eYs3D Windows SDK

1.5.8.8

Generated by Doxygen 1.9.1

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:

Chapter 3

File Index

3.1 File List

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

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: new_disp_vaule = disp_value ∗ depth_comp_pars[0] + 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 tye 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 effecitve 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 nMax←Count0, 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 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_GetPUPropVal (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_SetPUPropVal (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

pid	product id of the USB device
vid	vender id of the USB device
bAttached	TRUE if this is a USB device attached event, otherwise, it is a detached event.
pData	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 *img↩
Buf, int imgSize, int width, int height, int serialNumber, LONGLONG timestamp, void *pParam)
```

Callback function when video or data is ready.

Parameters

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

Returns

none

5.1.2.3 DEVINFORMATION

```
typedef struct tagDEVINFORMATION DEVINFORMATION
```

DEVINFORMATION.

device information

5.1.2.4 eSPCtrl_RectLogData

```
typedef struct eSPCtrl_RectLogData eSPCtrl_RectLogData
```

eSPCtrl_RectLogData

Rectified log data structure

5.1.2.5 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
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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	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

pHandleApcDI	CApcDI handler
pDevSelInfo	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

pHandleApcDI	CApcDI handler
pDevSelInfo	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

pHandleApcDI	CApcDI handler
pDevSelInfo	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

pHandleApcDI	CApcDI handler
pDevSelInfo	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
enable	true:enable, fales:diable

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

pHandleApcDI	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_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/ff5678 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.

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

∗pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
nId	specifies the member of the property set
pValue	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
pwType	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
pdevinfo	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

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

Returns

success: APC_OK, others: see eSPDI_ErrCode.h

Parameters

void	∗pHandleApcDI the pointer to the initilized ApcDI SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_ErrCode.h

Parameters

void	∗pHandleApcDI the pointer to the initilized ApcDI SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
const	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

pHandleApcDI	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

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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
length	length of IMU data to read, should be get from APC_GetFlexibleGyroLength
pGyroData	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
GyroLen	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
address	register address
pValue	pointer of value got from register address
flag	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
pszFwVersion	firmware version string
nBufferSize	input buffer length to receive FW version
pActualLength	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
nGPIOIndex	GPIO index, 1 or 2 is valid
pValue	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index

address	register address
pValue	pointer of value got from register address
flag	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
pwType	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
pwType	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store log data
BufferLength	input buffer length
pActualLength	actual length has written to buffer
index	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

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

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

Returns

success: APC_OK, others: see eSPDI_ErrCode.h

5.1.4.26 APC_GetPUPropVal()

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

get processing unit property value `https://msdn.microsoft.com/en-us/library/windows/hardware/ff56812`
`85).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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
nId	specifies the member of the property set
pValue	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
nId	sensor slave address. see SENSOR_TYPE_NAME enum definition
address	register address
pValue	pointer of value got from register address
flag	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
nSensorMode	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
address	register address
pValue	pointer of value got from register address
flag	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store log data
BufferLength	input buffer length
pActualLength	actual length has written to buffer
index	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
nId	sensor slave address. see SENSOR_TYPE_NAME enum definition
address	register address
pValue	pointer of value got from register address
flag	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
nSensorMode	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

ppHandleApcDI	a pointer of pointer to receive ApcDI SDK instance
bIsLogEnabled	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

ppHandleApcDI	a pointer of pointer to receive ApcDI SDK instance
bIsLogEnabled	set to true to generate log file, named log.txt in current folder
bEnableAutoRestart	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 initilize the ApcDI SDK. APC_Init2 adds the auto-restart function to the initilization 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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	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

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

Table 5.41 Image Control Mode

Mode	Description
0x01	color and depth frame output synchrously, 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

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

pHandleApcDI	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

pHandleApcDI	a pointer to ApcDI SDK instance
cbFunc	a callback function of type APC_DeviceEventFn that will receive USB cappure device events when the device is attached or detached.
pData	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

ppHandleApcDI	pointer of the pointer to the initilized 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

void	∗pHandleApcDI CApcDI handler
PDEVSELINFO	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/ff5678` `85).aspx` `https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089(v=vs.↩` `85).aspx`

Parameters

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
nId	specifies the member of the property set
nValue	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
wType	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
address	register address
nValue	register value to set
flag	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

nGPIOIndex	index of GPIO (1 ∼ 4)
nValue	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
nGPIOIndex	GPIO index, 1 or 2 is valid
nValue	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
filename	huffman table file, see jh_vga_422.dat sample file
bLogFile	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
address	register address
nValue	register value to set
flag	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
wType	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
wType	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store log data
BufferLength	input buffer length
pActualLength	actual length has written to buffer
index	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store log data
BufferLength	input buffer length
pActualLength	actual length has written to buffer
index	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

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

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

Returns

success: EtronDI_OK, others: see eSPDI_ErrCode.h

5.1.4.53 APC_SetPUPropVal()

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

get processing unit property value `https://msdn.microsoft.com/en-us/library/windows/hardware/ff5681` `85).aspx` `https://msdn.microsoft.com/en-us/library/windows/hardware/ff566089(v=vs.↩ 85).aspx`

Parameters

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
nId	specifies the member of the property set
nValue	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
filename	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

pHandleApcDI	CApcDI handler
stn	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

pHandleApcDI	CApcDI handler
pDevSelInfo	pointer of device select index
address	register address
nValue	register value to set
flag	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index

buffer	buffer to store log data
BufferLength	input buffer length
pActualLength	actual length has written to buffer
index	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store user data
BufferLength	input buffer length
usi	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.

