eYs3dSDK_Linux

4.0.1

Generated by Doxygen 1.8.13

Contents

Chapter 1

Introduction

This document describes the usage of Application Programming Interfaces of HD-DM-SDK

What's inside the SDK

Table 1.1 File List

Folder	Filename	Description
bin	All files	Sample executables on Linux x86_64 platform
eSPDI	eSPDI.h	Basic API declaration header
	eSPDI_def.h	Basic data struct declaration header
	eSPDI_version.h	SDK version declaration header
	libeSPDI_X86_64_4.0.0_pp.so	eSPDI shared library for Linux x86_64 platform
eSPDI/opencv	All files	OpenCV library
eSPDI/turbojpeg	All files	TurboJepg library
eSPDI/alsa-release	All files	Alsa library
eSPDI/DepthFilter	All files	Depth filter library
doc/html	index.html	This documentation
DMPreview		A sample project demonstrating how to open multiple devices in an application

Chapter 2

File Index

2.1 File List

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

eSPDI_source/eSPDI.h
: Etron SDK API export functions, data structure and variable definition Copyright: This file copyright (C) 2017 by . . . . . **??**

eSPDI_source/eSPDI_def.h
: Main data structure, variable and macro definition and error definition Copyright: This file copyright (C) 2017 by . . . . . **??**

Chapter 3

File Documentation

3.1 eSPDI_source/eSPDI.h File Reference

Etron SDK API export functions, data structure and variable definition Copyright: This file copyright (C) 2017 by.

Functions

- int EtronDI_Init (void **ppHandleEtronDI, bool bIsLogEnabled)

  entry point of Etron camera SDK including 1.create a CEtronDI class for accessing oncming APIs 2.find out Etron devices 3.create a CVideoDevice class for video streaming and hardware access
- int EtronDI_FindDevice (void *pHandleEtronDI)

  find out all Etron USB devices by PID, VID and ChipID, also remember device types
- void EtronDI_Release (void **ppHandleEtronDI)

  release resource that EtronDI_Init had allocated
- int EtronDI_RefreshDevice (void *pHandleEtronDI)

  refresh all Etron UVC devices
- int EtronDI_SwitchBaseline (int index)

  Swich the baseline index.
- bool EtronDI_IsMLBaseLine (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  Check the device is multiple baseline device.
- int EtronDI_DoFusion (unsigned char **pDepthBufList, double *pDepthMerge, unsigned char *pDepth↩ MergeFlag, int nDWidth, int nDHeight, double fFocus, double *pBaseline, double *pWRNear, double *pW↩ RFar, double *pWRFusion, int nMergeNum, bool bdepth2Byte11bit, int method)

  Do Fusion Merge.
- int EtronDI_GetDeviceNumber (void *pHandleEtronDI)

  get Etron USB device numbers
- int EtronDI_GetDeviceInfo (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, DEVINFORMATION *pdevinfo)

  get informations of Etron UVC devices, see DEVINFORMATION
- int EtronDI_GetDeviceInfoMBL_15cm (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, DEVINFORM↩ ATION *pdevinfo)

  get informations of Etron UVC devices, see DEVINFORMATION
- int EtronDI_SelectDevice (void *pHandleEtronDI, int dev_index)

  do not support currently
- bool EtronDI_IsInterleaveDevice (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

check module support interleave function or not

- int EtronDI_EnableInterleave (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, bool enable)

  enable or disable interleave function

- int EtronDI_SetControlCounterMode (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned char n↩ Value)

  enable or disable interleave function

- int EtronDI_GetControlCounterMode (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned char *nValue)

  enable or disable interleave function

- int EtronDI_GetSensorRegister (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nId, unsigned short address, unsigned short *pValue, int flag, SENSORMODE_INFO SensorMode)
- int EtronDI_SetSensorRegister (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nId, unsigned short address, unsigned short nValue, int flag, SENSORMODE_INFO SensorMode)

  set sensor register value

- int EtronDI_GetFWRegister (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short *pValue, int flag)

  get firmware register value

- int EtronDI_SetFWRegister (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short nValue, int flag)

  set firmware register value

- int EtronDI_GetHWRegister (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short *pValue, int flag)

  get hardware register value

- int EtronDI_SetHWRegister (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short nValue, int flag)

  set hardware register

- int EtronDI_GetFwVersion (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, char *pszFwVersion, int n↩ BufferSize, int *pActualLength)

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

- int EtronDI_GetPidVid (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short *pPidBuf, unsigned short *pVidBuf)

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

- int EtronDI_SetPidVid (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short *pPidBuf, unsigned short *pVidBuf)

  set PID and VID to device

- int EtronDI_GetSerialNumber (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned char *pData, int nbufferSize, int *pLen)

  get device serial number

- int EtronDI_SetSerialNumber (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned char *pData, int nLen)

  set serial number to device

- int EtronDI_GetYOffset (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)

  get Y offset (file ID 30+) value

- int EtronDI_GetRectifyTable (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int Buffer↩ Length, int *pActualLength, int index)

  get rectify values (file ID 40+) from flash

- int EtronDI_GetZDTable (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int Buffer↩ Length, int *pActualLength, PZDTABLEINFO pZDTableInfo)

  get disparity and Z values from flash

- int EtronDI_GetLogData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int Buffer↩ Length, int *pActualLength, int index, CALIBRATION_LOG_TYPE type)

  get log data from flash

- int EtronDI_GetUserData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, USERDATA_SECTION_INDEX usi)

  get user data from flash
- int EtronDI_SetYOffset (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)

  set Y offset values
- int EtronDI_SetRectifyTable (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)

  set rectify values to flash
- int EtronDI_SetZDTable (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, PZDTABLEINFO pZDTableInfo)

  set disparity and Z values to flash
- int EtronDI_SetLogData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)

  set log data to flash
- int EtronDI_SetUserData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, USERDATA_SECTION_INDEX usi)

  set user data to flash
- int EtronDI_ReadFlashData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, FLASH_DATA_TYPE fdt, BYTE *pBuffer, unsigned long int BufferLength, unsigned long int *pActualLength)

  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 EtronDI_WriteFlashData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, FLASH_DATA_TYPE fdt, BYTE *pBuffer, unsigned long int BufferLength, bool bIsDataVerify, KEEP_DATA_CTRL kdc)

  write firmware code(.bin) to flash The firmware code is the combination of boot loader, firmware body and plug-in data, also can keep original functions(Serial Number, Sensor Position, RectificationTable, ZD Table and CalibrationLog) on camera flash by KEEP_DATA_CTRL control
- int EtronDI_GetDevicePortType (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, USB_PORT_TYPE *pUSB_Port_Type)

  Get Device USB-port-type.
- int EtronDI_GetDeviceResolutionList (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nMaxCount, ETRONDI_STREAM_INFO *pStreamInfo0, int nMaxCvoidount1, ETRONDI_STREAM_INFO *pStreamInfo1)

  get the device resolution list
- int EtronDI_Setup_v4l2_requestbuffers (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int cnt)

  Setup v4l2 request buffers, default = 4.
- int EtronDI_OpenDevice (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nEP0Width, int nEP0Height, bool bEP0MJPG, int nEP1Width, int nEP1Height, DEPTH_TRANSFER_CTRL dtc=DEPTH_IMG_NON_TRANSFER, bool bIsOutputRGB24=false, void *phWndNotice=0, int *pFPS=0, CONTROL_MODE cm=IMAGE_SN_NONSYNC)

  the implement layer to open Etron camera device by V4L2(https://en.wikipedia.org/wiki/Video4Linux), can open color and depth at one time call, do functions as below,
- int EtronDI_OpenDevice2 (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nEP0Width, int nEP0Height, bool bEP0MJPG, int nEP1Width, int nEP1Height, DEPTH_TRANSFER_CTRL dtc=DEPTH_IMG_NON_TRANSFER, bool bIsOutputRGB24=false, void *phWndNotice=0, int *pFPS=0, CONTROL_MODE cm=IMAGE_SN_NONSYNC)

  the implement layer to open Etron camera device by V4L2(https://en.wikipedia.org/wiki/Video4Linux), can open color and depth at one time call, do functions as below,
- int EtronDI_OpenDeviceMBL (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nEP0Width, int nEP0Height, bool bEP0MJPG, int nEP1Width, int nEP1Height, DEPTH_TRANSFER_CTRL dtc=DEPTH_IMG_NON_TRANSFER, bool bIsOutputRGB24=false, void *phWndNotice=0, int *pFPS=0, CONTROL_MODE cm=IMAGE_SN_NONSYNC)

  the implement layer to open Multiple Base Line Etron camera device by V4L2(https://en.wikipedia.org/wiki/Video4Linux), can open color and depth at one time call, do functions as below,

