eYs3D Linux SDK

v5.0.0.5

Generated by Doxygen 1.8.13

Contents

Chapter 1

Introduction

This document describes the usage of eYs3D Linux SDK

What's inside the SDK

Table 1.1 File List

Folder	Filename	Description
bin	All files	sample executables on Linux platform
console_tester	All files	a console programm demonstrating how to use the APIs defined in eSPDI.h
cfg	All files	configration files
eSPDI	eSPDI.h	functions definitions
	eSPDI_def.h	error/data type definitions
	eSPDI_↩ version.h	SDK version declaration header
DMPreview	All files	a sample project demonstrating how to open multiple devices in an application

Chapter 2

Background

libjpeg-turbo is a JPEG image codec that uses SIMD instructions to accelerate baseline JPEG compression and decompression on x86, x86-64, ARM, PowerPC, and MIPS systems, as well as progressive JPEG compression on x86 and x86-64 systems. On such systems, libjpeg-turbo is generally 2-6x as fast as libjpeg, all else being equal. On other types of systems, libjpeg-turbo can still outperform libjpeg by a significant amount, by virtue of its highly-optimized Huffman coding routines. In many cases, the performance of libjpeg-turbo rivals that of proprietary high-speed JPEG codecs.

libjpeg-turbo implements both the traditional libjpeg API as well as the less powerful but more straightforward TurboJPEG API. libjpeg-turbo also features colorspace extensions that allow it to compress from/decompress to 32-bit and big-endian pixel buffers (RGBX, XBGR, etc.), as well as a full-featured Java interface.

libjpeg-turbo was originally based on libjpeg/SIMD, an MMX-accelerated derivative of libjpeg v6b developed by Miyasaka Masaru. The TigerVNC and VirtualGL projects made numerous enhancements to the codec in 2009, and in early 2010, libjpeg-turbo spun off into an independent project, with the goal of making high-speed JPEG compression/decompression technology available to a broader range of users and developers.

License

libjpeg-turbo is covered by three compatible BSD-style open source licenses. Refer to LICENSE.md for a roll-up of license terms.

Building libjpeg-turbo

Refer to BUILDING.md for complete instructions.

Using libjpeg-turbo

libjpeg-turbo includes two APIs that can be used to compress and decompress JPEG images:

- **TurboJPEG API**
  This API provides an easy-to-use interface for compressing and decompressing JPEG images in memory. It also provides some functionality that would not be straightforward to achieve using the underlying libjpeg API, such as generating planar YUV images and performing multiple simultaneous lossless transforms on an image. The Java interface for libjpeg-turbo is written on top of the TurboJPEG API. The TurboJPEG API is recommended for first-time users of libjpeg-turbo. Refer to `tjexample.c` and `TJExample.java` for examples of its usage and to `http://libjpeg-turbo.org/Documentation/Documentation` for API documentation.
- **libjpeg API**
  This is the de facto industry-standard API for compressing and decompressing JPEG images. It is more difficult to use than the TurboJPEG API but also more powerful. The libjpeg API implementation in libjpeg-turbo is both API/ABI-compatible and mathematically compatible with libjpeg v6b. It can also optionally be configured to be API/ABI-compatible with libjpeg v7 and v8 (see below.) Refer to `cjpeg.c` and `djpeg.c` for examples of its usage and to `libjpeg.txt` for API documentation.

There is no significant performance advantage to either API when both are used to perform similar operations.

Colorspace Extensions

libjpeg-turbo includes extensions that allow JPEG images to be compressed directly from (and decompressed directly to) buffers that use BGR, BGRX, RGBX, XBGR, and XRGB pixel ordering. This is implemented with ten new colorspace constants:

```
JCS_EXT_RGB   /* red/green/blue */
JCS_EXT_RGBX  /* red/green/blue/x */
JCS_EXT_BGR   /* blue/green/red */
JCS_EXT_BGRX  /* blue/green/red/x */
JCS_EXT_XBGR  /* x/blue/green/red */
JCS_EXT_XRGB  /* x/red/green/blue */
JCS_EXT_RGBA  /* red/green/blue/alpha */
JCS_EXT_BGRA  /* blue/green/red/alpha */
JCS_EXT_ABGR  /* alpha/blue/green/red */
JCS_EXT_ARGB  /* alpha/red/green/blue */
```

Setting `cinfo.in_color_space` (compression) or `cinfo.out_color_space` (decompression) to one of these values will cause libjpeg-turbo to read the red, green, and blue values from (or write them to) the appropriate position in the pixel when compressing from/decompressing to an RGB buffer.

Your application can check for the existence of these extensions at compile time with:

```
#ifdef JCS_EXTENSIONS
```

At run time, attempting to use these extensions with a libjpeg implementation that does not support them will result in a "Bogus input colorspace" error. Applications can trap this error in order to test whether run-time support is available for the colorspace extensions.

When using the RGBX, BGRX, XBGR, and XRGB colorspaces during decompression, the X byte is undefined, and in order to ensure the best performance, libjpeg-turbo can set that byte to whatever value it wishes. If an application expects the X byte to be used as an alpha channel, then it should specify `JCS_EXT_RGBA`, `JCS_EXT_BGRA`, `JCS_EXT_ABGR`, or `JCS_EXT_ARGB`. When these colorspace constants are used, the X byte is guaranteed to be 0xFF, which is interpreted as opaque.

Your application can check for the existence of the alpha channel colorspace extensions at compile time with:

```
#ifdef JCS_ALPHA_EXTENSIONS
```

`jcstest.c`, located in the libjpeg-turbo source tree, demonstrates how to check for the existence of the colorspace extensions at compile time and run time.

libjpeg v7 and v8 API/ABI Emulation

With libjpeg v7 and v8, new features were added that necessitated extending the compression and decompression structures. Unfortunately, due to the exposed nature of those structures, extending them also necessitated breaking backward ABI compatibility with previous libjpeg releases. Thus, programs that were built to use libjpeg v7 or v8 did not work with libjpeg-turbo, since it is based on the libjpeg v6b code base. Although libjpeg v7 and v8 are not as widely used as v6b, enough programs (including a few Linux distros) made the switch that there was a demand to emulate the libjpeg v7 and v8 ABIs in libjpeg-turbo. It should be noted, however, that this feature was added primarily so that applications that had already been compiled to use libjpeg v7+ could take advantage of accelerated baseline JPEG encoding/decoding without recompiling. libjpeg-turbo does not claim to support all of the libjpeg v7+ features, nor to produce identical output to libjpeg v7+ in all cases (see below.)

By passing an argument of `-DWITH_JPEG7=1` or `-DWITH_JPEG8=1` to `cmake`, you can build a version of libjpeg-turbo that emulates the libjpeg v7 or v8 ABI, so that programs that are built against libjpeg v7 or v8 can be run with libjpeg-turbo. The following section describes which libjpeg v7+ features are supported and which aren't.

Support for libjpeg v7 and v8 Features

Fully supported

- **libjpeg API: IDCT scaling extensions in decompressor**
  libjpeg-turbo supports IDCT scaling with scaling factors of 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 7/8, 9/8, 5/4, 11/8, 3/2, 13/8, 7/4, 15/8, and 2/1 (only 1/4 and 1/2 are SIMD-accelerated.)
- **libjpeg API: Arithmetic coding**
- **libjpeg API: In-memory source and destination managers**
  See notes below.
- **cjpeg: Separate quality settings for luminance and chrominance**
  Note that the libpjeg v7+ API was extended to accommodate this feature only for convenience purposes. It has always been possible to implement this feature with libjpeg v6b (see rdswitch.c for an example.)
- **cjpeg: 32-bit BMP support**
- **cjpeg: `-rgb` option**
- **jpegtran: Lossless cropping**
- **jpegtran: `-perfect` option**
- **jpegtran: Forcing width/height when performing lossless crop**
- **rdjpgcom: `-raw` option**
- **rdjpgcom: Locale awareness**

Not supported

NOTE: As of this writing, extensive research has been conducted into the usefulness of DCT scaling as a means of data reduction and SmartScale as a means of quality improvement. The reader is invited to peruse the research at `http://www.libjpeg-turbo.org/About/SmartScale` and draw his/her own conclusions, but it is the general belief of our project that these features have not demonstrated sufficient usefulness to justify inclusion in libjpeg-turbo.