```

#pragma once

#include <windows.h>
#include <io.h>

#ifndef APC_API
#ifdef __WEYE__
#define APC_API
#else
#ifdef APC_EXPORTS
#define APC_API __declspec(dllexport)
#else
#define APC_API __declspec(dllimport)
#endif
#endif
#endif

#ifndef BYTE
typedef unsigned char BYTE;
```

```
#endif //BYTE

#ifndef WORD
typedef unsigned short WORD;
#endif //WORD

#ifndef WCHAR
typedef wchar_t WCHAR;
#endif //WCHAR

#ifndef BOOL
typedef signed int BOOL;
#endif //BOOL

//
// C++ compatibility
//
#ifdef  __cplusplus
extern "C" {
#endif

#ifndef CALLBACK
#define CALLBACK __stdcall
#endif //CALLBACK


#define APC_MAX_STREAM_COUNT 64
#define APC_MAX_DEPTH_STREAM_COUNT 8
#pragma pack(push, 1)
typedef struct tagAPC_STREAM_INFO {
    int     nWidth;
    int     nHeight;
    BOOL    bFormatMJPG;
} APC_STREAM_INFO, *PAPC_STREAM_INFO;
#pragma pack(pop)

#ifndef WM_MYMSG_NOTICE_CAPTURE
#define WM_MYMSG_NOTICE_CAPTURE (WM_USER+101)
#endif

#include "eSPDI_ErrCode.h"

/* APC Stream Index */
#define APC_Stream_Color 0
#define APC_Stream_Track 4
#define APC_Stream_Kolor 5

/* APC VID */
#define APC_VID_0x1E4E 0x1E4E  // old VID
#define APC_VID_0x3438 0x3438

/* APC PID */
#define APC_PID_8029        0x0568
#define APC_PID_8030        APC_PID_8029
#define APC_PID_8039        APC_PID_8029
#define APC_PID_8031        0x0117
#define APC_PID_8032        0x0118
#define APC_PID_8036        0x0120
#define APC_PID_8037        0x0121
#define APC_PID_8038        0x0124
#define APC_PID_8038_M0     APC_PID_8038
#define APC_PID_8038_M1     0x0147
#define APC_PID_8040W       0x0130
#define APC_PID_8040S       0x0131
#define APC_PID_8040S_K     0x0149
#define APC_PID_8041        0x0126
#define APC_PID_8042        0x0127
#define APC_PID_8043        0x0128
#define APC_PID_8044        0x0129
#define APC_PID_8045K       0x0134
#define APC_PID_8046K       0x0135
#define APC_PID_8051        0x0136
#define APC_PID_8052        0x0137
#define APC_PID_8053        0x0138
#define APC_PID_8054        0x0139
#define APC_PID_8054_K      0x0143
#define APC_PID_8059        0x0146
#define APC_PID_8060        0x0152
#define APC_PID_8060_K      0x0150
#define APC_PID_8060_T      0x0151
#define APC_PID_AMBER       0x0112
#define APC_PID_SALLY       0x0158
#define APC_PID_8062        0x0162
#define APC_PID_8063        0x0164
#define APC_PID_8063_K      0x0165
#define APC_PID_HYPATIA     0x0160  // XY8071
#define APC_PID_HYPATIA2    0x0173
```

```
#define APC_PID_8072        0x0180
#define APC_PID_SANDRA      0x0167
#define APC_PID_NORA        0x0168
#define APC_PID_HELEN       0x0171
#define APC_PID_GRAPE       0x0202
#define APC_PID_IVY         0x0177
#define APC_PID_IVY2        0x0191
#define APC_PID_IVY3        0x0192
#define APC_PID_IVY2_S      0x0195
#define APC_PID_IVY4        0x0198
#define APC_PID_80362       0x0181
#define APC_PID_8077        0x0182
#define APC_PID_8081        0x0183
#define APC_PID_IRIS        0x0184
#define APC_PID_MARY        0x0174
#define APC_PID_FRANK       0x0187
#define APC_PID_STACY       0x0188
#define APC_PID_STACYJUNIOR 0x0189
#define APC_PID_TARYN       0x0199
#define APC_PID_HYPATIA4    0x0204

#define BIT_SET(a,b) ((a) |= (1«(b)))
#define BIT_CLEAR(a,b) ((a) &= ~(1«(b)))
#define BIT_FLIP(a,b) ((a) ^= (1«(b)))
#define BIT_CHECK(a,b) ((a) & (1«(b)))

#define FG_Address_1Byte 0x01
#define FG_Address_2Byte 0x02
#define FG_Value_1Byte   0x10
#define FG_Value_2Byte   0x20

// For Depth Data Type - 2016/12/14 by Sean
#define APC_DEPTH_DATA_DEFAULT                      0
#define APC_DEPTH_DATA_OFF_RAW                      0 // raw (depth off, only raw color)
#define APC_DEPTH_DATA_8_BITS                       1 // rectify
#define APC_DEPTH_DATA_14_BITS                      2 // rectify
#define APC_DEPTH_DATA_8_BITS_x80                   3 // rectify
#define APC_DEPTH_DATA_11_BITS                      4 // rectify
#define APC_DEPTH_DATA_OFF_RECTIFY                  5 // rectify (depth off, only rectify color)
#define APC_DEPTH_DATA_8_BITS_RAW                   6 // raw
#define APC_DEPTH_DATA_14_BITS_RAW                  7 // raw
#define APC_DEPTH_DATA_8_BITS_x80_RAW               8 // raw
#define APC_DEPTH_DATA_11_BITS_RAW                  9 // raw
#define APC_DEPTH_DATA_8_BITS_COMBINED_RECTIFY      10// multi-baseline
#define APC_DEPTH_DATA_14_BITS_COMBINED_RECTIFY     11// multi-baseline
#define APC_DEPTH_DATA_8_BITS_x80_COMBINED_RECTIFY  12// multi-baseline
#define APC_DEPTH_DATA_11_BITS_COMBINED_RECTIFY     13// multi-baseline