- int EtronDI_CloseDeviceMBL (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  close Multiple Base Linedevice and free resource
- int EtronDI_CloseDevice (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  close device and free resource
- int EtronDI_CloseDeviceEx (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  close device and free resource for warm reset
- int EtronDI_GetImage (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *pBuf, unsigned long int *pImageSize, int *pSerial=0, int nDepthDataType=0)

  get color or depth pin image by issuing V4L2's IOCTL to get frame data
- int EtronDI_GetColorImage (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *pBuf, unsigned long int *pImageSize, int *pSerial=0, int nDepthDataType=0)

  get color image by issuing V4L2's IOCTL to get frame data
- int EtronDI_GetDepthImage (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *pBuf, unsigned long int *pImageSize, int *pSerial=0, int nDepthDataType=0)

  get depth image by issuing V4L2's IOCTL to get frame data
- int EtronDI_SetupBlock (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, bool enable)

  get color or depth pin image by issuing V4L2's IOCTL to get frame data
- int EtronDI_Get_Color_30_mm_depth (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *pBuf, unsigned long int *pImageSize, int *pSerial=0, int nDepthDataType=0)

  get color or depth pin image by issuing V4L2's IOCTL to get frame data
- int EtronDI_Get_60_mm_depth (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *pBuf, unsigned long int *pImageSize, int *pSerial=0, int nDepthDataType=0)

  get color or depth pin image by issuing V4L2's IOCTL to get frame data
- int EtronDI_Get_150_mm_depth (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *pBuf, unsigned long int *pImageSize, int *pSerial=0, int nDepthDataType=0)

  get color or depth pin image by issuing V4L2's IOCTL to get frame data
- int EtronDI_Get2Image (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *pColorImgBuf, BYTE *pDepthImgBuf, unsigned long int *pColorImageSize, unsigned long int *pDepthImageSize, int *pSerial=0, int *pSerial2=0, int nDepthDataType=0)

  get color and/or depth pin images see EtronDI_GetImage for detailed description
- int EtronDI_GetExposureTime (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nSensorMode, float *pfExpTimeMS)

  get exposure time of ISP setting in millisecond the target sensor type was set in EtronDI_SetSensorTypeName()
- int EtronDI_SetExposureTime (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nSensorMode, float fExpTimeMS)

  set exposure time of ISP sensor setting the target sensor type was set in EtronDI_SetSensorTypeName()
- int EtronDI_GetGlobalGain (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nSensorMode, float *pfGlobalGain)

  get global gain of ISP setting the target sensor type was set in EtronDI_SetSensorTypeName()
- int EtronDI_SetGlobalGain (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nSensorMode, float fGlobalGain)

  set global gain of ISP sensor setting the target sensor type was set in EtronDI_SetSensorTypeName()
- int EtronDI_SetSensorTypeName (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, SENSOR_TYPE_NAME stn)

  set the sensor type you want to work on
- int EtronDI_GetColorGain (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nSensorMode, float *pfGainR, float *pfGainG, float *pfGainB)

  get color gain of ISP setting the target sensor type was set in EtronDI_SetSensorTypeName()
- int EtronDI_SetColorGain (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nSensorMode, float fGainR, float fGainG, float fGainB)

  set color gain of ISP
- int EtronDI_EnableAE (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  enable auto exposure(AE) function of ISP

- int EtronDI_DisableAE (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  disable auto exposure(AE) function of ISP
- int EtronDI_EnableAWB (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  enable auto white balance function of ISP
- int EtronDI_DisableAWB (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  disable auto white balance of ISP
- int EtronDI_GetAEStatus (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, PAE_STATUS pAEStatus)

  get auto exposure(AE) is enabled or disable
- int EtronDI_GetAWBStatus (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, PAWB_STATUS pAWB↩ Status)

  get auto white balance(AWB) is enabled or disable
- int EtronDI_GetGPIOValue (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE *pValue)

  get GPIO values
- int EtronDI_SetGPIOValue (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE n↩ Value)

  set GPIO values
- int EtronDI_SetGPIOCtrl (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE n↩ Value)

  set GPIO I/O control
- int EtronDI_GetCTPropVal (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nId, long int *pValue)

  get camera terminal(CT) property value By v4l2_control to get control value of camera terminal
- int EtronDI_SetCTPropVal (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nId, long int nValue)

  set camera terminal property values By v4l2_control to set
- int EtronDI_GetPUPropVal (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nId, long int *pValue)

  get processing unit property value by v4l2_control to get processing unit(PU) property value
- int EtronDI_SetPUPropVal (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nId, long int nValue)

  set processing unit property value by v4l2_control to set processing unit(PU) property value
- int EtronDI_GetCTRangeAndStep (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nId, int *pMax, int *pMin, int *pStep, int *pDefault, int *pFlags)

  set camera terminal property values By v4l2_queryctrl to get control values of camera terminal(CT) this enumeration contained the following properties: V4L2_CID_EXPOSURE_AUTO V4L2_CID_EXPOSURE_AUTO_PRIORI↩ TY V4L2_CID_EXPOSURE_ABSOLUTE V4L2_CID_EXPOSURE V4L2_CID_FOCUS_ABSOLUTE V4L2_CID_F↩ OCUS_RELATIVE V4L2_CID_FOCUS_AUTO V4L2_CID_IRIS_ABSOLUTE V4L2_CID_IRIS_RELATIVE V4L2_↩ CID_ZOOM_ABSOLUTE V4L2_CID_ZOOM_RELATIVE V4L2_CID_PAN_ABSOLUTE V4L2_CID_PAN_RELATIVE V4L2_CID_TILT_ABSOLUTE V4L2_CID_TILT_RELATIVE V4L2_CID_PRIVACY
- int EtronDI_GetPURangeAndStep (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nId, int *pMax, int *pMin, int *pStep, int *pDefault, int *pFlags)

  get processing unit property value By v4l2_queryctrl to get property values of processing unit(PU) this enumeration contained the following properties: V4L2_CID_BACKLIGHT_COMPENSATION V4L2_CID_BRIGHTNESS V4L2_↩ CID_CONTRAST V4L2_CID_GAIN V4L2_CID_POWER_LINE_FREQUENCY V4L2_CID_HUE V4L2_CID_HUE_↩ AUTO V4L2_CID_SATURATION V4L2_CID_SHARPNESS V4L2_CID_GAMMA V4L2_CID_WHITE_BALANCE_↩ TEMPERATURE V4L2_CID_AUTO_WHITE_BALANCE
- int EtronDI_SetDepthDataType (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short nValue)

  set depth data type, 11 bit for disparity data, 14 bit for Z data notice: only PUMA type IC can support this setting
- int EtronDI_GetDepthDataType (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short *p↩ Value)

  get current depth data type setting
- int EtronDI_SetCurrentIRValue (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short nValue)

  set infrared radiation(IR) value of PUMA type IC
- int EtronDI_GetCurrentIRValue (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short *p↩ Value)

  get infrared radiation(IR) value of PUMA type IC

- int EtronDI_GetIRMinValue (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short *pValue)

  get minimum IR value of camera module
- int EtronDI_GetIRMaxValue (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short *pValue)

  get maximum IR value of camera module
- int EtronDI_SetIRMode (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short nValue)

  enable or disable IRs
- int EtronDI_GetIRMode (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short *pValue)

  to check IR is turn on or off
- int EtronDI_GetRectifyLogData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, eSPCtrl_RectLogData *pData, int index)

  get rectify log data from flash, just for AXES1 device type
- int EtronDI_GetRectifyMatLogData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, eSPCtrl_Rect↩LogData *pData, int index)

  get rectify log data from flash, just for PUMA device type
- int EtronDI_EnablePostProcess (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, bool bEnable)

  Not support now.
- int EtronDI_PostInitial (void *pHandleEtronDI)

  Not support now.
- int EtronDI_PostEnd (void *pHandleEtronDI)

  Not support now.
- int EtronDI_ProcessFrame (void *pHandleEtronDI, unsigned char *pYUY2Buf, unsigned char *pDepthBuf, unsigned char *OutputBuf, int width, int height)

  Not support now.
- int EtronDI_PostSetParam (void *pHandleEtronDI, int Idx, int Val)

  Not support now.
- int EtronDI_PostGetParam (void *pHandleEtronDI, int Idx, int *pVal)

  Not support now.
- int EtronDI_CreateSwPostProc (int depthBits, void **handle)

  create a software post process class
- int EtronDI_ReleaseSwPostProc (void **handle)

  release a software post process class
- int EtronDI_DoSwPostProc (void *pHandleEtronDI, unsigned char *colorBuf, bool isColorRgb24, unsigned char *depthBuf, unsigned char *outputBuf, int width, int height)

  do software post process on a depth buffer
- int EtronDI_Convert_Depth_Y_To_Buffer (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned char *depth_y, unsigned char *rgb, unsigned int width, unsigned int height, bool color, unsigned short n↩DepthDataType)

  Convert Depth to RGB color or gray.
- int EtronDI_Convert_Depth_Y_To_Buffer_offset (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned char *depth_y, unsigned char *rgb, unsigned int width, unsigned int height, bool color, unsigned short nDepthDataType, int offset)

  Convert Depth to RGB color or gray, added offset for 3cm baseline.
- int EtronDI_EnableSensorIF (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, bool bIsEnable)

  enable or disable sensor IF
- int EtronDI_getUACNAME (char *input, char *output)

  Get Etron UAC Name.
- int EtronDI_InitialUAC (char *deviceName)

  UAC inital function.
- int EtronDI_WriteWaveHeader (int fd)

  Write Wave Header.
- int EtronDI_WriteWaveEnd (int fd, size_t length)

  Modified Wave Header.

