



**eYS3D**  
Microelectronics

eYs3D Windows SDK

1.5.2.5

Generated by Doxygen 1.9.1

<b>1 Introduction</b>	<b>3</b>
<b>2 Data Structure Index</b>	<b>5</b>
2.1 Data Structures	5
<b>3 File Index</b>	<b>7</b>
3.1 File List	7
<b>4 Data Structure Documentation</b>	<b>9</b>
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 wPID	10
4.1.2.5 wUsbNode	10
4.1.2.6 wVID	10
4.2 eSPCtrl_RectLogData	10
4.2.1 Detailed Description	11
4.3 ParalUT	11
4.3.1 Detailed Description	13
4.4 tagDEVINFORMATION	13
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 wPID	14
4.4.2.5 wUsbNode	15
4.4.2.6 wVID	15
<b>5 File Documentation</b>	<b>17</b>
5.1 eSPDI_Common.h File Reference	17
5.1.1 Detailed Description	21
5.1.2 Typedef Documentation	21
5.1.2.1 DEVINFORMATION	21
5.1.2.2 eSPCtrl_RectLogData	21
5.1.2.3 PARALUT	21
5.1.3 Enumeration Type Documentation	22
5.1.3.1 APC_DEVICE_TYPE	22
5.1.3.2 SENSOR_TYPE_NAME	22
5.1.3.3 USERDATA_SECTION_INDEX	22
5.1.4 Function Documentation	23

5.1.4.1 APC_CloseDevice()	23
5.1.4.2 APC_DisableAE()	23
5.1.4.3 APC_DisableAWB()	24
5.1.4.4 APC_EnableAE()	24
5.1.4.5 APC_EnableAWB()	24
5.1.4.6 APC_EnableGPUAcceleration()	25
5.1.4.7 APC_FindDevice()	25
5.1.4.8 APC_GetCTPropVal()	26
5.1.4.9 APC_GetCurrentIRValue()	26
5.1.4.10 APC_GetDepthFilterVersion()	27
5.1.4.11 APC_GetDeviceNumber()	27
5.1.4.12 APC_GetDeviceResolutionList()	27
5.1.4.13 APC_GetFlexibleGyroData()	28
5.1.4.14 APC_GetFlexibleGyroLength()	28
5.1.4.15 APC_GetFWRegister()	29
5.1.4.16 APC_GetFwVersion()	29
5.1.4.17 APC_GetGPIOValue()	30
5.1.4.18 APC_GetHWRegister()	30
5.1.4.19 APC_GetIRMaxValue()	31
5.1.4.20 APC_GetIRMinValue()	31
5.1.4.21 APC_GetLogData()	32
5.1.4.22 APC_GetPidVid()	32
5.1.4.23 APC_GetPUPropVal()	33
5.1.4.24 APC_GetSensorRegister()	34
5.1.4.25 APC_GetSlaveHWRegister()	34
5.1.4.26 APC_GetSlaveLogData()	35
5.1.4.27 APC_GetSlaveSensorRegister()	35
5.1.4.28 APC_Init()	36
5.1.4.29 APC_Init2()	36
5.1.4.30 APC_Is360Device()	37
5.1.4.31 APC_OpenDevice()	37
5.1.4.32 APC_ReadFlashData()	38
5.1.4.33 APC_RefreshDevice()	39
5.1.4.34 APC_RegisterDeviceEvents()	39
5.1.4.35 APC_Release()	39
5.1.4.36 APC_ResetUNPData()	40
5.1.4.37 APC_SetCTPropVal()	40
5.1.4.38 APC_SetCurrentIRValue()	41
5.1.4.39 APC_SetFWRegister()	41
5.1.4.40 APC_SetGPIOCtrl()	42
5.1.4.41 APC_SetGPIOValue()	42
5.1.4.42 APC_SetHuffmanTableData()	43