- **libjpeg API: DCT scaling in compressor**
  `cinfo.scale_num` and `cinfo.scale_denom` are silently ignored. There is no technical reason why DCT scaling could not be supported when emulating the libjpeg v7+ API/ABI, but without the SmartScale extension (see below), only scaling factors of 1/2, 8/15, 4/7, 8/13, 2/3, 8/11, 4/5, and 8/9 would be available, which is of limited usefulness.

- **libjpeg API: SmartScale**
  `cinfo.block_size` is silently ignored. SmartScale is an extension to the JPEG format that allows for DCT block sizes other than 8x8. Providing support for this new format would be feasible (particularly without full acceleration.) However, until/unless the format becomes either an official industry standard or, at minimum, an accepted solution in the community, we are hesitant to implement it, as there is no sense of whether or how it might change in the future. It is our belief that SmartScale has not demonstrated sufficient usefulness as a lossless format nor as a means of quality enhancement, and thus our primary interest in providing this feature would be as a means of supporting additional DCT scaling factors.
- **libjpeg API: Fancy downsampling in compressor**
  `cinfo.do_fancy_downsampling` is silently ignored. This requires the DCT scaling feature, which is not supported.
- **jpegtran: Scaling**
  This requires both the DCT scaling and SmartScale features, which are not supported.
- **Lossless RGB JPEG files**
  This requires the SmartScale feature, which is not supported.

What About libjpeg v9?

libjpeg v9 introduced yet another field to the JPEG compression structure (`color_transform`), thus making the ABI backward incompatible with that of libjpeg v8. This new field was introduced solely for the purpose of supporting lossless SmartScale encoding. Furthermore, there was actually no reason to extend the API in this manner, as the color transform could have just as easily been activated by way of a new JPEG colorspace constant, thus preserving backward ABI compatibility.

Our research (see link above) has shown that lossless SmartScale does not generally accomplish anything that can't already be accomplished better with existing, standard lossless formats. Therefore, at this time it is our belief that there is not sufficient technical justification for software projects to upgrade from libjpeg v8 to libjpeg v9, and thus there is not sufficient technical justification for us to emulate the libjpeg v9 ABI.

In-Memory Source/Destination Managers

By default, libjpeg-turbo 1.3 and later includes the `jpeg_mem_src()` and `jpeg_mem_dest()` functions, even when not emulating the libjpeg v8 API/ABI. Previously, it was necessary to build libjpeg-turbo from source with libjpeg v8 API/ABI emulation in order to use the in-memory source/destination managers, but several projects requested that those functions be included when emulating the libjpeg v6b API/ABI as well. This allows the use of those functions by programs that need them, without breaking ABI compatibility for programs that don't, and it allows those functions to be provided in the "official" libjpeg-turbo binaries.

Those who are concerned about maintaining strict conformance with the libjpeg v6b or v7 API can pass an argument of `-DWITH_MEM_SRCDST=0` to `cmake` prior to building libjpeg-turbo. This will restore the pre-1.3 behavior, in which `jpeg_mem_src()` and `jpeg_mem_dest()` are only included when emulating the libjpeg v8 API/ABI.

On Un∗x systems, including the in-memory source/destination managers changes the dynamic library version from 62.2.0 to 62.3.0 if using libjpeg v6b API/ABI emulation and from 7.2.0 to 7.3.0 if using libjpeg v7 API/ABI emulation.

Note that, on most Un∗x systems, the dynamic linker will not look for a function in a library until that function is actually used. Thus, if a program is built against libjpeg-turbo 1.3+ and uses `jpeg_mem_src()` or `jpeg_mem_dest()`, that program will not fail if run against an older version of libjpeg-turbo or against libjpeg v7- until the program actually tries to call `jpeg_mem_src()` or `jpeg_mem_dest()`. Such is not the case on Windows. If a program is built against the libjpeg-turbo 1.3+ DLL and uses `jpeg_mem_src()` or `jpeg_mem_dest()`, then it must use the libjpeg-turbo 1.3+ DLL at run time.

Both cjpeg and djpeg have been extended to allow testing the in-memory source/destination manager functions. See their respective man pages for more details.

Mathematical Compatibility

For the most part, libjpeg-turbo should produce identical output to libjpeg v6b. The one exception to this is when using the floating point DCT/IDCT, in which case the outputs of libjpeg v6b and libjpeg-turbo can differ for the following reasons:

- The SSE/SSE2 floating point DCT implementation in libjpeg-turbo is ever so slightly more accurate than the implementation in libjpeg v6b, but not by any amount perceptible to human vision (generally in the range of 0.01 to 0.08 dB gain in PNSR.)
- When not using the SIMD extensions, libjpeg-turbo uses the more accurate (and slightly faster) floating point IDCT algorithm introduced in libjpeg v8a as opposed to the algorithm used in libjpeg v6b. It should be noted, however, that this algorithm basically brings the accuracy of the floating point IDCT in line with the accuracy of the slow integer IDCT. The floating point DCT/IDCT algorithms are mainly a legacy feature, and they do not produce significantly more accuracy than the slow integer algorithms (to put numbers on this, the typical difference in PNSR between the two algorithms is less than 0.10 dB, whereas changing the quality level by 1 in the upper range of the quality scale is typically more like a 1.0 dB difference.)
- If the floating point algorithms in libjpeg-turbo are not implemented using SIMD instructions on a particular platform, then the accuracy of the floating point DCT/IDCT can depend on the compiler settings.

While libjpeg-turbo does emulate the libjpeg v8 API/ABI, under the hood it is still using the same algorithms as libjpeg v6b, so there are several specific cases in which libjpeg-turbo cannot be expected to produce the same output as libjpeg v8:

- When decompressing using scaling factors of 1/2 and 1/4, because libjpeg v8 implements those scaling algorithms differently than libjpeg v6b does, and libjpeg-turbo's SIMD extensions are based on the libjpeg v6b behavior.
- When using chrominance subsampling, because libjpeg v8 implements this with its DCT/IDCT scaling algorithms rather than with a separate downsampling/upsampling algorithm. In our testing, the subsampled/upsampled output of libjpeg v8 is less accurate than that of libjpeg v6b for this reason.
- When decompressing using a scaling factor $>$ 1 and merged (AKA "non-fancy" or "non-smooth") chrominance upsampling, because libjpeg v8 does not support merged upsampling with scaling factors $>$ 1.

Performance Pitfalls

Restart Markers

The optimized Huffman decoder in libjpeg-turbo does not handle restart markers in a way that makes the rest of the libjpeg infrastructure happy, so it is necessary to use the slow Huffman decoder when decompressing a JPEG image that has restart markers. This can cause the decompression performance to drop by as much as 20%, but the performance will still be much greater than that of libjpeg. Many consumer packages, such as Photoshop, use restart markers when generating JPEG images, so images generated by those programs will experience this issue.

Fast Integer Forward DCT at High Quality Levels

The algorithm used by the SIMD-accelerated quantization function cannot produce correct results whenever the fast integer forward DCT is used along with a JPEG quality of 98-100. Thus, libjpeg-turbo must use the non-SIMD quantization function in those cases. This causes performance to drop by as much as 40%. It is therefore strongly advised that you use the slow integer forward DCT whenever encoding images with a JPEG quality of 98 or higher.

Memory Debugger Pitfalls

Valgrind and Memory Sanitizer (MSan) can generate false positives (specifically, incorrect reports of uninitialized memory accesses) when used with libjpeg-turbo's SIMD extensions. It is generally recommended that the SIMD extensions be disabled, either by passing an argument of `-DWITH_SIMD=0` to `cmake` when configuring the build or by setting the environment variable `JSIMD_FORCENONE` to `1` at run time, when testing libjpeg-turbo with Valgrind, MSan, or other memory debuggers.