- int EtronDI_GetUACData (unsigned char *buffer, int length)

  UAC inital function.
- int EtronDI_ReleaseUAC (void)

  UAC inital function.
- int EtronDI_InitialFlexibleGyro (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  gyro sensor inital function
- int EtronDI_ReleaseFlexibleGyro (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  gyro sensor release function
- int EtronDI_GetFlexibleGyroData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, int length, unsigned char *pGyroData)

  getting gyro data function
- int EtronDI_GetFlexibleGyroLength (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short *GyroLen)

  getting length of gyro data function.
- int EtronDI_GetImageInterrupt (void)

  Get Image interrupt function Get the image interrupt and then read Gyro data.
- int EtronDI_InitialHidGyro (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  gyro sensor inital function
- int EtronDI_ReleaseHidGyro (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo)

  gyro sensor release function
- int EtronDI_GetHidGyro (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned char *pBuffer, int length)

  getting gyro data function
- int EtronDI_SetupHidGyro (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned char *pCmdBuf, int cmdlength)

  getting gyro data function
- int EtronDI_GetInfoHidGyro (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned char *pCmdBuf, int cmdlength, unsigned char *pResponseBuf, int *resplength)

  getting gyro data function
- int EtronDI_GenerateLutFile (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, const char *filename)

  generate look up table(LUT) for spherical display this function reads the camera user data and generate a LUT file using for 360 degree preview
- int EtronDI_SaveLutData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, const char *filename)

  Save LUT parameters in the specified file.
- int EtronDI_GetLutData (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, BYTE *buffer, int nSize)

  Read LUT parameters into the specified buffer.
- int EtronDI_EncryptMP4 (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, const char *filename)

  encrypt a H.264 video
- int EtronDI_DecryptMP4 (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, const char *filename)

  decrypt a H.264 video was generated by EtronDI_EncryptMP4()
- int EtronDI_GetAutoExposureMode (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short *mode)

  Get Auto Exposure Mode.
- int EtronDI_SetAutoExposureMode (void *pHandleEtronDI, PDEVSELINFO pDevSelInfo, unsigned short mode)

  Setup Auto Exposure Mode.
- int EtronDI_RotateImg90 (EtronDIImageType::Value imgType, int width, int height, unsigned char *src, unsigned char *dst, int len, bool clockwise)

  Rotate the image to 90 degree.
- int EtronDI_RotateImg180 (EtronDIImageType::Value imgType, int width, int height, unsigned char *src, unsigned char *dst, int len)

  Rotate the image to 180 degree.

- int EtronDI_ResizeImgToHalf (EtronDIImageType::Value imgType, int width, int height, unsigned char *src, unsigned char *dst, int len)

  Resize the image to half.
- int EtronDI_ImgMirro (EtronDIImageType::Value imgType, int width, int height, unsigned char *src, unsigned char *dst)

  Make the image to Mirro.
- int EtronDI_RGB2BMP (char *filename, int width, int height, unsigned char *data)

  RGB to BMP.
- int EtronDI_HoleFilled (unsigned short *pDImgIn, unsigned short *pDImgOut, int width, int height, int holeFilldiff)

  Hole Filled.
- int EtronDI_InitialCmdFiFo (const char *pfifoName, int *pFileDescrption, bool bRead)

  Cmd FiFo Initial function.
- int EtronDI_CloseCmdFiFo (int FileDescrption)

  Cmd FiFo Close function.
- int EtronDI_WriteCmdFiFo (int FileDescrption, unsigned char *pCmd, int len)

  Write Cmd FiFo function.
- int EtronDI_ReadCmdFiFo (int FileDescrption, unsigned char *pBuf, int len)

  Read Cmd FiFo function.
- int EtronDI_InitSRB (void **pSmbHandle, int QueueSize, char *queueName)

  Inital the SRB(Share Ring Buffering)
- int EtronDI_PutSRB (void *pSmbHandle, srb_packet_s *pPacket)

  Put Packet to SRB.
- int EtronDI_GetSRB (void *pSmbHandle, srb_packet_s *pPacket)

  Get Packet from SRB.

3.1.1 Detailed Description

Etron SDK API export functions, data structure and variable definition Copyright: This file copyright (C) 2017 by.

eYs3D an Etron 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 an Etron company.

3.1.2 Function Documentation

3.1.2.1 EtronDI_CloseCmdFiFo()

```
int EtronDI_CloseCmdFiFo (
            int FileDescrption )
```

Cmd FiFo Close function.

Parameters

int	FileDescrption File Description

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.2 EtronDI_CloseDevice()

```
int EtronDI_CloseDevice (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

close device and free resource

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.3 EtronDI_CloseDeviceEx()

```
int EtronDI_CloseDeviceEx (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

close device and free resource for warm reset

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.4 EtronDI_CloseDeviceMBL()

```
int EtronDI_CloseDeviceMBL (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

close Multiple Base Linedevice and free resource

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.5 EtronDI_Convert_Depth_Y_To_Buffer()

```
int EtronDI_Convert_Depth_Y_To_Buffer (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned char * depth_y,
            unsigned char * rgb,
            unsigned int width,
            unsigned int height,
            bool color,
            unsigned short nDepthDataType )
```

Convert Depth to RGB color or gray.

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char ∗depth_y depth data,
unsigned	char ∗rgb output data,
int	width image width,
int	height image height,

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.6 EtronDI_Convert_Depth_Y_To_Buffer_offset()

```
int EtronDI_Convert_Depth_Y_To_Buffer_offset (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned char * depth_y,
            unsigned char * rgb,
            unsigned int width,
            unsigned int height,
            bool color,
            unsigned short nDepthDataType,
            int offset )
```

Convert Depth to RGB color or gray, added offset for 3cm baseline.

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char ∗depth_y depth data,
unsigned	char ∗rgb output data,
int	width image width,
int	height image height,
int	offset dpeth_y offset,

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.7 EtronDI_CreateSwPostProc()

```
int EtronDI_CreateSwPostProc (
            int depthBits,
            void ** handle )
```

create a software post process class

Parameters

int	depthBits depth bit to set
void	∗∗handle handle pointer to this software post process class

Returns

success: EtronDI_OK, others: see eSPDI_ErrCode.h

3.1.2.8 EtronDI_DecryptMP4()

```
int EtronDI_DecryptMP4 (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            const char * filename )
```

decrypt a H.264 video was generated by EtronDI_EncryptMP4()

Parameters

void∗	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char ∗filename the input video file for decryption

Returns

success: EtronDI_OK, others:see En-Decrypt.h

3.1.2.9 EtronDI_DisableAE()

```
int EtronDI_DisableAE (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

disable auto exposure(AE) function of ISP

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.10 EtronDI_DisableAWB()

```
int EtronDI_DisableAWB (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

disable auto white balance of ISP

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.11 EtronDI_DoFusion()

```
int EtronDI_DoFusion (
            unsigned char ** pDepthBufList,
            double * pDepthMerge,
            unsigned char * pDepthMergeFlag,
            int nDWidth,
            int nDHeight,
            double fFocus,
            double * pBaseline,
            double * pWRNear,
            double * pWRFar,
            double * pWRFusion,
            int nMergeNum,
            bool bdepth2Byte11bit,
            int method )
```

Do Fusion Merge.

Parameters

unsigned	char ∗∗pDepthBufList Point to Depth Buffer List
double	∗pDepthMerge Point to Fusion output.
unsigned	char ∗pDepthMergeFlag Point to Fusion select fFocus Focus vale
int	nDWidth Image width
int	nDHeight Image Height
double	∗pBaseline Point to baseline array m_baselineDist[0] = 30.0; m_baselineDist[1] = 60.0; m_baselineDist[2] = 150.0;
double	∗pWRNear NearWorkingRange Vecror(Container)
double	∗pWRFar FarWorkingRange Vecror(Container)
double	∗pWRFusion FusionWorkingRange Vecror(Container)
int	nMergeNum Total merges
int	method method select 0: MBLBase 1: MBRbaseV0 2: MBRbaseV1

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.12 EtronDI_DoSwPostProc()

```
int EtronDI_DoSwPostProc (
            void * handle,
            unsigned char * colorBuf,
            bool isColorRgb24,
            unsigned char * depthBuf,
            unsigned char * outputBuf,
            int width,
            int height )
```

do software post process on a depth buffer

Parameters

void∗	handle handle of this software post process class
unsigned	char∗ colorBuf input color buffer
bool	isColorRgb24 is this color buffer RGB888
unsigned	char∗ depthBuf input depth buffer
unsigned	char∗ outputBuf output buffer
int	width image width
int	height image height

Returns

success: EtronDI_OK, others: see eSPDI_ErrCode.h

3.1.2.13 EtronDI_EnableAE()

```
int EtronDI_EnableAE (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

enable auto exposure(AE) function of ISP

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.14 EtronDI_EnableAWB()

```
int EtronDI_EnableAWB (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

enable auto white balance function of ISP

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.15 EtronDI_EnableInterleave()

```
int EtronDI_EnableInterleave (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            bool enable )
```

enable or disable interleave function

Parameters

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

Returns

success: EtronDI_OK, others: see eSPDI_ErrCode.h

3.1.2.16 EtronDI_EnableSensorIF()

```
int EtronDI_EnableSensorIF (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            bool bIsEnable )
```

enable or disable sensor IF

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
bool	bIsEnable true is enable, false is disable

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.17 EtronDI_EncryptMP4()

```
int EtronDI_EncryptMP4 (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            const char * filename )
```

encrypt a H.264 video

Parameters

void∗	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char ∗filename the input video file for encryption

Returns

success: EtronDI_OK, others:see En-Decrypt.h

3.1.2.18 EtronDI_FindDevice()