5.1.4.43 APC_SetHWRegister()	43
5.1.4.44 APC_SetIRMaxValue()	44
5.1.4.45 APC_SetLogData()	44
5.1.4.46 APC_SetLogData_Advanced()	45
5.1.4.47 APC_SetPidVid()	45
5.1.4.48 APC_SetPUPropVal()	46
5.1.4.49 APC_SetQuantizationTableData()	46
5.1.4.50 APC_SetSensorTypeName()	46
5.1.4.51 APC_SetSlaveHWRegister()	48
5.1.4.52 APC_SetSlaveLogData()	48
5.1.4.53 APC_SetUserData()	49
5.2 eSPDI_DM.h File Reference	49
5.2.1 Detailed Description	51
5.2.2 Function Documentation	51
5.2.2.1 APC_GetDepthDataType()	51
5.2.2.2 APC_GetRectifyMatLogData()	51
5.2.2.3 APC_GetRectifyTable()	52
5.2.2.4 APC_GetSlaveRectifyTable()	52
5.2.2.5 APC_GetSlaveYOffset()	53
5.2.2.6 APC_GetYOffset()	53
5.2.2.7 APC_GetZDTable()	54
5.2.2.8 APC_IsInterleaveDevice()	55
5.2.2.9 APC_SetDepthDataType()	55
5.2.2.10 APC_SetHWPostProcess()	56
5.2.2.11 APC_SetSlaveYOffset()	56
5.2.2.12 APC_SetYOffset()	57
5.2.2.13 APC_SetYOffset_Advanced()	57
5.3 eSPDI_ErrCode.h File Reference	58
5.3.1 Detailed Description	58
<b>Index</b>	<b>59</b>



# 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	<a href="#">eSPDI_Common.h</a>	Basic API declaration header
		<a href="#">eSPDI_DM.h</a>	Depth Map specific API declaration header
		<a href="#">eSPDI_ErrCode.h</a>	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:

<a href="#">DEVINFORMATIONEX</a>	9
<a href="#">eSPCtrl_RectLogData</a>	
<a href="#">ESPCtrl_RectLogData</a>	10
<a href="#">ParaLUT</a>	
<a href="#">ParaLUT</a>	11
<a href="#">tagDEVINFORMATION</a>	
<a href="#">DEVINFORMATION</a>	13





## Chapter 3

# File Index

### 3.1 File List

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

<a href="#">eSPDI_Common.h</a>	EYs3D SDK API export functions, data structure and variable definition . . . . .	17
<a href="#">eSPDI_DM.h</a>	EYs3D SDK API export functions, data structure and variable definition for depth map module .	49
<a href="#">eSPDI_ErrCode.h</a>	Definition of eYs3D SDK error code Copyright: This file copyright (C) 2017 by . . . . .	58



## Chapter 4

# Data Structure Documentation

### 4.1 DEVINFORMATIONEX

#### Data Fields

- unsigned short [wPID](#)
- unsigned short [wVID](#)
- char [strDevName](#) [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 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

#### 4.1.2.5 wUsbNode

```
unsigned short wUsbNode
```

USB Node

#### 4.1.2.6 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](#)

### 4.2.1 Detailed Description

#### [eSPCtrl\\_RectLogData](#)

Rectified log data structure

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

- [eSPDI\\_Common.h](#)

## 4.3 ParaLUT

[ParaLUT](#).

### 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: 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  $\text{img\_overlay\_LR(RL)} = \text{overlay\_value} + \text{overlay\_ratio}$
- 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.

