Index	7
3.1 File List	7
4 Data Structure Documentation	9
4.1 DEVINFORMATIONEX	9
4.1.1 Detailed Description	9
4.1.2 Field Documentation	9
4.1.2.1 nChipID	9
4.1.2.2 nDevType	9
4.1.2.3 strDevName	10
4.1.2.4 strDevPath	10
4.1.2.5 wPID	10
4.1.2.6 wUsbNode	10
4.1.2.7 wVID	10
4.2 eSPCtrl_RectLogData	11
4.2.1 Detailed Description	11
4.2.2 Field Documentation	11
4.2.2.1 Date	11
4.3 ParaLUT	11
4.3.1 Detailed Description	13
4.4 tagDEVINFORMATION	14
4.4.1 Detailed Description	14
4.4.2 Field Documentation	14
4.4.2.1 nChipID	14
4.4.2.2 nDevType	14
4.4.2.3 strDevName	14
4.4.2.4 strDevPath	14
4.4.2.5 wPID	14
4.4.2.6 wUsbNode	15
4.4.2.7 wVID	15
5 File Documentation	17
5.1 eSPDI_Common.h File Reference	17
5.1.1 Detailed Description	22
5.1.2 Typedef Documentation	22
5.1.2.1 APC_DeviceEventFn	22
5.1.2.2 APC_ImgCallbackFn	22
5.1.2.3 DEVINFORMATION	23
5.1.2.4 eSPCtrl_RectLogData	23

5.1.2.5 PARALUT	23
5.1.3 Enumeration Type Documentation	23
5.1.3.1 APC_DEVICE_TYPE	23
5.1.3.2 SENSOR_TYPE_NAME	23
5.1.3.3 USERDATA_SECTION_INDEX	23
5.1.4 Function Documentation	24
5.1.4.1 APC_CloseDevice()	24
5.1.4.2 APC_DisableAE()	24
5.1.4.3 APC_DisableAWB()	25
5.1.4.4 APC_EnableAE()	25
5.1.4.5 APC_EnableAWB()	25
5.1.4.6 APC_EnableGPUAcceleration()	26
5.1.4.7 APC_FindDevice()	27
5.1.4.8 APC_GetCTPropVal()	27
5.1.4.9 APC_GetCurrentIRValue()	28
5.1.4.10 APC_GetDepthFilterVersion()	28
5.1.4.11 APC_GetDeviceInfo()	28
5.1.4.12 APC_GetDeviceInfoEx()	29
5.1.4.13 APC_GetDeviceNumber()	29
5.1.4.14 APC_GetDeviceResolutionList()	30
5.1.4.15 APC_GetFlexibleGyroData()	30
5.1.4.16 APC_GetFlexibleGyroLength()	31
5.1.4.17 APC_GetFWRegister()	31
5.1.4.18 APC_GetFwVersion()	31
5.1.4.19 APC_GetGPIOValue()	32
5.1.4.20 APC_GetHWRegister()	32
5.1.4.21 APC_GetIRMaxValue()	33
5.1.4.22 APC_GetIRMinValue()	33
5.1.4.23 APC_GetLogData()	34
5.1.4.24 APC_GetMultiBytesHWRegister()	34
5.1.4.25 APC_GetPidVid()	35
5.1.4.26 APC_GetPUPropVal()	36
5.1.4.27 APC_GetSensorRegister()	37
5.1.4.28 APC_GetSlaveHWRegister()	37
5.1.4.29 APC_GetSlaveLogData()	38
5.1.4.30 APC_GetSlaveSensorRegister()	38
5.1.4.31 APC_Init()	39
5.1.4.32 APC_Init2()	39
5.1.4.33 APC_Is360Device()	39
5.1.4.34 APC_OpenDevice()	40
5.1.4.35 APC_ReadFlashData()	41
5.1.4.36 APC_RefreshDevice()	41

5.1.4.37 APC_RegisterDeviceEvents()	41
5.1.4.38 APC_Release()	42
5.1.4.39 APC_ResetUNPData()	42
5.1.4.40 APC_SetCTPropVal()	42
5.1.4.41 APC_SetCurrentIRValue()	43
5.1.4.42 APC_SetFWRegister()	43
5.1.4.43 APC_SetGPIOCtrl()	44
5.1.4.44 APC_SetGPIOValue()	44
5.1.4.45 APC_SetHuffmanTableData()	44
5.1.4.46 APC_SetHWRegister()	45
5.1.4.47 APC_SetIRMaxValue()	45
5.1.4.48 APC_SetIRMaxValueUnleashed()	46
5.1.4.49 APC_SetLogData()	46
5.1.4.50 APC_SetLogData_Advanced()	46
5.1.4.51 APC_SetMultiBytesHWRegister()	47
5.1.4.52 APC_SetPidVid()	47
5.1.4.53 APC_SetPUPPropVal()	48
5.1.4.54 APC_SetQuantizationTableData()	48
5.1.4.55 APC_SetSensorTypeName()	48
5.1.4.56 APC_SetSlaveHWRegister()	49
5.1.4.57 APC_SetSlaveLogData()	49
5.1.4.58 APC_SetUserData()	50
5.2 eSPDI_Common.h	50
5.3 eSPDI_DM.h File Reference	62
5.3.1 Detailed Description	64
5.3.2 Function Documentation	64
5.3.2.1 APC_GetDepthDataType()	64
5.3.2.2 APC_GetFileData()	64
5.3.2.3 APC_GetFWFSLength()	65
5.3.2.4 APC_GetHWPostProcess()	65
5.3.2.5 APC_GetRectifyMatLogData()	65
5.3.2.6 APC_GetRectifyTable()	66
5.3.2.7 APC_GetSlaveRectifyTable()	66
5.3.2.8 APC_GetSlaveYOffset()	67
5.3.2.9 APC_GetYOffset()	67
5.3.2.10 APC_GetZDTable()	68
5.3.2.11 APC_IsInterleaveDevice()	68
5.3.2.12 APC_SetDepthDataType()	69
5.3.2.13 APC_SetHWPostProcess()	69
5.3.2.14 APC_SetSlaveYOffset()	70
5.3.2.15 APC_SetYOffset()	70
5.3.2.16 APC_SetYOffset_Advanced()	71

5.4 eSPDI_DM.h	71
5.5 eSPDI_ErrCode.h File Reference	73
5.5.1 Detailed Description	73
5.6 eSPDI_ErrCode.h	74
Index	75

Chapter 1

Introduction

This document describes the usage of Application Programming Interfaces of eYs3D Windows SDK

What's inside the SDK

Table 1.1 File List

Folder	Subfolder	Filename	Description
bin	Win32	All files	Sample executables on Win32 platform
	x64	All files	Sample executables on Windows 64-bits platform
eSPDI	include	eSPDI_Common.h	Basic API declaration header
		eSPDI_DM.h	Depth Map specific API declaration header
		eSPDI_ErrCode.h	Error code definitions
	Win32	eSPDI_DM.dll	eSPDI dynamical linked library for Win32 platform
		eSPDI_DM.lib	eSPDI static linked library for Win32 platform
	x64	eSPDI_DM.dll	eSPDI dynamical linked library for Windows 64-bits
		eSPDI_DM.lib	eSPDI static linked library for Windows 64-bits
doc	html	index.html	This documentation
DMPreview			A sample VC++ project demonstrating how to open multiple devices in an application

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

DEVINFORMATIONEX	9
eSPCtrl_RectLogData	
ESPCtrl_RectLogData	11
ParaLUT	
ParaLUT	11
tagDEVINFORMATION	
DEVINFORMATION	14

Chapter 3

File Index

3.1 File List

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

eSPDI_Common.h	EYs3D SDK API export functions, data structure and variable definition	17
eSPDI_DM.h	EYs3D SDK API export functions, data structure and variable definition for depth map module .	62
eSPDI_ErrCode.h	Definition of eYs3D SDK error code Copyright: This file copyright (C) 2017 by	73

Chapter 4

Data Structure Documentation

4.1 DEVINFORMATIONEX

```
#include <eSPDI_Common.h>
```

Data Fields

- unsigned short [wPID](#)
- unsigned short [wVID](#)
- char [strDevName](#) [512]
- char [strDevPath](#) [512]
- unsigned short [nChipID](#)
- unsigned short [nDevType](#)
- unsigned short [wUsbNode](#)

4.1.1 Detailed Description

extended device information class

4.1.2 Field Documentation

4.1.2.1 nChipID

```
unsigned short nChipID
```

chip ID, 0x18 for AXES1, 0x1C for KIWI, 0x15 for PUMA

4.1.2.2 nDevType

```
unsigned short nDevType
```

chip enum value, see [APC_DEVICE_TYPE](#)

4.1.2.3 strDevName

```
char strDevName[512]
```

device name

4.1.2.4 strDevPath

```
char strDevPath[512]
```

device path

4.1.2.5 wPID

```
unsigned short wPID
```

product ID

Table 4.1 PID List

Chip Name	Chip ID	PID
AXES1	0x18	0x0568
		0x0668
		0x0113
		0x0115
		0x0116
KIWI	0x1C	0x0118
PUMA	0x15	0x0112
		0x0120

4.1.2.6 wUsbNode

```
unsigned short wUsbNode
```

USB Node

4.1.2.7 wVID

```
unsigned short wVID
```

vender ID, 0x1E4E for ApcDI device

The documentation for this class was generated from the following file:

- [eSPDI_Common.h](#)

4.2 eSPCtrl_RectLogData

[eSPCtrl_RectLogData](#)

```
#include <eSPDI_Common.h>
```

4.2.1 Detailed Description

[eSPCtrl_RectLogData](#)

Rectified log data structure

4.2.2 Field Documentation

4.2.2.1 Date

long Date

pars for compensating disparity value, Formula: $\text{new_disp_vaule} = \text{disp_value} * \text{depth_comp_pars}[0] + \text{depth_comp_pars}[1]$

The documentation for this struct was generated from the following file:

- [eSPDI_Common.h](#)

4.3 ParaLUT

[ParaLUT](#).

```
#include <eSPDI_Common.h>
```

Data Fields

- long long **file_ID_header**
[00]-[000] File ID header : 2230
- long long **file_ID_version**
[01]-[008] File ID version : 4
- double **FOV**
[02]-[016] Field of view with degree
- long long **semi_FOV_pixels**
[03]-[024] Pixels for semi-FOV
- long long **img_src_cols**
[04]-[032] Width for source image (single image)
- long long **img_src_rows**
[05]-[040] Height for source image
- double **img_L_src_col_center**
[06]-[048] Center of width for L side source image

- double **img_L_src_row_center**
[07]-[056] Center of height for L side source image
- double **img_R_src_col_center**
[08]-[064] Center of width for R side source image
- double **img_R_src_row_center**
[09]-[072] Center of height for R side source image
- double **img_L_rotation**
[10]-[080] Rotation for L side image
- double **img_R_rotation**
[11]-[088] Rotation for R side image
- double **spline_control_v1**
[12]-[096] Spline control value for row = DIV x 0 pixel, DIV = rows/6
- double **spline_control_v2**
[13]-[104] Spline control value for row = DIV x 1 pixel, DIV = rows/6
- double **spline_control_v3**
[14]-[112] Spline control value for row = DIV x 2 pixel, DIV = rows/6
- double **spline_control_v4**
[15]-[120] Spline control value for row = DIV x 3 pixel, DIV = rows/6
- double **spline_control_v5**
[16]-[128] Spline control value for row = DIV x 4 pixel, DIV = rows/6
- double **spline_control_v6**
[17]-[136] Spline control value for row = DIV x 5 pixel, DIV = rows/6
- double **spline_control_v7**
[18]-[144] Spline control value for row = DIV x 6 pixel, DIV = rows/6
- long long **img_dst_cols**
[19]-[152] Width for output image (single image), according to "Original" parameters
- long long **img_dst_rows**
[20]-[160] Height for output image, according to "Original" parameters
- long long **img_L_dst_shift**
[21]-[168] Output L side image shift in row
- long long **img_R_dst_shift**
[22]-[176] Output R side image shift in row
- long long **img_overlay_LR**
[23]-[184] Overlay between L/R in pixels, far field, (YUV must be even)
- long long **img_overlay_RL**
[24]-[192] Overlay between R/L in pixels, far field, (YUV must be even)
- long long **img_stream_cols**
[25]-[200] Output image stream of cols
- long long **img_stream_rows**
[26]-[208] Output image stream of rows
- long long **video_stream_cols**
[27]-[216] Output video stream of cols
- long long **video_stream_rows**
[28]-[224] Output video stream of rows
- long long **usb_type**
[29]-[232] 2 for usb2, 3 for usb3
- long long **img_type**
[30]-[240] 1 for yuv422, 2 for BGR, 3 for RGB
- long long **lut_type**
[31]-[248] Output LUT type eys::LutModes
- long long **blending_type**

- [32]-[256] 0 for choosed by function, 1 for alpha-blending, 2 for Laplacian pyramid blending*
- double **overlay_ratio**
 - [33]-[264] far field overlay value is equal to [img_overlay_LR\(RL\)](#) = overlay_value + overlay_ratio*
- long long **serial_number_date0**
 - [34]-[272] 8 bytes, yyyy-mm-dd*
- long long **serial_number_date1**
 - [35]-[280] 8 bytes, hh-mm-ss-xxx, xxx for machine number*
- double **unit_sphere_radius**
 - [36]-[288] Original : Unit spherical radius for dewarping get x and y*
- double **min_col**
 - [37]-[296] Original : Parameters of min position of image width*
- double **max_col**
 - [38]-[304] Original : Parameters of max position of image width*
- double **min_row**
 - [39]-[312] Original : Parameters of min position of image height*
- double **max_row**
 - [40]-[320] Original : Parameters of max position of image height*
- long long **AGD_LR**
 - [41]-[328] Err : Average gray-level value discrepancy at LR boundary*
- long long **AGD_RL**
 - [42]-[336] Err : Average gray-level value discrepancy at RL boundary*
- long long **out_img_resolution**
 - [43]-[344] Set output resolution eys::ImgResolutionModes*
- long long **out_lut_cols**
 - [44]-[352] Output side-by-side lut width, according to the set of out_img_resolution*
- long long **out_lut_rows**
 - [45]-[360] Output lut height, according to the set of out_img_resolution*
- long long **out_lut_cols_eff**
 - [46]-[368] Output effective pixels in out_lut_cols, 0 is for all*
- long long **out_lut_rows_eff**
 - [47]-[376] Output effective pixels in out_lut_rows, 0 is for all*
- long long **out_img_cols**
 - [48]-[384] Output side-by-side image width after dewarping and stitching, according to the set of out_img_resolution*
- long long **out_img_rows**
 - [49]-[392] Output image height, according to the set of out_img_resolution*
- long long **out_overlay_LR**
 - [50]-[340] Output L/R overlay value, according to the set of out_img_resolution*
- long long **out_overlay_RL**
 - [51]-[408] Output R/L overlay value, according to the set of out_img_resolution*
- long long **reserve** [44]
 - [52]-[416] Reserve 44 parameter to use*

4.3.1 Detailed Description

[ParaLUT](#).

Spherical look-up table conversion parameters

The documentation for this struct was generated from the following file:

- [eSPDI_Common.h](#)

4.4 tagDEVINFORMATION

[DEVINFORMATION.](#)

```
#include <eSPDI_Common.h>
```

Data Fields

- unsigned short [wPID](#)
- unsigned short [wVID](#)
- char * [strDevName](#)
- char * [strDevPath](#)
- unsigned short [nChipID](#)
- unsigned short [nDevType](#)
- unsigned short [wUsbNode](#)

4.4.1 Detailed Description

[DEVINFORMATION.](#)

device information

4.4.2 Field Documentation

4.4.2.1 nChipID

unsigned short nChipID

chip ID, 0x18 for AXES1, 0x1C for KIWI, 0x15 for PUMA

4.4.2.2 nDevType

unsigned short nDevType

chip enum value,

See also

[APC_DEVICE_TYPE](#)

4.4.2.3 strDevName

char* strDevName

pointer to device name stored inside the SDK

4.4.2.4 strDevPath

char* strDevPath

pointer to device path stored inside the SDK

4.4.2.5 wPID

unsigned short wPID

product ID

Table 4.2 PID List

Chip Name	Chip ID	PID
AXES1	0x18	0x0568
		0x0668
		0x0113
		0x0115
		0x0116
KIWI	0x1C	0x0118
PUMA	0x15	0x0112
		0x0120

4.4.2.6 wUsbNode

unsigned short wUsbNode

USB Node

4.4.2.7 wVID

unsigned short wVID

vender ID, 0x1E4E for ApcDI device

The documentation for this struct was generated from the following file:

- [eSPDI_Common.h](#)

Chapter 5

File Documentation

5.1 eSPDI_Common.h File Reference

eYs3D SDK API export functions, data structure and variable definition

Data Structures

- struct [eSPCtrl_RectLogData](#)
eSPCtrl_RectLogData
- struct [ParaLUT](#)
ParaLUT.
- struct [tagDEVINFORMATION](#)
DEVINFORMATION.
- class [DEVINFORMATIONEX](#)

Typedefs

- typedef struct [eSPCtrl_RectLogData](#) [eSPCtrl_RectLogData](#)
eSPCtrl_RectLogData
- typedef struct [ParaLUT](#) [PARALUT](#)
ParaLUT.
- typedef void(* [APC_ImgCallbackFn](#)) (APCImageType::Value imgType, int imgId, unsigned char *imgBuf, int imgSize, int width, int height, int serialNumber, LONGLONG timestamp, void *pParam)
Callback function when video or data is ready.
- typedef struct [tagDEVINFORMATION](#) [DEVINFORMATION](#)
DEVINFORMATION.
- typedef void(* [APC_DeviceEventFn](#)) (UINT pid, UINT vid, BOOL bAttached, void *pData)
Callback function to receive any USB capture device attachment or detachment events.