Chapter 3

Background

libjpeg-turbo is a JPEG image codec that uses SIMD instructions to accelerate baseline JPEG compression and decompression on x86, x86-64, ARM, PowerPC, and MIPS systems, as well as progressive JPEG compression on x86 and x86-64 systems. On such systems, libjpeg-turbo is generally 2-6x as fast as libjpeg, all else being equal. On other types of systems, libjpeg-turbo can still outperform libjpeg by a significant amount, by virtue of its highly-optimized Huffman coding routines. In many cases, the performance of libjpeg-turbo rivals that of proprietary high-speed JPEG codecs.

libjpeg-turbo implements both the traditional libjpeg API as well as the less powerful but more straightforward TurboJPEG API. libjpeg-turbo also features colorspace extensions that allow it to compress from/decompress to 32-bit and big-endian pixel buffers (RGBX, XBGR, etc.), as well as a full-featured Java interface.

libjpeg-turbo was originally based on libjpeg/SIMD, an MMX-accelerated derivative of libjpeg v6b developed by Miyasaka Masaru. The TigerVNC and VirtualGL projects made numerous enhancements to the codec in 2009, and in early 2010, libjpeg-turbo spun off into an independent project, with the goal of making high-speed JPEG compression/decompression technology available to a broader range of users and developers.

License

libjpeg-turbo is covered by three compatible BSD-style open source licenses. Refer to LICENSE.md for a roll-up of license terms.

Building libjpeg-turbo

Refer to BUILDING.md for complete instructions.

Using libjpeg-turbo

libjpeg-turbo includes two APIs that can be used to compress and decompress JPEG images:

- **TurboJPEG API**
  This API provides an easy-to-use interface for compressing and decompressing JPEG images in memory. It also provides some functionality that would not be straightforward to achieve using the underlying libjpeg API, such as generating planar YUV images and performing multiple simultaneous lossless transforms on an image. The Java interface for libjpeg-turbo is written on top of the TurboJPEG API. The TurboJPEG API is recommended for first-time users of libjpeg-turbo. Refer to `tjexample.c` and `TJExample.java` for examples of its usage and to `http://libjpeg-turbo.org/Documentation/Documentation` for API documentation.
- **libjpeg API**
  This is the de facto industry-standard API for compressing and decompressing JPEG images. It is more difficult to use than the TurboJPEG API but also more powerful. The libjpeg API implementation in libjpeg-turbo is both API/ABI-compatible and mathematically compatible with libjpeg v6b. It can also optionally be configured to be API/ABI-compatible with libjpeg v7 and v8 (see below.) Refer to `cjpeg.c` and `djpeg.c` for examples of its usage and to `libjpeg.txt` for API documentation.

There is no significant performance advantage to either API when both are used to perform similar operations.

Colorspace Extensions

libjpeg-turbo includes extensions that allow JPEG images to be compressed directly from (and decompressed directly to) buffers that use BGR, BGRX, RGBX, XBGR, and XRGB pixel ordering. This is implemented with ten new colorspace constants:

```
JCS_EXT_RGB   /* red/green/blue */
JCS_EXT_RGBX  /* red/green/blue/x */
JCS_EXT_BGR   /* blue/green/red */
JCS_EXT_BGRX  /* blue/green/red/x */
JCS_EXT_XBGR  /* x/blue/green/red */
JCS_EXT_XRGB  /* x/red/green/blue */
JCS_EXT_RGBA  /* red/green/blue/alpha */
JCS_EXT_BGRA  /* blue/green/red/alpha */
JCS_EXT_ABGR  /* alpha/blue/green/red */
JCS_EXT_ARGB  /* alpha/red/green/blue */
```

Setting `cinfo.in_color_space` (compression) or `cinfo.out_color_space` (decompression) to one of these values will cause libjpeg-turbo to read the red, green, and blue values from (or write them to) the appropriate position in the pixel when compressing from/decompressing to an RGB buffer.

Your application can check for the existence of these extensions at compile time with:

```
#ifdef JCS_EXTENSIONS
```

At run time, attempting to use these extensions with a libjpeg implementation that does not support them will result in a "Bogus input colorspace" error. Applications can trap this error in order to test whether run-time support is available for the colorspace extensions.

When using the RGBX, BGRX, XBGR, and XRGB colorspaces during decompression, the X byte is undefined, and in order to ensure the best performance, libjpeg-turbo can set that byte to whatever value it wishes. If an application expects the X byte to be used as an alpha channel, then it should specify `JCS_EXT_RGBA`, `JCS_EXT_BGRA`, `JCS_EXT_ABGR`, or `JCS_EXT_ARGB`. When these colorspace constants are used, the X byte is guaranteed to be 0xFF, which is interpreted as opaque.

Your application can check for the existence of the alpha channel colorspace extensions at compile time with:

```
#ifdef JCS_ALPHA_EXTENSIONS
```

`jcstest.c`, located in the libjpeg-turbo source tree, demonstrates how to check for the existence of the colorspace extensions at compile time and run time.

libjpeg v7 and v8 API/ABI Emulation

With libjpeg v7 and v8, new features were added that necessitated extending the compression and decompression structures. Unfortunately, due to the exposed nature of those structures, extending them also necessitated breaking backward ABI compatibility with previous libjpeg releases. Thus, programs that were built to use libjpeg v7 or v8 did not work with libjpeg-turbo, since it is based on the libjpeg v6b code base. Although libjpeg v7 and v8 are not as widely used as v6b, enough programs (including a few Linux distros) made the switch that there was a demand to emulate the libjpeg v7 and v8 ABIs in libjpeg-turbo. It should be noted, however, that this feature was added primarily so that applications that had already been compiled to use libjpeg v7+ could take advantage of accelerated baseline JPEG encoding/decoding without recompiling. libjpeg-turbo does not claim to support all of the libjpeg v7+ features, nor to produce identical output to libjpeg v7+ in all cases (see below.)

By passing an argument of `-DWITH_JPEG7=1` or `-DWITH_JPEG8=1` to `cmake`, you can build a version of libjpeg-turbo that emulates the libjpeg v7 or v8 ABI, so that programs that are built against libjpeg v7 or v8 can be run with libjpeg-turbo. The following section describes which libjpeg v7+ features are supported and which aren't.

Support for libjpeg v7 and v8 Features

Fully supported

- **libjpeg API: IDCT scaling extensions in decompressor**
  libjpeg-turbo supports IDCT scaling with scaling factors of 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 7/8, 9/8, 5/4, 11/8, 3/2, 13/8, 7/4, 15/8, and 2/1 (only 1/4 and 1/2 are SIMD-accelerated.)
- **libjpeg API: Arithmetic coding**
- **libjpeg API: In-memory source and destination managers**
  See notes below.
- **cjpeg: Separate quality settings for luminance and chrominance**
  Note that the libpjeg v7+ API was extended to accommodate this feature only for convenience purposes. It has always been possible to implement this feature with libjpeg v6b (see rdswitch.c for an example.)
- **cjpeg: 32-bit BMP support**
- **cjpeg: `-rgb` option**
- **jpegtran: Lossless cropping**
- **jpegtran: `-perfect` option**
- **jpegtran: Forcing width/height when performing lossless crop**
- **rdjpgcom: `-raw` option**
- **rdjpgcom: Locale awareness**

Not supported

NOTE: As of this writing, extensive research has been conducted into the usefulness of DCT scaling as a means of data reduction and SmartScale as a means of quality improvement. The reader is invited to peruse the research at `http://www.libjpeg-turbo.org/About/SmartScale` and draw his/her own conclusions, but it is the general belief of our project that these features have not demonstrated sufficient usefulness to justify inclusion in libjpeg-turbo.

- **libjpeg API: DCT scaling in compressor**
  `cinfo.scale_num` and `cinfo.scale_denom` are silently ignored. There is no technical reason why DCT scaling could not be supported when emulating the libjpeg v7+ API/ABI, but without the SmartScale extension (see below), only scaling factors of 1/2, 8/15, 4/7, 8/13, 2/3, 8/11, 4/5, and 8/9 would be available, which is of limited usefulness.