### Data Fields

- unsigned short [wPID](#)
- unsigned short [wVID](#)
- char \* [strDevName](#)
- 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 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

#### 4.4.2.5 wUsbNode

unsigned short wUsbNode

USB Node

#### 4.4.2.6 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 struct [tagDEVINFORMATION](#) [DEVINFORMATION](#)  
*DEVINFORMATION.*

#### 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\\_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\\_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\\_GetSlaveHWRRegister](#) (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\\_SetSlaveHWRRegister](#) (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\\_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\\_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 bIsEnable)

*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](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/ff568122(85).aspx) [https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089\(v=vs.11.0\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089(v=vs.11.0).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](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/ff567802(85).aspx) [https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089\(v=vs.11.0\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089(v=vs.11.0).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\\_ResetUNPData](#) (void \*pHandleApcDI, PDEVSELINFO pDevSelInfo)  
*Reset the UNProtection area's datum.*
- 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 DEVINFORMATION

```
typedef struct tagDEVINFORMATION DEVINFORMATION
```

DEVINFORMATION.

device information

#### 5.1.2.2 eSPCtrl\_RectLogData

```
typedef struct eSPCtrl_RectLogData eSPCtrl_RectLogData
```

[eSPCtrl\\_RectLogData](#)

Rectified log data structure

#### 5.1.2.3 PARALUT

```
typedef struct ParaLUT 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

## Enumerator

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_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_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_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_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_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 initilized ApcDI SDK instance
<i>pDevSelInfo</i>	pointer of device select index
<i>enable</i>	true:enable, fales:diable

## Returns

success: APC\_OK, others: see [eSPDI\\_ErrCode.h](#)

**5.1.4.7 APC\_FindDevice()**

```
int 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 initilized ApcDI SDK instance
---------------------	--

## Returns

success: APC\_OK, others: see [eSPDI\\_ErrCode.h](#)

#### 5.1.4.8 APC\_GetCTPropVal()

```
int 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_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()**

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

get depth filter version

**Parameters**

<i>pHandleApcDI</i>	the pointer to the initialized 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\_GetDeviceNumber()**

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

get eYs3D USB device numbers

**Parameters**

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

**Returns**

number of eYs3D device

**5.1.4.12 APC\_GetDeviceResolutionList()**

```
int 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](#)

#### 5.1.4.13 APC\_GetFlexibleGyroData()

```

int 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

#### 5.1.4.14 APC\_GetFlexibleGyroLength()

```

int 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](#)

**5.1.4.15 APC\_GetFWRegister()**

```
int 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](#)

**5.1.4.16 APC\_GetFwVersion()**

```
int 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](#)

**5.1.4.17 APC\_GetGPIOValue()**

```
int 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](#)

**5.1.4.18 APC\_GetHWRegister()**

```
int 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](#)

**5.1.4.19 APC\_GetIRMaxValue()**

```
int 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](#)

**5.1.4.20 APC\_GetIRMinValue()**

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

get minimum IR value the module support

**Parameters**

<i>pHandleApcDI</i>	the pointer to the initialized 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](#)

**5.1.4.21 APC\_GetLogData()**

```
int 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](#)

**5.1.4.22 APC\_GetPidVid()**

```
int 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](#)

## 5.1.4.23 APC\_GetPUPPropVal()

```
int 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](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](#)

#### 5.1.4.24 APC\_GetSensorRegister()

```
int 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 <a href="#">SENSOR_TYPE_NAME</a> 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](#)

#### 5.1.4.25 APC\_GetSlaveHWRegister()

```
int 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](#)

**5.1.4.26 APC\_GetSlaveLogData()**

```
int 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](#)

**5.1.4.27 APC\_GetSlaveSensorRegister()**

```
int 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 <a href="#">SENSOR_TYPE_NAME</a> enum definition

## Parameters

<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\_Init()

```
int 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.

## 5.1.4.29 APC\_Init2()

```
int 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.