Enumerations

- enum [APC_DEVICE_TYPE](#) { [OTHERS](#) = 0 , [AXES1](#) , [PUMA](#) , [PLUM](#) , [GRAPE_FPGA](#) }
- enum [USERDATA_SECTION_INDEX](#) {
[USERDATA_SECTION_0](#) = 0 , [USERDATA_SECTION_1](#) , [USERDATA_SECTION_2](#) , [USERDATA_SECTION_3](#)
, [USERDATA_SECTION_4](#) , [USERDATA_SECTION_5](#) , [USERDATA_SECTION_6](#) , [USERDATA_SECTION_7](#) ,
[USERDATA_SECTION_8](#) , [USERDATA_SECTION_9](#) , [USERDATA_SECTION_10](#) , [USERDATA_SECTION_NUM](#)
}
- enum [SENSOR_TYPE_NAME](#) {
[APC_SENSOR_TYPE_H22](#) = 0 , [APC_SENSOR_TYPE_OV7740](#) , [APC_SENSOR_TYPE_AR0134](#) ,
[APC_SENSOR_TYPE_AR0135](#) ,
[APC_SENSOR_TYPE_AR0144](#) , [APC_SENSOR_TYPE_OV9714](#) , [APC_SENSOR_TYPE_OV9282](#) ,
[APC_SENSOR_TYPE_AR0330](#) ,
[APC_SENSOR_TYPE_AR1335](#) , [APC_SENSOR_TYPE_H65](#) , [APC_SENSOR_TYPE_AR0522](#) , [APC_SENSOR_TYPE_OV2740](#) ,
[APC_SENSOR_TYPE_OC0SA10](#) , [APC_SENSOR_TYPE_VD56G3](#) , [APC_SENSOR_TYPE_VD66GY](#) ,
[APC_SENSOR_TYPE_H68](#) ,
[APC_SENSOR_TYPE_UNKOWN](#) = 0xffff }

Functions

- int APC_API [APC_Init](#) (void **ppHandleApcDI, bool bIsLogEnabled)
entry point of eYs3D camera SDK. This API allocates resource and find all the eSPI camera devices connected to the system.
- int APC_API [APC_Init2](#) (void **ppHandleApcDI, bool bIsLogEnabled, bool bAutoRestart)
entry point of eYs3D camera SDK. This API allocates resource and find all the eSPI camera devices connected to the system.
- int APC_API [APC_RegisterDeviceEvents](#) (void *pHandleApcDI, [APC_DeviceEventFn](#) cbFunc, void *pData)
Register the USB device plug or unplug events. Any USB capture device attachment or detachment events will call the callback function cbFunc.
- void APC_API [APC_Release](#) (void **ppHandleApcDI)
release all resource that APC_Init had allocated
- int APC_API [APC_FindDevice](#) (void *pHandleApcDI)
find out all eYs3D USB devices by PID, VID and ChipID, also remember device types
- int APC_API [APC_RefreshDevice](#) (void *pHandleApcDI)
refresh all eYs3D UVC devices
- int APC_API [APC_GetDeviceNumber](#) (void *pHandleApcDI)
get eYs3D USB device numbers
- int APC_API [APC_GetDeviceInfo](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, [DEVINFORMATION](#) *pdevinfo)
get informations of eYs3D UVC devices, see @DEVINFORMATION
- int APC_API [APC_GetDeviceInfoEx](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, [DEVINFORMATIONEX](#) *pdevinfo)
get target device info,
- int APC_API [APC_GetSlaveSensorRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, unsigned short address, unsigned short *pValue, int flag, int nSensorMode)
get value from sensor register
- int APC_API [APC_GetSensorRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, unsigned short address, unsigned short *pValue, int flag, int nSensorMode)
get value from sensor register
- int APC_API [APC_GetFWRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short *pValue, int flag)
get firmware register value

- int APC_API [APC_SetFWRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short nValue, int flag)
set firmware register value
- int APC_API [APC_GetSlaveHWRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short *pValue, int flag)
get hardware register value
- int APC_API [APC_GetHWRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short *pValue, int flag)
get hardware register value
- int APC_API [APC_SetSlaveHWRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short nValue, int flag)
set hardware register
- int APC_API [APC_SetHWRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short nValue, int flag)
set hardware register
- int APC_API [APC_GetMultiBytesHWRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, UCHAR *data, int address, int size)
get hardware register with multibytes
- int APC_API [APC_SetMultiBytesHWRegister](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, UCHAR *data, int address, int size)
set hardware register with multibytes
- int APC_API [APC_GetFwVersion](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, char *pszFwVersion, int nBufferSize, int *pActualLength)
get the firmware version of device, the version is a string
- int APC_API [APC_GetPidVid](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short *pPidBuf, unsigned short *pVidBuf)
get PID(product ID) and VID(vendor ID) of device
- int APC_API [APC_SetPidVid](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short *pPidBuf, unsigned short *pVidBuf)
set PID and VID to device
- int APC_API [APC_GetSlaveLogData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
get log data from flash
- int APC_API [APC_GetLogData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
get log data from flash
- int APC_API [APC_SetSlaveLogData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
set log data to flash
- int APC_API [APC_SetLogData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
set log data to flash
- int APC_API [APC_SetLogData_Advanced](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
set log data to flash
- int APC_API [APC_SetUserData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, [USERDATA_SECTION_INDEX](#) usi)
set user data to flash
- int APC_API [APC_ReadFlashData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, FLASH_DATA_TYPE fdt, BYTE *pBuffer, unsigned long int nLengthOfBuffer, unsigned long int *pActualBufferLen)
read firmware code(.bin) form flash The firmware code is the combination of boot loader, firmware body and plug-in data. This input buffer length has to match with the flash data type

- int APC_API [APC_OpenDevice](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int colorStreamIndex, int depthStreamIndex, int depthStreamSwitch, int iFps, [APC_ImgCallbackFn](#) callbackFn, void *pCallbackParam, int pid=-1)
open camera device with image callback support
- int APC_API [APC_GetColorImage](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *pBuf, unsigned long int *pImageSize, int *pSerial=NULL)
get color image
- int APC_API [APC_CloseDevice](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo)
close device and stop video render
- int APC_API [APC_GetDeviceResolutionList](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nMaxCount0, APC_STREAM_INFO *pStreamInfo0, int nMaxCount1, APC_STREAM_INFO *pStreamInfo1)
get the device resolution list
- bool APC_API [APC_Is360Device](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo)
check module is spherical device or not
- int APC_API [APC_GetSerialNumberFromLog](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, char *pSerialNum, int nBufferSize, int *pActualLength)
get the module serial number
- int APC_API [APC_SetCurrentIRValue](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType)
set current infrared radiation(IR) value
- int APC_API [APC_GetCurrentIRValue](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType)
get current infrared radiation(IR) value
- int APC_API [APC_GetIRMinValue](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType)
get minimum IR value the module support
- int APC_API [APC_SetIRMaxValue](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType)
set maximum IR value the module support
- int APC_API [APC_SetIRMaxValueUnleashed](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType)
set maximum IR value the module support without any limitation
- int APC_API [APC_GetIRMaxValue](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType)
get maximum IR value the module support
- int APC_API [APC_SetIRMode](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType)
set IR mode, left, right or both
- int APC_API [APC_GetIRMode](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType)
set IR mode, left, right or both
- int APC_API [APC_EnableSensorIF](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool blsEnable)
turn on/off sensor IF function
- int APC_API [APC_SetSensorTypeName](#) (void *pHandleApcDI, [SENSOR_TYPE_NAME](#) stn)
select which sensor to operate
- int APC_API [APC_EnableAE](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo)
enable auto exposure function of ISP
- int APC_API [APC_DisableAE](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo)
disable auto exposure function of ISP
- int APC_API [APC_EnableAWB](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo)
enable auto white balance function of ISP
- int APC_API [APC_DisableAWB](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo)
disable auto white balance of ISP
- int APC_API [APC_GetGPIOValue](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE *pValue)
get general purpose IO value
- int APC_API [APC_SetGPIOValue](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE nValue)
set GPIO value

- int APC_API [APC_SetGPIOCtrl](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE nValue)
set GPIO control address
- int APC_API [APC_GetPUPPropVal](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int *pValue)
get processing unit property value <https://msdn.microsoft.com/en-us/library/windows/hardware/ff568122-85> .aspx The PROPSETID_VIDCAP_VIDEOPROCAMP property set controls devices that can adjust image color attributes of analog or digital signals.
- int APC_API [APC_SetPUPPropVal](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int nValue)
get processing unit property value <https://msdn.microsoft.com/en-us/library/windows/hardware/ff568122-85> .aspx <https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089> (v=vs.↵85) .aspx
- int APC_API [APC_GetCTPropVal](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int *pValue)
set control terminal property value <https://msdn.microsoft.com/en-us/library/windows/hardware/ff567802-85> .aspx The PROPSETID_VIDCAP_CAMERACONTROL property set controls camera device settings. The controls it provides are a subset of the ITU T.RDC standard.
- int APC_API [APC_SetCTPropVal](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int nValue)
get control terminal property value <https://msdn.microsoft.com/en-us/library/windows/hardware/ff567802-85> .aspx <https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089> (v=vs.↵85) .aspx
- int APC_API **APC_GetAutoExposureMode** (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short *mode)
misc function : get auto exposure mode
- int APC_API **APC_SetAutoExposureMode** (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short mode)
misc function : set auto exposure mode
- int APC_API [APC_GetFlexibleGyroData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int length, BYTE *pGyroData)
get IMU(Gyro) data
- int APC_API [APC_GetFlexibleGyroLength](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short *GyroLen)
get the IMU(Gyro) data length
- int APC_API [APC_SetHuffmanTableData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, const char *filename, bool bLogFile)
set huffman table data for jpeg encode
- int APC_API [APC_SetQuantizationTableData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, const char *filename)
set quantication table data for jpeg encode
- int APC_API **APC_SetPlumAR0330** (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool bEnable)
Set Plum Sensor AR0330.
- int APC_API **APC_SetRootCipher** (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, const char *cipher)
Set Root Cipher to write the file id 30/40/50/240.
- int APC_API [APC_ResetUNPData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo)
Reset the UNProtection area's datum.
- int APC_API **APC_GetDevicePortType** (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, USB_PORT↵_TYPE *pUSB_Port_Type)
Get Device USB-port-type.
- int APC_API [APC_EnableGPUAcceleration](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool enable)
enable depth filter with GPU acceleration or not
- APC_API char * [APC_GetDepthFilterVersion](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo)
get depth filter version

5.1.1 Detailed Description

eYs3D SDK API export functions, data structure and variable definition

Copyright

This file copyright (C) 2017 by eYs3D company

An unpublished work. All rights reserved. This file is proprietary information, and may not be disclosed or copied without the prior permission of eYs3D company.

5.1.2 Typedef Documentation

5.1.2.1 APC_DeviceEventFn

```
typedef void(* APC_DeviceEventFn) (UINT pid, UINT vid, BOOL bAttached, void *pData)
```

Callback function to receive any USB capture device attachment or detachment events.

Parameters

<i>pid</i>	product id of the USB device
<i>vid</i>	vender id of the USB device
<i>bAttached</i>	TRUE if this is a USB device attached event, otherwise, it is a detached event.
<i>pData</i>	user defined data to pass to the callback function

Returns

none

5.1.2.2 APC_ImgCallbackFn

```
typedef void(* APC_ImgCallbackFn) (APCImageType::Value imgType, int imgId, unsigned char *imgBuf, int imgSize, int width, int height, int serialNumber, LONGLONG timestamp, void *pParam)
```

Callback function when video or data is ready.

Parameters

<i>pid</i>	product id of the USB device
<i>vid</i>	vender id of the USB device
<i>bAttached</i>	TRUE if this is a USB device attached event, otherwise, it is a detached event.
<i>pData</i>	user defined data to pass to the callback function

Returns

none

5.1.2.3 DEVINFORMATION

typedef struct tagDEVINFORMATION DEVINFORMATION

device information

5.1.2.4 eSPCtrl_RectLogData

typedef struct eSPCtrl_RectLogData eSPCtrl_RectLogData

Rectified log data structure

5.1.2.5 PARALUT

typedef struct ParaLUT PARALUT

Spherical look-up table conversion parameters

5.1.3 Enumeration Type Documentation

5.1.3.1 APC_DEVICE_TYPE

enum APC_DEVICE_TYPE

chip enum value

Enumerator

OTHERS	Other
AXES1	AXIS1
PUMA	PUMA

5.1.3.2 SENSOR_TYPE_NAME

enum SENSOR_TYPE_NAME

Enumerator

APC_SENSOR_TYPE_H22	H22
APC_SENSOR_TYPE_OV7740	OV7740
APC_SENSOR_TYPE_AR0134	AR0134
APC_SENSOR_TYPE_AR0135	AR0135
APC_SENSOR_TYPE_AR0144	AR0144
APC_SENSOR_TYPE_OV9714	OV9714
APC_SENSOR_TYPE_OV9282	OV9282
APC_SENSOR_TYPE_AR0330	AR0330
APC_SENSOR_TYPE_AR1335	AR1335

5.1.3.3 USERDATA_SECTION_INDEX

enum USERDATA_SECTION_INDEX

Enumerator

USERDATA_SECTION_0	Section 0
USERDATA_SECTION_1	Section 1
USERDATA_SECTION_2	Section 2
USERDATA_SECTION_3	Section 3
USERDATA_SECTION_4	Section 4
USERDATA_SECTION_5	Section 5
USERDATA_SECTION_6	Section 6
USERDATA_SECTION_7	Section 7
USERDATA_SECTION_8	Section 8
USERDATA_SECTION_9	Section 9
USERDATA_SECTION_10	Section 10
USERDATA_SECTION_NUM	Total Section Number

5.1.4 Function Documentation

5.1.4.1 APC_CloseDevice()

```
int APC_API APC_CloseDevice (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo)
```

close device and stop video render

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index

Returns

success:APC_OK, others:see [eSPDI_ErrCode.h](#)

5.1.4.2 APC_DisableAE()

```
int APC_API APC_DisableAE (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo)
```

disable auto exposure function of ISP

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.3 APC_DisableAWB()

```
int APC_API APC_DisableAWB (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo)
```

disable auto white balance of ISP

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.4 APC_EnableAE()

```
int APC_API APC_EnableAE (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo)
```

enable auto exposure function of ISP

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.5 APC_EnableAWB()

```
int APC_API APC_EnableAWB (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo)
```

enable auto white balance function of ISP

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.6 APC_EnableGPUAcceleration()

```
int APC_API APC_EnableGPUAcceleration (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    bool enable)
```

enable depth filter with GPU acceleration or not

Parameters

<i>pHandleApcDI</i>	the pointer to the initialized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>enable</i>	true:enable, false:disable

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.7 APC_FindDevice()

```
int APC_API APC_FindDevice (
    void * pHandleApcDI)
```

find out all eYs3D USB devices by PID, VID and ChipID, also remember device types

Parameters

<i>pHandleApcDI</i>	the pointer to the initialized ApcDI SDK instance
---------------------	---

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.8 APC_GetCTPropVal()

```
int APC_API APC_GetCTPropVal (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int nId,
    int * pValue)
```

set control terminal property value <https://msdn.microsoft.com/en-us/library/windows/hardware/ff567885> .aspx The PROPSETID_VIDCAP_CAMERACONTROL property set controls camera device settings. The controls it provides are a subset of the ITU T.RDC standard.

The KSPROPERTY_VIDCAP_CAMERACONTROL enumeration in Ksmedia.h specifies the properties of this set.

Support for this property set is optional and should be implemented only by minidrivers of devices that offer camera control settings. For more information, see the ITU website.