```
int EtronDI_FindDevice (
            void * pHandleEtronDI )
```

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

Parameters

void	∗pHandleEtronDI CEtronDI handler

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.19 EtronDI_GenerateLutFile()

```
int EtronDI_GenerateLutFile (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            const char * filename )
```

generate look up table(LUT) for spherical display this function reads the camera user data and generate a LUT file using for 360 degree preview

Parameters

void∗	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char∗ filename output LUT file name

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.20 EtronDI_Get2Image()

```
int EtronDI_Get2Image (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * pColorImgBuf,
            BYTE * pDepthImgBuf,
            unsigned long int * pColorImageSize,
            unsigned long int * pDepthImageSize,
            int * pColorSerial = 0,
            int * pDepthSerial = 0,
            int nDepthDataType = 0 )
```

get color and/or depth pin images see EtronDI_GetImage for detailed description

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗pColorImgBuf buffer to store color image
BYTE	∗pDepthImgBuf buffer to store depth image
unsigned	long int ∗pColorImageSize the actual color buffer size
unsigned	long int ∗pDepthImageSize the actual depth buffer size
int	∗pColorSerial color serial number
int	∗pDepthSerial depth serial number
int	nDepthDataType the depth data type, see definition in eSPDI_def.h

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.21 EtronDI_Get_150_mm_depth()

```
int EtronDI_Get_150_mm_depth (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * pBuf,
            unsigned long int * pImageSize,
            int * pSerial = 0,
            int nDepthDataType = 0 )
```

get color or depth pin image by issuing V4L2's IOCTL to get frame data

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗pDepthImgBuf buffer to store image data
unsigned	long int ∗pImageSize the actual buffer size getting from device
int	∗pDepthSerial the serial number for synchronizing depth image
int	nDepthDataType the depth data type, see definition in eSPDI_def.h

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.22 EtronDI_Get_60_mm_depth()

```
int EtronDI_Get_60_mm_depth (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * pBuf,
            unsigned long int * pImageSize,
            int * pSerial = 0,
            int nDepthDataType = 0 )
```

get color or depth pin image by issuing V4L2's IOCTL to get frame data

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗pBuf buffer to store image data
unsigned	long int ∗pImageSize the actual buffer size getting from device
int	∗pSerial the serial number for synchronizing color and depth image
int	nDepthDataType the depth data type, see definition in eSPDI_def.h

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.23 EtronDI_Get_Color_30_mm_depth()

```
int EtronDI_Get_Color_30_mm_depth (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * pBuf,
            unsigned long int * pImageSize,
            int * pSerial = 0,
            int nDepthDataType = 0 )
```

get color or depth pin image by issuing V4L2's IOCTL to get frame data

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗pBuf buffer to store image data
unsigned	long int ∗pImageSize the actual buffer size getting from device
int	∗pSerial the serial number for synchronizing color and depth image
int	nDepthDataType the depth data type, see definition in eSPDI_def.h

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.24 EtronDI_GetAEStatus()

```
int EtronDI_GetAEStatus (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            PAE_STATUS pAEStatus )
```

get auto exposure(AE) is enabled or disable

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
PAE_STATUS	pAEStatus see enum definition as to AE_STATUS in eSPDI_def.h

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.25 EtronDI_GetAutoExposureMode()

```
int EtronDI_GetAutoExposureMode (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short * mode )
```

Get Auto Exposure Mode.

Parameters

void∗	pHandleEtronDI CEtronDI handler.
PDEVSELINFO	pDevSelInfo pointer of device select index.
unsigned	short∗ mode pointer of the mode value. 0: Average, 1: Left (or Front) camera, 2: Right (or Back) camera

Returns

success: EtronDI_OK, others:eSPDI_def.h

3.1.2.26 EtronDI_GetAWBStatus()

```
int EtronDI_GetAWBStatus (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            PAWB_STATUS pAWBStatus )
```

get auto white balance(AWB) is enabled or disable

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
PAWB_STATUS	pAWBStatus see enum definition as to AWB_STATUS in eSPDI_def.h

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.27 EtronDI_GetColorGain()

```
int EtronDI_GetColorGain (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float * pfGainR,
            float * pfGainG,
            float * pfGainB )
```

get color gain of ISP setting the target sensor type was set in EtronDI_SetSensorTypeName()

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nSensorMode which sensor(sensor A, B or Both) to get A is 0, B is 1, Both is 2
float	∗pfGainR pointer of red gain value of ISP setting
float	∗pfGainG pointer of green gain value of ISP setting
float	∗pfGainB pointer of blue gain value of ISP setting

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.28 EtronDI_GetColorImage()

```
int EtronDI_GetColorImage (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * pBuf,
            unsigned long int * pImageSize,
            int * pSerial = 0,
            int nDepthDataType = 0 )
```

get color image by issuing V4L2's IOCTL to get frame data

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗pBuf buffer to store image data
unsigned	long int ∗pImageSize the actual buffer size getting from device
int	∗pSerial the serial number for synchronizing color and depth image
int	nDepthDataType reserved, no used.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.29 EtronDI_GetControlCounterMode()

```
int EtronDI_GetControlCounterMode (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned char * nValue )
```

enable or disable interleave function

Parameters

pHandleEtronDI	the pointer to the initilized EtronDI SDK instance
pDevSelInfo	pointer of device select index
∗nValue	pointer to frame counter mode value, 0: Frame Counter Mode, 1: Serial Counter Mode,

Returns

success: EtronDI_OK, others: see eSPDI_ErrCode.h

3.1.2.30 EtronDI_GetCTPropVal()

```
int EtronDI_GetCTPropVal (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nId,
            long int * pValue )
```

get camera terminal(CT) property value By v4l2_control to get control value of camera terminal

this enumeration contained the following properties: V4L2_CID_EXPOSURE_AUTO; V4L2_CID_EXPOSURE_A↩
UTO_PRIORITY V4L2_CID_EXPOSURE_ABSOLUTE V4L2_CID_EXPOSURE V4L2_CID_FOCUS_ABSOLUTE
V4L2_CID_FOCUS_RELATIVE V4L2_CID_FOCUS_AUTO V4L2_CID_IRIS_ABSOLUTE V4L2_CID_IRIS_REL↩
ATIVE V4L2_CID_ZOOM_ABSOLUTE V4L2_CID_ZOOM_RELATIVE V4L2_CID_PAN_ABSOLUTE V4L2_CID↩
_PAN_RELATIVE V4L2_CID_TILT_ABSOLUTE V4L2_CID_TILT_RELATIVE V4L2_CID_PRIVACY

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nId specifies the member of the property set, see CT Property ID defined in eSPDI_def.h
int	∗pValue pointer of store CT property value

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.31 EtronDI_GetCTRangeAndStep()

```
int EtronDI_GetCTRangeAndStep (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nId,
            int * pMax,
            int * pMin,
            int * pStep,
            int * pDefault,
            int * pFlags )
```

set camera terminal property values By v4l2_queryctrl to get control values of camera terminal(CT) this enumeration contained the following properties: V4L2_CID_EXPOSURE_AUTO V4L2_CID_EXPOSURE_AUTO_PRIOR↩ ITY V4L2_CID_EXPOSURE_ABSOLUTE V4L2_CID_EXPOSURE V4L2_CID_FOCUS_ABSOLUTE V4L2_CID↩ _FOCUS_RELATIVE V4L2_CID_FOCUS_AUTO V4L2_CID_IRIS_ABSOLUTE V4L2_CID_IRIS_RELATIVE V4↩ L2_CID_ZOOM_ABSOLUTE V4L2_CID_ZOOM_RELATIVE V4L2_CID_PAN_ABSOLUTE V4L2_CID_PAN_RE↩ LATIVE V4L2_CID_TILT_ABSOLUTE V4L2_CID_TILT_RELATIVE V4L2_CID_PRIVACY

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nId specifies the member of the property set, see CT Property ID defined in eSPDI_def.h
long	int ∗pMax maximum value, inclusive. This field gives an upper bound for the control
long	int ∗pMin minimum value, inclusive. This field gives a lower bound for the control
long	int ∗pStep This field gives a step size for the control see enum https://www.↩ linuxtv.org/downloads/v4l-dvb-apis-old/vidioc-queryctrl.↩ html#v4l2-ctrl-type how the step value is to be used for each possible control type. Note that this an unsigned 32-bit value
long	int ∗pDefault The default value of a V4L2_CTRL_TYPE_INTEGER, _BOOLEAN, _BITMASK, _MENU or _INTEGER_MENU control. Not valid for other types of controls. Note that drivers reset controls to their default value only when the driver is first loaded, never afterwards.
long	int ∗pFlags control flags, see https://www.linuxtv.↩ org/downloads/v4l-dvb-apis-old/vidioc-queryctrl.↩ html#control-flags

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.32 EtronDI_GetCurrentIRValue()

```
int EtronDI_GetCurrentIRValue (
            void * pHandleEtronDI,
```

```
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

get infrared radiation(IR) value of PUMA type IC

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short ∗pValue current 1 byte IR value setting

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.33 EtronDI_GetDepthDataType()

```
int EtronDI_GetDepthDataType (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

get current depth data type setting

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
WORD	∗pValue pointer of current depth data type in device

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.34 EtronDI_GetDepthImage()