#### 5.1.4.30 APC\_Is360Device()

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

check module is spherical device or not

### Parameters

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

### Returns

true: module support 360, false: not support

#### 5.1.4.31 APC\_OpenDevice()

```
int 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 initialized 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



## Parameters

<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.  <div style="text-align: center;"><b>Table 5.34 Image Control Mode</b></div> <table border="1"> <thead> <tr> <th>Mode</th><th>Description</th></tr> </thead> <tbody> <tr> <td>0x01</td><td>color and depth frame output synchronously, for depth map module only</td></tr> <tr> <td>0x02</td><td>enable post-process, for Depth Map module only</td></tr> <tr> <td>0x04</td><td>stitch images if this bit is set, for fisheye spherical module only</td></tr> <tr> <td>0x08</td><td>use OpenCL in stitching. This bit effective only when bit-2 is set.</td></tr> </tbody> </table>	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.
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](#)

## 5.1.4.32 APC\_ReadFlashData()

```
int 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](#)

#### 5.1.4.33 APC\_RefreshDevice()

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

refresh all eYs3D UVC devices

##### Parameters

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

##### Returns

success: APC\_OK, others: see [eSPDI\\_ErrCode.h](#)

#### 5.1.4.34 APC\_RegisterDeviceEvents()

```
int 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](#)

#### 5.1.4.35 APC\_Release()

```
void 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.

**5.1.4.36 APC\_ResetUNPData()**

```
int 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](#)

**5.1.4.37 APC\_SetCTPropVal()**

```
int 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](#)

**5.1.4.38 APC\_SetCurrentIRValue()**

```
int 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](#)

**5.1.4.39 APC\_SetFWRegister()**

```
int 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](#)

**5.1.4.40 APC\_SetGPIOCtrl()**

```
int 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](#)

**5.1.4.41 APC\_SetGPIOValue()**

```
int 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](#)

#### 5.1.4.42 APC\_SetHuffmanTableData()

```
int 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_huffman.dat file is generated

##### Returns

success: APC\_OK, others: see [eSPDI\\_ErrCode.h](#)

#### 5.1.4.43 APC\_SetHWRegister()

```
int 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](#)

#### 5.1.4.44 APC\_SetIRMaxValue()

```
int 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](#)

#### 5.1.4.45 APC\_SetLogData()

```
int 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](#)

#### 5.1.4.46 APC\_SetLogData\_Advanced()

```
int 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 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](#)

#### 5.1.4.47 APC\_SetPidVid()

```
int 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](#)



#### 5.1.4.48 APC\_SetPUPPropVal()

```
int 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](#)

#### 5.1.4.49 APC\_SetQuantizationTableData()

```
int 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](#)

#### 5.1.4.50 APC\_SetSensorTypeName()

```
int 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

**5.1.4.51 APC\_SetSlaveHWRegister()**

```
int 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](#)

**5.1.4.52 APC\_SetSlaveLogData()**

```
int 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](#)

## 5.1.4.53 APC\_SetUserData()

```
int 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\_DM.h File Reference

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

## Functions

- 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*
- bool APC\_API [APC\\_IsInterleaveDevice](#) (void \*pHandleApcDI, PDEVSELINFO pDevSelInfo)  
*enable or disable interleave function*

## 5.2.1 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.

## 5.2.2 Function Documentation

### 5.2.2.1 APC\_GetDepthDataType()

```
int 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](#)

### 5.2.2.2 APC\_GetRectifyMatLogData()

```
int 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 <a href="#">eSPCtrl_RectLogData</a> for detailed members
<i>index</i>	index to identify rectify log data for corresponding depth

## Returns

success: APC\_OK, others: see [eSPDI\\_ErrCode.h](#)

## 5.2.2.3 APC\_GetRectifyTable()

```
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](#)

## 5.2.2.4 APC\_GetSlaveRectifyTable()

```
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](#)

## 5.2.2.5 APC\_GetSlaveYOffset()

```
int 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](#)

## 5.2.2.6 APC\_GetYOffset()

```
int 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](#)

### 5.2.2.7 APC\_GetZDTable()

```

int 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](#)

**5.2.2.8 APC\_IsInterleaveDevice()**

```
bool 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

**5.2.2.9 APC\_SetDepthDataType()**

```
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> Generated by Doxygen	depth data type you want to set, see APC_DEPTH_DATA_xxx in APC_O.h \output success: APC_OK, others: see <a href="#">eSPDI_ErrCode.h</a>

### 5.2.2.10 APC\_SetHWPostProcess()