Prior to USB video class, this enumeration contained the following properties: KSPROPERTY_CAMERACONTROL↵
_EXPOSURE KSPROPERTY_CAMERACONTROL_FOCUS KSPROPERTY_CAMERACONTROL_IRIS KSPROPERTY↵
_CAMERACONTROL_ZOOM KSPROPERTY_CAMERACONTROL_PAN KSPROPERTY_CAMERACONTROL↵
_ROLL KSPROPERTY_CAMERACONTROL_TILT

<https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089> (v=vs.↵
85) .aspx

Parameters

<i>*pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>nId</i>	specifies the member of the property set
<i>pValue</i>	pointer of store CT property value

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.9 APC_GetCurrentIRValue()

```
int APC_API APC_GetCurrentIRValue (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    WORD * pwType)
```

get current infrared radiation(IR) value

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>pwType</i>	value of current IR

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.10 APC_GetDepthFilterVersion()

```
APC_API char * APC_GetDepthFilterVersion (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo)
```

get depth filter version

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index

Returns

success: get version string, others: get N/A string

5.1.4.11 APC_GetDeviceInfo()

```
int APC_API APC_GetDeviceInfo (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    DEVINFORMATION * pdevinfo)
```

get informations of eYs3D UVC devices, see @DEVINFORMATION

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>pdevinfo</i>	pointer of device information

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.12 APC_GetDeviceInfoEx()

```
int APC_API APC_GetDeviceInfoEx (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    DEVINFORMATIONEX * pdevinfo)
```

get target device info,

get target device info

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>pdevinfo</i>	pointer of buffer to store DEVINFORMATIONEX

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

Parameters

<i>void</i>	*pHandleApcDI the pointer to the initilized ApcDI SDK instance
<i>PDEVSELINFO</i>	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

Parameters

<i>void</i>	*pHandleApcDI the pointer to the initilized ApcDI SDK instance
<i>PDEVSELINFO</i>	pDevSelInfo pointer of device select index
<i>const</i>	char* cipher cipher to get root authority for device unprotection

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.13 APC_GetDeviceNumber()

```
int APC_API APC_GetDeviceNumber (
    void * pHandleApcDI)
```

get eYs3D USB device numbers

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
---------------------	--

Returns

number of eYs3D device

5.1.4.14 APC_GetDeviceResolutionList()

```
int APC_API APC_GetDeviceResolutionList (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int nMaxCount0,
    APC_STREAM_INFO * pStreamInfo0,
    int nMaxCount1,
    APC_STREAM_INFO * pStreamInfo1)
```

get the device resolution list

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>nMaxCount0</i>	max count of endpoint1 resolutions
<i>pStreamInfo0</i>	resolution infos of endpoint1
<i>nMaxCount1</i>	max count of endpoint2 resolutions
<i>pStreamInfo1</i>	resolutions infos of endpoint2

Returns

success: nCount0*256+nCount1, others: see [eSPDI_ErrCode.h](#)

5.1.4.15 APC_GetFlexibleGyroData()

```
int APC_API APC_GetFlexibleGyroData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int length,
    BYTE * pGyroData)
```

get IMU(Gyro) data

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>length</i>	length of IMU data to read, should be get from APC_GetFlexibleGyroLength
<i>pGyroData</i>	data buffer to store IMU data

5.1.4.16 APC_GetFlexibleGyroLength()

```
int APC_API APC_GetFlexibleGyroLength (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    unsigned short * GyroLen)
```

get the IMU(Gyro) data length

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>GyroLen</i>	pointer to store IMU data length

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.17 APC_GetFWRegister()

```
int APC_API APC_GetFWRegister (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    unsigned short address,
    unsigned short * pValue,
    int flag)
```

get firmware register value

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>address</i>	register address
<i>pValue</i>	pointer of value got from register address
<i>flag</i>	address and value data length(2 or 1 byte) ie FG_Address_2Byte FG_Value_2Byte is 2 byte address and 2 byte value #define FG_Address_1Byte 0x01 #define FG_Address_2Byte 0x02 #define FG_Value_1Byte 0x10 #define FG_Value_2Byte 0x20

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.18 APC_GetFwVersion()

```
int APC_API APC_GetFwVersion (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    char * pszFwVersion,
    int nBufferSize,
    int * pActualLength)
```

get the firmware version of device, the version is a string

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>pszFwVersion</i>	firmware version string
<i>nBufferSize</i>	input buffer length to receive FW version
<i>pActualLength</i>	the actual length of FW version in byte

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.19 APC_GetGPIOValue()

```
int APC_API APC_GetGPIOValue (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int nGPIOIndex,
    BYTE * pValue)
```

get general purpose IO value

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>nGPIOIndex</i>	GPIO index, 1 or 2 is valid
<i>pValue</i>	pointer of GPIO value

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.20 APC_GetHWRegister()

```
int APC_API APC_GetHWRegister (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    unsigned short address,
    unsigned short * pValue,
    int flag)
```

get hardware register value

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index

<i>address</i>	register address
<i>pValue</i>	pointer of value got from register address
<i>flag</i>	address and value data length(2 or 1 byte) ie FG_Address_2Byte FG_Value_2Byte is 2 byte address and 2 byte value #define FG_Address_1Byte 0x01 #define FG_Address_2Byte 0x02 #define FG_Value_1Byte 0x10 #define FG_Value_2Byte 0x20

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.21 APC_GetIRMaxValue()

```
int APC_API APC_GetIRMaxValue (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo,  
    WORD * pwType)
```

get maximum IR value the module support

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>pwType</i>	pointer strors maximum IR value

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.22 APC_GetIRMinValue()

```
int APC_API APC_GetIRMinValue (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo,  
    WORD * pwType)
```

get minimum IR value the module support

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>pwType</i>	pointer strors minimum IR value

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.23 APC_GetLogData()

```
int APC_API APC_GetLogData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

get log data from flash

Parameters

<i>pHandleApcDI</i>	the pointer to the initialized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store log data
<i>BufferLength</i>	input buffer length
<i>pActualLength</i>	actual length has written to buffer
<i>index</i>	index to identify log data for corresponding depth

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.24 APC_GetMultiBytesHWRegister()

```
int APC_API APC_GetMultiBytesHWRegister (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    UCHAR * data,
    int address,
    int size)
```

get hardware register with multibytes

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>data</i>	buffer to keep firmware read back
<i>address</i>	flash address start to read
<i>size</i>	buffer length

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.25 APC_GetPidVid()

```
int APC_API APC_GetPidVid (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    unsigned short * pPidBuf,
    unsigned short * pVidBuf)
```

get PID(product ID) and VID(vendor ID) of device

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>pPidBuf</i>	4 byte buffer to store PID value
<i>pVidBuf</i>	4 byte buffer to store VID value

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.26 APC_GetPUPPropVal()

```
int APC_API APC_GetPUPPropVal (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int nId,
    int * pValue)
```

get processing unit property value <https://msdn.microsoft.com/en-us/library/windows/hardware/ff568185> .aspx The PROPSETID_VIDCAP_VIDEOPROCAMP property set controls devices that can adjust image color attributes of analog or digital signals.

The KSPROPERTY_VIDCAP_VIDEOPROCAMP enumeration in ksmedia.h specifies the properties of this set.

Support for this property set is optional and should be implemented only by devices that allow adjustment of brightness, contrast, hue, and other image quality settings.

Prior to USB video class, this enumeration contained the following property items: KSPROPERTY_VIDEOPROCAMP_BACKLIGHT_COMPENSATION KSPROPERTY_VIDEOPROCAMP_BRIGHTNESS KSPROPERTY_VIDEOPROCAMP_COLOREnable KSPROPERTY_VIDEOPROCAMP_CONTRAST KSPROPERTY_VIDEOPROCAMP_GAMMA KSPROPERTY_VIDEOPROCAMP_HUE KSPROPERTY_VIDEOPROCAMP_SATURATION KSPROPERTY_VIDEOPROCAMP_SHARPNESS KSPROPERTY_VIDEOPROCAMP_WHITEBALANCE KSPROPERTY_VIDEOPROCAMP_GAIN

<https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089> (v=vs.85) .aspx The KSPROPERTY_VIDEOPROCAMP_S structure describes filter-based property settings in the PROPSETID_VIDCAP_VIDEOPROCAMP property set.

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>nId</i>	specifies the member of the property set
<i>pValue</i>	pointer of store PU property value

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.27 APC_GetSensorRegister()

```
int APC_API APC_GetSensorRegister (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int nId,
    unsigned short address,
    unsigned short * pValue,
    int flag,
    int nSensorMode)
```

get value from sensor register

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>nId</i>	sensor slave address. see SENSOR_TYPE_NAME enum definition
<i>address</i>	register address
<i>pValue</i>	pointer of value got from register address
<i>flag</i>	address and value data length(2 or 1 byte) ie FG_Address_2Byte FG_Value_2Byte is 2 byte address and 2 byte value #define FG_Address_1Byte 0x01 #define FG_Address_2Byte 0x02 #define FG_Value_1Byte 0x10 #define FG_Value_2Byte 0x20
<i>nSensorMode</i>	sensor mode(sensor A, B or Both) A is 0, B is 1, Both is 2

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.28 APC_GetSlaveHWRegister()

```
int APC_API APC_GetSlaveHWRegister (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    unsigned short address,
    unsigned short * pValue,
    int flag)
```

get hardware register value

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>address</i>	register address
<i>pValue</i>	pointer of value got from register address
<i>flag</i>	address and value data length(2 or 1 byte) ie FG_Address_2Byte FG_Value_2Byte is 2 byte address and 2 byte value #define FG_Address_1Byte 0x01 #define FG_Address_2Byte 0x02 #define FG_Value_1Byte 0x10 #define FG_Value_2Byte 0x20

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.29 APC_GetSlaveLogData()

```
int APC_API APC_GetSlaveLogData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

get log data from flash

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store log data
<i>BufferLength</i>	input buffer length
<i>pActualLength</i>	actual length has written to buffer
<i>index</i>	index to identify log data for corresponding depth

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.30 APC_GetSlaveSensorRegister()

```
int APC_API APC_GetSlaveSensorRegister (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int nId,
    unsigned short address,
    unsigned short * pValue,
    int flag,
    int nSensorMode)
```

get value from sensor register

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>nId</i>	sensor slave address. see SENSOR_TYPE_NAME enum definition
<i>address</i>	register address
<i>pValue</i>	pointer of value got from register address
<i>flag</i>	address and value data length(2 or 1 byte) ie FG_Address_2Byte FG_Value_2Byte is 2 byte address and 2 byte value #define FG_Address_1Byte 0x01 #define FG_Address_2Byte 0x02 #define FG_Value_1Byte 0x10 #define FG_Value_2Byte 0x20
<i>nSensorMode</i>	sensor mode(sensor A, B or Both) A is 0, B is 1, Both is 2

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.31 APC_Init()

```
int APC_API APC_Init (
    void ** ppHandleApcDI,
    bool bIsLogEnabled)
```

entry point of eYs3D camera SDK. This API allocates resource and find all the eSPI camera devices connected to the system.

Parameters

<i>ppHandleApcDI</i>	a pointer of pointer to receive ApcDI SDK instance
<i>bIsLogEnabled</i>	set to true to generate log file, named log.txt in current folder

Returns

success: none negative integer to indicate numbers of devices found in the system.

5.1.4.32 APC_Init2()

```
int APC_API APC_Init2 (
    void ** ppHandleApcDI,
    bool bIsLogEnabled,
    bool bEnableAutoRestart)
```

entry point of eYs3D camera SDK. This API allocates resource and find all the eSPI camera devices connected to the system.

Parameters

<i>ppHandleApcDI</i>	a pointer of pointer to receive ApcDI SDK instance
<i>bIsLogEnabled</i>	set to true to generate log file, named log.txt in current folder
<i>bEnableAutoRestart</i>	set true to auto-restart the device if the device was detached and attached again.

Returns

success: none negative integer to indicate numbers of devices found in the system.

Note

Calls APC_Init or APC_Init2 to initialize the ApcDI SDK. APC_Init2 adds the auto-restart function to the initialization options. If you call APC_Init, the bEnableAutoRestart is set as disabled.

5.1.4.33 APC_Is360Device()

```
bool APC_API APC_Is360Device (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo)
```

check module is spherical device or not

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index

Returns

true: module support 360, false: not support

5.1.4.34 APC_OpenDevice()

```
int APC_API APC_OpenDevice (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int colorStreamIndex,
    int depthStreamIndex,
    int depthStreamSwitch,
    int iFps,
    APC_ImgCallbackFn callbackFn,
    void * pCallbackParam,
    int pid = -1)
```

open camera device with image callback support

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>colorStreamIndex</i>	index of the desired color stream
<i>depthStreamIndex</i>	index of the desired sdepth tream
<i>depthStreamSwitch</i>	depth switch for S0, S1 or S2
<i>iFps</i>	pointer to the desired frame rate, returns the actual frame rate.
<i>callbackFn</i>	set image callback function
<i>pCallbackParam</i>	the data to associate with the callback function
<i>pid</i>	Specify device pid.

Table 5.41 Image Control Mode	
Mode	Description
0x01	color and depth frame output synchronously, for depth map module only
0x02	enable post-process, for Depth Map module only
0x04	stitch images if this bit is set, for fisheye spherical module only
0x08	use OpenCL in stitching. This bit effective only when bit-2 is set.

Returns

success:APC_OK, others:see [eSPDI_ErrCode.h](#)

5.1.4.35 APC_ReadFlashData()

```
int APC_API APC_ReadFlashData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    FLASH_DATA_TYPE fdt,
    BYTE * pBuffer,
    unsigned long int nLengthOfBuffer,
    unsigned long int * pActualBufferLen)
```

read firmware code(.bin) form flash The firmware code is the combination of boot loader, firmware body and plug-in data. This input buffer length has to match with the flash data type

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>fdt</i>	segment type of flash be read
<i>pBuffer</i>	buffer to store firmware code
<i>nLengthOfBuffer</i>	input buffer length
<i>pActualBufferLen</i>	actual length has written to pBuffer

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.36 APC_RefreshDevice()

```
int APC_API APC_RefreshDevice (
    void * pHandleApcDI)
```

refresh all eYs3D UVC devices

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
---------------------	--

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.37 APC_RegisterDeviceEvents()

```
int APC_API APC_RegisterDeviceEvents (
    void * pHandleApcDI,
    APC_DeviceEventFn cbFunc,
    void * pData)
```

Register the USB device plug or unplug events. Any USB capture device attachment or detachment events will call the callback function cbFunc.

Parameters

<i>pHandleApcDI</i>	a pointer to ApcDI SDK instance
<i>cbFunc</i>	a callback function of type APC_DeviceEventFn that will receive USB capture device events when the device is attached or detached.
<i>pData</i>	user defined data which will send to the callback function

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.38 APC_Release()

```
void APC_API APC_Release (
    void ** ppHandleApcDI)
```

release all resource that APC_Init had allocated

Parameters

<i>ppHandleApcDI</i>	pointer of the pointer to the initialized ApcDI SDK instance.
----------------------	---

Returns

none

Note

the pointer to ppHandleApcDI will be set to NULL when this call returns successfully.

5.1.4.39 APC_ResetUNPData()

```
int APC_API APC_ResetUNPData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo)
```

Reset the UNProtection area's datum.

Parameters

<i>void</i>	*pHandleApcDI CApcDI handler
<i>PDEVSELINFO</i>	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.40 APC_SetCTPropVal()

```
int APC_API APC_SetCTPropVal (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int nId,
    int nValue)
```

get control terminal property value <https://msdn.microsoft.com/en-us/library/windows/hardware/ff567885.aspx> [https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089(v=vs.85).aspx)

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>nId</i>	specifies the member of the property set
<i>nValue</i>	CT property value to set

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.41 APC_SetCurrentIRValue()

```
int APC_API APC_SetCurrentIRValue (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    WORD wType)
```

set current infrared radiation(IR) value

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>wType</i>	value to set

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.42 APC_SetFWRegister()

```
int APC_API APC_SetFWRegister (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    unsigned short address,
    unsigned short nValue,
    int flag)
```

set firmware register value

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>address</i>	register address
<i>nValue</i>	register value to set
<i>flag</i>	address and value data length(2 or 1 byte) ie FG_Address_1Byte FG_Value_1Byte is 1 byte address and 1 byte value #define FG_Address_1Byte 0x01 #define FG_Address_2Byte 0x02 #define FG_Value_1Byte 0x10 #define FG_Value_2Byte 0x20

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.43 APC_SetGPIOCtrl()

```
int APC_API APC_SetGPIOCtrl (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo,  
    int nGPIOIndex,  
    BYTE nValue)
```

set GPIO control address

Parameters

<i>nGPIOIndex</i>	index of GPIO (1 ~ 4)
<i>nValue</i>	register value to set

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.44 APC_SetGPIOValue()

```
int APC_API APC_SetGPIOValue (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo,  
    int nGPIOIndex,  
    BYTE nValue)
```

set GPIO value

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>nGPIOIndex</i>	GPIO index, 1 or 2 is valid
<i>nValue</i>	GPIO value to set

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.45 APC_SetHuffmanTableData()

```
int APC_API APC_SetHuffmanTableData (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo,  
    const char * filename,  
    bool bLogFile)
```

set huffman table data for jpeg encode

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>filename</i>	huffman table file, see jh_vga_422.dat sample file
<i>bLogFile</i>	if true then puma_htable.dat file is generated

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.46 APC_SetHWRegister()

```
int APC_API APC_SetHWRegister (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo,  
    unsigned short address,  
    unsigned short nValue,  
    int flag)
```

set hardware register

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>address</i>	register address
<i>nValue</i>	register value to set
<i>flag</i>	address and value data length(2 or 1 byte) ie FG_Address_1Byte FG_Value_1Byte is 1 byte address and 1 byte value #define FG_Address_1Byte 0x01 #define FG_Address_2Byte 0x02 #define FG_Value_1Byte 0x10 #define FG_Value_2Byte 0x20

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.47 APC_SetIRMaxValue()

```
int APC_API APC_SetIRMaxValue (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo,  
    WORD wType)
```

set maximum IR value the module support

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>wType</i>	pointer strors maximum IR value

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.48 APC_SetIRMaxValueUnleashed()

```
int APC_API APC_SetIRMaxValueUnleashed (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    WORD wType)
```

set maximum IR value the module support without any limitation

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>wType</i>	pointer strors maximum IR value

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.49 APC_SetLogData()

```
int APC_API APC_SetLogData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

set log data to flash

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store log data
<i>BufferLength</i>	input buffer length
<i>pActualLength</i>	actual length has written to buffer
<i>index</i>	index to identify log data for corresponding depth

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.50 APC_SetLogData_Advanced()

```
int APC_API APC_SetLogData_Advanced (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

set log data to flash

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store log data
<i>BufferLength</i>	input buffer length
<i>pActualLength</i>	actual length has written to buffer
<i>index</i>	index to identify log data for corresponding depth