- **libjpeg API: SmartScale**
  `cinfo.block_size` is silently ignored. SmartScale is an extension to the JPEG format that allows for DCT block sizes other than 8x8. Providing support for this new format would be feasible (particularly without full acceleration.) However, until/unless the format becomes either an official industry standard or, at minimum, an accepted solution in the community, we are hesitant to implement it, as there is no sense of whether or how it might change in the future. It is our belief that SmartScale has not demonstrated sufficient usefulness as a lossless format nor as a means of quality enhancement, and thus our primary interest in providing this feature would be as a means of supporting additional DCT scaling factors.
- **libjpeg API: Fancy downsampling in compressor**
  `cinfo.do_fancy_downsampling` is silently ignored. This requires the DCT scaling feature, which is not supported.
- **jpegtran: Scaling**
  This requires both the DCT scaling and SmartScale features, which are not supported.
- **Lossless RGB JPEG files**
  This requires the SmartScale feature, which is not supported.

What About libjpeg v9?

libjpeg v9 introduced yet another field to the JPEG compression structure (`color_transform`), thus making the ABI backward incompatible with that of libjpeg v8. This new field was introduced solely for the purpose of supporting lossless SmartScale encoding. Furthermore, there was actually no reason to extend the API in this manner, as the color transform could have just as easily been activated by way of a new JPEG colorspace constant, thus preserving backward ABI compatibility.

Our research (see link above) has shown that lossless SmartScale does not generally accomplish anything that can't already be accomplished better with existing, standard lossless formats. Therefore, at this time it is our belief that there is not sufficient technical justification for software projects to upgrade from libjpeg v8 to libjpeg v9, and thus there is not sufficient technical justification for us to emulate the libjpeg v9 ABI.

In-Memory Source/Destination Managers

By default, libjpeg-turbo 1.3 and later includes the `jpeg_mem_src()` and `jpeg_mem_dest()` functions, even when not emulating the libjpeg v8 API/ABI. Previously, it was necessary to build libjpeg-turbo from source with libjpeg v8 API/ABI emulation in order to use the in-memory source/destination managers, but several projects requested that those functions be included when emulating the libjpeg v6b API/ABI as well. This allows the use of those functions by programs that need them, without breaking ABI compatibility for programs that don't, and it allows those functions to be provided in the "official" libjpeg-turbo binaries.

Those who are concerned about maintaining strict conformance with the libjpeg v6b or v7 API can pass an argument of `-DWITH_MEM_SRCDST=0` to `cmake` prior to building libjpeg-turbo. This will restore the pre-1.3 behavior, in which `jpeg_mem_src()` and `jpeg_mem_dest()` are only included when emulating the libjpeg v8 API/ABI.

On Un∗x systems, including the in-memory source/destination managers changes the dynamic library version from 62.2.0 to 62.3.0 if using libjpeg v6b API/ABI emulation and from 7.2.0 to 7.3.0 if using libjpeg v7 API/ABI emulation.

Note that, on most Un∗x systems, the dynamic linker will not look for a function in a library until that function is actually used. Thus, if a program is built against libjpeg-turbo 1.3+ and uses `jpeg_mem_src()` or `jpeg_mem_dest()`, that program will not fail if run against an older version of libjpeg-turbo or against libjpeg v7- until the program actually tries to call `jpeg_mem_src()` or `jpeg_mem_dest()`. Such is not the case on Windows. If a program is built against the libjpeg-turbo 1.3+ DLL and uses `jpeg_mem_src()` or `jpeg_mem_dest()`, then it must use the libjpeg-turbo 1.3+ DLL at run time.

Both cjpeg and djpeg have been extended to allow testing the in-memory source/destination manager functions. See their respective man pages for more details.

Mathematical Compatibility

For the most part, libjpeg-turbo should produce identical output to libjpeg v6b. The one exception to this is when using the floating point DCT/IDCT, in which case the outputs of libjpeg v6b and libjpeg-turbo can differ for the following reasons:

- The SSE/SSE2 floating point DCT implementation in libjpeg-turbo is ever so slightly more accurate than the implementation in libjpeg v6b, but not by any amount perceptible to human vision (generally in the range of 0.01 to 0.08 dB gain in PNSR.)
- When not using the SIMD extensions, libjpeg-turbo uses the more accurate (and slightly faster) floating point IDCT algorithm introduced in libjpeg v8a as opposed to the algorithm used in libjpeg v6b. It should be noted, however, that this algorithm basically brings the accuracy of the floating point IDCT in line with the accuracy of the slow integer IDCT. The floating point DCT/IDCT algorithms are mainly a legacy feature, and they do not produce significantly more accuracy than the slow integer algorithms (to put numbers on this, the typical difference in PNSR between the two algorithms is less than 0.10 dB, whereas changing the quality level by 1 in the upper range of the quality scale is typically more like a 1.0 dB difference.)
- If the floating point algorithms in libjpeg-turbo are not implemented using SIMD instructions on a particular platform, then the accuracy of the floating point DCT/IDCT can depend on the compiler settings.

While libjpeg-turbo does emulate the libjpeg v8 API/ABI, under the hood it is still using the same algorithms as libjpeg v6b, so there are several specific cases in which libjpeg-turbo cannot be expected to produce the same output as libjpeg v8:

- When decompressing using scaling factors of 1/2 and 1/4, because libjpeg v8 implements those scaling algorithms differently than libjpeg v6b does, and libjpeg-turbo's SIMD extensions are based on the libjpeg v6b behavior.
- When using chrominance subsampling, because libjpeg v8 implements this with its DCT/IDCT scaling algorithms rather than with a separate downsampling/upsampling algorithm. In our testing, the subsampled/upsampled output of libjpeg v8 is less accurate than that of libjpeg v6b for this reason.
- When decompressing using a scaling factor $>$ 1 and merged (AKA "non-fancy" or "non-smooth") chrominance upsampling, because libjpeg v8 does not support merged upsampling with scaling factors $>$ 1.

Performance Pitfalls

Restart Markers

The optimized Huffman decoder in libjpeg-turbo does not handle restart markers in a way that makes the rest of the libjpeg infrastructure happy, so it is necessary to use the slow Huffman decoder when decompressing a JPEG image that has restart markers. This can cause the decompression performance to drop by as much as 20%, but the performance will still be much greater than that of libjpeg. Many consumer packages, such as Photoshop, use restart markers when generating JPEG images, so images generated by those programs will experience this issue.

Fast Integer Forward DCT at High Quality Levels

The algorithm used by the SIMD-accelerated quantization function cannot produce correct results whenever the fast integer forward DCT is used along with a JPEG quality of 98-100. Thus, libjpeg-turbo must use the non-SIMD quantization function in those cases. This causes performance to drop by as much as 40%. It is therefore strongly advised that you use the slow integer forward DCT whenever encoding images with a JPEG quality of 98 or higher.

Memory Debugger Pitfalls

Valgrind and Memory Sanitizer (MSan) can generate false positives (specifically, incorrect reports of uninitialized memory accesses) when used with libjpeg-turbo's SIMD extensions. It is generally recommended that the SIMD extensions be disabled, either by passing an argument of `-DWITH_SIMD=0` to `cmake` when configuring the build or by setting the environment variable `JSIMD_FORCENONE` to `1` at run time, when testing libjpeg-turbo with Valgrind, MSan, or other memory debuggers.

Chapter 4

Background

libjpeg-turbo is a JPEG image codec that uses SIMD instructions to accelerate baseline JPEG compression and decompression on x86, x86-64, ARM, PowerPC, and MIPS systems, as well as progressive JPEG compression on x86 and x86-64 systems. On such systems, libjpeg-turbo is generally 2-6x as fast as libjpeg, all else being equal. On other types of systems, libjpeg-turbo can still outperform libjpeg by a significant amount, by virtue of its highly-optimized Huffman coding routines. In many cases, the performance of libjpeg-turbo rivals that of proprietary high-speed JPEG codecs.

libjpeg-turbo implements both the traditional libjpeg API as well as the less powerful but more straightforward TurboJPEG API. libjpeg-turbo also features colorspace extensions that allow it to compress from/decompress to 32-bit and big-endian pixel buffers (RGBX, XBGR, etc.), as well as a full-featured Java interface.

libjpeg-turbo was originally based on libjpeg/SIMD, an MMX-accelerated derivative of libjpeg v6b developed by Miyasaka Masaru. The TigerVNC and VirtualGL projects made numerous enhancements to the codec in 2009, and in early 2010, libjpeg-turbo spun off into an independent project, with the goal of making high-speed JPEG compression/decompression technology available to a broader range of users and developers.