// For Inter-Leave-Mode Depth Data Type
#define APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET           16
#define APC_DEPTH_DATA_ILM_DEFAULT                      16
#define APC_DEPTH_DATA_ILM_OFF_RAW                      16 // raw (depth off, only raw color)
#define APC_DEPTH_DATA_ILM_8_BITS                       17 // rectify
#define APC_DEPTH_DATA_ILM_14_BITS                      18 // rectify
#define APC_DEPTH_DATA_ILM_8_BITS_x80                   19 // rectify
#define APC_DEPTH_DATA_ILM_11_BITS                      20 // rectify
#define APC_DEPTH_DATA_ILM_OFF_RECTIFY                  21 // rectify (depth off, only rectify color)
#define APC_DEPTH_DATA_ILM_8_BITS_RAW                   22 // raw
#define APC_DEPTH_DATA_ILM_14_BITS_RAW                  23 // raw
#define APC_DEPTH_DATA_ILM_8_BITS_x80_RAW               24 // raw
#define APC_DEPTH_DATA_ILM_11_BITS_RAW                  25 // raw
#define APC_DEPTH_DATA_ILM_8_BITS_COMBINED_RECTIFY      26// multi-baseline
#define APC_DEPTH_DATA_ILM_14_BITS_COMBINED_RECTIFY     27// multi-baseline
#define APC_DEPTH_DATA_ILM_8_BITS_x80_COMBINED_RECTIFY  28// multi-baseline
#define APC_DEPTH_DATA_ILM_11_BITS_COMBINED_RECTIFY     29// multi-baseline

#define APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET 32
#define APC_DEPTH_DATA_SCALE_DOWN_OFF_RAW               (APC_DEPTH_DATA_OFF_RAW +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET)/* raw (depth off, only raw color) */
#define APC_DEPTH_DATA_SCALE_DOWN_DEFAULT               (APC_DEPTH_DATA_DEFAULT +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET)  /* raw (depth off, only raw color) */
#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 */
#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 */
#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 */
#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 */
#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*/
#define APC_DEPTH_DATA_SCALE_DOWN_8_BITS_RAW            (APC_DEPTH_DATA_8_BITS_RAW +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* raw */
#define APC_DEPTH_DATA_SCALE_DOWN_14_BITS_RAW           (APC_DEPTH_DATA_14_BITS_RAW +
APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* raw */
#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 */
```

```
#define APC_DEPTH_DATA_SCALE_DOWN_11_BITS_RAW           (APC_DEPTH_DATA_11_BITS_RAW + APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /* raw */
#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*/
#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*/

// For Interleave mode depth data type
#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) */
#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) */
#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 */
#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 */
#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 */
#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 */
#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) */
#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 */
#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 */
#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 */
#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 */
#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) //
#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


// For Flash Read/Write
// Firmware (size in KBytes)
#define APC_READ_FLASH_TOTAL_SIZE           128
#define APC_READ_FLASH_FW_PLUGIN_SIZE       104
#define APC_WRITE_FLASH_TOTAL_SIZE          128
#define APC_READ_FLASH_TOTAL_SIZE_256       256
#define APC_WRITE_FLASH_TOTAL_SIZE_256      256

/*
    The group 1 is the factory settings which are calibrated before shipment.
    The group 2 is the factory settings after post calibration.
    FW Register 0xF6 is the offset.
    The default offset is set as 5 which means 10 divided by 2 groups.
*/
#define FW_FID_GROUP_OFFSET                 5
#define MD5_SIGNATURE_BYTE_SIZE             32
#define FW_PROTECT_STRUCT_LEN_OF_STI        17

// PlugIn data (size in bytes)
#define APC_Y_OFFSET_FILE_ID_0              30
#define APC_Y_OFFSET_FILE_SIZE              256
#define APC_RECTIFY_FILE_ID_0               40
#define APC_RECTIFY_FILE_SIZE               1024
#define APC_ZD_TABLE_FILE_ID_0              50
#define APC_ZD_TABLE_FILE_SIZE              4096
#define APC_CALIB_LOG_FILE_ID_0             240
#define APC_CALIB_LOG_FILE_SIZE             4096
#define APC_USER_DATA_FILE_ID_0             200
#define APC_USER_DATA_FILE_SIZE_0           1024
#define APC_USER_DATA_FILE_SIZE_1           4096
#define APC_USER_DATA_FILE_SIZE_2           128
#define APC_USER_DATA_FILE_SIZE_3           1024
#define APC_USER_DATA_FILE_SIZE_4           4096

//===================================================================
//
// Property Type
//
#define PROP_TYPE_PU    0
#define PROP_TYPE_CT    1
//
// PU Property ID
//
#define    PU_PROPERTY_ID_BRIGHTNESS                0
#define    PU_PROPERTY_ID_CONTRAST                  1
#define    PU_PROPERTY_ID_HUE                       2
#define    PU_PROPERTY_ID_SATURATION                3
#define    PU_PROPERTY_ID_SHARPNESS                 4
```

```
#define    PU_PROPERTY_ID_GAMMA                     5
#define    PU_PROPERTY_ID_COLORENABLE               6
#define    PU_PROPERTY_ID_WHITEBALANCE              7
#define    PU_PROPERTY_ID_BACKLIGHT_COMPENSATION    8
#define    PU_PROPERTY_ID_GAIN                      9
#define    PU_PROPERTY_ID_DIGITAL_MULTIPLIER        10
#define    PU_PROPERTY_ID_DIGITAL_MULTIPLIER_LIMIT  11
#define    PU_PROPERTY_ID_WHITEBALANCE_COMPONENT    12
#define    PU_PROPERTY_ID_POWERLINE_FREQUENCY       13
//
// CT Property ID
//
#define    CT_PROPERTY_ID_PAN                            0
#define    CT_PROPERTY_ID_TILT                          1
#define    CT_PROPERTY_ID_ROLL                          2
#define    CT_PROPERTY_ID_ZOOM                          3
#define    CT_PROPERTY_ID_EXPOSURE                      4
#define    CT_PROPERTY_ID_IRIS                          5
#define    CT_PROPERTY_ID_FOCUS                         6
#define    CT_PROPERTY_ID_SCANMODE                      7
#define    CT_PROPERTY_ID_PRIVACY                       8
#define    CT_PROPERTY_ID_PANTILT                       9
#define    CT_PROPERTY_ID_PAN_RELATIVE                  10
#define    CT_PROPERTY_ID_TILT_RELATIVE                 11
#define    CT_PROPERTY_ID_ROLL_RELATIVE                 12
#define    CT_PROPERTY_ID_ZOOM_RELATIVE                 13
#define    CT_PROPERTY_ID_EXPOSURE_RELATIVE             14
#define    CT_PROPERTY_ID_IRIS_RELATIVE                 15
#define    CT_PROPERTY_ID_FOCUS_RELATIVE                16
#define    CT_PROPERTY_ID_PANTILT_RELATIVE              17
#define    CT_PROPERTY_ID_AUTO_EXPOSURE_PRIORITY        19