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.51 APC_SetMultiBytesHWRegister()

```
int APC_API APC_SetMultiBytesHWRegister (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    UCHAR * data,
    int address,
    int size)
```

set hardware register with multibytes

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>data</i>	buffer to keep firmware want to write
<i>address</i>	flash address start to write
<i>size</i>	buffer length

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.52 APC_SetPidVid()

```
int APC_API APC_SetPidVid (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    unsigned short * pPidBuf,
    unsigned short * pVidBuf)
```

set PID and VID to device

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>pPidBuf</i>	4 byte PID value buffer to set
<i>pVidBuf</i>	4 byte VID value buffer to set

Returns

success: EtronDI_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.53 APC_SetPUPPropVal()

```
int APC_API APC_SetPUPPropVal (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int nId,
    int nValue)
```

get processing unit property value <https://msdn.microsoft.com/en-us/library/windows/hardware/ff568185.aspx> [https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089(v=vs.85).aspx)

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>nId</i>	specifies the member of the property set
<i>nValue</i>	value to set

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.54 APC_SetQuantizationTableData()

```
int APC_API APC_SetQuantizationTableData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    const char * filename)
```

set quantication table data for jpeg encode

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>filename</i>	quantization table file, see FS_DEF_010.txt sample file

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.55 APC_SetSensorTypeName()

```
int APC_API APC_SetSensorTypeName (
    void * pHandleApcDI,
    SENSOR_TYPE_NAME stn)
```

select which sensor to operate

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>stn</i>	sensor type

Returns

APC_OK

5.1.4.56 APC_SetSlaveHWRegister()

```
int APC_API APC_SetSlaveHWRegister (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    unsigned short address,
    unsigned short nValue,
    int flag)
```

set hardware register

Parameters

<i>pHandleApcDI</i>	CApcDI handler
<i>pDevSelInfo</i>	pointer of device select index
<i>address</i>	register address
<i>nValue</i>	register value to set
<i>flag</i>	address and value data length(2 or 1 byte) ie FG_Address_1Byte FG_Value_1Byte is 1 byte address and 1 byte value #define FG_Address_1Byte 0x01 #define FG_Address_2Byte 0x02 #define FG_Value_1Byte 0x10 #define FG_Value_2Byte 0x20

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.57 APC_SetSlaveLogData()

```
int APC_API APC_SetSlaveLogData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

set log data to flash

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index

<i>buffer</i>	buffer to store log data
<i>BufferLength</i>	input buffer length
<i>pActualLength</i>	actual length has written to buffer
<i>index</i>	index to identify log data for corresponding depth

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.1.4.58 APC_SetUserData()

```
int APC_API APC_SetUserData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    USERDATA_SECTION_INDEX usi)
```

set user data to flash

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store user data
<i>BufferLength</i>	input buffer length
<i>usi</i>	which user index data to select

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.2 eSPDI_Common.h

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

```
00001
00009 #pragma once
00010
00011 #include <windows.h>
00012 #include <io.h>
00013
00014 #ifndef APC_API
00015 #ifdef __WEYE__
00016 #define APC_API
00017 #else
00018 #ifdef APC_EXPORTS
00019 #define APC_API __declspec(dllexport)
00020 #else
00021 #define APC_API __declspec(dllimport)
00022 #endif
00023 #endif
00024 #endif
00025
00026 #ifndef BYTE
00027 typedef unsigned char BYTE;
```

```

00028 #endif //BYTE
00029
00030 #ifndef WORD
00031 typedef unsigned short WORD;
00032 #endif //WORD
00033
00034 #ifndef WCHAR
00035 typedef wchar_t WCHAR;
00036 #endif //WCHAR
00037
00038 #ifndef BOOL
00039 typedef signed int BOOL;
00040 #endif //BOOL
00041
00042 //
00043 // C++ compatibility
00044 //
00045 #ifdef __cplusplus
00046 extern "C" {
00047 #endif
00048
00049 #ifndef CALLBACK
00050 #define CALLBACK __stdcall
00051 #endif //CALLBACK
00052
00053
00054 #define APC_MAX_STREAM_COUNT 64
00055 #define APC_MAX_DEPTH_STREAM_COUNT 8
00056 #pragma pack(push, 1)
00057 typedef struct tagAPC_STREAM_INFO {
00058     int        nWidth;
00059     int        nHeight;
00060     BOOL       bFormatMJPEG;
00061 } APC_STREAM_INFO, *PAPC_STREAM_INFO;
00062 #pragma pack(pop)
00063
00064 #ifndef WM_MYMSG_NOTICE_CAPTURE
00065 #define WM_MYMSG_NOTICE_CAPTURE (WM_USER+101)
00066 #endif
00067
00068 #include "eSPDI_ErrCode.h"
00069
00070 /* APC Stream Index */
00071 #define APC_Stream_Color 0
00072 #define APC_Stream_Track 4
00073 #define APC_Stream_Kolor 5
00074
00075 /* APC VID */
00076 #define APC_VID_0x1E4E 0x1E4E // old VID
00077 #define APC_VID_0x3438 0x3438
00078
00079 /* APC PID */
00080 #define APC_PID_8029      0x0568
00081 #define APC_PID_8030      APC_PID_8029
00082 #define APC_PID_8039      APC_PID_8029
00083 #define APC_PID_8031      0x0117
00084 #define APC_PID_8032      0x0118
00085 #define APC_PID_8036      0x0120
00086 #define APC_PID_8037      0x0121
00087 #define APC_PID_8038      0x0124
00088 #define APC_PID_8038_M0   APC_PID_8038
00089 #define APC_PID_8038_M1   0x0147
00090 #define APC_PID_8040W      0x0130
00091 #define APC_PID_8040S      0x0131
00092 #define APC_PID_8040S_K    0x0149
00093 #define APC_PID_8041      0x0126
00094 #define APC_PID_8042      0x0127
00095 #define APC_PID_8043      0x0128
00096 #define APC_PID_8044      0x0129
00097 #define APC_PID_8045K      0x0134
00098 #define APC_PID_8046K      0x0135
00099 #define APC_PID_8051      0x0136
00100 #define APC_PID_8052      0x0137
00101 #define APC_PID_8053      0x0138
00102 #define APC_PID_8054      0x0139
00103 #define APC_PID_8054_K    0x0143
00104 #define APC_PID_8059      0x0146
00105 #define APC_PID_8060      0x0152
00106 #define APC_PID_8060_K    0x0150
00107 #define APC_PID_8060_T    0x0151
00108 #define APC_PID_AMBER      0x0112
00109 #define APC_PID_SALLY      0x0158
00110 #define APC_PID_8062      0x0162
00111 #define APC_PID_8063      0x0164
00112 #define APC_PID_8063_K    0x0165
00113 #define APC_PID_HYPATIA    0x0160 // XY8071
00114 #define APC_PID_HYPATIA2   0x0173

```

```

00115 #define APC_PID_8072 0x0180
00116 #define APC_PID_SANDRA 0x0167
00117 #define APC_PID_NORA 0x0168
00118 #define APC_PID_HELEN 0x0171
00119 #define APC_PID_GRAPE 0x0202
00120 #define APC_PID_IVY 0x0177
00121 #define APC_PID_IVY2 0x0191
00122 #define APC_PID_IVY3 0x0192
00123 #define APC_PID_IVY2_S 0x0195
00124 #define APC_PID_IVY4 0x0198
00125 #define APC_PID_80362 0x0181
00126 #define APC_PID_8077 0x0182
00127 #define APC_PID_8081 0x0183
00128 #define APC_PID_IRIS 0x0184
00129 #define APC_PID_MARY 0x0174
00130 #define APC_PID_FRANK 0x0187
00131 #define APC_PID_STACY 0x0188
00132 #define APC_PID_STACYJUNIOR 0x0189
00133 #define APC_PID_TARYN 0x0199
00134 #define APC_PID_HYPATIA4 0x0204
00135
00136 #define BIT_SET(a,b) ((a) |= (1<<(b)))
00137 #define BIT_CLEAR(a,b) ((a) &= ~(1<<(b)))
00138 #define BIT_FLIP(a,b) ((a) ^= (1<<(b)))
00139 #define BIT_CHECK(a,b) ((a) & (1<<(b)))
00140
00141 #define FG_Address_1Byte 0x01
00142 #define FG_Address_2Byte 0x02
00143 #define FG_Value_1Byte 0x10
00144 #define FG_Value_2Byte 0x20
00145
00146 // For Depth Data Type - 2016/12/14 by Sean
00147 #define APC_DEPTH_DATA_DEFAULT 0
00148 #define APC_DEPTH_DATA_OFF_RAW 0 // raw (depth off, only raw color)
00149 #define APC_DEPTH_DATA_8_BITS 1 // rectify
00150 #define APC_DEPTH_DATA_14_BITS 2 // rectify
00151 #define APC_DEPTH_DATA_8_BITS_x80 3 // rectify
00152 #define APC_DEPTH_DATA_11_BITS 4 // rectify
00153 #define APC_DEPTH_DATA_OFF_RECTIFY 5 // rectify (depth off, only rectify color)
00154 #define APC_DEPTH_DATA_8_BITS_RAW 6 // raw
00155 #define APC_DEPTH_DATA_14_BITS_RAW 7 // raw
00156 #define APC_DEPTH_DATA_8_BITS_x80_RAW 8 // raw
00157 #define APC_DEPTH_DATA_11_BITS_RAW 9 // raw
00158 #define APC_DEPTH_DATA_8_BITS_COMBINED_RECTIFY 10// multi-baseline
00159 #define APC_DEPTH_DATA_14_BITS_COMBINED_RECTIFY 11// multi-baseline
00160 #define APC_DEPTH_DATA_8_BITS_x80_COMBINED_RECTIFY 12// multi-baseline
00161 #define APC_DEPTH_DATA_11_BITS_COMBINED_RECTIFY 13// multi-baseline
00162
00163 // For Inter-Leave-Mode Depth Data Type
00164 #define APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET 16
00165 #define APC_DEPTH_DATA_ILM_DEFAULT 16
00166 #define APC_DEPTH_DATA_ILM_OFF_RAW 16 // raw (depth off, only raw color)
00167 #define APC_DEPTH_DATA_ILM_8_BITS 17 // rectify
00168 #define APC_DEPTH_DATA_ILM_14_BITS 18 // rectify
00169 #define APC_DEPTH_DATA_ILM_8_BITS_x80 19 // rectify
00170 #define APC_DEPTH_DATA_ILM_11_BITS 20 // rectify
00171 #define APC_DEPTH_DATA_ILM_OFF_RECTIFY 21 // rectify (depth off, only rectify color)
00172 #define APC_DEPTH_DATA_ILM_8_BITS_RAW 22 // raw
00173 #define APC_DEPTH_DATA_ILM_14_BITS_RAW 23 // raw
00174 #define APC_DEPTH_DATA_ILM_8_BITS_x80_RAW 24 // raw
00175 #define APC_DEPTH_DATA_ILM_11_BITS_RAW 25 // raw
00176 #define APC_DEPTH_DATA_ILM_8_BITS_COMBINED_RECTIFY 26// multi-baseline
00177 #define APC_DEPTH_DATA_ILM_14_BITS_COMBINED_RECTIFY 27// multi-baseline
00178 #define APC_DEPTH_DATA_ILM_8_BITS_x80_COMBINED_RECTIFY 28// multi-baseline
00179 #define APC_DEPTH_DATA_ILM_11_BITS_COMBINED_RECTIFY 29// multi-baseline
00180
00181 #define APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET 32
00182 #define APC_DEPTH_DATA_SCALE_DOWN_MODE_OFF_RAW (APC_DEPTH_DATA_OFF_RAW +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET)/* raw (depth off, only raw color) */
00183 #define APC_DEPTH_DATA_SCALE_DOWN_DEFAULT (APC_DEPTH_DATA_DEFAULT +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* raw (depth off, only raw color) */
00184 #define APC_DEPTH_DATA_SCALE_DOWN_8_BITS (APC_DEPTH_DATA_8_BITS +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET)/* rectify, 1 byte per pixel */
00185 #define APC_DEPTH_DATA_SCALE_DOWN_14_BITS (APC_DEPTH_DATA_14_BITS +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* rectify, 2 byte per pixel */
00186 #define APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80 (APC_DEPTH_DATA_8_BITS_x80 +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* rectify, 2 byte per pixel but using 1 byte only */
00187 #define APC_DEPTH_DATA_SCALE_DOWN_11_BITS (APC_DEPTH_DATA_11_BITS +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET)/* rectify, 2 byte per pixel but using 11 bit only */
00188 #define APC_DEPTH_DATA_SCALE_DOWN_OFF_RECTIFY (APC_DEPTH_DATA_OFF_RECTIFY +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* Rule 0.4b Reserved unused in any firmware*/
00189 #define APC_DEPTH_DATA_SCALE_DOWN_8_BITS_RAW (APC_DEPTH_DATA_8_BITS_RAW +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* raw */
00190 #define APC_DEPTH_DATA_SCALE_DOWN_14_BITS_RAW (APC_DEPTH_DATA_14_BITS_RAW +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* raw */
00191 #define APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80_RAW (APC_DEPTH_DATA_8_BITS_x80_RAW +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* raw */

```



```

00192 #define APC_DEPTH_DATA_SCALE_DOWN_11_BITS_RAW          (APC_DEPTH_DATA_11_BITS_RAW +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* raw */
00193 #define APC_DEPTH_DATA_SCALE_DOWN_14_BITS_COMBINED_RECTIFY
(APC_DEPTH_DATA_14_BITS_COMBINED_RECTIFY + APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* Rule 0.4b
Reserved unused in any firmware*/
00194 #define APC_DEPTH_DATA_SCALE_DOWN_11_BITS_COMBINED_RECTIFY
(APC_DEPTH_DATA_11_BITS_COMBINED_RECTIFY + APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* Rule 0.4b
Reserved unused in any firmware*/
00195
00196 // For Interleave mode depth data type
00197 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_OFF_RAW          (APC_DEPTH_DATA_SCALE_DOWN_OFF_RAW +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* raw (depth off, only raw color) */
00198 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_DEFAULT          (APC_DEPTH_DATA_SCALE_DOWN_DEFAULT +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* raw (depth off, only raw color) */
00199 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS          (APC_DEPTH_DATA_SCALE_DOWN_8_BITS +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* rectify, 1 byte per pixel */
00200 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS          (APC_DEPTH_DATA_SCALE_DOWN_14_BITS +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* rectify, 2 byte per pixel */
00201 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_x80      (APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80 +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* rectify, 2 byte per pixel but using 1 byte only */
00202 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS          (APC_DEPTH_DATA_SCALE_DOWN_11_BITS +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* rectify, 2 byte per pixel but using 11 bit only */
00203 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_OFF_RECTIFY      (APC_DEPTH_DATA_SCALE_DOWN_OFF_RECTIFY +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* rectify (depth off, only rectify color) */
00204 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_RAW      (APC_DEPTH_DATA_SCALE_DOWN_8_BITS_RAW +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* raw */
00205 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS_RAW      (APC_DEPTH_DATA_SCALE_DOWN_14_BITS_RAW +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* raw */
00206 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_x80_RAW   (APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80_RAW +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* raw */
00207 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS_RAW      (APC_DEPTH_DATA_SCALE_DOWN_11_BITS_RAW +
APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /* raw */
00208 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS_COMBINED_RECTIFY
(APC_DEPTH_DATA_SCALE_DOWN_14_BITS_COMBINED_RECTIFY + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) //
00209 #define APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS_COMBINED_RECTIFY
(APC_DEPTH_DATA_SCALE_DOWN_11_BITS_COMBINED_RECTIFY + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) //
multi-baseline
00210
00211
00212 // For Flash Read/Write
00213 // Firmware (size in KBytes)
00214 #define APC_READ_FLASH_TOTAL_SIZE          128
00215 #define APC_READ_FLASH_FW_PLUGIN_SIZE      104
00216 #define APC_WRITE_FLASH_TOTAL_SIZE         128
00217 #define APC_READ_FLASH_TOTAL_SIZE_256      256
00218 #define APC_WRITE_FLASH_TOTAL_SIZE_256     256
00219
00220 /*
00221     The group 1 is the factory settings which are calibrated before shipment.
00222     The group 2 is the factory settings after post calibration.
00223     FW Register 0xF6 is the offset.
00224     The default offset is set as 5 which means 10 divided by 2 groups.
00225 */
00226 #define FW_FID_GROUP_OFFSET          5
00227 #define MD5_SIGNATURE_BYTE_SIZE      32
00228 #define FW_PROTECT_STRUCT_LEN_OF_STI 17
00229
00230 // PlugIn data (size in bytes)
00231 #define APC_Y_OFFSET_FILE_ID_0       30
00232 #define APC_Y_OFFSET_FILE_SIZE       256
00233 #define APC_RECTIFY_FILE_ID_0        40
00234 #define APC_RECTIFY_FILE_SIZE        1024
00235 #define APC_ZD_TABLE_FILE_ID_0       50
00236 #define APC_ZD_TABLE_FILE_SIZE       4096
00237 #define APC_CALIB_LOG_FILE_ID_0      240
00238 #define APC_CALIB_LOG_FILE_SIZE      4096
00239 #define APC_USER_DATA_FILE_ID_0      200
00240 #define APC_USER_DATA_FILE_SIZE_0    1024
00241 #define APC_USER_DATA_FILE_SIZE_1    4096
00242 #define APC_USER_DATA_FILE_SIZE_2    128
00243 #define APC_USER_DATA_FILE_SIZE_3    1024
00244 #define APC_USER_DATA_FILE_SIZE_4    4096
00245
00246 //=====
00247 //
00248 // Property Type
00249 //
00250 #define PROP_TYPE_PU          0
00251 #define PROP_TYPE_CT          1
00252 //
00253 // PU Property ID
00254 //
00255 #define PU_PROPERTY_ID_BRIGHTNESS          0
00256 #define PU_PROPERTY_ID_CONTRAST            1
00257 #define PU_PROPERTY_ID_HUE                 2
00258 #define PU_PROPERTY_ID_SATURATION          3
00259 #define PU_PROPERTY_ID_SHARPNESS           4

```