```
int EtronDI_GetDepthImage (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * pBuf,
            unsigned long int * pImageSize,
            int * pSerial = 0,
            int nDepthDataType = 0 )
```

get depth image by issuing V4L2's IOCTL to get frame data

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗pBuf buffer to store image data
unsigned	long int ∗pImageSize the actual buffer size getting from device
int	∗pSerial the serial number for synchronizing color and depth image
int	nDepthDataType the depth data type, see definition in eSPDI_def.h

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.35 EtronDI_GetDeviceInfo()

```
int EtronDI_GetDeviceInfo (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            DEVINFORMATION * pdevinfo )
```

get informations of Etron UVC devices, see DEVINFORMATION

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
DEVINFORMATION∗	pdevinfo pointer of device information

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.36 EtronDI_GetDeviceInfoMBL_15cm()

```
int EtronDI_GetDeviceInfoMBL_15cm (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            DEVINFORMATION * pdevinfo )
```

get informations of Etron UVC devices, see DEVINFORMATION

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
DEVINFORMATION∗	pdevinfo pointer of device information

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.37 EtronDI_GetDeviceNumber()

```
int EtronDI_GetDeviceNumber (
            void * pHandleEtronDI )
```

get Etron USB device numbers

Parameters

void	∗pHandleEtronDI CEtronDI handler

Returns

number of Etron device

3.1.2.38 EtronDI_GetDeviceResolutionList()

```
int EtronDI_GetDeviceResolutionList (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nMaxCount0,
            ETRONDI_STREAM_INFO * pStreamInfo0,
            int nMaxCount1,
            ETRONDI_STREAM_INFO * pStreamInfo1 )
```

get the device resolution list

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nMaxCount0 max count of endpoint1 resolutions
ETRONDI_STREAM_INFO	∗pStreamInfo0 resolution infos of endpoint1
int	nMaxCount1 max count of endpoint2 resolutions
ETRONDI_STREAM_INFO	∗pStreamInfo1 resolutions infos of endpoint2

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.39 EtronDI_GetExposureTime()

```
int EtronDI_GetExposureTime (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float * pfExpTimeMS )
```

get exposure time of ISP setting in millisecond the target sensor type was set in EtronDI_SetSensorTypeName()

Parameters

void	∗pHandleEtronDI pHandleEtronDI
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nSensorMode which sensor(sensor A, B or Both) to get A is 0, B is 1, Both is 2
float	∗pfExpTimeMS pointer of getting exposure time in millisecond by pixel clock, pixel per line, exposure line to get exposure time

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.40 EtronDI_GetFlexibleGyroData()

```
int EtronDI_GetFlexibleGyroData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int length,
            unsigned char * pGyroData )
```

getting gyro data function

Parameters

void∗	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	length Gyro Data Length
unsigned	char ∗pGyroData pointer of Gyro Data.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.41 EtronDI_GetFlexibleGyroLength()

```
int EtronDI_GetFlexibleGyroLength (
            void * pHandleEtronDI,
```

```
            PDEVSELINFO pDevSelInfo,
            unsigned short * GyroLen )
```

getting length of gyro data function.

Parameters

void∗	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short∗ GyroLen pointer of Gyro Data Lenhth.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.42 EtronDI_GetFWRegister()

```
int EtronDI_GetFWRegister (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short address,
            unsigned short * pValue,
            int flag )
```

get firmware register value

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short address register address
unsigned	short ∗pValue pointer of value got from register address
int	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: EtronDI_OK, others: see eSPDI_def.h

3.1.2.43 EtronDI_GetFwVersion()

```
int EtronDI_GetFwVersion (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
```

```
            char * pszFwVersion,
            int nBufferSize,
            int * pActualLength )
```

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

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
char	∗pszFwVersion firmware version string
int	nBufferSize input buffer length to receive FW version
int	∗pActualLength the actual length of FW version in byte

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.44 EtronDI_GetGlobalGain()

```
int EtronDI_GetGlobalGain (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float * pfGlobalGain )
```

get global gain of ISP setting the target sensor type was set in EtronDI_SetSensorTypeName()

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nSensorMode which sensor(sensor A, B or Both) to get A is 0, B is 1, Both is 2
float	∗pfGlobalGain pointer of global gain value

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.45 EtronDI_GetHidGyro()

```
int EtronDI_GetHidGyro (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pBuffer,
            int length )
```

getting gyro data function

Parameters

void∗	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char ∗pGyroData pointer of Gyro Data Buffer.
int	length Input buffer Length, should be >= 24

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.46 EtronDI_GetHWRegister()

```
int EtronDI_GetHWRegister (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short address,
            unsigned short * pValue,
            int flag )
```

get hardware register value

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short address register address
unsigned	short ∗pValue pointer of value got from register address
int	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: EtronDI_OK, others: see eSPDI_ErrCode.h

3.1.2.47 EtronDI_GetImage()

```
int EtronDI_GetImage (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * pBuf,
            unsigned long int * pImageSize,
            int * pSerial = 0,
            int nDepthDataType = 0 )
```

get color or depth pin image by issuing V4L2's IOCTL to get frame data

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗pBuf buffer to store image data
unsigned	long int ∗pImageSize the actual buffer size getting from device
int	∗pSerial the serial number for synchronizing color and depth image
int	nDepthDataType the depth data type, see definition in eSPDI_def.h

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.48 EtronDI_GetImageInterrupt()

```
int EtronDI_GetImageInterrupt (
            void  )
```

Get Image interrupt function Get the image interrupt and then read Gyro data.

Returns

success: 0, others: not got interrupt

3.1.2.49 EtronDI_GetInfoHidGyro()

```
int EtronDI_GetInfoHidGyro (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pCmdBuf,
            int cmdlength,
            unsigned char * pResponseBuf,
            int * resplength )
```

getting gyro data function

Parameters

void∗	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char ∗pCmdBuf pointer of Gyro Cmd Buffer.
int	cmdlength Command Lehgth.
unsigned	char ∗pResponseBuf pointer of ResponseBuffer.
int	resplength Response Length

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.50 EtronDI_GetIRMaxValue()

```
int EtronDI_GetIRMaxValue (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

get maximum IR value of camera module

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short ∗pValue the maximum 1 byte IR value can be set

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.51 EtronDI_GetIRMinValue()

```
int EtronDI_GetIRMinValue (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

get minimum IR value of camera module

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short ∗pValue the minimum 1 byte IR value can be set

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.52 EtronDI_GetIRMode()

```
int EtronDI_GetIRMode (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

to check IR is turn on or off

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short ∗pValue get IR was enabled or not D[7:4]: Reserved D3: Channel 3 D2: Channel 2 D1: Channel 1 D0: Channel 0 1: Enable Channel 0: Disable Channel If want to control ch0 and ch1, ubMode[3:0] must set to 0x03

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.53 EtronDI_GetLogData()

```
int EtronDI_GetLogData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            int index,
            CALIBRATION_LOG_TYPE type )
```

get log data from flash

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer buffer to store log data
int	BufferLength input buffer length, must be 4096
int	∗pActualLength actual length has written to buffer
int	index index to identify log data for corresponding depth
CALIBRATION_LOG_TYPE	type which calibration log to get

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.54 EtronDI_GetLutData()

```
int EtronDI_GetLutData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int nSize )
```

Read LUT parameters into the specified buffer.

Parameters

void∗	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE∗	buffer memory to store LUT data
int	nSize length of buffer in bytes

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.55 EtronDI_GetPidVid()

```
int EtronDI_GetPidVid (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pPidBuf,
            unsigned short * pVidBuf )
```

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

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short ∗pPidBuf 4 byte buffer to store PID value
unsigned	short ∗pVidBuf 4 byte buffer to store VID value

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.56 EtronDI_GetPUPropVal()

```
int EtronDI_GetPUPropVal (
            void * pHandleEtronDI,
```

```
            PDEVSELINFO pDevSelInfo,
            int nId,
            long int * pValue )
```

get processing unit property value by v4l2_control to get processing unit(PU) property value

this enumeration contained the following properties: V4L2_CID_BACKLIGHT_COMPENSATION V4L2_CID_BR↩IGHTNESS V4L2_CID_CONTRAST V4L2_CID_GAIN V4L2_CID_POWER_LINE_FREQUENCY V4L2_CID_HUE V4L2_CID_HUE_AUTO V4L2_CID_SATURATION V4L2_CID_SHARPNESS V4L2_CID_GAMMA V4L2_CID_W↩HITE_BALANCE_TEMPERATURE V4L2_CID_AUTO_WHITE_BALANCE

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nId specifies the member of the property set see PU property ID defined in eSPDI_def.h
long	int ∗pValue pointer of store PU property value

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.57 EtronDI_GetPURangeAndStep()

```
int EtronDI_GetPURangeAndStep (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nId,
            int * pMax,
            int * pMin,
            int * pStep,
            int * pDefault,
            int * pFlags )
```

get processing unit property value By v4l2_queryctrl to get property values of processing unit(PU) this enumeration contained the following properties: V4L2_CID_BACKLIGHT_COMPENSATION V4L2_CID_BRIGHTNESS V4L2↩_CID_CONTRAST V4L2_CID_GAIN V4L2_CID_POWER_LINE_FREQUENCY V4L2_CID_HUE V4L2_CID_HU↩E_AUTO V4L2_CID_SATURATION V4L2_CID_SHARPNESS V4L2_CID_GAMMA V4L2_CID_WHITE_BALAN↩CE_TEMPERATURE V4L2_CID_AUTO_WHITE_BALANCE