//=================================================================


typedef struct tagZDTableInfo
{
    int nIndex;
    int nDataType;
} ZDTABLEINFO, *PZDTABLEINFO;

typedef struct eSPCtrl_RectLogData
{
    union {
        BYTE uByteArray[1024];
        struct {
            WORD    InImgWidth;
            WORD    InImgHeight;
            WORD    OutImgWidth;
            WORD    OutImgHeight;
            int     RECT_ScaleEnable;
            int     RECT_CropEnable;
            WORD    RECT_ScaleWidth;
            WORD    RECT_ScaleHeight;
            float   CamMat1[9];
            float   CamDist1[8];
            float   CamMat2[9];
            float   CamDist2[8];
            float   RotaMat[9];
            float   TranMat[3];
            float   LRotaMat[9];
            float   RRotaMat[9];
            float   NewCamMat1[12];
            float   NewCamMat2[12];
            WORD    RECT_Crop_Row_BG;
            WORD    RECT_Crop_Row_ED;
            WORD    RECT_Crop_Col_BG_L;
            WORD    RECT_Crop_Col_ED_L;
            BYTE    RECT_Scale_Col_M;
            BYTE    RECT_Scale_Col_N;
            BYTE    RECT_Scale_Row_M;
            BYTE    RECT_Scale_Row_N;
            float   RECT_AvgErr;
            WORD    nLineBuffers;
            float   ReProjectMat[16];
            float   ParameterRatio[2]; // Ratio for distortion K6
            float   LR_cam_K_temperature[2];
            float   LR_cam_thermal_variation_rate_of_focal[2];
            float depth_comp_pars[2];
            long    Date; // Calibration Date
            char    type; // Calibartion type
            char    version[4]; // Calibration Version
        };
    };
} eSPCtrl_RectLogData;

typedef struct ParaLUT
```

```
{
    long long file_ID_header;
    long long file_ID_version;
    double    FOV;
    long long semi_FOV_pixels;
    long long img_src_cols;
    long long img_src_rows;
    double    img_L_src_col_center;
    double    img_L_src_row_center;
    double    img_R_src_col_center;
    double    img_R_src_row_center;
    double    img_L_rotation;
    double    img_R_rotation;
    double    spline_control_v1;
    double    spline_control_v2;
    double    spline_control_v3;
    double    spline_control_v4;
    double    spline_control_v5;
    double    spline_control_v6;
    double    spline_control_v7;
    long long img_dst_cols;
    long long img_dst_rows;
    long long img_L_dst_shift;
    long long img_R_dst_shift;
    long long img_overlay_LR;
    long long img_overlay_RL;
    long long img_stream_cols;
    long long img_stream_rows;
    long long video_stream_cols;
    long long video_stream_rows;
    long long usb_type;
    long long img_type;
    long long lut_type;
    long long blending_type;
    double    overlay_ratio;

    long long serial_number_date0;
    long long serial_number_date1;

    double    unit_sphere_radius;
    double    min_col;
    double    max_col;
    double    min_row;
    double    max_row;

    long long AGD_LR;
    long long AGD_RL;

    long long out_img_resolution;
    long long out_lut_cols;
    long long out_lut_rows;
    long long out_lut_cols_eff;
    long long out_lut_rows_eff;
    long long out_img_cols;
    long long out_img_rows;
    long long out_overlay_LR;
    long long out_overlay_RL;
    long long reserve[44];
    BYTE      serial_number[256];
} PARALUT, *PPARALUT;

struct APCImageType
{
    enum Value
    {
        IMAGE_UNKNOWN = -1,
        COLOR_YUY2 = 0,
        COLOR_Y12Bits,
        COLOR_RGB24,
        COLOR_MJPG,
        DEPTH_8BITS = 100,
        DEPTH_8BITS_0x80,
        DEPTH_11BITS,
        DEPTH_14BITS
    };

    static bool IsImageColor(APCImageType::Value type)
    {
        return (type == COLOR_YUY2 || type == COLOR_RGB24 || type == COLOR_MJPG || type ==
COLOR_Y12Bits);
    }

    static bool IsImageDepth(APCImageType::Value type)
    {
        return (type != IMAGE_UNKNOWN && !IsImageColor(type));
    }