License

libjpeg-turbo is covered by three compatible BSD-style open source licenses. Refer to LICENSE.md for a roll-up of license terms.

Building libjpeg-turbo

Refer to BUILDING.md for complete instructions.

Using libjpeg-turbo

libjpeg-turbo includes two APIs that can be used to compress and decompress JPEG images:

- **TurboJPEG API**
  This API provides an easy-to-use interface for compressing and decompressing JPEG images in memory. It also provides some functionality that would not be straightforward to achieve using the underlying libjpeg API, such as generating planar YUV images and performing multiple simultaneous lossless transforms on an image. The Java interface for libjpeg-turbo is written on top of the TurboJPEG API. The TurboJPEG API is recommended for first-time users of libjpeg-turbo. Refer to `tjexample.c` and `TJExample.java` for examples of its usage and to `http://libjpeg-turbo.org/Documentation/Documentation` for API documentation.
- **libjpeg API**
  This is the de facto industry-standard API for compressing and decompressing JPEG images. It is more difficult to use than the TurboJPEG API but also more powerful. The libjpeg API implementation in libjpeg-turbo is both API/ABI-compatible and mathematically compatible with libjpeg v6b. It can also optionally be configured to be API/ABI-compatible with libjpeg v7 and v8 (see below.) Refer to `cjpeg.c` and `djpeg.c` for examples of its usage and to `libjpeg.txt` for API documentation.

There is no significant performance advantage to either API when both are used to perform similar operations.

Colorspace Extensions

libjpeg-turbo includes extensions that allow JPEG images to be compressed directly from (and decompressed directly to) buffers that use BGR, BGRX, RGBX, XBGR, and XRGB pixel ordering. This is implemented with ten new colorspace constants:

```
JCS_EXT_RGB   /* red/green/blue */
JCS_EXT_RGBX  /* red/green/blue/x */
JCS_EXT_BGR   /* blue/green/red */
JCS_EXT_BGRX  /* blue/green/red/x */
JCS_EXT_XBGR  /* x/blue/green/red */
JCS_EXT_XRGB  /* x/red/green/blue */
JCS_EXT_RGBA  /* red/green/blue/alpha */
JCS_EXT_BGRA  /* blue/green/red/alpha */
JCS_EXT_ABGR  /* alpha/blue/green/red */
JCS_EXT_ARGB  /* alpha/red/green/blue */
```

Setting `cinfo.in_color_space` (compression) or `cinfo.out_color_space` (decompression) to one of these values will cause libjpeg-turbo to read the red, green, and blue values from (or write them to) the appropriate position in the pixel when compressing from/decompressing to an RGB buffer.

Your application can check for the existence of these extensions at compile time with:

```
#ifdef JCS_EXTENSIONS
```

At run time, attempting to use these extensions with a libjpeg implementation that does not support them will result in a "Bogus input colorspace" error. Applications can trap this error in order to test whether run-time support is available for the colorspace extensions.

When using the RGBX, BGRX, XBGR, and XRGB colorspaces during decompression, the X byte is undefined, and in order to ensure the best performance, libjpeg-turbo can set that byte to whatever value it wishes. If an application expects the X byte to be used as an alpha channel, then it should specify `JCS_EXT_RGBA`, `JCS_EXT_BGRA`, `JCS_EXT_ABGR`, or `JCS_EXT_ARGB`. When these colorspace constants are used, the X byte is guaranteed to be 0xFF, which is interpreted as opaque.

Your application can check for the existence of the alpha channel colorspace extensions at compile time with:

```
#ifdef JCS_ALPHA_EXTENSIONS
```

`jcstest.c`, located in the libjpeg-turbo source tree, demonstrates how to check for the existence of the colorspace extensions at compile time and run time.

libjpeg v7 and v8 API/ABI Emulation

With libjpeg v7 and v8, new features were added that necessitated extending the compression and decompression structures. Unfortunately, due to the exposed nature of those structures, extending them also necessitated breaking backward ABI compatibility with previous libjpeg releases. Thus, programs that were built to use libjpeg v7 or v8 did not work with libjpeg-turbo, since it is based on the libjpeg v6b code base. Although libjpeg v7 and v8 are not as widely used as v6b, enough programs (including a few Linux distros) made the switch that there was a demand to emulate the libjpeg v7 and v8 ABIs in libjpeg-turbo. It should be noted, however, that this feature was added primarily so that applications that had already been compiled to use libjpeg v7+ could take advantage of accelerated baseline JPEG encoding/decoding without recompiling. libjpeg-turbo does not claim to support all of the libjpeg v7+ features, nor to produce identical output to libjpeg v7+ in all cases (see below.)

By passing an argument of `-DWITH_JPEG7=1` or `-DWITH_JPEG8=1` to `cmake`, you can build a version of libjpeg-turbo that emulates the libjpeg v7 or v8 ABI, so that programs that are built against libjpeg v7 or v8 can be run with libjpeg-turbo. The following section describes which libjpeg v7+ features are supported and which aren't.

Support for libjpeg v7 and v8 Features

Fully supported

- **libjpeg API: IDCT scaling extensions in decompressor**
  libjpeg-turbo supports IDCT scaling with scaling factors of 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 7/8, 9/8, 5/4, 11/8, 3/2, 13/8, 7/4, 15/8, and 2/1 (only 1/4 and 1/2 are SIMD-accelerated.)
- **libjpeg API: Arithmetic coding**
- **libjpeg API: In-memory source and destination managers**
  See notes below.
- **cjpeg: Separate quality settings for luminance and chrominance**
  Note that the libpjeg v7+ API was extended to accommodate this feature only for convenience purposes. It has always been possible to implement this feature with libjpeg v6b (see rdswitch.c for an example.)
- **cjpeg: 32-bit BMP support**
- **cjpeg: `-rgb` option**
- **jpegtran: Lossless cropping**
- **jpegtran: `-perfect` option**
- **jpegtran: Forcing width/height when performing lossless crop**
- **rdjpgcom: `-raw` option**
- **rdjpgcom: Locale awareness**

Not supported

NOTE: As of this writing, extensive research has been conducted into the usefulness of DCT scaling as a means of data reduction and SmartScale as a means of quality improvement. The reader is invited to peruse the research at `http://www.libjpeg-turbo.org/About/SmartScale` and draw his/her own conclusions, but it is the general belief of our project that these features have not demonstrated sufficient usefulness to justify inclusion in libjpeg-turbo.

- **libjpeg API: DCT scaling in compressor**
  `cinfo.scale_num` and `cinfo.scale_denom` are silently ignored. There is no technical reason why DCT scaling could not be supported when emulating the libjpeg v7+ API/ABI, but without the SmartScale extension (see below), only scaling factors of 1/2, 8/15, 4/7, 8/13, 2/3, 8/11, 4/5, and 8/9 would be available, which is of limited usefulness.

- **libjpeg API: SmartScale**
  `cinfo.block_size` is silently ignored. SmartScale is an extension to the JPEG format that allows for DCT block sizes other than 8x8. Providing support for this new format would be feasible (particularly without full acceleration.) However, until/unless the format becomes either an official industry standard or, at minimum, an accepted solution in the community, we are hesitant to implement it, as there is no sense of whether or how it might change in the future. It is our belief that SmartScale has not demonstrated sufficient usefulness as a lossless format nor as a means of quality enhancement, and thus our primary interest in providing this feature would be as a means of supporting additional DCT scaling factors.
- **libjpeg API: Fancy downsampling in compressor**
  `cinfo.do_fancy_downsampling` is silently ignored. This requires the DCT scaling feature, which is not supported.
- **jpegtran: Scaling**
  This requires both the DCT scaling and SmartScale features, which are not supported.
- **Lossless RGB JPEG files**
  This requires the SmartScale feature, which is not supported.