```
int 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](#)

### 5.2.2.11 APC\_SetSlaveYOffset()

```
int 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](#)

**5.2.2.12 APC\_SetYOffset()**

```
int 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](#)

**5.2.2.13 APC\_SetYOffset\_Advanced()**

```
int 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](#)

## 5.3 eSPDI\_ErrCode.h File Reference

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

### 5.3.1 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.

# Index

APC\_CloseDevice  
    eSPDI\_Common.h, [23](#)

APC\_DEVICE\_TYPE  
    eSPDI\_Common.h, [22](#)

APC\_DisableAE  
    eSPDI\_Common.h, [23](#)

APC\_DisableAWB  
    eSPDI\_Common.h, [24](#)

APC\_EnableAE  
    eSPDI\_Common.h, [24](#)

APC\_EnableAWB  
    eSPDI\_Common.h, [24](#)

APC\_EnableGPUAcceleration  
    eSPDI\_Common.h, [25](#)

APC\_FindDevice  
    eSPDI\_Common.h, [25](#)

APC\_GetCTPropVal  
    eSPDI\_Common.h, [25](#)

APC\_GetCurrentIRValue  
    eSPDI\_Common.h, [26](#)

APC\_GetDepthDataType  
    eSPDI\_DM.h, [51](#)

APC\_GetDepthFilterVersion  
    eSPDI\_Common.h, [27](#)

APC\_GetDeviceNumber  
    eSPDI\_Common.h, [27](#)

APC\_GetDeviceResolutionList  
    eSPDI\_Common.h, [27](#)

APC\_GetFlexibleGyroData  
    eSPDI\_Common.h, [28](#)

APC\_GetFlexibleGyroLength  
    eSPDI\_Common.h, [28](#)

APC\_GetFWRegister  
    eSPDI\_Common.h, [29](#)

APC\_GetFwVersion  
    eSPDI\_Common.h, [29](#)

APC\_GetGPIOValue  
    eSPDI\_Common.h, [30](#)

APC\_GetHWRegister  
    eSPDI\_Common.h, [30](#)

APC\_GetIRMaxValue  
    eSPDI\_Common.h, [31](#)

APC\_GetIRMinValue  
    eSPDI\_Common.h, [31](#)

APC\_GetLogData  
    eSPDI\_Common.h, [32](#)

APC\_GetPidVid  
    eSPDI\_Common.h, [32](#)

APC\_GetPUPropVal  
    eSPDI\_Common.h, [33](#)

APC\_GetRectifyMatLogData  
    eSPDI\_DM.h, [51](#)

APC\_GetRectifyTable  
    eSPDI\_DM.h, [52](#)

APC\_GetSensorRegister  
    eSPDI\_Common.h, [33](#)

APC\_GetSlaveHWRegister  
    eSPDI\_Common.h, [34](#)

APC\_GetSlaveLogData  
    eSPDI\_Common.h, [35](#)

APC\_GetSlaveRectifyTable  
    eSPDI\_DM.h, [52](#)

APC\_GetSlaveSensorRegister  
    eSPDI\_Common.h, [35](#)

APC\_GetSlaveYOffset  
    eSPDI\_DM.h, [53](#)

APC\_GetYOffset  
    eSPDI\_DM.h, [53](#)

APC\_GetZDTable  
    eSPDI\_DM.h, [54](#)

APC\_Init  