    static APCImageType::Value DepthDataTypeToDepthImageType(WORD dataType)
    {
        switch (dataType)
        {
        case APC_DEPTH_DATA_8_BITS:
        case APC_DEPTH_DATA_8_BITS_RAW:
        case APC_DEPTH_DATA_ILM_8_BITS:
        case APC_DEPTH_DATA_ILM_8_BITS_RAW:
        case APC_DEPTH_DATA_SCALE_DOWN_8_BITS:
        case APC_DEPTH_DATA_SCALE_DOWN_8_BITS_RAW:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_RAW:
            return APCImageType::DEPTH_8BITS;
        case APC_DEPTH_DATA_8_BITS_x80:
        case APC_DEPTH_DATA_8_BITS_x80_RAW:
        case APC_DEPTH_DATA_ILM_8_BITS_x80:
        case APC_DEPTH_DATA_ILM_8_BITS_x80_RAW:
        case APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80:
        case APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80_RAW:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_x80:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_x80_RAW:
            return APCImageType::DEPTH_8BITS_0x80;
        case APC_DEPTH_DATA_11_BITS:
        case APC_DEPTH_DATA_11_BITS_RAW:
        case APC_DEPTH_DATA_11_BITS_COMBINED_RECTIFY:
        case APC_DEPTH_DATA_ILM_11_BITS:
        case APC_DEPTH_DATA_ILM_11_BITS_RAW:
        case APC_DEPTH_DATA_ILM_11_BITS_COMBINED_RECTIFY:
        case APC_DEPTH_DATA_SCALE_DOWN_11_BITS:
        case APC_DEPTH_DATA_SCALE_DOWN_11_BITS_RAW:
        case APC_DEPTH_DATA_SCALE_DOWN_11_BITS_COMBINED_RECTIFY:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS_RAW:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS_COMBINED_RECTIFY:
            return APCImageType::DEPTH_11BITS;
        case APC_DEPTH_DATA_14_BITS:
        case APC_DEPTH_DATA_14_BITS_RAW:
        case APC_DEPTH_DATA_14_BITS_COMBINED_RECTIFY:
        case APC_DEPTH_DATA_ILM_14_BITS:
        case APC_DEPTH_DATA_ILM_14_BITS_RAW:
        case APC_DEPTH_DATA_ILM_14_BITS_COMBINED_RECTIFY:
        case APC_DEPTH_DATA_SCALE_DOWN_14_BITS:
        case APC_DEPTH_DATA_SCALE_DOWN_14_BITS_RAW:
        case APC_DEPTH_DATA_SCALE_DOWN_14_BITS_COMBINED_RECTIFY:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS_RAW:
        case APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS_COMBINED_RECTIFY:
            return APCImageType::DEPTH_14BITS;
        default: return APCImageType::IMAGE_UNKNOWN;
        }
    }

    static bool IsSupportedDisparityCompensateDevice(int pid, WORD dataType) {
        return (pid == APC_PID_HYPATIA2 || pid == APC_PID_8072) &&
(APCImageType::DepthDataTypeToDepthImageType(dataType) == APCImageType::DEPTH_11BITS);
    }
};

struct ApcDIDepthSwitch
{
    enum Value
    {
        Depth0 = 0x01,
        Depth1 = 0x02,
        Depth2 = 0x04
    };

    static bool IsOn(ApcDIDepthSwitch::Value target, int setting)
    {
        return ((target & setting) != 0);
    }
};
typedef void(*APC_ImgCallbackFn)(APCImageType::Value imgType, int imgId, unsigned char* imgBuf, int
imgSize,
    int width, int height, int serialNumber, LONGLONG timestamp, void* pParam);

typedef enum {
    OTHERS = 0,
    AXES1,
    PUMA,
    PLUM,
    GRAPE_FPGA

}APC_DEVICE_TYPE;

struct APC_SensorMode
{
    enum Value
    {
        Sensor1 = 0,
        Sensor2 = 1,
        SensorAll = 2,
        Sensor3 = 3,
        Sensor4 = 4
    };
};
typedef struct tagDEVINFORMATION {
    unsigned short wPID;
    unsigned short wVID;
    char *strDevName;
    char *strDevPath;
    unsigned short  nChipID;
    unsigned short  nDevType;
    unsigned short wUsbNode;
} DEVINFORMATION;


class DEVINFORMATIONEX
{
public:
    DEVINFORMATIONEX()
    {
        wPID = wVID = nChipID = 0;
        nDevType = OTHERS;
        strDevName[0] = '\0';
        strDevPath[0] = '\0';
        wUsbNode = -1;
    }

    DEVINFORMATIONEX& operator=(const DEVINFORMATIONEX& rhs)
    {
        wPID = rhs.wPID;
        wVID = rhs.wVID;
        strcpy_s(strDevName, rhs.strDevName);
        strcpy_s(strDevPath, rhs.strDevPath);
        nChipID = rhs.nChipID;
        nDevType = rhs.nDevType;
        wUsbNode = rhs.wUsbNode;

        return *this;
    }

    DEVINFORMATIONEX& operator=(const DEVINFORMATION& rhs)
    {
        wPID = rhs.wPID;
        wVID = rhs.wVID;
        strcpy_s(strDevName, rhs.strDevName);
        strcpy_s(strDevPath, rhs.strDevPath);
        nChipID = rhs.nChipID;
        nDevType = rhs.nDevType;
        wUsbNode = rhs.wUsbNode;

        return *this;
    }

    DEVINFORMATIONEX(const DEVINFORMATIONEX& rhs)
    {
        *this = rhs;
    }

    unsigned short wPID;
    unsigned short wVID;
    char strDevName[512];
    char strDevPath[512];
    unsigned short nChipID;
    unsigned short nDevType;
    unsigned short wUsbNode;
};
typedef struct tagDEVSEL
{
  int index;
} DEVSELINFO, *PDEVSELINFO;

typedef enum
{
  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
} USERDATA_SECTION_INDEX;

// for total and fw+plugin read/write +
typedef enum
{
    Total = 0,
    FW_PLUGIN,
    Total_Slave,        /* total for first slave device */
    FW_PLUGIN_Slave,    /* fw_plugin for first slave device */
    UNP                 /* UNProtection Area */
} FLASH_DATA_TYPE;

typedef enum
{
    USB_PORT_TYPE_2_0 = 2,
    USB_PORT_TYPE_3_0,
    USB_PORT_TYPE_UNKNOW
} USB_PORT_TYPE;

typedef struct tagKEEP_DATA_CTRL {
    bool  bIsSerialNumberKeep;
    bool  bIsSensorPositionKeep;
    bool  bIsRectificationTableKeep;
    bool  bIsZDTableKeep;
    bool  bIsCalibrationLogKeep;
} KEEP_DATA_CTRL;
// for total and fw+plugin read/write -

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

int  APC_API APC_Init2( void **ppHandleApcDI, bool bIsLogEnabled, bool bAutoRestart);

int  APC_API APC_Init3( void **ppHandleApcDI, bool bIsLogEnabled, bool bEnableAutoRestart, bool bMonitorUSBEvent);