What About libjpeg v9?

libjpeg v9 introduced yet another field to the JPEG compression structure (`color_transform`), thus making the ABI backward incompatible with that of libjpeg v8. This new field was introduced solely for the purpose of supporting lossless SmartScale encoding. Furthermore, there was actually no reason to extend the API in this manner, as the color transform could have just as easily been activated by way of a new JPEG colorspace constant, thus preserving backward ABI compatibility.

Our research (see link above) has shown that lossless SmartScale does not generally accomplish anything that can't already be accomplished better with existing, standard lossless formats. Therefore, at this time it is our belief that there is not sufficient technical justification for software projects to upgrade from libjpeg v8 to libjpeg v9, and thus there is not sufficient technical justification for us to emulate the libjpeg v9 ABI.

In-Memory Source/Destination Managers

By default, libjpeg-turbo 1.3 and later includes the `jpeg_mem_src()` and `jpeg_mem_dest()` functions, even when not emulating the libjpeg v8 API/ABI. Previously, it was necessary to build libjpeg-turbo from source with libjpeg v8 API/ABI emulation in order to use the in-memory source/destination managers, but several projects requested that those functions be included when emulating the libjpeg v6b API/ABI as well. This allows the use of those functions by programs that need them, without breaking ABI compatibility for programs that don't, and it allows those functions to be provided in the "official" libjpeg-turbo binaries.

Those who are concerned about maintaining strict conformance with the libjpeg v6b or v7 API can pass an argument of `-DWITH_MEM_SRCDST=0` to `cmake` prior to building libjpeg-turbo. This will restore the pre-1.3 behavior, in which `jpeg_mem_src()` and `jpeg_mem_dest()` are only included when emulating the libjpeg v8 API/ABI.

On Un∗x systems, including the in-memory source/destination managers changes the dynamic library version from 62.2.0 to 62.3.0 if using libjpeg v6b API/ABI emulation and from 7.2.0 to 7.3.0 if using libjpeg v7 API/ABI emulation.

Note that, on most Un∗x systems, the dynamic linker will not look for a function in a library until that function is actually used. Thus, if a program is built against libjpeg-turbo 1.3+ and uses `jpeg_mem_src()` or `jpeg_mem_dest()`, that program will not fail if run against an older version of libjpeg-turbo or against libjpeg v7- until the program actually tries to call `jpeg_mem_src()` or `jpeg_mem_dest()`. Such is not the case on Windows. If a program is built against the libjpeg-turbo 1.3+ DLL and uses `jpeg_mem_src()` or `jpeg_mem_dest()`, then it must use the libjpeg-turbo 1.3+ DLL at run time.

Both cjpeg and djpeg have been extended to allow testing the in-memory source/destination manager functions. See their respective man pages for more details.

Mathematical Compatibility

For the most part, libjpeg-turbo should produce identical output to libjpeg v6b. The one exception to this is when using the floating point DCT/IDCT, in which case the outputs of libjpeg v6b and libjpeg-turbo can differ for the following reasons:

- The SSE/SSE2 floating point DCT implementation in libjpeg-turbo is ever so slightly more accurate than the implementation in libjpeg v6b, but not by any amount perceptible to human vision (generally in the range of 0.01 to 0.08 dB gain in PNSR.)
- When not using the SIMD extensions, libjpeg-turbo uses the more accurate (and slightly faster) floating point IDCT algorithm introduced in libjpeg v8a as opposed to the algorithm used in libjpeg v6b. It should be noted, however, that this algorithm basically brings the accuracy of the floating point IDCT in line with the accuracy of the slow integer IDCT. The floating point DCT/IDCT algorithms are mainly a legacy feature, and they do not produce significantly more accuracy than the slow integer algorithms (to put numbers on this, the typical difference in PNSR between the two algorithms is less than 0.10 dB, whereas changing the quality level by 1 in the upper range of the quality scale is typically more like a 1.0 dB difference.)
- If the floating point algorithms in libjpeg-turbo are not implemented using SIMD instructions on a particular platform, then the accuracy of the floating point DCT/IDCT can depend on the compiler settings.

While libjpeg-turbo does emulate the libjpeg v8 API/ABI, under the hood it is still using the same algorithms as libjpeg v6b, so there are several specific cases in which libjpeg-turbo cannot be expected to produce the same output as libjpeg v8:

- When decompressing using scaling factors of 1/2 and 1/4, because libjpeg v8 implements those scaling algorithms differently than libjpeg v6b does, and libjpeg-turbo's SIMD extensions are based on the libjpeg v6b behavior.
- When using chrominance subsampling, because libjpeg v8 implements this with its DCT/IDCT scaling algorithms rather than with a separate downsampling/upsampling algorithm. In our testing, the subsampled/upsampled output of libjpeg v8 is less accurate than that of libjpeg v6b for this reason.
- When decompressing using a scaling factor $>$ 1 and merged (AKA "non-fancy" or "non-smooth") chrominance upsampling, because libjpeg v8 does not support merged upsampling with scaling factors $>$ 1.

Performance Pitfalls

Restart Markers

The optimized Huffman decoder in libjpeg-turbo does not handle restart markers in a way that makes the rest of the libjpeg infrastructure happy, so it is necessary to use the slow Huffman decoder when decompressing a JPEG image that has restart markers. This can cause the decompression performance to drop by as much as 20%, but the performance will still be much greater than that of libjpeg. Many consumer packages, such as Photoshop, use restart markers when generating JPEG images, so images generated by those programs will experience this issue.

Fast Integer Forward DCT at High Quality Levels

The algorithm used by the SIMD-accelerated quantization function cannot produce correct results whenever the fast integer forward DCT is used along with a JPEG quality of 98-100. Thus, libjpeg-turbo must use the non-SIMD quantization function in those cases. This causes performance to drop by as much as 40%. It is therefore strongly advised that you use the slow integer forward DCT whenever encoding images with a JPEG quality of 98 or higher.

Memory Debugger Pitfalls

Valgrind and Memory Sanitizer (MSan) can generate false positives (specifically, incorrect reports of uninitialized memory accesses) when used with libjpeg-turbo's SIMD extensions. It is generally recommended that the SIMD extensions be disabled, either by passing an argument of `-DWITH_SIMD=0` to `cmake` when configuring the build or by setting the environment variable `JSIMD_FORCENONE` to `1` at run time, when testing libjpeg-turbo with Valgrind, MSan, or other memory debuggers.

Chapter 5

Background

libjpeg-turbo is a JPEG image codec that uses SIMD instructions to accelerate baseline JPEG compression and decompression on x86, x86-64, ARM, PowerPC, and MIPS systems, as well as progressive JPEG compression on x86 and x86-64 systems. On such systems, libjpeg-turbo is generally 2-6x as fast as libjpeg, all else being equal. On other types of systems, libjpeg-turbo can still outperform libjpeg by a significant amount, by virtue of its highly-optimized Huffman coding routines. In many cases, the performance of libjpeg-turbo rivals that of proprietary high-speed JPEG codecs.

libjpeg-turbo implements both the traditional libjpeg API as well as the less powerful but more straightforward TurboJPEG API. libjpeg-turbo also features colorspace extensions that allow it to compress from/decompress to 32-bit and big-endian pixel buffers (RGBX, XBGR, etc.), as well as a full-featured Java interface.

libjpeg-turbo was originally based on libjpeg/SIMD, an MMX-accelerated derivative of libjpeg v6b developed by Miyasaka Masaru. The TigerVNC and VirtualGL projects made numerous enhancements to the codec in 2009, and in early 2010, libjpeg-turbo spun off into an independent project, with the goal of making high-speed JPEG compression/decompression technology available to a broader range of users and developers.

License

libjpeg-turbo is covered by three compatible BSD-style open source licenses. Refer to LICENSE.md for a roll-up of license terms.