    eSPDI\_Common.h, [36](#)

APC\_Init2  
    eSPDI\_Common.h, [36](#)

APC\_Is360Device  
    eSPDI\_Common.h, [37](#)

APC\_IsInterleaveDevice  
    eSPDI\_DM.h, [55](#)

APC\_OpenDevice  
    eSPDI\_Common.h, [37](#)

APC\_ReadFlashData  
    eSPDI\_Common.h, [38](#)

APC\_RefreshDevice  
    eSPDI\_Common.h, [38](#)

APC\_RegisterDeviceEvents  
    eSPDI\_Common.h, [39](#)

APC\_Release  
    eSPDI\_Common.h, [39](#)

APC\_ResetUNPData  
    eSPDI\_Common.h, [40](#)

APC\_SENSOR\_TYPE\_AR0134  
    eSPDI\_Common.h, [22](#)

APC\_SENSOR\_TYPE\_AR0135  
    eSPDI\_Common.h, [22](#)

APC\_SENSOR\_TYPE\_AR0144  
    eSPDI\_Common.h, [22](#)

APC\_SENSOR\_TYPE\_AR0330  
    eSPDI\_Common.h, [22](#)

- APC\_SENSOR\_TYPE\_AR1335
  - eSPDI\_Common.h, [22](#)
- APC\_SENSOR\_TYPE\_H22
  - eSPDI\_Common.h, [22](#)
- APC\_SENSOR\_TYPE\_OV7740
  - eSPDI\_Common.h, [22](#)
- APC\_SENSOR\_TYPE\_OV9282
  - eSPDI\_Common.h, [22](#)
- APC\_SENSOR\_TYPE\_OV9714
  - eSPDI\_Common.h, [22](#)
- APC\_SetCTPropVal
  - eSPDI\_Common.h, [40](#)
- APC\_SetCurrentIRValue
  - eSPDI\_Common.h, [41](#)
- APC\_SetDepthDataType
  - eSPDI\_DM.h, [55](#)
- APC\_SetFWRegister
  - eSPDI\_Common.h, [41](#)
- APC\_SetGPIOCtrl
  - eSPDI\_Common.h, [42](#)
- APC\_SetGPIOValue
  - eSPDI\_Common.h, [42](#)
- APC\_SetHuffmanTableData
  - eSPDI\_Common.h, [42](#)
- APC\_SetHWPostProcess
  - eSPDI\_DM.h, [56](#)
- APC\_SetHWRegister
  - eSPDI\_Common.h, [43](#)
- APC\_SetIRMaxValue
  - eSPDI\_Common.h, [43](#)
- APC\_SetLogData
  - eSPDI\_Common.h, [44](#)
- APC\_SetLogData\_Advanced
  - eSPDI\_Common.h, [44](#)
- APC\_SetPidVid
  - eSPDI\_Common.h, [45](#)
- APC\_SetPUPropVal
  - eSPDI\_Common.h, [45](#)
- APC\_SetQuantizationTableData
  - eSPDI\_Common.h, [46](#)
- APC\_SetSensorTypeName
  - eSPDI\_Common.h, [46](#)
- APC\_SetSlaveHWRegister
  - eSPDI\_Common.h, [48](#)
- APC\_SetSlaveLogData
  - eSPDI\_Common.h, [48](#)
- APC\_SetSlaveYOffset
  - eSPDI\_DM.h, [56](#)
- APC\_SetUserData
  - eSPDI\_Common.h, [49](#)
- APC\_SetYOffset
  - eSPDI\_DM.h, [56](#)
- APC\_SetYOffset\_Advanced
  - eSPDI\_DM.h, [57](#)
- AXES1
  - eSPDI\_Common.h, [22](#)
- DEVINFORMATION
  - eSPDI\_Common.h, [21](#)
- DEVINFORMATIONEX, [9](#)
  - nChipID, [9](#)
  - nDevType, [9](#)
  - strDevName, [9](#)
  - wPID, [10](#)
  - wUsbNode, [10](#)
  - wVID, [10](#)
- eSPCtrl\_RectLogData, [10](#)