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nId nId specifies the member of the property set, see CT Property ID defined in eSPDI_def.h
long	int ∗pMax maximum value, inclusive. This field gives an upper bound for the control
long	int ∗pMin minimum value, inclusive. This field gives a lower bound for the control
long	int ∗pStep This field gives a step size for the control see enum https://www.↩linuxtv.org/downloads/v4l-dvb-apis-old/vidioc-queryctrl.↩html#v4l2-ctrl-type how the step value is to be used for each possible control type. Note that this an unsigned 32-bit value

Parameters

long	int *pDefault The default value of a V4L2_CTRL_TYPE_INTEGER, _BOOLEAN, _BITMASK, _MENU or _INTEGER_MENU control. Not valid for other types of controls. Note that drivers reset controls to their default value only when the driver is first loaded, never afterwards.
long	int *pFlags control flags, see https://www.linuxtv.org/downloads/v4l-dvb-apis-old/vidioc-queryctrl.html#control-flags

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.58 EtronDI_GetRectifyLogData()

```
int EtronDI_GetRectifyLogData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            eSPCtrl_RectLogData * pData,
            int index )
```

get rectify log data from flash, just for AXES1 device type

Parameters

void	*pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
eSPCtrl_RectLogData	*pData 4096 bytes of rectify log data, see eSPCtrl_RectLogData for detailed members
index,user	data section from 0 ~ 9

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.59 EtronDI_GetRectifyMatLogData()

```
int EtronDI_GetRectifyMatLogData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            eSPCtrl_RectLogData * pData,
            int index )
```

get rectify log data from flash, just for PUMA device type

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
eSPCtrl_RectLogData	∗pData 4096 bytes of rectify log data, see eSPCtrl_RectLogData for detailed members
index,user	data section from 0 ∼ 9

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.60 EtronDI_GetRectifyTable()

```
int EtronDI_GetRectifyTable (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            int index )
```

get rectify values (file ID 40+) from flash

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer buffer to store rectify table data
int	BufferLength input buffer length, must be 1024
int	∗pActualLength actual length has written to buffer
int	index index(from 0 ∼ 9) to identify rectify table for corresponding depth

Returns

success:EtronDI_OK, others: see eSPDI_def.h

3.1.2.61 EtronDI_GetSensorRegister()

```
int EtronDI_GetSensorRegister (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nId,
            unsigned short address,
            unsigned short * pValue,
            int flag,
            SENSORMODE_INFO SensorMode )
```

get value from sensor register

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nId sensor slave address see Videodevice.h for sensor slave address setting
unsigned	short address register address
unsigned	short ∗pValue pointer of value got from register address
int	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
SENSORMODE_INFO	SensorMode sensor mode(sensor A, B or Both) A is 0, B is 1, Both is 2

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.62 EtronDI_GetSerialNumber()

```
int EtronDI_GetSerialNumber (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pData,
            int nbufferSize,
            int * pLen )
```

get device serial number

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗pData output buffer to store serial number string
int	nbufferSize pData buffer length in byte, 2 byte(WideChar) is a unit
int	∗pLen pointer of actual serial number length

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.63 EtronDI_GetSRB()

```
int EtronDI_GetSRB (
            void * pSrbHandle,
            srb_packet_s * pPacket )
```

Get Packet from SRB.

Parameters

void	*pSrbHandle pointer to SRB class
packet_s	*pPacket Input Packet

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.64 EtronDI_GetUACData()

```
int EtronDI_GetUACData (
            unsigned char * buffer,
            int length )
```

UAC inital function.

Parameters

unsigned	char *buffer pointer of UAC buffer
int	length UAC buffer length

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.65 EtronDI_getUACNAME()

```
int EtronDI_getUACNAME (
            char * input,
            char * output )
```

Get Etron UAC Name.

Parameters

char	*input Point to device Address.
char	*output Point to device Name.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.66 EtronDI_GetUserData()

```
int EtronDI_GetUserData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            USERDATA_SECTION_INDEX usi )
```

get user data from flash

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer buffer to store user data
int	BufferLength input buffer length
USERDATA_SECTION_INDEX	usi which user index data to select

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.67 EtronDI_GetYOffset()

```
int EtronDI_GetYOffset (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            int index )
```

get Y offset (file ID 30+) value

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer buffer to store Y offset values
int	BufferLength must be 256
int	∗pActualLength the buffer length, always be 256
int	index index value to file ID 30

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.68 EtronDI_GetZDTable()

```
int EtronDI_GetZDTable (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            PZDTABLEINFO pZDTableInfo )
```

get disparity and Z values from flash

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer bufer to store ZD table
int	BufferLength input buffer length
int	∗pActualLength actual length has written to buffer
PZDTABLEINFO	pZDTableInfo index to identify ZD table and data type for corrresponding depth

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.69 EtronDI_HoleFilled()

```
int EtronDI_HoleFilled (
            unsigned short * pDImgIn,
            unsigned short * pDImgOut,
            int width,
            int height,
            int holeFilldiff )
```

Hole Filled.

Parameters

unsigned	short ∗pDImgIn Image Input
unsigned	short ∗pDImgOut Image Output
int	width image width
int	height image height
int	holeFilldiff Hole filled strangth, value from 0 to 2047.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.70 EtronDI_ImgMirro()

```
int EtronDI_ImgMirro (
            EtronDIImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dstBuf )
```

Make the image to Mirro.

Parameters

EtronDIImageType::Value	imgType Image Type
int	width image width
int	height image height
unsigned	char ∗src image source
unsigned	char ∗dstBuf image desteration

Returns

success: EtronDI_OK, others: see eSPDI_def.h

Parameters

pHandleEtronDI	the pointer to the initilized EtronDI SDK instance
pDevSelInfo	pointer of device select index
EtronDIImageType::Value	imgType Image Type
int	width image width
int	height image height
unsigned	char ∗src image source
unsigned	char ∗dstBuf image desteration

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.71 EtronDI_Init()

```
int EtronDI_Init (
            void ** ppHandleEtronDI,
            bool bIsLogEnabled )
```

entry point of Etron camera SDK including 1.create a CEtronDI class for accessing oncming APIs 2.find out Etron devices 3.create a CVideoDevice class for video streaming and hardware access

Parameters

***ppHandleEtronDI*	a pointer of pointer to access CEtronDI class
bIsLogEnabled	generate log or not

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.72 EtronDI_InitialCmdFiFo()

```
int EtronDI_InitialCmdFiFo (
            const char * pfifoName,
            int * pFileDescrption,
            bool bRead )
```

Cmd FiFo Initial function.

Parameters

const	char *pfifoName Point to the cmd fifo name
int	*pFileDescrption Point to the file description
bRead	Indicate Read or Write Cmd fifo

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.73 EtronDI_InitialFlexibleGyro()

```
int EtronDI_InitialFlexibleGyro (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

gyro sensor inital function

Parameters

*void**	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.74 EtronDI_InitialHidGyro()

```
int EtronDI_InitialHidGyro (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

gyro sensor inital function

Parameters

*void**	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.75 EtronDI_InitialUAC()

```
int EtronDI_InitialUAC (
            char * deviceName )
```

UAC inital function.

Parameters

char	*deviceName Point to device Name.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.76 EtronDI_InitSRB()

```
int EtronDI_InitSRB (
            void ** pSrbHandle,
            int QueueSize,
            char * queueName )
```

Inital the SRB(Share Ring Buffering)

Parameters

void	**pSrbHandle a pointer of pointer to SRB class
int	QueueSize
char	srbName SRM Name

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.77 EtronDI_IsInterleaveDevice()

```
bool EtronDI_IsInterleaveDevice (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

check module support interleave function or not

Parameters

pHandleEtronDI	the pointer to the initilized EtronDI SDK instance
pDevSelInfo	pointer of device select index

Returns

true: support interleave, false: not support

3.1.2.78 EtronDI_IsMLBaseLine()

```
bool EtronDI_IsMLBaseLine (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

Check the device is multiple baseline device.

Parameters

void	*pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

true: multiplies baseline device, false: normally device.