#ifndef APC_DeviceEventFn_
typedef void(*APC_DeviceEventFn)(UINT pid, UINT vid, BOOL bAttached, void* pData);
#define APC_DeviceEventFn_
#endif

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

void APC_API APC_Release( void **ppHandleApcDI);

int  APC_API APC_FindDevice( void *pHandleApcDI);

int  APC_API APC_RefreshDevice( void *pHandleApcDI);

int  APC_API APC_GetDeviceNumber( void *pHandleApcDI);

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

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

// register APIs +
int APC_API APC_GetSlaveSensorRegister(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, unsigned short address, unsigned short *pValue, int flag, int nSensorMode);

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

int APC_API APC_SetSlaveSensorRegister(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, unsigned short address, unsigned short nValue, int flag, int nSensorMode);

int APC_API APC_SetSensorRegister ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, unsigned short address, unsigned short nValue,  int flag, int nSensorMode);

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

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

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

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

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

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

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

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

// register APIs -

// File ID +
int APC_API APC_GetFwVersion    ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, char *pszFwVersion, int nBufferSize, int *pActualLength);

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

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

int APC_API APC_GetSerialNumber( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *pSerialNum, int nBufferSize, int *pACtualSNLenByByte);

int APC_API APC_SetSerialNumber(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *pSerialNum, int nBufferSize);

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

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

int APC_API APC_GetUserData     ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, USERDATA_SECTION_INDEX usi);

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

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

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

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


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


int APC_API APC_WriteFlashData  ( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, FLASH_DATA_TYPE fdt, BYTE *pBuffer,
                                          unsigned long int nLengthOfBuffer, BOOL bIsDataVerify, KEEP_DATA_CTRL kdc);

int APC_API APC_GetStructLenOfSTI(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);

int APC_API APC_GetUnpAreaStartSec(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);

int APC_API APC_IsBPX4bitsClear(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);

int APC_API APC_UnprotectFlash(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);

int APC_API APC_UnprotectFlashByCipher(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, const char* cipher);

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 );

int APC_API APC_GetColorImage(void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
    BYTE *pBuf, unsigned long int *pImageSize, int *pSerial = NULL);

int APC_API APC_GetDepthImage(void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
    BYTE *pBuf, unsigned long int *pImageSize, int *pSerial = NULL, int nDepthDataType = 0);

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

int APC_API APC_GetDeviceResolutionListEx( void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
                                                  int nMaxCount0, APC_STREAM_INFO *pStreamInfo0,
                                                        int nMaxCount1, APC_STREAM_INFO *pStreamInfo1,
                                                        int pid);

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

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

int APC_API APC_GetSerialNumberFromLog( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, char *pSerialNum,
int nBufferSize, int *pActualLength);

// IR support
int APC_API APC_SetCurrentIRValue(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType);

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

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

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

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

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

int APC_API APC_SetIRMode(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType);

int APC_API APC_GetIRMode(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD *pwType);

// for sensorif
int APC_API APC_EnableSensorIF( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool bIsEnable);


//int APC_API APC_GetMotorCurrentState( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool*
bIsRunning);

// for Gyro

typedef enum
{
    DPS_245 = 0,
    DPS_500,
    DPS_2000
} SENSITIVITY_LEVEL_L3G;

// bPowerMode :
//      true  : Normal
//      false : Power Down
// bIsZEnable :
//      true  : Enable
//      false : Disable
// bIsYEnable :
//      true  : Enable
//      false : Disable
// bIsXEnable :
//      true  : Enable
//      false : Disable

//typedef struct GyroTag
//{
//   short x;
//   short y;
//   short z;
//} GYRO_ANGULAR_RATE_DATA;


typedef enum
{
    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
} SENSOR_TYPE_NAME;

```

```
//
// Sensor Mode: Left, Right, or Both
//
#define ESPAEAWB_SENSOR_MODE_LEFT       0
#define ESPAEAWB_SENSOR_MODE_RIGHT      1
#define ESPAEAWB_SENSOR_MODE_BOTH       2


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

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

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

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

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


int APC_API APC_GetExposureTime( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid, float *pfExpTimeMS);

int APC_API APC_SetExposureTime( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid, float fExpTimeMS);

int APC_API APC_GetGlobalGain( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid, float *pfGlobalGain);

int APC_API APC_SetGlobalGain( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid, float fGlobalGain);

int APC_API APC_SetAnalogGain(void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid, float fGlobalGain);

int APC_API APC_GetAnalogGain(void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid, float *pfGlobalGain);

int APC_API APC_SetDigitalGain(void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid, float fGlobalGain);

int APC_API APC_GetDigitalGain(void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, int pid, float *pfGlobalGain);

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

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

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

int APC_API APC_GetAccMeterValue( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int *pX, int *pY, int *pZ);

int APC_API APC_GetPUPropVal( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int *pValue);

int APC_API APC_SetPUPropVal( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int nId, int nValue);

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

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

// For AEAWB - 2015/01/28 by Wolf

int APC_API APC_EncryptMP4(void* pHandleApcDI, PDEVSELINFO pDevSelInfo, const char* filename);
int APC_API APC_DecryptMP4(void* pHandleApcDI, PDEVSELINFO pDevSelInfo, const char* filename);
int APC_API APC_RetrieveMp4ExtraData(void* pHandleApcDI, PDEVSELINFO pDevSelInfo,
    const char* filename, char* dataBuf, int* dataSize);
int APC_API APC_FlushMp4ExtraData(void* pHandleApcDI, PDEVSELINFO pDevSelInfo,
    const char* filename, const char* dataBuf, int dataSize);
int APC_API APC_GetAutoExposureMode(void* pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short* mode);

int APC_API APC_SetAutoExposureMode(void* pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned short mode);

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

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

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

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

int APC_API APC_SetPlumAR0330(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool bEnable);

int APC_API APC_SetRootCipher(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, const char* cipher);