```

00260 #define      PU_PROPERTY_ID_GAMMA                      5
00261 #define      PU_PROPERTY_ID_COLORENABLE                 6
00262 #define      PU_PROPERTY_ID_WHITEBALANCE                7
00263 #define      PU_PROPERTY_ID_BACKLIGHT_COMPENSATION      8
00264 #define      PU_PROPERTY_ID_GAIN                       9
00265 #define      PU_PROPERTY_ID_DIGITAL_MULTIPLIER         10
00266 #define      PU_PROPERTY_ID_DIGITAL_MULTIPLIER_LIMIT   11
00267 #define      PU_PROPERTY_ID_WHITEBALANCE_COMPONENT     12
00268 #define      PU_PROPERTY_ID_POWERLINE_FREQUENCY        13
00269 //
00270 // CT Property ID
00271 //
00272 #define      CT_PROPERTY_ID_PAN                        0
00273 #define      CT_PROPERTY_ID_TILT                      1
00274 #define      CT_PROPERTY_ID_ROLL                      2
00275 #define      CT_PROPERTY_ID_ZOOM                      3
00276 #define      CT_PROPERTY_ID_EXPOSURE                 4
00277 #define      CT_PROPERTY_ID_IRIS                     5
00278 #define      CT_PROPERTY_ID_FOCUS                    6
00279 #define      CT_PROPERTY_ID_SCANMODE                 7
00280 #define      CT_PROPERTY_ID_PRIVACY                  8
00281 #define      CT_PROPERTY_ID_PANTILT                   9
00282 #define      CT_PROPERTY_ID_PAN_RELATIVE             10
00283 #define      CT_PROPERTY_ID_TILT_RELATIVE            11
00284 #define      CT_PROPERTY_ID_ROLL_RELATIVE            12
00285 #define      CT_PROPERTY_ID_ZOOM_RELATIVE            13
00286 #define      CT_PROPERTY_ID_EXPOSURE_RELATIVE        14
00287 #define      CT_PROPERTY_ID_IRIS_RELATIVE            15
00288 #define      CT_PROPERTY_ID_FOCUS_RELATIVE           16
00289 #define      CT_PROPERTY_ID_PANTILT_RELATIVE         17
00290 #define      CT_PROPERTY_ID_AUTO_EXPOSURE_PRIORITY   19
00291
00292 //=====
00293
00294
00295 typedef struct tagZDTableInfo
00296 {
00297     int nIndex;
00298     int nDataType;
00299 } ZDTABLEINFO, *PZDTABLEINFO;
00300
00306 typedef struct eSPCtrl_RectLogData
00307 {
00308     union {
00309         BYTE uByteArray[1024];
00310         struct {
00311             WORD    InImgWidth;
00312             WORD    InImgHeight;
00313             WORD    OutImgWidth;
00314             WORD    OutImgHeight;
00315             int     RECT_ScaleEnable;
00316             int     RECT_CropEnable;
00317             WORD    RECT_ScaleWidth;
00318             WORD    RECT_ScaleHeight;
00319             float   CamMat1[9];
00320             float   CamDist1[8];
00321             float   CamMat2[9];
00322             float   CamDist2[8];
00323             float   RotaMat[9];
00324             float   TranMat[3];
00325             float   LRotaMat[9];
00326             float   RRotaMat[9];
00327             float   NewCamMat1[12];
00328             float   NewCamMat2[12];
00329             WORD    RECT_Crop_Row_BG;
00330             WORD    RECT_Crop_Row_ED;
00331             WORD    RECT_Crop_Col_BG_L;
00332             WORD    RECT_Crop_Col_ED_L;
00333             BYTE    RECT_Scale_Col_M;
00334             BYTE    RECT_Scale_Col_N;
00335             BYTE    RECT_Scale_Row_M;
00336             BYTE    RECT_Scale_Row_N;
00337             float   RECT_AvgErr;
00338             WORD    nLineBuffers;
00339             float   ReProjectMat[16];
00340             float   ParameterRatio[2]; // Ratio for distortion K6
00341             float   LR_cam_K_temperature[2];
00342             float   LR_cam_thermal_variation_rate_of_focal[2];
00343             float   depth_comp_pars[2];
00344             long    Date; // Calibration Date
00345             char    type; // Calibration type
00346             char    version[4]; // Calibration Version
00347         };
00348     };
00349 };
00350
00351 } eSPCtrl_RectLogData;
00352
00358 typedef struct ParaLUT

```

```

00359 {
00360     long long file_ID_header;
00361     long long file_ID_version;
00362     double FOV;
00363     long long semi_FOV_pixels;
00364     long long img_src_cols;
00365     long long img_src_rows;
00366     double img_L_src_col_center;
00367     double img_L_src_row_center;
00368     double img_R_src_col_center;
00369     double img_R_src_row_center;
00370     double img_L_rotation;
00371     double img_R_rotation;
00372     double spline_control_v1;
00373     double spline_control_v2;
00374     double spline_control_v3;
00375     double spline_control_v4;
00376     double spline_control_v5;
00377     double spline_control_v6;
00378     double spline_control_v7;
00379     long long img_dst_cols;
00380     long long img_dst_rows;
00381     long long img_L_dst_shift;
00382     long long img_R_dst_shift;
00383     long long img_overlay_LR;
00384     long long img_overlay_RL;
00385     long long img_stream_cols;
00386     long long img_stream_rows;
00387     long long video_stream_cols;
00388     long long video_stream_rows;
00389     long long usb_type;
00390     long long img_type;
00391     long long lut_type;
00392     long long blending_type;
00393     double overlay_ratio;
00394
00395     long long serial_number_date0;
00396     long long serial_number_date1;
00397
00398     double unit_sphere_radius;
00399     double min_col;
00400     double max_col;
00401     double min_row;
00402     double max_row;
00403
00404     long long AGD_LR;
00405     long long AGD_RL;
00406
00407     long long out_img_resolution;
00408     long long out_lut_cols;
00409     long long out_lut_rows;
00410     long long out_lut_cols_eff;
00411     long long out_lut_rows_eff;
00412     long long out_img_cols;
00413     long long out_img_rows;
00414     long long out_overlay_LR;
00415     long long out_overlay_RL;
00416     long long reserve[44];
00417     BYTE serial_number[256];
00418 } PARALUT, *PPARALUT;
00419
00420 struct APCImageType
00421 {
00422     enum Value
00423     {
00424         IMAGE_UNKNOWN = -1,
00425         COLOR_YUY2 = 0,
00426         COLOR_Y12Bits,
00427         COLOR_RGB24,
00428         COLOR_MJPG,
00429         DEPTH_8BITS = 100,
00430         DEPTH_8BITS_0x80,
00431         DEPTH_11BITS,
00432         DEPTH_14BITS
00433     };
00434
00435     static bool IsImageColor(APCImageType::Value type)
00436     {
00437         return (type == COLOR_YUY2 || type == COLOR_RGB24 || type == COLOR_MJPG || type ==
COLOR_Y12Bits);
00438     }
00439
00440     static bool IsImageDepth(APCImageType::Value type)
00441     {
00442         return (type != IMAGE_UNKNOWN && !IsImageColor(type));
00443     }
00444 }

```

```

00445     static APCHandleType::Value DepthDataTypeToDepthImageType (WORD dataType)
00446     {
00447         switch (dataType)
00448         {
00449             case APC_DEPTH_DATA_8_BITS:
00450             case APC_DEPTH_DATA_8_BITS_RAW:
00451             case APC_DEPTH_DATA_ILM_8_BITS:
00452             case APC_DEPTH_DATA_ILM_8_BITS_RAW:
00453             case APC_DEPTH_DATA_SCALE_DOWN_8_BITS:
00454             case APC_DEPTH_DATA_SCALE_DOWN_8_BITS_RAW:
00455             case APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS:
00456             case APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_RAW:
00457                 return APCHandleType::DEPTH_8BITS;
00458             case APC_DEPTH_DATA_8_BITS_x80:
00459             case APC_DEPTH_DATA_8_BITS_x80_RAW:
00460             case APC_DEPTH_DATA_ILM_8_BITS_x80:
00461             case APC_DEPTH_DATA_ILM_8_BITS_x80_RAW:
00462             case APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80:
00463             case APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80_RAW:
00464             case APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_x80:
00465             case APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_x80_RAW:
00466                 return APCHandleType::DEPTH_8BITS_0x80;
00467             case APC_DEPTH_DATA_11_BITS:
00468             case APC_DEPTH_DATA_11_BITS_RAW:
00469             case APC_DEPTH_DATA_11_BITS_COMBINED_RECTIFY:
00470             case APC_DEPTH_DATA_ILM_11_BITS:
00471             case APC_DEPTH_DATA_ILM_11_BITS_RAW:
00472             case APC_DEPTH_DATA_ILM_11_BITS_COMBINED_RECTIFY:
00473             case APC_DEPTH_DATA_SCALE_DOWN_11_BITS:
00474             case APC_DEPTH_DATA_SCALE_DOWN_11_BITS_RAW:
00475             case APC_DEPTH_DATA_SCALE_DOWN_11_BITS_COMBINED_RECTIFY:
00476             case APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS:
00477             case APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS_RAW:
00478             case APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS_COMBINED_RECTIFY:
00479                 return APCHandleType::DEPTH_11BITS;
00480             case APC_DEPTH_DATA_14_BITS:
00481             case APC_DEPTH_DATA_14_BITS_RAW:
00482             case APC_DEPTH_DATA_14_BITS_COMBINED_RECTIFY:
00483             case APC_DEPTH_DATA_ILM_14_BITS:
00484             case APC_DEPTH_DATA_ILM_14_BITS_RAW:
00485             case APC_DEPTH_DATA_ILM_14_BITS_COMBINED_RECTIFY:
00486             case APC_DEPTH_DATA_SCALE_DOWN_14_BITS:
00487             case APC_DEPTH_DATA_SCALE_DOWN_14_BITS_RAW:
00488             case APC_DEPTH_DATA_SCALE_DOWN_14_BITS_COMBINED_RECTIFY:
00489             case APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS:
00490             case APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS_RAW:
00491             case APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS_COMBINED_RECTIFY:
00492                 return APCHandleType::DEPTH_14BITS;
00493             default: return APCHandleType::IMAGE_UNKNOWN;
00494         }
00495     }
00496
00497     static bool IsSupportedDisparityCompensateDevice(int pid, WORD dataType) {
00498         return (pid == APC_PID_HYPATIA2 || pid == APC_PID_8072) &&
00499             (APCHandleType::DepthDataTypeToDepthImageType(dataType) == APCHandleType::DEPTH_11BITS);
00500     };
00501
00502     struct ApcDIDepthSwitch
00503     {
00504         enum Value
00505         {
00506             Depth0 = 0x01,
00507             Depth1 = 0x02,
00508             Depth2 = 0x04
00509         };
00510
00511         static bool IsOn(ApcDIDepthSwitch::Value target, int setting)
00512         {
00513             return ((target & setting) != 0);
00514         }
00515     };
00516
00517     typedef void(*APC_ImgCallbackFn)(APCHandleType::Value imgType, int imgId, unsigned char* imgBuf, int
00518     imgSize,
00519     int width, int height, int serialNumber, LONGLONG timestamp, void* pParam);
00520
00521     typedef enum {
00522         OTHERS = 0,
00523         AXES1,
00524         PUMA,
00525         PLUM,
00526         GRAPE_FPGA
00527     }APC_DEVICE_TYPE;
00528
00529     struct APC_SensorMode
00530     {

```

```

00544     enum Value
00545     {
00546         Sensor1 = 0,
00547         Sensor2 = 1,
00548         SensorAll = 2,
00549         Sensor3 = 3,
00550         Sensor4 = 4
00551     };
00552 };
00553 typedef struct tagDEVINFORMATION {
00554     unsigned short wPID;
00555     unsigned short wVID;
00556     char *strDevName;
00557     char *strDevPath;
00558     unsigned short nChipID;
00559     unsigned short nDevType;
00560     unsigned short wUsbNode;
00561 } DEVINFORMATION;
00562
00563 class DEVINFORMATIONEX
00564 {
00565 public:
00566     DEVINFORMATIONEX()
00567     {
00568         wPID = wVID = nChipID = 0;
00569         nDevType = OTHERS;
00570         strDevName[0] = '\0';
00571         strDevPath[0] = '\0';
00572         wUsbNode = -1;
00573     }
00574
00575     DEVINFORMATIONEX& operator=(const DEVINFORMATIONEX& rhs)
00576     {
00577         wPID = rhs.wPID;
00578         wVID = rhs.wVID;
00579         strcpy_s(strDevName, rhs.strDevName);
00580         strcpy_s(strDevPath, rhs.strDevPath);
00581         nChipID = rhs.nChipID;
00582         nDevType = rhs.nDevType;
00583         wUsbNode = rhs.wUsbNode;
00584
00585         return *this;
00586     }
00587
00588     DEVINFORMATIONEX& operator=(const DEVINFORMATION& rhs)
00589     {
00590         wPID = rhs.wPID;
00591         wVID = rhs.wVID;
00592         strcpy_s(strDevName, rhs.strDevName);
00593         strcpy_s(strDevPath, rhs.strDevPath);
00594         nChipID = rhs.nChipID;
00595         nDevType = rhs.nDevType;
00596         wUsbNode = rhs.wUsbNode;
00597
00598         return *this;
00599     }
00600
00601     DEVINFORMATIONEX(const DEVINFORMATIONEX& rhs)
00602     {
00603         *this = rhs;
00604     }
00605
00606     unsigned short wPID;
00607     unsigned short wVID;
00608     char strDevName[512];
00609     char strDevPath[512];
00610     unsigned short nChipID;
00611     unsigned short nDevType;
00612     unsigned short wUsbNode;
00613 };
00614
00615 typedef struct tagDEVSEL
00616 {
00617     int index;
00618 } DEVSELINFO, *PDEVSELINFO;
00619
00620 typedef enum
00621 {
00622     USERDATA_SECTION_0 = 0,
00623     USERDATA_SECTION_1,
00624     USERDATA_SECTION_2,
00625     USERDATA_SECTION_3,
00626     USERDATA_SECTION_4,
00627     USERDATA_SECTION_5,
00628     USERDATA_SECTION_6,
00629     USERDATA_SECTION_7,
00630     USERDATA_SECTION_8,

```

```

00669     USERDATA_SECTION_9,
00670     USERDATA_SECTION_10,
00671     USERDATA_SECTION_NUM
00672 } USERDATA_SECTION_INDEX;
00673
00674 // for total and fw+plugin read/write +
00675 typedef enum
00676 {
00677     Total = 0,
00678     FW_PLUGIN,
00679     Total_Slave,      /* total for first slave device */
00680     FW_PLUGIN_Slave,  /* fw_plugin for first slave device */
00681     UNP                /* UNProtection Area */
00682 } FLASH_DATA_TYPE;
00683
00684 typedef enum
00685 {
00686     USB_PORT_TYPE_2_0 = 2,
00687     USB_PORT_TYPE_3_0,
00688     USB_PORT_TYPE_UNKNOW
00689 } USB_PORT_TYPE;
00690
00691 typedef struct tagKEEP_DATA_CTRL {
00692     bool    bIsSerialNumberKeep;
00693     bool    bIsSensorPositionKeep;
00694     bool    bIsRectificationTableKeep;
00695     bool    bIsZDTableKeep;
00696     bool    bIsCalibrationLogKeep;
00697 } KEEP_DATA_CTRL;
00698 // for total and fw+plugin read/write -
00699
00700 int  APC_API APC_Init( void **ppHandleApcDI, bool bIsLogEnabled);
00710
00723 int  APC_API APC_Init2( void **ppHandleApcDI, bool bIsLogEnabled, bool bAutoRestart);
00724
00738 int  APC_API APC_Init3( void **ppHandleApcDI, bool bIsLogEnabled, bool bEnableAutoRestart, bool
bMonitorUSBEvent);
00739
00748 #ifndef APC_DeviceEventFn_
00749 typedef void(*APC_DeviceEventFn)(UINT pid, UINT vid, BOOL bAttached, void* pData);
00750 #define APC_DeviceEventFn_
00751 #endif
00752
00762 int  APC_API APC_RegisterDeviceEvents(void *pHandleApcDI, APC_DeviceEventFn cbFunc, void *pData);
00763
00771 void APC_API APC_Release( void **ppHandleApcDI);
00772
00779 int  APC_API APC_FindDevice( void *pHandleApcDI);
00780
00787 int  APC_API APC_RefreshDevice( void *pHandleApcDI);
00788
00795 int  APC_API APC_GetDeviceNumber( void *pHandleApcDI);
00796
00807 int  APC_API APC_GetDeviceInfo( void *pHandleApcDI, PDEVSELINFO pDevSelInfo ,DEVINFORMATION*
pdevinfo);
00808
00819 int  APC_API APC_GetDeviceInfoEx( void *pHandleApcDI, PDEVSELINFO pDevSelInfo ,DEVINFORMATIONEX*
pdevinfo);
00820
00821 // register APIs +
00846 int APC_API APC_GetSlaveSensorRegister(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, unsigned
short address, unsigned short *pValue, int flag, int nSensorMode);
00847
00872 int APC_API APC_GetSensorRegister ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, unsigned
short address, unsigned short *pValue, int flag, int nSensorMode);
00873
00898 int APC_API APC_SetSlaveSensorRegister(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, unsigned
short address, unsigned short nValue, int flag, int nSensorMode);
00899
00924 int APC_API APC_SetSensorRegister ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, unsigned
short address, unsigned short nValue, int flag, int nSensorMode);
00925
00945 int APC_API APC_GetFWRegister ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short
address, unsigned short *pValue, int flag);
00946
00966 int APC_API APC_SetFWRegister ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short
address, unsigned short nValue, int flag);
00967
00987 int APC_API APC_GetSlaveHWRRegister(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short
address, unsigned short *pValue, int flag);
00988
01008 int APC_API APC_GetHWRRegister ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short
address, unsigned short *pValue, int flag);
01009
01029 int APC_API APC_SetSlaveHWRRegister(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short
address, unsigned short nValue, int flag);
01030

```