3.1.2.79 EtronDI_OpenDevice()

```
int EtronDI_OpenDevice (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nEP0Width,
            int nEP0Height,
            bool bEP0MJPG,
            int nEP1Width,
            int nEP1Height,
            DEPTH_TRANSFER_CTRL dtc = DEPTH_IMG_NON_TRANSFER,
            bool bIsOutputRGB24 = false,
            void * phWndNotice = 0,
            int * pFPS = 0,
            CONTROL_MODE cm = IMAGE_SN_NONSYNC )
```

the implement layer to open Etron camera device by V4L2(https://en.wikipedia.org/wiki/Video4↩Linux), can open color and depth at one time call, do functions as below,

1. initialize the USB device by V4L2 protocol 1.1 query device v4l2 capability 1.2 must have video capability 1.3 must have streaming capability 1.4 issue resolution mode to UVC driver and check result 1.5 initialize memory buffer mapping from kernel to user mode
2. enumerate frame interval to set frame rate
3. start video capture processes

Parameters

void	*pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nEP0Width width of endpoint1(color) resolution
int	nEP0Height height of endpoint1(color) resolution
bool	bEP0MJPG endpoint1 output is MJPEG ?
int	*pFPS input frame rate setting

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.80 EtronDI_OpenDevice2()

```
int EtronDI_OpenDevice2 (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nEP0Width,
            int nEP0Height,
            bool bEP0MJPG,
            int nEP1Width,
            int nEP1Height,
            DEPTH_TRANSFER_CTRL dtc = DEPTH_IMG_NON_TRANSFER,
```

```
            bool bIsOutputRGB24 = false,
            void * phWndNotice = 0,
            int * pFPS = 0,
            CONTROL_MODE cm = IMAGE_SN_NONSYNC )
```

the implement layer to open Etron camera device by V4L2(https://en.wikipedia.org/wiki/Video4↩Linux), can open color and depth at one time call, do functions as below,

1. initialize the USB device by V4L2 protocol 1.1 query device v4l2 capability 1.2 must have video capability 1.3 must have streaming capability 1.4 issue resolution mode to UVC driver and check result 1.5 initialize memory buffer mapping from kernel to user mode
2. enumerate frame interval to set frame rate
3. start video capture processes

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nEP0Width width of endpoint1(color) resolution
int	nEP0Height height of endpoint1(color) resolution
bool	bEP0MJPG endpoint1 output is MJPEG ?
int	nEP1Width width of endpoint2(depth) resolution
int	nEP1Height height of endpoint2(depth) resolution
DEPTH_TRANSFER_CTRL	dtc depth image output transfer

1. default is transferred to color(DEPTH_IMG_COLORFUL_TRANSFER) by calling from EtronDI_OpenDevice()
2. DEPTH_IMG_GRAY_TRANSFER : transfer to gray
3. DEPTH_IMG_NON_TRANSFER : no transfer

Parameters

bool	bIsOutputRGB24 output color image is RGB format
void	∗phWndNotice reserved, not use
int	∗pFPS input frame rate setting
CONTROL_MODE	cm reserved, not use

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.81 EtronDI_OpenDeviceMBL()

```
int EtronDI_OpenDeviceMBL (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nEP0Width,
```

```
int nEP0Height,
bool bEP0MJPG,
int nEP1Width,
int nEP1Height,
DEPTH_TRANSFER_CTRL dtc = DEPTH_IMG_NON_TRANSFER,
bool bIsOutputRGB24 = false,
void * phWndNotice = 0,
int * pFPS = 0,
CONTROL_MODE cm = IMAGE_SN_NONSYNC )
```

the implement layer to open Multiple Base Line Etron camera device by V4L2(https://en.wikipedia.org/wiki/Video4Linux), can open color and depth at one time call, do functions as below,

1. initialize the USB device by V4L2 protocol 1.1 query device v4l2 capability 1.2 must have video capability 1.3 must have streaming capability 1.4 issue resolution mode to UVC driver and check result 1.5 initialize memory buffer mapping from kernel to user mode
2. enumerate frame interval to set frame rate
3. start video capture processes

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nEP0Width width of endpoint1(color) resolution
int	nEP0Height height of endpoint1(color) resolution
bool	bEP0MJPG endpoint1 output is MJPEG ?
int	nEP1Width width of endpoint2(depth) resolution
int	nEP1Height height of endpoint2(depth) resolution
DEPTH_TRANSFER_CTRL	dtc depth image output transfer

1. default is transferred to color(DEPTH_IMG_COLORFUL_TRANSFER) by calling from EtronDI_OpenDevice()
2. DEPTH_IMG_GRAY_TRANSFER : transfer to gray
3. DEPTH_IMG_NON_TRANSFER : no transfer

Parameters

bool	bIsOutputRGB24 output color image is RGB format
void	∗phWndNotice reserved, not use
int	∗pFPS input frame rate setting
CONTROL_MODE	cm reserved, not use

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.82 EtronDI_PutSRB()

```
int EtronDI_PutSRB (
            void * pSrbHandle,
            srb_packet_s * pPacket )
```

Put Packet to SRB.

Parameters

void	*pSrbHandle pointer to SRB class
packet_s	*pPacket Input Packet

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.83 EtronDI_ReadCmdFiFo()

```
EtronDI_ReadCmdFiFo (
            int FileDescrption,
            unsigned char * pBuf,
            int len )
```

Read Cmd FiFo function.

Parameters

int	FileDescrption File description
unsigned	char *pCmd Point to the cmd buffer
int	lenIndicate the cmd lemgth.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.84 EtronDI_ReadFlashData()

```
int EtronDI_ReadFlashData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            FLASH_DATA_TYPE fdt,
            BYTE * pBuffer,
            unsigned long int BufferLength,
            unsigned long int * pActualLength )
```

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

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
FLASH_DATA_TYPE	fdt segment type of flash be read
BYTE	∗pBuffer buffer to store firmware code
unsigned	long int BufferLength input buffer length
unsigned	long int ∗pActualLength actual length has written to pBuffer

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.85 EtronDI_RefreshDevice()

```
int EtronDI_RefreshDevice (
            void * pHandleEtronDI )
```

refresh all Etron UVC devices

Parameters

void	∗pHandleEtronDI CEtronDI handler

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.86 EtronDI_Release()

```
void EtronDI_Release (
            void ** ppHandleEtronDI )
```

release resource that EtronDI_Init had allocated

Parameters

void	∗∗ppHandleEtronDI array of CEtronDI class handlers

Returns

none

3.1.2.87 EtronDI_ReleaseFlexibleGyro()

```
int EtronDI_ReleaseFlexibleGyro (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

gyro sensor release function

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.88 EtronDI_ReleaseHidGyro()

```
int EtronDI_ReleaseHidGyro (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo )
```

gyro sensor release function

Parameters

*void**	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.89 EtronDI_ReleaseSwPostProc()

```
int EtronDI_ReleaseSwPostProc (
            void ** handle )
```

release a software post process class

Parameters

*void***	handle handle pointer to this software post process class

Returns

success: EtronDI_OK, others: see eSPDI_ErrCode.h

3.1.2.90 EtronDI_ReleaseUAC()

```
int EtronDI_ReleaseUAC (
            void  )
```

UAC inital function.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.91 EtronDI_ResizeImgToHalf()

```
int EtronDI_ResizeImgToHalf (
            EtronDIImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dst,
            int len )
```

Resize the image to half.

Parameters

EtronDIImageType::Value	mgType Image Type
int	width image width
int	height image height
unsigned	char *src image source
unsigned	char *dst image desteration
int	len desteration buffer length

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.92 EtronDI_RGB2BMP()

```
int EtronDI_RGB2BMP (
            char * filename,
            int width,
            int height,
            unsigned char * data )
```

RGB to BMP.

Parameters

∗filename	Ouput BMP file name
int	width image width
int	height image height
∗data	input RGB buffer.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.93 EtronDI_RotateImg180()

```
int EtronDI_RotateImg180 (
            EtronDIImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dstBuf,
            int len )
```

Rotate the image to 180 degree.

Parameters

EtronDIImageType::Value	mgType Image Type
int	width image width
int	height image height
unsigned	char ∗src image source
unsigned	char ∗dstBuf image desteration
int	len desteration buffer length

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.94 EtronDI_RotateImg90()

```
int EtronDI_RotateImg90 (
            EtronDIImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dstBuf,
            int len,
            bool clockwise )
```

Rotate the image to 90 degree.

Parameters

EtronDIImageType::Value	mgType Image Type
int	width image width
int	height image height
unsigned	char *src image source
unsigned	char *dstBuf image desteration
int	len desteration buffer length
bClockwise,false	not supported.
bOpencv	useage, not supported.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.95 EtronDI_SaveLutData()

```
int EtronDI_SaveLutData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            const char * filename )
```

Save LUT parameters in the specified file.

Parameters

*void**	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char* filename output LUT file name

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.96 EtronDI_SelectDevice()

```
int EtronDI_SelectDevice (
            void * pHandleEtronDI,
            int dev_index )
```

do not support currently

Returns

ETronDI_NotSupport

3.1.2.97 EtronDI_SetAutoExposureMode()

```
int EtronDI_SetAutoExposureMode (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short mode )
```

Setup Auto Exposure Mode.

Parameters

void∗	pHandleEtronDI CEtronDI handler.
PDEVSELINFO	pDevSelInfo pointer of device select index.
unsigned	short mode The setup mode value. 0: Average, 1: Left (or Front) camera, 2: Right (or Back) camera

Returns

success: EtronDI_OK, others:eSPDI_def.h

3.1.2.98 EtronDI_SetColorGain()

```
int EtronDI_SetColorGain (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float fGainR,
            float fGainG,
            float fGainB )
```

set color gain of ISP

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nSensorMode which sensor(sensor A, B or Both) to get A is 0, B is 1, Both is 2
float	fGainR Red channel color gain value
float	fGainG Green channel color gain value
float	fGainB Blue channel color gain value

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.99 EtronDI_SetControlCounterMode()

```
int EtronDI_SetControlCounterMode (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned char nValue )
```

enable or disable interleave function

Parameters

pHandleEtronDI	the pointer to the initilized EtronDI SDK instance
pDevSelInfo	pointer of device select index
nValue	0: Frame Counter Mode, 1: Serial Counter Mode,

Returns

success: EtronDI_OK, others: see eSPDI_ErrCode.h

3.1.2.100 EtronDI_SetCTPropVal()