Building libjpeg-turbo

Refer to BUILDING.md for complete instructions.

Using libjpeg-turbo

libjpeg-turbo includes two APIs that can be used to compress and decompress JPEG images:

- **TurboJPEG API**
  This API provides an easy-to-use interface for compressing and decompressing JPEG images in memory. It also provides some functionality that would not be straightforward to achieve using the underlying libjpeg API, such as generating planar YUV images and performing multiple simultaneous lossless transforms on an image. The Java interface for libjpeg-turbo is written on top of the TurboJPEG API. The TurboJPEG API is recommended for first-time users of libjpeg-turbo. Refer to `tjexample.c` and `TJExample.java` for examples of its usage and to `http://libjpeg-turbo.org/Documentation/Documentation` for API documentation.
- **libjpeg API**
  This is the de facto industry-standard API for compressing and decompressing JPEG images. It is more difficult to use than the TurboJPEG API but also more powerful. The libjpeg API implementation in libjpeg-turbo is both API/ABI-compatible and mathematically compatible with libjpeg v6b. It can also optionally be configured to be API/ABI-compatible with libjpeg v7 and v8 (see below.) Refer to `cjpeg.c` and `djpeg.c` for examples of its usage and to `libjpeg.txt` for API documentation.

There is no significant performance advantage to either API when both are used to perform similar operations.

Colorspace Extensions

libjpeg-turbo includes extensions that allow JPEG images to be compressed directly from (and decompressed directly to) buffers that use BGR, BGRX, RGBX, XBGR, and XRGB pixel ordering. This is implemented with ten new colorspace constants:

```
JCS_EXT_RGB   /* red/green/blue */
JCS_EXT_RGBX  /* red/green/blue/x */
JCS_EXT_BGR   /* blue/green/red */
JCS_EXT_BGRX  /* blue/green/red/x */
JCS_EXT_XBGR  /* x/blue/green/red */
JCS_EXT_XRGB  /* x/red/green/blue */
JCS_EXT_RGBA  /* red/green/blue/alpha */
JCS_EXT_BGRA  /* blue/green/red/alpha */
JCS_EXT_ABGR  /* alpha/blue/green/red */
JCS_EXT_ARGB  /* alpha/red/green/blue */
```

Setting `cinfo.in_color_space` (compression) or `cinfo.out_color_space` (decompression) to one of these values will cause libjpeg-turbo to read the red, green, and blue values from (or write them to) the appropriate position in the pixel when compressing from/decompressing to an RGB buffer.

Your application can check for the existence of these extensions at compile time with:

```
#ifdef JCS_EXTENSIONS
```

At run time, attempting to use these extensions with a libjpeg implementation that does not support them will result in a "Bogus input colorspace" error. Applications can trap this error in order to test whether run-time support is available for the colorspace extensions.

When using the RGBX, BGRX, XBGR, and XRGB colorspaces during decompression, the X byte is undefined, and in order to ensure the best performance, libjpeg-turbo can set that byte to whatever value it wishes. If an application expects the X byte to be used as an alpha channel, then it should specify `JCS_EXT_RGBA`, `JCS_EXT_BGRA`, `JCS_EXT_ABGR`, or `JCS_EXT_ARGB`. When these colorspace constants are used, the X byte is guaranteed to be 0xFF, which is interpreted as opaque.

Your application can check for the existence of the alpha channel colorspace extensions at compile time with:

```
#ifdef JCS_ALPHA_EXTENSIONS
```

`jcstest.c`, located in the libjpeg-turbo source tree, demonstrates how to check for the existence of the colorspace extensions at compile time and run time.

libjpeg v7 and v8 API/ABI Emulation

With libjpeg v7 and v8, new features were added that necessitated extending the compression and decompression structures. Unfortunately, due to the exposed nature of those structures, extending them also necessitated breaking backward ABI compatibility with previous libjpeg releases. Thus, programs that were built to use libjpeg v7 or v8 did not work with libjpeg-turbo, since it is based on the libjpeg v6b code base. Although libjpeg v7 and v8 are not as widely used as v6b, enough programs (including a few Linux distros) made the switch that there was a demand to emulate the libjpeg v7 and v8 ABIs in libjpeg-turbo. It should be noted, however, that this feature was added primarily so that applications that had already been compiled to use libjpeg v7+ could take advantage of accelerated baseline JPEG encoding/decoding without recompiling. libjpeg-turbo does not claim to support all of the libjpeg v7+ features, nor to produce identical output to libjpeg v7+ in all cases (see below.)

By passing an argument of `-DWITH_JPEG7=1` or `-DWITH_JPEG8=1` to `cmake`, you can build a version of libjpeg-turbo that emulates the libjpeg v7 or v8 ABI, so that programs that are built against libjpeg v7 or v8 can be run with libjpeg-turbo. The following section describes which libjpeg v7+ features are supported and which aren't.

Support for libjpeg v7 and v8 Features

Fully supported

- **libjpeg API: IDCT scaling extensions in decompressor**
  libjpeg-turbo supports IDCT scaling with scaling factors of 1/8, 1/4, 3/8, 1/2, 5/8, 3/4, 7/8, 9/8, 5/4, 11/8, 3/2, 13/8, 7/4, 15/8, and 2/1 (only 1/4 and 1/2 are SIMD-accelerated.)
- **libjpeg API: Arithmetic coding**
- **libjpeg API: In-memory source and destination managers**
  See notes below.
- **cjpeg: Separate quality settings for luminance and chrominance**
  Note that the libpjeg v7+ API was extended to accommodate this feature only for convenience purposes. It has always been possible to implement this feature with libjpeg v6b (see rdswitch.c for an example.)
- **cjpeg: 32-bit BMP support**
- **cjpeg: `-rgb` option**
- **jpegtran: Lossless cropping**
- **jpegtran: `-perfect` option**
- **jpegtran: Forcing width/height when performing lossless crop**
- **rdjpgcom: `-raw` option**
- **rdjpgcom: Locale awareness**

Not supported

NOTE: As of this writing, extensive research has been conducted into the usefulness of DCT scaling as a means of data reduction and SmartScale as a means of quality improvement. The reader is invited to peruse the research at `http://www.libjpeg-turbo.org/About/SmartScale` and draw his/her own conclusions, but it is the general belief of our project that these features have not demonstrated sufficient usefulness to justify inclusion in libjpeg-turbo.

- **libjpeg API: DCT scaling in compressor**
  `cinfo.scale_num` and `cinfo.scale_denom` are silently ignored. There is no technical reason why DCT scaling could not be supported when emulating the libjpeg v7+ API/ABI, but without the SmartScale extension (see below), only scaling factors of 1/2, 8/15, 4/7, 8/13, 2/3, 8/11, 4/5, and 8/9 would be available, which is of limited usefulness.

- **libjpeg API: SmartScale**
  `cinfo.block_size` is silently ignored. SmartScale is an extension to the JPEG format that allows for DCT block sizes other than 8x8. Providing support for this new format would be feasible (particularly without full acceleration.) However, until/unless the format becomes either an official industry standard or, at minimum, an accepted solution in the community, we are hesitant to implement it, as there is no sense of whether or how it might change in the future. It is our belief that SmartScale has not demonstrated sufficient usefulness as a lossless format nor as a means of quality enhancement, and thus our primary interest in providing this feature would be as a means of supporting additional DCT scaling factors.
- **libjpeg API: Fancy downsampling in compressor**
  `cinfo.do_fancy_downsampling` is silently ignored. This requires the DCT scaling feature, which is not supported.
- **jpegtran: Scaling**
  This requires both the DCT scaling and SmartScale features, which are not supported.
- **Lossless RGB JPEG files**
  This requires the SmartScale feature, which is not supported.

What About libjpeg v9?

libjpeg v9 introduced yet another field to the JPEG compression structure (`color_transform`), thus making the ABI backward incompatible with that of libjpeg v8. This new field was introduced solely for the purpose of supporting lossless SmartScale encoding. Furthermore, there was actually no reason to extend the API in this manner, as the color transform could have just as easily been activated by way of a new JPEG colorspace constant, thus preserving backward ABI compatibility.