```

01050 int APC_API APC_SetHWRegister      ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short
address, unsigned short nValue,  int flag);
01051
01066 int APC_API APC_GetMultiBytesHWRegister(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, UCHAR *data, int
address, int size);
01067
01082 int APC_API APC_SetMultiBytesHWRegister(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, UCHAR *data, int
address, int size);
01083
01084 // register APIs -
01085
01086 // File ID +
01101 int APC_API APC_GetFwVersion      ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, char *pszFwVersion, int
nBufferSize, int *pActualLength);
01102
01115 int APC_API APC_GetPidVid        ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short
*pPidBuf, unsigned short *pVidBuf );
01116
01129 int APC_API APC_SetPidVid(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short *pPidBuf,
unsigned short *pVidBuf);
01130
01140 int APC_API APC_GetSerialNumber( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *pSerialNum, int
nBufferSize, int *pActualSNLenByByte);
01141
01150 int APC_API APC_SetSerialNumber(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *pSerialNum, int
nBufferSize);
01151
01168 int APC_API APC_GetSlaveLogData(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
01169
01186 int APC_API APC_GetLogData      ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
01187
01202 int APC_API APC_GetUserData      ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, USERDATA_SECTION_INDEX usi);
01203
01214 int APC_API APC_SetSlaveLogData(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
01215
01226 int APC_API APC_SetLogData      ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
01227
01238 int APC_API APC_SetLogData_Advanced(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
01239
01249 int APC_API APC_SetUserData      ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, USERDATA_SECTION_INDEX usi);
01250
01251
01270 int APC_API APC_ReadFlashData   ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, FLASH_DATA_TYPE fdt,
BYTE *pBuffer,
01271                                     unsigned long int nLengthOfBuffer, unsigned long int
*pActualBufferLen);
01272
01273
01274 int APC_API APC_WriteFlashData  ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, FLASH_DATA_TYPE fdt,
BYTE *pBuffer,
01275                                     unsigned long int nLengthOfBuffer, BOOL bIsDataVerify,
KEEP_DATA_CTRL kdc);
01276
01282 int APC_API APC_GetStructLenOfSTI(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01283
01289 int APC_API APC_GetUnpAreaStartSec(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01290
01296 int APC_API APC_IsBPX4bitsClear(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01297
01307 int APC_API APC_UnprotectFlash(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01308
01319 int APC_API APC_UnprotectFlashByCipher(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, const char*
cipher);
01320
01351 int APC_API APC_OpenDevice( void* pHandleApcDI, PDEVSELINFO pDevSelInfo,
01352                             int colorStreamIndex, int depthStreamIndex, int depthStreamSwitch,
int iFps,
01353                             APC_ImgCallbackFn callbackFn, void* pCallbackParam, int pid = -1
);
01354
01358 int APC_API APC_GetColorImage(void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
01359     BYTE *pBuf, unsigned long int *pImageSize, int *pSerial = NULL);
01360
01364 int APC_API APC_GetDepthImage(void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
01365     BYTE *pBuf, unsigned long int *pImageSize, int *pSerial = NULL, int nDepthDataType = 0);
01366
01375 int APC_API APC_CloseDevice( void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01376
01393 int APC_API APC_GetDeviceResolutionListEx( void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
01394     int nMaxCount0, APC_STREAM_INFO *pStreamInfo0,

```

```

01395                                     int nMaxCount1, APC_STREAM_INFO *pStreamInfo1,
01396                                     int pid);
01397
01414 int APC_API APC_GetDeviceResolutionList( void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
01415                                     int nMaxCount0, APC_STREAM_INFO *pStreamInfo0,
01416                                     int nMaxCount1, APC_STREAM_INFO *pStreamInfo1);
01417
01426 bool APC_API APC_Is360Device(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01427
01431 int APC_API APC_GetSerialNumberFromLog( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, char *pSerialNum,
01432                                     int nBufferSize, int *pActualLength);
01433
01433 // IR support
01441 int APC_API APC_SetCurrentIRValue(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType);
01442
01450 int APC_API APC_GetCurrentIRValue(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType);
01451
01459 int APC_API APC_GetIRMinValue(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType);
01460
01468 int APC_API APC_SetIRMaxValue(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType);
01469
01477 int APC_API APC_SetIRMaxValueUnleashed(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType);
01478
01486 int APC_API APC_GetIRMaxValue(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType);
01487
01491 int APC_API APC_SetIRMode(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType);
01492
01496 int APC_API APC_GetIRMode(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType);
01497
01498 // for sensorif
01502 int APC_API APC_EnableSensorIF( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool bIsEnable);
01503
01504
01505 //int APC_API APC_GetMotorCurrentState( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool*
01506     bIsRunning);
01507
01507 // for Gyro
01508
01509 typedef enum
01510 {
01511     DPS_245 = 0,
01512     DPS_500,
01513     DPS_2000
01514 } SENSITIVITY_LEVEL_L3G;
01515
01516 // bPowerMode :
01517 //     true : Normal
01518 //     false : Power Down
01519 // bIsZEnable :
01520 //     true : Enable
01521 //     false : Disable
01522 // bIsYEnable :
01523 //     true : Enable
01524 //     false : Disable
01525 // bIsXEnable :
01526 //     true : Enable
01527 //     false : Disable
01528
01529 //typedef struct GyroTag
01530 //{
01531 //    short x;
01532 //    short y;
01533 //    short z;
01534 //} GYRO_ANGULAR_RATE_DATA;
01535
01536
01538 typedef enum
01539 {
01540     APC_SENSOR_TYPE_H22 = 0,
01541     APC_SENSOR_TYPE_OV7740,
01542     APC_SENSOR_TYPE_AR0134,
01543     APC_SENSOR_TYPE_AR0135,
01544     APC_SENSOR_TYPE_AR0144,
01545     APC_SENSOR_TYPE_OV9714,
01546     APC_SENSOR_TYPE_OV9282,
01547     APC_SENSOR_TYPE_AR0330,
01548     APC_SENSOR_TYPE_AR1335,
01549     APC_SENSOR_TYPE_H65,
01550     APC_SENSOR_TYPE_AR0522,
01551     APC_SENSOR_TYPE_OV2740,
01552     APC_SENSOR_TYPE_OC0SA10,
01553     APC_SENSOR_TYPE_VD56G3,
01554     APC_SENSOR_TYPE_VD66GY,
01555     APC_SENSOR_TYPE_H68,
01556     APC_SENSOR_TYPE_UNKOWN = 0xffff
01557 } SENSOR_TYPE_NAME;
01558

```



```

01559 //
01560 // Sensor Mode: Left, Right, or Both
01561 //
01562 #define ESPAEAWB_SENSOR_MODE_LEFT      0
01563 #define ESPAEAWB_SENSOR_MODE_RIGHT     1
01564 #define ESPAEAWB_SENSOR_MODE_BOTH      2
01565
01566
01575 int APC_API APC_SetSensorTypeName( void *pHandleApcDI, SENSOR_TYPE_NAME stn);
01576
01585 int APC_API APC_EnableAE ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01586
01595 int APC_API APC_DisableAE ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01596
01605 int APC_API APC_EnableAWB ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01606
01615 int APC_API APC_DisableAWB( void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
01616
01617
01633 int APC_API APC_GetExposureTime( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nSensorMode, int
pid, float *pfExpTimeMS);
01634
01650 int APC_API APC_SetExposureTime( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nSensorMode, int
pid, float fExpTimeMS);
01651
01668 int APC_API APC_GetGlobalGain( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid,
float *pfGlobalGain);
01669
01686 int APC_API APC_SetGlobalGain( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid,
float fGlobalGain);
01687
01703 int APC_API APC_SetAnalogGain(void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid,
float fGlobalGain);
01704
01720 int APC_API APC_GetAnalogGain(void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid,
float *pfGlobalGain);
01721
01737 int APC_API APC_SetDigitalGain(void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid,
float fGlobalGain);
01738
01754 int APC_API APC_GetDigitalGain(void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid,
float *pfGlobalGain);
01755
01768 int APC_API APC_GetGPIOValue( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE
*pValue);
01769
01782 int APC_API APC_SetGPIOValue( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE
nValue);
01783
01790 int APC_API APC_SetGPIOCtrl(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE nValue);
01791
01792 int APC_API APC_GetAccMeterValue( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int *pX, int *pY, int
*pZ);
01793
01831 int APC_API APC_GetPUPPropVal( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int *pValue);
01832
01847 int APC_API APC_SetPUPPropVal( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int nValue);
01848
01881 int APC_API APC_GetCTPropVal( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int *pValue);
01882
01897 int APC_API APC_SetCTPropVal( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int nValue);
01898
01899 // For AEAWB - 2015/01/28 by Wolf
01900
01901 int APC_API APC_EncryptMP4(void* pHandleApcDI, PDEVSELINFO pDevSelInfo, const char* filename);
01902 int APC_API APC_DecryptMP4(void* pHandleApcDI, PDEVSELINFO pDevSelInfo, const char* filename);
01903 int APC_API APC_RetrieveMp4ExtraData(void* pHandleApcDI, PDEVSELINFO pDevSelInfo,
const char* filename, char* dataBuf, int* dataSize);
01904
01905 int APC_API APC_FlushMp4ExtraData(void* pHandleApcDI, PDEVSELINFO pDevSelInfo,
const char* filename, const char* dataBuf, int dataSize);
01906
01910 int APC_API APC_GetAutoExposureMode(void* pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short*
mode);
01911
01915 int APC_API APC_SetAutoExposureMode(void* pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short mode);
01916
01925 int APC_API APC_GetFlexibleGyroData(void * pHandleApcDI, PDEVSELINFO pDevSelInfo,
int length, BYTE *pGyroData);
01926
01927
01935 int APC_API APC_GetFlexibleGyroLength(void* pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short*
GyroLen);
01936
01949 int APC_API APC_SetHuffmanTableData(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, const char *filename,
bool bLogFile);
01950
01961 int APC_API APC_SetQuantizationTableData(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, const char
*filename);
01962

```

```

01966 int APC_API APC_SetPlumAR0330(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool bEnable);
01967
01971 int APC_API APC_SetRootCipher(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, const char* cipher);
01972
01981 int APC_API APC_ResetUNPData(void* pHandleApcDI, PDEVSELINFO pDevSelInfo);
01982
01986 int APC_API APC_GetDevicePortType(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, USB_PORT_TYPE*
    pUSB_Port_Type);
01987
01990 int APC_API APC_SubSample(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char** SubSample,
    unsigned char* depthBuf, int bytesPerPixel, int width, int height, int& new_width, int& new_height,
    int mode = 0, int factor = 3);
01991
01994 int APC_API APC_HoleFill(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char* depthBuf, int
    bytesPerPixel, int kernel_size, int width, int height, int level, bool horizontal);
01995
01998 int APC_API APC_TemporalFilter(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char* depthBuf,
    int bytesPerPixel, int width, int height, float alpha, int history);
01999
02002 int APC_API APC_EdgePreServingFilter(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char*
    depthBuf, int type, int width, int height, int level, float sigma, float lumda);
02003
02006 int APC_API APC_ApplyFilters(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char* depthBuf,
    unsigned char* subDisparity, int bytesPerPixel, int width, int height, int sub_w, int sub_h, int
    threshold=64);
02007
02015 int APC_API APC_EnableGPUAcceleration(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool enable);
02016
02023 APC_API char* APC_GetDepthFilterVersion(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
02024
02027 int APC_API APC_ResetFilters(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
02028
02041 int APC_API APC_TableToData(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int width, int height, int
    TableSize, unsigned short* Table, unsigned short* Src, unsigned short* Dst);
02042
02055 int APC_API APC_ColorFormat_to_RGB24( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char*
    ImgDst, unsigned char* ImgSrc, int SrcSize, int width, int height, APCImageType::Value type, int
    colorNum=0 );
02056
02069 int APC_API APC_ColorFormat_to_BGR24(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char*
    ImgDst, unsigned char* ImgSrc, int SrcSize, int width, int height, APCImageType::Value type);
02070
02071 int APC_API APC_PropertyPU_GetRange(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty, long
    * pMin, long * pMax, long * pStep, long * pDefault, long * pCapsFlag, int pid);
02072 int APC_API APC_PropertyCT_GetRange(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty, long
    * pMin, long * pMax, long * pStep, long * pDefault, long * pCapsFlag, int pid);
02073 int APC_API APC_PropertyPU_GetCurrent(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty,
    long *pCur, long *pCur2, long *pCapsFlag, int pid);
02074 int APC_API APC_PropertyCT_GetCurrent(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty,
    long *pCur, long *pCur2, long *pCapsFlag, int pid);
02075 int APC_API APC_PropertyPU_SetCurrent(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty,
    long nCur, long nCur2, long nCapsFlag, int pid);
02076 int APC_API APC_PropertyCT_SetCurrent(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty,
    long nCur, long nCur2, long nCapsFlag, int pid);
02077 int APC_API APC_PropertyItem_Write(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, REFGUID guid, int
    nPropertyItem, LONG nValue, int pid);
02078 int APC_API APC_PropertyItem_Read(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, REFGUID guid, int
    nPropertyItem, LONG *pValue, int pid);
02079
02080 int APC_API APC_ShowPropertyPage( void * pHandleApcDI, PDEVSELINFO pDevSelInfo, int pid );
02081
02082 #ifdef __cplusplus
02083 }
02084 #endif

```

5.3 eSPDI_DM.h File Reference

eYs3D SDK API export functions, data structure and variable definition for depth map module

Functions

- int APC_API [APC_GetFWFSLength](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int iFWFSIndex, int &iFWFSLength)

get length for specific file id

- int APC_API [APC_GetFileData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nID, int DataSize, BYTE *lpData)
get file data for specific file id
- int APC_API [APC_GetSlaveYOffset](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
get Y offset data
- int APC_API [APC_GetYOffset](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
get Y offset data
- int APC_API [APC_GetSlaveRectifyTable](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
get rectify values from flash
- int APC_API [APC_GetRectifyTable](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
get rectify values from flash
- int APC_API [APC_GetZDTable](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, PZDTABLEINFO pZDTableInfo)
get disparity and Z values from flash
- int APC_API [APC_SetSlaveYOffset](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
set Y offset data
- int APC_API [APC_SetYOffset](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
set Y offset data
- int APC_API [APC_SetYOffset_Advanced](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
set Y offset data in both Groups #1 and #2 when the firmware has the flash protection.
- int APC_API [APC_SetSlaveRectifyTable](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
set rectify data to flash, see APC_SetRectifyTable except set
- int APC_API [APC_SetRectifyTable](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
set rectify data to flash, see APC_SetRectifyTable except set
- int APC_API [APC_SetRectifyTable_Advanced](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)
set rectify data to flash, see APC_SetRectifyTable except set
- int APC_API [APC_SetZDTable](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, PZDTABLEINFO pZDTableInfo)
set disparity and Z values to flash, see APC_GetZDTable except get
- int APC_API [APC_SetZDTable_Advanced](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, PZDTABLEINFO pZDTableInfo)
set disparity and Z values to flash, see APC_GetZDTable except get
- int APC_API [APC_GetRectifyMatLogData](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, [eSPCtrl_RectLogData](#) *pData, int index)
get rectify log data from flash for Puma IC
- int APC_API [APC_SetDepthDataType](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType)
set depth data type, 11 bit for disparity data, 14 bit for Z data notice: only PUMA type IC can support this setting
- int APC_API [APC_GetDepthDataType](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType)
get current depth data type setting
- int APC_API [APC_SetHWPostProcess](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool enable)
enable or disable internal chip post processing function
- int APC_API [APC_GetHWPostProcess](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool *enable)
get hardware post processing status
- bool APC_API [APC_IsInterleaveDevice](#) (void *pHandleApcDI, PDEVSELINFO pDevSelInfo)
enable or disable interleave function

5.3.1 Detailed Description

eYs3D SDK API export functions, data structure and variable definition for depth map module

Copyright

This file copyright (C) 2017 by eYs3D company

An unpublished work. All rights reserved. This file is proprietary information, and may not be disclosed or copied without the prior permission of eYs3D company.

5.3.2 Function Documentation

5.3.2.1 APC_GetDepthDataType()