```
int EtronDI_SetCTPropVal (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nId,
            long int nValue )
```

set camera terminal property values By v4l2_control to set

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nId specifies the member of the property set see CT Property ID defined in eSPDI_def.h
long	int nValue CT property value to set

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.101 EtronDI_SetCurrentIRValue()

```
t EtronDI_SetCurrentIRValue (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short nValue )
```

set infrared radiation(IR) value of PUMA type IC

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short nValue 1 byte IR value to set

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.102 EtronDI_SetDepthDataType()

```
EtronDI_SetDepthDataType (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short nValue )
```

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

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short nValue depth data type you want to set, see ETronDI_DEPTH_DATA_xxx in eSPDI_def.h

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.103 EtronDI_SetExposureTime()

```
int EtronDI_SetExposureTime (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float fExpTimeMS )
```

set exposure time of ISP sensor setting the target sensor type was set in EtronDI_SetSensorTypeName()

EtronDI_SetExposureTime(void ∗pHandleEtronDI, PDEVSELINFO pDevSelInfo, int nSensorMode, float fExp↩ TimeMS)

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nSensorMode which sensor(sensor A, B or Both) to set A is 0, B is 1, Both is 2
float	fExpTimeMS pointer of setting exposure time in millisecond check sensor spec for detailed setting, we need pixel clock, pixel per line, V blank and exposure line

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.104 EtronDI_SetFWRegister()

```
int EtronDI_SetFWRegister (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short address,
            unsigned short nValue,
            int flag )
```

set firmware register value

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short address register address
unsigned	short nValue register value to set
int	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: EtronDI_OK, others: see eSPDI_def.h

3.1.2.105 EtronDI_SetGlobalGain()

```
int EtronDI_SetGlobalGain (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float fGlobalGain )
```

set global gain of ISP sensor setting the target sensor type was set in EtronDI_SetSensorTypeName()

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nSensorMode which sensor(sensor A, B or Both) to get A is 0, B is 1, Both is 2
float	fGlobalGain pointer of global gain value

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.106 EtronDI_SetHWRegister()

```
int EtronDI_SetHWRegister (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short address,
            unsigned short nValue,
            int flag )
```

set hardware register

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short address register address
unsigned	short nValue register value to set
int	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: EtronDI_OK, others: see eSPDI_ErrCode.h

3.1.2.107 EtronDI_SetIRMode()

```
EtronDI_SetIRMode (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short nValue )
```

enable or disable IRs

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short nValue 8 bit definition as below to turn on/off IR D[7:4]: Reserved D3: Channel 3 D2: Channel 2 D1: Channel 1 D0: Channel 0 1: Enable Channel 0: Disable Channel If want to control ch0 and ch1, ubMode[3:0] must set to 0x03

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.108 EtronDI_SetLogData()

```
int EtronDI_SetLogData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            int index )
```

set log data to flash

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer log data to set
int	BufferLength buffer length, must be 4096
int	∗pActualLength always return 4096
int	index index to identify log data for corresponding depth

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.109 EtronDI_SetPidVid()

```
int EtronDI_SetPidVid (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pPidBuf,
            unsigned short * pVidBuf )
```

set PID and VID to device

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short ∗pPidBuf 4 byte PID value buffer to set
unsigned	short ∗pVidBuf 4 byte VID value buffer to set

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.110 EtronDI_SetPUPropVal()

```
int EtronDI_SetPUPropVal (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nId,
            long int nValue )
```

set processing unit property value by v4l2_control to set processing unit(PU) property value

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nId specifies the member of the property set see PU Property ID defined in eSPDI_def.h
int	nValue value to set

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.111 EtronDI_SetRectifyTable()

```
int EtronDI_SetRectifyTable (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            int index )
```

set rectify values to flash

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer rectify values to set
int	BufferLength bufer length, must be 1024
int	∗pActualLength always return 1024
int	index index(from 0 ∼ 9) to identify rectify table for corresponding depth

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.112 EtronDI_SetSensorRegister()

```
int EtronDI_SetSensorRegister (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int nId,
            unsigned short address,
            unsigned short nValue,
            int flag,
            SENSORMODE_INFO SensorMode )
```

set sensor register value

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nId sensor slave address see Videodevice.h for sensor slave address setting
unsigned	short address register address
unsigned	short nValue value to set
int	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
SENSORMODE_INFO	SensorMode sensor mode(sensor A, B or Both) A is 0, B is 1, Both is 2

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.113 EtronDI_SetSensorTypeName()

```
int EtronDI_SetSensorTypeName (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            SENSOR_TYPE_NAME stn )
```

set the sensor type you want to work on

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
SENSOR_TYPE_NAME	stn which sensor you want to work on

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.114 EtronDI_SetSerialNumber()

```
int EtronDI_SetSerialNumber (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pData,
            int nLen )
```

set serial number to device

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗pData pointer of buffer to store serial number, it is WildChar
int	nLen pData length in byte

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.115 EtronDI_Setup_v4l2_requestbuffers()

```
int EtronDI_Setup_v4l2_requestbuffers (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            int cnt )
```

Setup v4l2 request buffers, default = 4.

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
int	cnt Should be >= 0

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.116 EtronDI_SetupBlock()

```
int EtronDI_SetupBlock (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            bool enable )
```

get color or depth pin image by issuing V4L2's IOCTL to get frame data

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
bool	enable Enable the Blocking mode or not)

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.117 EtronDI_SetupHidGyro()

```
int EtronDI_SetupHidGyro (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pCmdBuf,
            int cmdlength )
```

getting gyro data function

Parameters

void∗	pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char ∗pGyroData pointer of Gyro Data Buffer.
int	length Input buffer Length, shoul

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.118 EtronDI_SetUserData()

```
int EtronDI_SetUserData (
            void * pHandleEtronDI,
```

```
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            USERDATA_SECTION_INDEX usi )
```

set user data to flash

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer user buffer data to set
int	BufferLength buffer length to write
USERDATA_SECTION_INDEX	usi which user section data to set

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.119 EtronDI_SetYOffset()

```
int EtronDI_SetYOffset (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            int index )
```

set Y offset values

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer buffer data to set
int	BufferLength buffer length
int	∗pActualLength always return 256
int	index index value to file ID 30

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.120 EtronDI_SetZDTable()

```
int EtronDI_SetZDTable (
            void * pHandleEtronDI,
```

```
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            PZDTABLEINFO pZDTableInfo )
```

set disparity and Z values to flash

Parameters

void	∗pHandleEtronDI CEtronDI handler
PDEVSELINFO	pDevSelInfo pointer of device select index
BYTE	∗buffer ZD values to set
int	BufferLength corresponding length of ZD table in buffer
int	∗pActualLength buffer lenth written to flash, should be same as BufferLength
PZDTABLEINFO	pZDTableInfo index and depth type of this ZD

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.121 EtronDI_SwitchBaseline()

```
int EtronDI_SwitchBaseline (
            int index )
```

Swich the baseline index.

Parameters

int	index Baseline index 1: 30 mm 2: 60 mm 3: 150 mm

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.122 EtronDI_WriteCmdFiFo()

```
int EtronDI_WriteCmdFiFo (
            int FileDescrption,
            unsigned char * pCmd,
            int len )
```

Write Cmd FiFo function.

Parameters

int	FileDescrption File description
unsigned	char *pCmd Point to the cmd buffer
int	lenIndicate the cmd lemgth.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.123 EtronDI_WriteFlashData()

```
int EtronDI_WriteFlashData (
            void * pHandleEtronDI,
            PDEVSELINFO pDevSelInfo,
            FLASH_DATA_TYPE fdt,
            BYTE * pBuffer,
            unsigned long int BufferLength,
            bool bIsDataVerify,
            KEEP_DATA_CTRL kdc )
```

write firmware code(.bin) to flash The firmware code is the combination of boot loader, firmware body and plug-in data, also can keep original functions(Serial Number, Sensor Position, RectificationTable, ZD Table and CalibrationLog) on camera flash by KEEP_DATA_CTRL control

Parameters

void	*pHandleEtronDI CEronDI class
PDEVSELINFO	pDevSelInfo pointer of device select index
FLASH_DATA_TYPE	fdt segment type of flash be wrote
BYTE	*pBuffer buffer of firmware code
unsigned	long int BufferLength Buffer length to be wrote
BOOL	bIsDataVerify write data verification flag, if true this function will read data again and do a byte to byte comparison
KEEP_DATA_CTRL	kdc keep function flags

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.124 EtronDI_WriteWaveEnd()

```
int EtronDI_WriteWaveEnd (
            int fd,
            size_t length )
```

Modified Wave Header.

Parameters

int	fd wave file descript.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.1.2.125 EtronDI_WriteWaveHeader()

```
int EtronDI_WriteWaveHeader (
            int fd )
```

Write Wave Header.

Parameters

int	fd wave file descript.

Returns

success: EtronDI_OK, others: see eSPDI_def.h

3.2 eSPDI_source/eSPDI_def.h File Reference

main data structure, variable and macro definition and error definition Copyright: This file copyright (C) 2017 by

3.2.1 Detailed Description

main data structure, variable and macro definition and error definition Copyright: This file copyright (C) 2017 by

eYs3D an Etron company