  - eSPDI\_Common.h, [21](#)
- eSPDI\_Common.h, [17](#)
  - APC\_CloseDevice, [23](#)
  - APC\_DEVICE\_TYPE, [22](#)
  - APC\_DisableAE, [23](#)
  - APC\_DisableAWB, [24](#)
  - APC\_EnableAE, [24](#)
  - APC\_EnableAWB, [24](#)
  - APC\_EnableGPUAcceleration, [25](#)
  - APC\_FindDevice, [25](#)
  - APC\_GetCTPropVal, [25](#)
  - APC\_GetCurrentIRValue, [26](#)
  - APC\_GetDepthFilterVersion, [27](#)
  - APC\_GetDeviceNumber, [27](#)
  - APC\_GetDeviceResolutionList, [27](#)
  - APC\_GetFlexibleGyroData, [28](#)
  - APC\_GetFlexibleGyroLength, [28](#)
  - APC\_GetFWRegister, [29](#)
  - APC\_GetFwVersion, [29](#)
  - APC\_GetGPIOValue, [30](#)
  - APC\_GetHWRegister, [30](#)
  - APC\_GetIRMaxValue, [31](#)
  - APC\_GetIRMinValue, [31](#)
  - APC\_GetLogData, [32](#)
  - APC\_GetPidVid, [32](#)
  - APC\_GetPUPropVal, [33](#)
  - APC\_GetSensorRegister, [33](#)
  - APC\_GetSlaveHWRegister, [34](#)
  - APC\_GetSlaveLogData, [35](#)
  - APC\_GetSlaveSensorRegister, [35](#)
  - APC\_Init, [36](#)
  - APC\_Init2, [36](#)
  - APC\_Is360Device, [37](#)
  - APC\_OpenDevice, [37](#)
  - APC\_ReadFlashData, [38](#)
  - APC\_RefreshDevice, [38](#)
  - APC\_RegisterDeviceEvents, [39](#)
  - APC\_Release, [39](#)
  - APC\_ResetUNPData, [40](#)
  - APC\_SENSOR\_TYPE\_AR0134, [22](#)
  - APC\_SENSOR\_TYPE\_AR0135, [22](#)
  - APC\_SENSOR\_TYPE\_AR0144, [22](#)
  - APC\_SENSOR\_TYPE\_AR0330, [22](#)
  - APC\_SENSOR\_TYPE\_AR1335, [22](#)
  - APC\_SENSOR\_TYPE\_H22, [22](#)
  - APC\_SENSOR\_TYPE\_OV7740, [22](#)
  - APC\_SENSOR\_TYPE\_OV9282, [22](#)
  - APC\_SENSOR\_TYPE\_OV9714, [22](#)
  - APC\_SetCTPropVal, [40](#)
  - APC\_SetCurrentIRValue, [41](#)

APC\_SetFWRegister, [41](#)  
 APC\_SetGPIOCtrl, [42](#)  
 APC\_SetGPIOValue, [42](#)  
 APC\_SetHuffmanTableData, [42](#)  
 APC\_SetHWRegister, [43](#)  
 APC\_SetIRMaxValue, [43](#)  
 APC\_SetLogData, [44](#)  
 APC\_SetLogData\_Advanced, [44](#)  
 APC\_SetPidVid, [45](#)  
 APC\_SetPUPPropVal, [45](#)  
 APC\_SetQuantizationTableData, [46](#)  
 APC\_SetSensorTypeName, [46](#)  
 APC\_SetSlaveHWRegister, [48](#)  
 APC\_SetSlaveLogData, [48](#)  
 APC\_SetUserData, [49](#)  
 AXES1, [22](#)  
 DEVINFORMATION, [21](#)  
 eSPCtrl\_RectLogData, [21](#)  
 OTHERS, [22](#)  
 PARALUT, [21](#)  
 PUMA, [22](#)  
 SENSOR\_TYPE\_NAME, [22](#)  
 USERDATA\_SECTION\_0, [22](#)  
 USERDATA\_SECTION\_1, [22](#)  