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

int APC_API APC_GetDevicePortType(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, USB_PORT_TYPE*
pUSB_Port_Type);

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);

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);

int APC_API APC_TemporalFilter(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char* depthBuf,
int bytesPerPixel, int width, int height, float alpha, int history);

int APC_API APC_EdgePreServingFilter(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char*
depthBuf, int type, int width, int height, int level, float sigma, float lumda);

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);

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

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

int APC_API APC_ResetFilters(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);

int APC_API APC_TableToData(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int width, int height, int
TableSize, unsigned short* Table, unsigned short* Src, unsigned short* Dst);

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 );

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);

int APC_API APC_PropertyPU_GetRange(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty, long
* pMin, long * pMax, long * pStep, long * pDefault, long * pCapsFlag, int pid);
int APC_API APC_PropertyCT_GetRange(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty, long
* pMin, long * pMax, long * pStep, long * pDefault, long * pCapsFlag, int pid);
int APC_API APC_PropertyPU_GetCurrent(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty,
long *pCur, long *pCur2, long *pCapsFlag, int pid);
int APC_API APC_PropertyCT_GetCurrent(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty,
long *pCur, long *pCur2, long *pCapsFlag, int pid);
int APC_API APC_PropertyPU_SetCurrent(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty,
long nCur, long nCur2, long nCapsFlag, int pid);
int APC_API APC_PropertyCT_SetCurrent(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, long nProperty,
long nCur, long nCur2, long nCapsFlag, int pid);
int APC_API APC_PropertyItem_Write(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, REFGUID guid, int
nPropertyItem, LONG nValue, int pid);
int APC_API APC_PropertyItem_Read(void * pHandleApcDI, PDEVSELINFO pDevSelInfo, REFGUID guid, int
nPropertyItem, LONG *pValue, int pid);

int APC_API APC_ShowPropertyPage( void * pHandleApcDI, PDEVSELINFO pDevSelInfo, int pid );

#ifdef __cplusplus
}
#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 Buffer←Length, 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 Buffer←Length, 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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
pwType	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
nID	actual file id
lpData	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
iFWFSIndex	actual file id
iFWFSLength	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
enable	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

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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store rectify table data
BufferLength	input buffer length
pActualLength	actual length has written to buffer
index	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store rectify table data
BufferLength	input buffer length
pActualLength	actual length has written to buffer
index	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store
BufferLength	length of buffer
pActualLength	actual byte of reading
index	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store
BufferLength	length of buffer
pActualLength	actual byte of reading
index	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

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

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

Returns

success: APC_OK, others: see eSPDI_ErrCode.h

Parameters

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
wType	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
enable	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store
BufferLength	length of buffer
pActualLength	actual byte of reading
index	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

pHandleApcDI	the pointer to the initilized ApcDI SDK instance
pDevSelInfo	pointer of device select index
buffer	buffer to store
BufferLength	length of buffer
pActualLength	actual byte of reading
index	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

void	∗pHandleApcDI the pointer to the initilized ApcDI SDK instance
PDEVSELINFO	pDevSelInfo CApcDI handler
BYTE	∗buffer buffer to store
int	BufferLength length of buffer
int	∗pActualLength actual byte of reading
int	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.

```
#pragma once
#include "eSPDI_Common.h"

// for APC_PostSetParam/APC_PostGetParam
#define POSTPAR_HR_MODE         5
#define POSTPAR_HR_CURVE_0      6
#define POSTPAR_HR_CURVE_1      7
#define POSTPAR_HR_CURVE_2      8
#define POSTPAR_HR_CURVE_3      9
#define POSTPAR_HR_CURVE_4     10
#define POSTPAR_HR_CURVE_5     11
#define POSTPAR_HR_CURVE_6     12
#define POSTPAR_HR_CURVE_7     13
#define POSTPAR_HR_CURVE_8     14
#define POSTPAR_HF_MODE        17
#define POSTPAR_DC_MODE        20
#define POSTPAR_DC_CNT_THD     21
#define POSTPAR_DC_GRAD_THD    22
#define POSTPAR_SEG_MODE       23
#define POSTPAR_SEG_THD_SUB    24
#define POSTPAR_SEG_THD_SLP    25
#define POSTPAR_SEG_THD_MAX    26
#define POSTPAR_SEG_THD_MIN    27
#define POSTPAR_SEG_FILL_MODE 28
#define POSTPAR_HF2_MODE       31
#define POSTPAR_GRAD_MODE      34
#define POSTPAR_TEMP0_MODE     37
#define POSTPAR_TEMP0_THD      38
#define POSTPAR_TEMP1_MODE     41
#define POSTPAR_TEMP1_LEVEL    42
#define POSTPAR_TEMP1_THD      43
#define POSTPAR_FC_MODE        46
#define POSTPAR_FC_EDGE_THD    47
#define POSTPAR_FC_AREA_THD    48
#define POSTPAR_MF_MODE        51
#define POSTPAR_ZM_MODE        52
```

```
#define POSTPAR_RF_MODE       53
#define POSTPAR_RF_LEVEL      54

//
// C++ compatibility
//
#ifdef  __cplusplus
extern "C" {
#endif

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

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

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

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

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

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

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

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

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

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

int APC_API APC_SetSlaveRectifyTable(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);

int APC_API APC_SetRectifyTable(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int
BufferLength, int *pActualLength, int index);

int APC_API APC_SetRectifyTable_Advanced(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, BYTE *buffer,
int BufferLength, int *pActualLength, int index);

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

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

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

int APC_API APC_GetRectifyMatLogDataSlave(void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
eSPCtrl_RectLogData *pData, int index);

int APC_API APC_SetDepthDataTypeEx(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, WORD wType, int pid);

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

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

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

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

int APC_API APC_EnableInterleave(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool enable);

bool APC_API APC_IsInterleaveDevice(void *pHandleApcDI, PDEVSELINFO pDevSelInfo);

int APC_API APC_EnableSerialCount( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, bool enable );

struct PointCloudInfo
{
//normal data
    float centerX;
    float centerY;
    float focalLength;
    float disparityToW[ 2048 ];
    int   disparity_len;
    WORD  wDepthType;
    //float centerXP;
    //float centerYP;
    //float focalXP;
    //float focalYP;
    //float baseline;
    float fx1;
    float fy1;
    float fx2;
    float fy2;
    float cx1;
    float cy1;
    float cx2;
    float cy2;
    float Tx;

//multi-lens data
    float focalLength_K;
    float baseline_K;
    float diff_K;
//slave data
    float   CamMat2[9];
    float   RotaMat[9];
    float   TranMat[3];

    PointCloudInfo() { memset( this, NULL, sizeof( PointCloudInfo ) ); }
};

int APC_API APC_GetPointCloud( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char* ImgColor, int CW, int CH,
                                                                                           unsigned char* ImgDepth, int DW, int DH,
                                                                                           PointCloudInfo* pPointCloudInfo,
                                                                                           unsigned char* pPointCloudRGB, float* pPointCloudXYZ, float Near, float Far );
int APC_API APC_FlyingDepthCancellation_D8( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char* pdepthD8, int width, int height );

int APC_API APC_FlyingDepthCancellation_D11( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char* pdepthD11, int width, int height );



int APC_API APC_DepthMerge( void *pHandleApcDI, PDEVSELINFO pDevSelInfo, unsigned char** pDepthBufList, float *pDepthMergeOut,
    unsigned char *pDepthMergeFlag, int nDWidth, int nDHeight, float fFocus, float * pBaseline, float * pWRNear, float * pWRFar, float * pWRFusion, int nMergeNum );

int APC_API APC_AdjustFocalLengthFromFlash(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int width, int height);

int APC_API APC_AdjustFocalLength(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int width, int height, int pixelUnit);

int APC_API APC_GetDeviceFocalLength(void *pHandleApcDI, PDEVSELINFO pDevSelInfo,
    int *pLeftFx, int *pLeftFy, int *pRightFx, int *pRightFy);

int APC_API APC_GetFlashFocalLength(void *pHandleApcDI, PDEVSELINFO pDevSelInfo, int width, int height,
    int *pLeftFx, int *pLeftFy, int *pRightFx, int *pRightFy, int *pPixelUnit);

#ifdef __cplusplus
}
#endif
```