```
int APC_API APC_GetDepthDataType (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo,  
    WORD * pwType)
```

get current depth data type setting

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>pwType</i>	pointer of current depth data type in device

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.3.2.2 APC_GetFileData()

```
int APC_API APC_GetFileData (  
    void * pHandleApcDI,  
    PDEVSELINFO pDevSelInfo,  
    int nID,  
    int DataSize,  
    BYTE * lpData)
```

get file data for specific file id

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>nID</i>	actual file id
<i>lpData</i>	the pointer to file data buffer

Returns

success:APC_OK, others:see [eSPDI_ErrCode.h](#)

5.3.2.3 APC_GetFWFSLength()

```
int APC_API APC_GetFWFSLength (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    int iFWFSIndex,
    int & iFWFSLength)
```

get length for specific file id

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>iFWFSIndex</i>	actual file id
<i>iFWFSLength</i>	length of file id

Returns

success:APC_OK, others:see [eSPDI_ErrCode.h](#)

5.3.2.4 APC_GetHWPostProcess()

```
int APC_API APC_GetHWPostProcess (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    bool * enable)
```

get hardware post processing status

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>enable</i>	returns current hardware post-process status

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.3.2.5 APC_GetRectifyMatLogData()

```
int APC_API APC_GetRectifyMatLogData (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    eSPCtrl_RectLogData * pData,
    int index)
```

get rectify log data from flash for Puma IC

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>pData</i>	rectify log data, its buffer size is 4096 bytes see eSPCtrl_RectLogData for detailed members
<i>index</i>	index to identify rectify log data for corresponding depth

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.3.2.6 APC_GetRectifyTable()

```
int APC_API APC_GetRectifyTable (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

get rectify values from flash

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store rectify table data
<i>BufferLength</i>	input buffer length
<i>pActualLength</i>	actual length has written to buffer
<i>index</i>	index to identify rectify table for corresponding depth

Returns

success:APC_OK, others: see [eSPDI_ErrCode.h](#)

5.3.2.7 APC_GetSlaveRectifyTable()

```
int APC_API APC_GetSlaveRectifyTable (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

get rectify values from flash

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store rectify table data
<i>BufferLength</i>	input buffer length
<i>pActualLength</i>	actual length has written to buffer
<i>index</i>	index to identify rectify table for corresponding depth

Returns

success:APC_OK, others: see [eSPDI_ErrCode.h](#)

5.3.2.8 APC_GetSlaveYOffset()

```
int APC_API APC_GetSlaveYOffset (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

get Y offset data

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store
<i>BufferLength</i>	length of buffer
<i>pActualLength</i>	actual byte of reading
<i>index</i>	index of Y offset file ID

Returns

success:APC_OK, others:see [eSPDI_ErrCode.h](#)

5.3.2.9 APC_GetYOffset()

```
int APC_API APC_GetYOffset (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

get Y offset data

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store
<i>BufferLength</i>	length of buffer
<i>pActualLength</i>	actual byte of reading
<i>index</i>	index of Y offset file ID

Returns

success:APC_OK, others:see [eSPDI_ErrCode.h](#)

5.3.2.10 APC_GetZDTable()

```
int APC_API APC_GetZDTable (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    PZDTABLEINFO pZDTableInfo)
```

get disparity and Z values from flash

1. if depth data type is APC_DEPTH_DATA_14_BITS then just get Z value from depth buffer
2. if depth data type is APC_ZD_TABLE_FILE_SIZE_11_BITS then using depth buffer value as a index to get Z value inside ZD table
3. see GetZValue() of example.c to get Z value from different depth data type

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	bufer to store ZD table
<i>BufferLength</i>	input buffer length
<i>pActualLength</i>	actual length has written to buffer
<i>pZDTableInfo</i>	index to identify ZD table and data type for corresponding depth

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.3.2.11 APC_IsInterleaveDevice()

```
bool APC_API APC_IsInterleaveDevice (
    void * pHandleEYSD,
    PDEVSELINFO pDevSelInfo)
```

enable or disable interleave function

check module support interleave function or not

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>enable</i>	set true to enable interleave, or set false to disable interleave

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index

Returns

true: support interleave, false: not support

5.3.2.12 APC_SetDepthDataType()

```
int APC_API APC_SetDepthDataType (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    WORD wType)
```

set depth data type, 11 bit for disparity data, 14 bit for Z data notice: only PUMA type IC can support this setting

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>wType</i>	depth data type you want to set, see APC_DEPTH_DATA_xxx in APC_O.h \output success: APC_OK, others: see eSPDI_ErrCode.h

5.3.2.13 APC_SetHWPostProcess()

```
int APC_API APC_SetHWPostProcess (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    bool enable)
```

enable or disable internal chip post processing function

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>enable</i>	set true to enable post-process, or set false to disable post-process

Returns

success: APC_OK, others: see [eSPDI_ErrCode.h](#)

5.3.2.14 APC_SetSlaveYOffset()

```
int APC_API APC_SetSlaveYOffset (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

set Y offset data

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store
<i>BufferLength</i>	length of buffer
<i>pActualLength</i>	actual byte of reading
<i>index</i>	index of Y offset file ID

Returns

success:APC_OK, others:see [eSPDI_ErrCode.h](#)

5.3.2.15 APC_SetYOffset()

```
int APC_API APC_SetYOffset (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

set Y offset data

Parameters

<i>pHandleApcDI</i>	the pointer to the initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>buffer</i>	buffer to store
<i>BufferLength</i>	length of buffer
<i>pActualLength</i>	actual byte of reading
<i>index</i>	index of Y offset file ID

Returns

success:APC_OK, others:see [eSPDI_ErrCode.h](#)

5.3.2.16 APC_SetYOffset_Advanced()

```
int APC_API APC_SetYOffset_Advanced (
    void * pHandleApcDI,
    PDEVSELINFO pDevSelInfo,
    BYTE * buffer,
    int BufferLength,
    int * pActualLength,
    int index)
```

set Y offset data in both Groups #1 and #2 when the firmware has the flash protection.

Parameters

<i>void</i>	*pHandleApcDI the pointer to the initilized ApcDI SDK instance
<i>PDEVSELINFO</i>	pDevSelInfo CApcDI handler
<i>BYTE</i>	*buffer buffer to store
<i>int</i>	BufferLength length of buffer
<i>int</i>	*pActualLength actual byte of reading
<i>int</i>	index index of Y offset file ID

Returns

success:APC_OK, others:see [eSPDI_ErrCode.h](#)

5.4 eSPDI_DM.h

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

```
00001
00009 #pragma once
00010 #include "eSPDI_Common.h"
00011
00012 // for APC_PostSetParam/APC_PostGetParam
00013 #define POSTPAR_HR_MODE 5
00014 #define POSTPAR_HR_CURVE_0 6
00015 #define POSTPAR_HR_CURVE_1 7
00016 #define POSTPAR_HR_CURVE_2 8
00017 #define POSTPAR_HR_CURVE_3 9
00018 #define POSTPAR_HR_CURVE_4 10
00019 #define POSTPAR_HR_CURVE_5 11
00020 #define POSTPAR_HR_CURVE_6 12
00021 #define POSTPAR_HR_CURVE_7 13
00022 #define POSTPAR_HR_CURVE_8 14
00023 #define POSTPAR_HF_MODE 17
00024 #define POSTPAR_DC_MODE 20
00025 #define POSTPAR_DC_CNT_THD 21
00026 #define POSTPAR_DC_GRAD_THD 22
00027 #define POSTPAR_SEG_MODE 23
00028 #define POSTPAR_SEG_THD_SUB 24
00029 #define POSTPAR_SEG_THD_SLP 25
00030 #define POSTPAR_SEG_THD_MAX 26
00031 #define POSTPAR_SEG_THD_MIN 27
00032 #define POSTPAR_SEG_FILL_MODE 28
00033 #define POSTPAR_HF2_MODE 31
00034 #define POSTPAR_GRAD_MODE 34
00035 #define POSTPAR_TEMP0_MODE 37
00036 #define POSTPAR_TEMP0_THD 38
00037 #define POSTPAR_TEMP1_MODE 41
00038 #define POSTPAR_TEMP1_LEVEL 42
00039 #define POSTPAR_TEMP1_THD 43
00040 #define POSTPAR_FC_MODE 46
00041 #define POSTPAR_FC_EDGE_THD 47
00042 #define POSTPAR_FC_AREA_THD 48
00043 #define POSTPAR_MF_MODE 51
00044 #define POSTPAR_ZM_MODE 52
```

```

00045 #define POSTPAR_RF_MODE          53
00046 #define POSTPAR_RF_LEVEL          54
00047
00048 //
00049 // C++ compatibility
00050 //
00051 #ifdef __cplusplus
00052 extern "C" {
00053 #endif
00054
00067 int APC_API APC_GetFWFSLength(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int iFWFSIndex, int&
iFWFSLength);
00068
00082 int APC_API APC_GetFileData(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nID, int DataSize, BYTE
*lpData);
00083
00100 int APC_API APC_GetSlaveYOffset(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
00101
00118 int APC_API APC_GetYOffset(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
00119
00136 int APC_API APC_GetSlaveRectifyTable(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
00137
00154 int APC_API APC_GetRectifyTable(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
00155
00177 int APC_API APC_GetZDTable(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, PZDTABLEINFO pZDTableInfo);
00178
00195 int APC_API APC_SetSlaveYOffset(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
00196
00213 int APC_API APC_SetYOffset(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
00214
00231 int APC_API APC_SetYOffset_Advanced(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
00232
00236 int APC_API APC_SetSlaveRectifyTable(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
00237
00241 int APC_API APC_SetRectifyTable(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);
00242
00246 int APC_API APC_SetRectifyTable_Advanced(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer,
int BufferLength, int *pActualLength, int index);
00247
00251 int APC_API APC_SetZDTable(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, PZDTABLEINFO pZDTableInfo);
00252
00256 int APC_API APC_SetZDTable_Advanced(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, PZDTABLEINFO pZDTableInfo);
00257
00271 int APC_API APC_GetRectifyMatLogData(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, eSPCtrl_RectLogData
*pData, int index);
00272
00273 int APC_API APC_GetRectifyMatLogDataSlave(void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
eSPCtrl_RectLogData *pData, int index);
00274
00288 int APC_API APC_SetDepthDataTypeEx(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType, int pid);
00289
00302 int APC_API APC_SetDepthDataType(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType);
00303
00314 int APC_API APC_GetDepthDataType(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType);
00315
00323 int APC_API APC_SetHWPostProcess(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool enable);
00324
00332 int APC_API APC_GetHWPostProcess(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool* enable);
00333
00343 int APC_API APC_EnableInterleave(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool enable);
00344
00353 bool APC_API APC_IsInterleaveDevice(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);
00354
00364 int APC_API APC_EnableSerialCount( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool enable );
00365
00385 struct PointCloudInfo
00386 {
00387     //normal data
00388     float centerX;
00389     float centerY;
00390     float focallength;
00391     float disparityToW[ 2048 ];
00392     int disparity_len;
00393     WORD wDepthType;
00394     //float centerXP;

```

```

00395     //float centerYP;
00396     //float focalXP;
00397     //float focalYP;
00398     //float baseline;
00399     float fx1;
00400     float fy1;
00401     float fx2;
00402     float fy2;
00403     float cx1;
00404     float cy1;
00405     float cx2;
00406     float cy2;
00407     float Tx;
00408
00409     //multi-lens data
00410     float focalLength_K;
00411     float baseline_K;
00412     float diff_K;
00413     //slave data
00414     float CamMat2[9];
00415     float RotaMat[9];
00416     float TranMat[3];
00417
00418     PointCloudInfo() { memset( this, NULL, sizeof( PointCloudInfo ) ); }
00419 };
00420
00421 int APC_API APC_GetPointCloud( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char* ImgColor,
00422                               int CW, int CH,
00423                               unsigned char*
00424                               ImgDepth, int DW, int DH,
00425                               PointCloudInfo*
00426                               pPointCloudInfo,
00427                               unsigned char*
00428                               pPointCloudRGB, float* pPointCloudXYZ, float Near, float Far );
00429 int APC_API APC_FlyingDepthCancellation_D8( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned
00430 char* pdepthD8, int width, int height );
00431
00432 int APC_API APC_FlyingDepthCancellation_D11( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned
00433 char* pdepthD11, int width, int height );
00434
00435
00436
00437 int APC_API APC_DepthMerge( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char**
00438 pDepthBufList, float *pDepthMergeOut,
00439 unsigned char *pDepthMergeFlag, int nDWidth, int nDHeight, float fFocus, float * pBaseline, float
00440 * pWRNear, float * pWRFar, float * pWRFusion, int nMergeNum );
00441
00442 int APC_API APC_AdjustFocalLengthFromFlash( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int width, int
00443 height );
00444
00445 int APC_API APC_AdjustFocalLength( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int width, int height,
00446 int pixelUnit );
00447
00448 int APC_API APC_GetDeviceFocalLength( void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
00449 int *pLeftFx, int *pLeftFy, int *pRightFx, int *pRightFy );
00450
00451 int APC_API APC_GetFlashFocalLength( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int width, int
00452 height,
00453 int *pLeftFx, int *pLeftFy, int *pRightFx, int *pRightFy, int *pPixelUnit );
00454
00455
00456
00457
00458
00459
00460
00461
00462
00463
00464
00465
00466
00467
00468
00469
00470
00471
00472
00473
00474
00475
00476
00477
00478
00479
00480
00481
00482
00483
00484
00485
00486
00487
00488
00489
00490
00491
00492
00493
00494
00495
00496
00497
00498
00499
00500
00501
00502
00503
00504
00505
00506
00507
00508
00509
00510
00511
00512
00513
00514
00515
00516
00517
00518
00519
00520
00521
00522
00523
00524
00525
00526
00527
00528
00529
00530
00531
00532
00533
00534
00535
00536
00537
00538
00539
00540
00541
00542
00543
00544
00545
00546
00547
00548
00549
00550
00551
00552
00553
00554
00555
00556
00557
00558
00559
00560
00561
00562
00563
00564
00565
00566
00567
00568
00569
00570
00571
00572
00573
00574
00575
00576
00577
00578
00579
00580
00581
00582
00583
00584
00585
00586
00587
00588
00589
00590
00591
00592
00593
00594
00595
00596
00597
00598
00599
00600
00601
00602
00603
00604
00605
00606
00607
00608
00609
00610
00611
00612
00613
00614
00615
00616
00617
00618
00619
00620
00621
00622
00623
00624
00625
00626
00627
00628
00629
00630
00631
00632
00633
00634
00635
00636
00637
00638
00639
00640
00641
00642
00643
00644
00645
00646
00647
00648
00649
00650
00651
00652
00653
00654
00655
00656
00657
00658
00659
00660
00661
00662
00663
00664
00665
00666
00667
00668
00669
00670
00671
00672
00673
00674
00675
00676
00677
00678
00679
00680
00681
00682
00683
00684
00685
00686
00687
00688
00689
00690
00691
00692
00693
00694
00695
00696
00697
00698
00699
00700
00701
00702
00703
00704
00705
00706
00707
00708
00709
00710
00711
00712
00713
00714
00715
00716
00717
00718
00719
00720
00721
00722
00723
00724
00725
00726
00727
00728
00729
00730
00731
00732
00733
00734
00735
00736
00737
00738
00739
00740
00741
00742
00743
00744
00745
00746
00747
00748
00749
00750
00751
00752
00753
00754
00755
00756
00757
00758
00759
00760
00761
00762
00763
00764
00765
00766
00767
00768
00769
00770
00771
00772
00773
00774
00775
00776
00777
00778
00779
00780
00781
00782
00783
00784
00785
00786
00787
00788
00789
00790
00791
00792
00793
00794
00795
00796
00797
00798
00799
00800
00801
00802
00803
00804
00805
00806
00807
00808
00809
00810
00811
00812
00813
00814
00815
00816
00817
00818
00819
00820
00821
00822
00823
00824
00825
00826
00827
00828
00829
00830
00831
00832
00833
00834
00835
00836
00837
00838
00839
00840
00841
00842
00843
00844
00845
00846
00847
00848
00849
00850
00851
00852
00853
00854
00855
00856
00857
00858
00859
00860
00861
00862
00863
00864
00865
00866
00867
00868
00869
00870
00871
00872
00873
00874
00875
00876
00877
00878
00879
00880
00881
00882
00883
00884
00885
00886
00887
00888
00889
00890
00891
00892
00893
00894
00895
00896
00897
00898
00899
00900
00901
00902
00903
00904
00905
00906
00907
00908
00909
00910
00911
00912
00913
00914
00915
00916
00917
00918
00919
00920
00921
00922
00923
00924
00925
00926
00927
00928
00929
00930
00931
00932
00933
00934
00935
00936
00937
00938
00939
00940
00941
00942
00943
00944
00945
00946
00947
00948
00949
00950
00951
00952
00953
00954
00955
00956
00957
00958
00959
00960
00961
00962
00963
00964
00965
00966
00967
00968
00969
00970
00971
00972
00973
00974
00975
00976
00977
00978
00979
00980
00981
00982
00983
00984
00985
00986
00987
00988
00989
00990
00991
00992
00993
00994
00995
00996
00997
00998
00999

```

5.5 eSPDI_ErrCode.h File Reference

definition of eYs3D SDK error code Copyright: This file copyright (C) 2017 by

5.5.1 Detailed Description

definition of eYs3D SDK error code Copyright: This file copyright (C) 2017 by

eYs3D company

An unpublished work. All rights reserved.

This file is proprietary information, and may not be disclosed or copied without the prior permission of eYs3D company.

5.6 eSPDI_ErrCode.h

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