 USERDATA\_SECTION\_10, [23](#)  
 USERDATA\_SECTION\_2, [22](#)  
 USERDATA\_SECTION\_3, [22](#)  
 USERDATA\_SECTION\_4, [22](#)  
 USERDATA\_SECTION\_5, [23](#)  
 USERDATA\_SECTION\_6, [23](#)  
 USERDATA\_SECTION\_7, [23](#)  
 USERDATA\_SECTION\_8, [23](#)  
 USERDATA\_SECTION\_9, [23](#)  
 USERDATA\_SECTION\_INDEX, [22](#)  
 USERDATA\_SECTION\_NUM, [23](#)  
 eSPDI\_DM.h, [49](#)  
   APC\_GetDepthDataType, [51](#)  
   APC\_GetRectifyMatLogData, [51](#)  
   APC\_GetRectifyTable, [52](#)  
   APC\_GetSlaveRectifyTable, [52](#)  
   APC\_GetSlaveYOffset, [53](#)  
   APC\_GetYOffset, [53](#)  
   APC\_GetZDTable, [54](#)  
   APC\_IsInterleaveDevice, [55](#)  
   APC\_SetDepthDataType, [55](#)  
   APC\_SetHWPostProcess, [56](#)  
   APC\_SetSlaveYOffset, [56](#)  
   APC\_SetYOffset, [56](#)  
   APC\_SetYOffset\_Advanced, [57](#)  
 eSPDI\_ErrCode.h, [58](#)  
 nChipID  
   DEVINFORMATIONEX, [9](#)  
   tagDEVINFORMATION, [14](#)  
 nDevType  
   DEVINFORMATIONEX, [9](#)  
   tagDEVINFORMATION, [14](#)  
 OTHERS  
   eSPDI\_Common.h, [22](#)  
 PARALUT  
   eSPDI\_Common.h, [21](#)  
 ParaLUT, [11](#)  
 PUMA  
   eSPDI\_Common.h, [22](#)  
 SENSOR\_TYPE\_NAME  
   eSPDI\_Common.h, [22](#)  
 strDevName  
   DEVINFORMATIONEX, [9](#)  
   tagDEVINFORMATION, [14](#)  
 tagDEVINFORMATION, [13](#)  
   nChipID, [14](#)  
   nDevType, [14](#)  
   strDevName, [14](#)  
   wPID, [14](#)  
   wUsbNode, [15](#)  
   wVID, [15](#)  
 USERDATA\_SECTION\_0  
   eSPDI\_Common.h, [22](#)  
 USERDATA\_SECTION\_1  
   eSPDI\_Common.h, [22](#)  
 USERDATA\_SECTION\_10  
   eSPDI\_Common.h, [23](#)  
 USERDATA\_SECTION\_2  
   eSPDI\_Common.h, [22](#)  
 USERDATA\_SECTION\_3  
   eSPDI\_Common.h, [22](#)  
 USERDATA\_SECTION\_4  
   eSPDI\_Common.h, [22](#)  
 USERDATA\_SECTION\_5  
   eSPDI\_Common.h, [23](#)  
 USERDATA\_SECTION\_6  
   eSPDI\_Common.h, [23](#)  
 USERDATA\_SECTION\_7  
   eSPDI\_Common.h, [23](#)  
 USERDATA\_SECTION\_8  
   eSPDI\_Common.h, [23](#)  
 USERDATA\_SECTION\_9  
   eSPDI\_Common.h, [23](#)  
 USERDATA\_SECTION\_INDEX  
   eSPDI\_Common.h, [22](#)  
 USERDATA\_SECTION\_NUM  
   eSPDI\_Common.h, [23](#)  
 wPID  
   DEVINFORMATIONEX, [10](#)  
   tagDEVINFORMATION, [14](#)  
 wUsbNode  
   DEVINFORMATIONEX, [10](#)  
   tagDEVINFORMATION, [15](#)  
 wVID  
   DEVINFORMATIONEX, [10](#)  
   tagDEVINFORMATION, [15](#)