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.

```


//define Error Code by Wolf 2013/08/30
#define  APC_OK                              0
#define  APC_NoDevice                       -1
#define  APC_NullPtr                        -2
#define  APC_ErrBufLen                      -3
#define  APC_Init_Fail                      -4
#define  APC_NoZDTable                      -5
#define  APC_READFLASHFAIL                  -6
#define  APC_WRITEFLASHFAIL                 -7
#define  APC_VERIFY_DATA_FAIL               -8
#define  APC_KEEP_DATA_FAIL                 -9
#define  APC_RECT_DATA_LEN_FAIL            -10
#define  APC_RECT_DATA_PARSING_FAIL        -11
#define  APC_RET_BAD_PARAM                 -12
#define  APC_RET_OPEN_FILE_FAIL            -13
#define  APC_NO_CALIBRATION_LOG            -14
#define  APC_POSTPROCESS_INIT_FAIL         -15
#define  APC_POSTPROCESS_NOT_INIT          -16
#define  APC_POSTPROCESS_FRAME_FAIL        -17
#define  APC_NotSupport                    -18
#define  APC_OpenFileFail                  -19
#define  APC_READ_REG_FAIL                 -20
#define  APC_WRITE_REG_FAIL                -21
#define  APC_SET_FPS_FAIL                  -22
#define  APC_VIDEO_RENDER_FAIL             -23
#define  APC_OPEN_DEVICE_FAIL              -24
#define  APC_FIND_DEVICE_FAIL              -25
#define  APC_GET_IMAGE_FAIL                -26
#define  APC_NOT_SUPPORT_RES               -27
#define  APC_CALLBACK_REGISTER_FAIL        -28
#define  APC_DEVICE_NOT_SUPPORT            -29

// for 3D Scanner +
#define  APC_ILLEGAL_ANGLE                 -30
#define  APC_ILLEGAL_STEP                  -31
#define  APC_ILLEGAL_TIMEPERSTEP           -32
#define  APC_MOTOR_RUNNING                 -33
#define  APC_GETSENSORREG_FAIL             -34
#define  APC_SETSENSORREG_FAIL             -35
#define  APC_READ_X_AXIS_FAIL              -36
#define  APC_READ_Y_AXIS_FAIL              -37
#define  APC_READ_Z_AXIS_FAIL              -38
#define  APC_READ_PRESS_DATA_FAIL          -39
#define  APC_READ_TEMPERATURE_FAIL         -40
#define  APC_RETURNHOME_RUNNING            -41
#define  APC_MOTOTSTOP_BY_HOME_INDEX       -42
#define  APC_MOTOTSTOP_BY_PROTECT_SCHEME   -43
#define  APC_MOTOTSTOP_BY_NORMAL           -44
#define  APC_ILLEGAL_FIRMWARE_VERSION      -45
#define  APC_ILLEGAL_STEPPERTIME           -46
// for 3D Scanner -

// For AEAWB + 2015/01/28 by Wolf
#define  APC_GET_PU_PROP_VAL               -50
#define  APC_SET_PU_PROP_VAL               -51
#define  APC_GET_CT_PROP_VAL               -52
#define  APC_SET_CT_PROP_VAL               -53
// For AEAWB - 2015/01/28 by Wolf

// for Dewarping + Stitching +
#define  APC_INVALID_USERDATA              -70
#define  APC_MAP_LUT_FAIL                  -71
#define  APC_APPEND_TO_FILE_FRONT_FAIL     -72
// for Dewarping + Stitching -

#define APC_TOO_MANY_DEVICE                -80
#define APC_ACCESS_MP4_EXTRA_DATA_FAIL     -81
```

Index