```

00001
00013
00014 //define Error Code by Wolf 2013/08/30
00015 #define APC_OK 0
00016 #define APC_NoDevice -1
00017 #define APC_NullPtr -2
00018 #define APC_ErrBufLen -3
00019 #define APC_Init_Fail -4
00020 #define APC_NoZDTable -5
00021 #define APC_READFLASHFAIL -6
00022 #define APC_WRITEFLASHFAIL -7
00023 #define APC_VERIFY_DATA_FAIL -8
00024 #define APC_KEEP_DATA_FAIL -9
00025 #define APC_RECT_DATA_LEN_FAIL -10
00026 #define APC_RECT_DATA_PARSING_FAIL -11
00027 #define APC_RET_BAD_PARAM -12
00028 #define APC_RET_OPEN_FILE_FAIL -13
00029 #define APC_NO_CALIBRATION_LOG -14
00030 #define APC_POSTPROCESS_INIT_FAIL -15
00031 #define APC_POSTPROCESS_NOT_INIT -16
00032 #define APC_POSTPROCESS_FRAME_FAIL -17
00033 #define APC_NotSupport -18
00034 #define APC_OpenFileFail -19
00035 #define APC_READ_REG_FAIL -20
00036 #define APC_WRITE_REG_FAIL -21
00037 #define APC_SET_FPS_FAIL -22
00038 #define APC_VIDEO_RENDER_FAIL -23
00039 #define APC_OPEN_DEVICE_FAIL -24
00040 #define APC_FIND_DEVICE_FAIL -25
00041 #define APC_GET_IMAGE_FAIL -26
00042 #define APC_NOT_SUPPORT_RES -27
00043 #define APC_CALLBACK_REGISTER_FAIL -28
00044 #define APC_DEVICE_NOT_SUPPORT -29
00045
00046 // for 3D Scanner +
00047 #define APC_ILLEGAL_ANGLE -30
00048 #define APC_ILLEGAL_STEP -31
00049 #define APC_ILLEGAL_TIMEPERSTEP -32
00050 #define APC_MOTOR_RUNNING -33
00051 #define APC_GETSENSORREG_FAIL -34
00052 #define APC_SETSENSORREG_FAIL -35
00053 #define APC_READ_X_AXIS_FAIL -36
00054 #define APC_READ_Y_AXIS_FAIL -37
00055 #define APC_READ_Z_AXIS_FAIL -38
00056 #define APC_READ_PRESS_DATA_FAIL -39
00057 #define APC_READ_TEMPERATURE_FAIL -40
00058 #define APC_RETURNHOME_RUNNING -41
00059 #define APC_MOTOTSTOP_BY_HOME_INDEX -42
00060 #define APC_MOTOTSTOP_BY_PROTECT_SCHEME -43
00061 #define APC_MOTOTSTOP_BY_NORMAL -44
00062 #define APC_ILLEGAL_FIRMWARE_VERSION -45
00063 #define APC_ILLEGAL_STEPPERTIME -46
00064 // for 3D Scanner -
00065
00066 // For AEAWB + 2015/01/28 by Wolf
00067 #define APC_GET_PU_PROP_VAL -50
00068 #define APC_SET_PU_PROP_VAL -51
00069 #define APC_GET_CT_PROP_VAL -52
00070 #define APC_SET_CT_PROP_VAL -53
00071 // For AEAWB - 2015/01/28 by Wolf
00072
00073 // for Dewarping + Stitching +
00074 #define APC_INVALID_USERDATA -70
00075 #define APC_MAP_LUT_FAIL -71
00076 #define APC_APPEND_TO_FILE_FRONT_FAIL -72
00077 // for Dewarping + Stitching -
00078
00079 #define APC_TOO_MANY_DEVICE -80
00080 #define APC_ACCESS_MP4_EXTRA_DATA_FAIL -81

```

Index

APC_CloseDevice
 eSPDI_Common.h, [24](#)

APC_DEVICE_TYPE
 eSPDI_Common.h, [23](#)

APC_DeviceEventFn
 eSPDI_Common.h, [22](#)

APC_DisableAE
 eSPDI_Common.h, [24](#)

APC_DisableAWB
 eSPDI_Common.h, [24](#)

APC_EnableAE
 eSPDI_Common.h, [25](#)

APC_EnableAWB
 eSPDI_Common.h, [25](#)

APC_EnableGPUAcceleration
 eSPDI_Common.h, [25](#)

APC_FindDevice
 eSPDI_Common.h, [27](#)

APC_GetCTPropVal
 eSPDI_Common.h, [27](#)

APC_GetCurrentIRValue
 eSPDI_Common.h, [28](#)

APC_GetDepthDataType
 eSPDI_DM.h, [64](#)

APC_GetDepthFilterVersion
 eSPDI_Common.h, [28](#)

APC_GetDeviceInfo
 eSPDI_Common.h, [28](#)

APC_GetDeviceInfoEx
 eSPDI_Common.h, [29](#)

APC_GetDeviceNumber
 eSPDI_Common.h, [29](#)

APC_GetDeviceResolutionList
 eSPDI_Common.h, [30](#)

APC_GetFileData
 eSPDI_DM.h, [64](#)

APC_GetFlexibleGyroData
 eSPDI_Common.h, [30](#)

APC_GetFlexibleGyroLength
 eSPDI_Common.h, [30](#)

APC_GetFWFSLength
 eSPDI_DM.h, [64](#)

APC_GetFWRegister
 eSPDI_Common.h, [31](#)

APC_GetFwVersion
 eSPDI_Common.h, [31](#)

APC_GetGPIOValue
 eSPDI_Common.h, [32](#)

APC_GetHWPostProcess
 eSPDI_DM.h, [65](#)

APC_GetHWRegister
 eSPDI_Common.h, [32](#)

APC_GetIRMaxValue
 eSPDI_Common.h, [33](#)

APC_GetIRMinValue
 eSPDI_Common.h, [33](#)

APC_GetLogData
 eSPDI_Common.h, [33](#)

APC_GetMultiBytesHWRegister
 eSPDI_Common.h, [34](#)

APC_GetPidVid
 eSPDI_Common.h, [34](#)

APC_GetPUPropVal
 eSPDI_Common.h, [36](#)

APC_GetRectifyMatLogData
 eSPDI_DM.h, [65](#)

APC_GetRectifyTable
 eSPDI_DM.h, [66](#)

APC_GetSensorRegister
 eSPDI_Common.h, [36](#)

APC_GetSlaveHWRegister
 eSPDI_Common.h, [37](#)

APC_GetSlaveLogData
 eSPDI_Common.h, [37](#)

APC_GetSlaveRectifyTable
 eSPDI_DM.h, [66](#)

APC_GetSlaveSensorRegister
 eSPDI_Common.h, [38](#)

APC_GetSlaveYOffset
 eSPDI_DM.h, [67](#)

APC_GetYOffset
 eSPDI_DM.h, [67](#)

APC_GetZDTable
 eSPDI_DM.h, [68](#)

APC_ImgCallbackFn
 eSPDI_Common.h, [22](#)

APC_Init
 eSPDI_Common.h, [38](#)

APC_Init2
 eSPDI_Common.h, [39](#)

APC_Is360Device
 eSPDI_Common.h, [39](#)

APC_IsInterleaveDevice
 eSPDI_DM.h, [68](#)

APC_OpenDevice
 eSPDI_Common.h, [40](#)

APC_ReadFlashData
 eSPDI_Common.h, [40](#)

- APC_RefreshDevice
 - eSPDI_Common.h, [41](#)
- APC_RegisterDeviceEvents
 - eSPDI_Common.h, [41](#)
- APC_Release
 - eSPDI_Common.h, [42](#)
- APC_ResetUNPData
 - eSPDI_Common.h, [42](#)
- APC_SENSOR_TYPE_AR0134
 - eSPDI_Common.h, [23](#)
- APC_SENSOR_TYPE_AR0135
 - eSPDI_Common.h, [23](#)
- APC_SENSOR_TYPE_AR0144
 - eSPDI_Common.h, [23](#)
- APC_SENSOR_TYPE_AR0330
 - eSPDI_Common.h, [23](#)
- APC_SENSOR_TYPE_AR1335
 - eSPDI_Common.h, [23](#)
- APC_SENSOR_TYPE_H22
 - eSPDI_Common.h, [23](#)
- APC_SENSOR_TYPE_OV7740
 - eSPDI_Common.h, [23](#)
- APC_SENSOR_TYPE_OV9282
 - eSPDI_Common.h, [23](#)
- APC_SENSOR_TYPE_OV9714
 - eSPDI_Common.h, [23](#)
- APC_SetCTPropVal
 - eSPDI_Common.h, [42](#)
- APC_SetCurrentIRValue
 - eSPDI_Common.h, [43](#)
- APC_SetDepthDataType
 - eSPDI_DM.h, [69](#)
- APC_SetFWRegister
 - eSPDI_Common.h, [43](#)
- APC_SetGPIOCtrl
 - eSPDI_Common.h, [43](#)
- APC_SetGPIOValue
 - eSPDI_Common.h, [44](#)
- APC_SetHuffmanTableData
 - eSPDI_Common.h, [44](#)
- APC_SetHWPostProcess
 - eSPDI_DM.h, [69](#)
- APC_SetHWRegister
 - eSPDI_Common.h, [45](#)
- APC_SetIRMaxValue
 - eSPDI_Common.h, [45](#)
- APC_SetIRMaxValueUnleashed
 - eSPDI_Common.h, [45](#)
- APC_SetLogData
 - eSPDI_Common.h, [46](#)
- APC_SetLogData_Advanced
 - eSPDI_Common.h, [46](#)
- APC_SetMultiBytesHWRegister
 - eSPDI_Common.h, [47](#)
- APC_SetPidVid
 - eSPDI_Common.h, [47](#)
- APC_SetPUPPropVal
 - eSPDI_Common.h, [48](#)
- APC_SetQuantizationTableData
 - eSPDI_Common.h, [48](#)
- APC_SetSensorTypeName
 - eSPDI_Common.h, [48](#)
- APC_SetSlaveHWRegister
 - eSPDI_Common.h, [49](#)
- APC_SetSlaveLogData
 - eSPDI_Common.h, [49](#)
- APC_SetSlaveYOffset
 - eSPDI_DM.h, [69](#)
- APC_SetUserData
 - eSPDI_Common.h, [50](#)
- APC_SetYOffset
 - eSPDI_DM.h, [70](#)
- APC_SetYOffset_Advanced
 - eSPDI_DM.h, [70](#)
- AXES1
 - eSPDI_Common.h, [23](#)
- Date
 - eSPCtrl_RectLogData, [11](#)
- DEVINFORMATION
 - eSPDI_Common.h, [22](#)
- DEVINFORMATIONEX, [9](#)
 - nChipID, [9](#)
 - nDevType, [9](#)
 - strDevName, [9](#)
 - strDevPath, [10](#)
 - wPID, [10](#)
 - wUsbNode, [10](#)
 - wVID, [10](#)
- eSPCtrl_RectLogData, [11](#)
 - Date, [11](#)
 - eSPDI_Common.h, [23](#)
- eSPDI_Common.h, [17](#)
 - APC_CloseDevice, [24](#)
 - APC_DEVICE_TYPE, [23](#)
 - APC_DeviceEventFn, [22](#)
 - APC_DisableAE, [24](#)
 - APC_DisableAWB, [24](#)
 - APC_EnableAE, [25](#)
 - APC_EnableAWB, [25](#)
 - APC_EnableGPUAcceleration, [25](#)
 - APC_FindDevice, [27](#)
 - APC_GetCTPropVal, [27](#)
 - APC_GetCurrentIRValue, [28](#)
 - APC_GetDepthFilterVersion, [28](#)
 - APC_GetDeviceInfo, [28](#)
 - APC_GetDeviceInfoEx, [29](#)
 - APC_GetDeviceNumber, [29](#)
 - APC_GetDeviceResolutionList, [30](#)
 - APC_GetFlexibleGyroData, [30](#)
 - APC_GetFlexibleGyroLength, [30](#)
 - APC_GetFWRegister, [31](#)
 - APC_GetFwVersion, [31](#)
 - APC_GetGPIOValue, [32](#)
 - APC_GetHWRegister, [32](#)
 - APC_GetIRMaxValue, [33](#)

APC_GetIRMinValue, 33
 APC_GetLogData, 33
 APC_GetMultiBytesHWRegister, 34
 APC_GetPidVid, 34
 APC_GetPUPPropVal, 36
 APC_GetSensorRegister, 36
 APC_GetSlaveHWRegister, 37
 APC_GetSlaveLogData, 37
 APC_GetSlaveSensorRegister, 38
 APC_ImgCallbackFn, 22
 APC_Init, 38
 APC_Init2, 39
 APC_Is360Device, 39
 APC_OpenDevice, 40
 APC_ReadFlashData, 40
 APC_RefreshDevice, 41
 APC_RegisterDeviceEvents, 41
 APC_Release, 42
 APC_ResetUNPData, 42
 APC_SENSOR_TYPE_AR0134, 23
 APC_SENSOR_TYPE_AR0135, 23
 APC_SENSOR_TYPE_AR0144, 23
 APC_SENSOR_TYPE_AR0330, 23
 APC_SENSOR_TYPE_AR1335, 23
 APC_SENSOR_TYPE_H22, 23
 APC_SENSOR_TYPE_OV7740, 23
 APC_SENSOR_TYPE_OV9282, 23
 APC_SENSOR_TYPE_OV9714, 23
 APC_SetCTPropVal, 42
 APC_SetCurrentIRValue, 43
 APC_SetFWRegister, 43
 APC_SetGPIOCtrl, 43
 APC_SetGPIOValue, 44
 APC_SetHuffmanTableData, 44
 APC_SetHWRegister, 45
 APC_SetIRMaxValue, 45
 APC_SetIRMaxValueUnleashed, 45
 APC_SetLogData, 46
 APC_SetLogData_Advanced, 46
 APC_SetMultiBytesHWRegister, 47
 APC_SetPidVid, 47
 APC_SetPUPPropVal, 48
 APC_SetQuantizationTableData, 48
 APC_SetSensorTypeName, 48
 APC_SetSlaveHWRegister, 49
 APC_SetSlaveLogData, 49
 APC_SetUserData, 50
 AXES1, 23
 DEVINFORMATION, 22
 eSPCtrl_RectLogData, 23
 OTHERS, 23
 PARALUT, 23
 PUMA, 23
 SENSOR_TYPE_NAME, 23
 USERDATA_SECTION_0, 24
 USERDATA_SECTION_1, 24
 USERDATA_SECTION_10, 24
 USERDATA_SECTION_2, 24
 USERDATA_SECTION_3, 24
 USERDATA_SECTION_4, 24
 USERDATA_SECTION_5, 24
 USERDATA_SECTION_6, 24
 USERDATA_SECTION_7, 24
 USERDATA_SECTION_8, 24
 USERDATA_SECTION_9, 24
 USERDATA_SECTION_INDEX, 23
 USERDATA_SECTION_NUM, 24
 eSPDI_DM.h, 62
 APC_GetDepthDataType, 64
 APC_GetFileData, 64
 APC_GetFWFSLength, 64
 APC_GetHWPostProcess, 65
 APC_GetRectifyMatLogData, 65
 APC_GetRectifyTable, 66
 APC_GetSlaveRectifyTable, 66
 APC_GetSlaveYOffset, 67
 APC_GetYOffset, 67
 APC_GetZDTable, 68
 APC_IsInterleaveDevice, 68
 APC_SetDepthDataType, 69
 APC_SetHWPostProcess, 69
 APC_SetSlaveYOffset, 69
 APC_SetYOffset, 70
 APC_SetYOffset_Advanced, 70
 eSPDI_ErrCode.h, 73
 Introduction, 3
 nChipID
 DEVINFORMATIONEX, 9
 tagDEVINFORMATION, 14
 nDevType
 DEVINFORMATIONEX, 9
 tagDEVINFORMATION, 14
 OTHERS
 eSPDI_Common.h, 23
 PARALUT
 eSPDI_Common.h, 23
 ParaLUT, 11
 PUMA
 eSPDI_Common.h, 23
 SENSOR_TYPE_NAME
 eSPDI_Common.h, 23
 strDevName
 DEVINFORMATIONEX, 9
 tagDEVINFORMATION, 14
 strDevPath
 DEVINFORMATIONEX, 10
 tagDEVINFORMATION, 14
 tagDEVINFORMATION, 14
 nChipID, 14
 nDevType, 14
 strDevName, 14
 strDevPath, 14

- wPID, [14](#)
- wUsbNode, [15](#)
- wVID, [15](#)

- USERDATA_SECTION_0
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_1
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_10
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_2
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_3
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_4
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_5
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_6
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_7
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_8
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_9
 - eSPDI_Common.h, [24](#)
- USERDATA_SECTION_INDEX
 - eSPDI_Common.h, [23](#)
- USERDATA_SECTION_NUM
 - eSPDI_Common.h, [24](#)

- wPID
 - DEVINFORMATIONEX, [10](#)
 - tagDEVINFORMATION, [14](#)
- wUsbNode
 - DEVINFORMATIONEX, [10](#)
 - tagDEVINFORMATION, [15](#)
- wVID
 - DEVINFORMATIONEX, [10](#)
 - tagDEVINFORMATION, [15](#)