Our research (see link above) has shown that lossless SmartScale does not generally accomplish anything that can't already be accomplished better with existing, standard lossless formats. Therefore, at this time it is our belief that there is not sufficient technical justification for software projects to upgrade from libjpeg v8 to libjpeg v9, and thus there is not sufficient technical justification for us to emulate the libjpeg v9 ABI.

In-Memory Source/Destination Managers

By default, libjpeg-turbo 1.3 and later includes the `jpeg_mem_src()` and `jpeg_mem_dest()` functions, even when not emulating the libjpeg v8 API/ABI. Previously, it was necessary to build libjpeg-turbo from source with libjpeg v8 API/ABI emulation in order to use the in-memory source/destination managers, but several projects requested that those functions be included when emulating the libjpeg v6b API/ABI as well. This allows the use of those functions by programs that need them, without breaking ABI compatibility for programs that don't, and it allows those functions to be provided in the "official" libjpeg-turbo binaries.

Those who are concerned about maintaining strict conformance with the libjpeg v6b or v7 API can pass an argument of `-DWITH_MEM_SRCDST=0` to `cmake` prior to building libjpeg-turbo. This will restore the pre-1.3 behavior, in which `jpeg_mem_src()` and `jpeg_mem_dest()` are only included when emulating the libjpeg v8 API/ABI.

On Un∗x systems, including the in-memory source/destination managers changes the dynamic library version from 62.2.0 to 62.3.0 if using libjpeg v6b API/ABI emulation and from 7.2.0 to 7.3.0 if using libjpeg v7 API/ABI emulation.

Note that, on most Un∗x systems, the dynamic linker will not look for a function in a library until that function is actually used. Thus, if a program is built against libjpeg-turbo 1.3+ and uses `jpeg_mem_src()` or `jpeg_mem_dest()`, that program will not fail if run against an older version of libjpeg-turbo or against libjpeg v7- until the program actually tries to call `jpeg_mem_src()` or `jpeg_mem_dest()`. Such is not the case on Windows. If a program is built against the libjpeg-turbo 1.3+ DLL and uses `jpeg_mem_src()` or `jpeg_mem_dest()`, then it must use the libjpeg-turbo 1.3+ DLL at run time.

Both cjpeg and djpeg have been extended to allow testing the in-memory source/destination manager functions. See their respective man pages for more details.

Mathematical Compatibility

For the most part, libjpeg-turbo should produce identical output to libjpeg v6b. The one exception to this is when using the floating point DCT/IDCT, in which case the outputs of libjpeg v6b and libjpeg-turbo can differ for the following reasons:

- The SSE/SSE2 floating point DCT implementation in libjpeg-turbo is ever so slightly more accurate than the implementation in libjpeg v6b, but not by any amount perceptible to human vision (generally in the range of 0.01 to 0.08 dB gain in PNSR.)
- When not using the SIMD extensions, libjpeg-turbo uses the more accurate (and slightly faster) floating point IDCT algorithm introduced in libjpeg v8a as opposed to the algorithm used in libjpeg v6b. It should be noted, however, that this algorithm basically brings the accuracy of the floating point IDCT in line with the accuracy of the slow integer IDCT. The floating point DCT/IDCT algorithms are mainly a legacy feature, and they do not produce significantly more accuracy than the slow integer algorithms (to put numbers on this, the typical difference in PNSR between the two algorithms is less than 0.10 dB, whereas changing the quality level by 1 in the upper range of the quality scale is typically more like a 1.0 dB difference.)
- If the floating point algorithms in libjpeg-turbo are not implemented using SIMD instructions on a particular platform, then the accuracy of the floating point DCT/IDCT can depend on the compiler settings.

While libjpeg-turbo does emulate the libjpeg v8 API/ABI, under the hood it is still using the same algorithms as libjpeg v6b, so there are several specific cases in which libjpeg-turbo cannot be expected to produce the same output as libjpeg v8:

- When decompressing using scaling factors of 1/2 and 1/4, because libjpeg v8 implements those scaling algorithms differently than libjpeg v6b does, and libjpeg-turbo's SIMD extensions are based on the libjpeg v6b behavior.
- When using chrominance subsampling, because libjpeg v8 implements this with its DCT/IDCT scaling algorithms rather than with a separate downsampling/upsampling algorithm. In our testing, the subsampled/upsampled output of libjpeg v8 is less accurate than that of libjpeg v6b for this reason.
- When decompressing using a scaling factor $>$ 1 and merged (AKA "non-fancy" or "non-smooth") chrominance upsampling, because libjpeg v8 does not support merged upsampling with scaling factors $>$ 1.

Performance Pitfalls

Restart Markers

The optimized Huffman decoder in libjpeg-turbo does not handle restart markers in a way that makes the rest of the libjpeg infrastructure happy, so it is necessary to use the slow Huffman decoder when decompressing a JPEG image that has restart markers. This can cause the decompression performance to drop by as much as 20%, but the performance will still be much greater than that of libjpeg. Many consumer packages, such as Photoshop, use restart markers when generating JPEG images, so images generated by those programs will experience this issue.

Fast Integer Forward DCT at High Quality Levels

The algorithm used by the SIMD-accelerated quantization function cannot produce correct results whenever the fast integer forward DCT is used along with a JPEG quality of 98-100. Thus, libjpeg-turbo must use the non-SIMD quantization function in those cases. This causes performance to drop by as much as 40%. It is therefore strongly advised that you use the slow integer forward DCT whenever encoding images with a JPEG quality of 98 or higher.

Memory Debugger Pitfalls

Valgrind and Memory Sanitizer (MSan) can generate false positives (specifically, incorrect reports of uninitialized memory accesses) when used with libjpeg-turbo's SIMD extensions. It is generally recommended that the SIMD extensions be disabled, either by passing an argument of `-DWITH_SIMD=0` to `cmake` when configuring the build or by setting the environment variable `JSIMD_FORCENONE` to `1` at run time, when testing libjpeg-turbo with Valgrind, MSan, or other memory debuggers.

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 7

File Index

7.1 File List

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

Chapter 8

Class Documentation

8.1 AccelerationTag Struct Reference

Public Attributes

- short **x**
- short **y**
- short **z**

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

- eSPDI/eSPDI_def.h

8.2 APCImageType Struct Reference

Public Types

- enum **Value** {
  IMAGE_UNKNOWN = -1, **COLOR_YUY2** = 0, **COLOR_RGB24**, **COLOR_MJPG**,
  DEPTH_8BITS = 100, **DEPTH_8BITS_0x80**, **DEPTH_11BITS**, **DEPTH_14BITS** }

Static Public Member Functions

- static bool **IsImageColor** (APCImageType::Value type)
- static bool **IsImageDepth** (APCImageType::Value type)
- static APCImageType::Value **DepthDataTypeToDepthImageType** (WORD dataType)

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

- eSPDI/eSPDI_def.h

8.3 CompassTag Struct Reference

Public Attributes

- short **x**
- short **y**
- short **z**

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

- eSPDI/eSPDI_def.h

8.4 eSPCtrl_RectLogData Struct Reference

Public Attributes

-
```
union {
  unsigned char uByteArray [1024]
  struct {
    unsigned short InImgWidth
    unsigned short InImgHeight
    unsigned short OutImgWidth
    unsigned short OutImgHeight
    int RECT_ScaleEnable
    int RECT_CropEnable
    unsigned short RECT_ScaleWidth
    unsigned short RECT_ScaleHeight
    float CamMat1 [9]
    float CamDist1 [8]
    float CamMat2 [9]
    float CamDist2 [8]
    float RotaMat [9]
    float TranMat [3]
    float LRotaMat [9]
    float RRotaMat [9]
    float NewCamMat1 [12]
    float NewCamMat2 [12]
    unsigned short RECT_Crop_Row_BG
    unsigned short RECT_Crop_Row_ED
    unsigned short RECT_Crop_Col_BG_L
    unsigned short RECT_Crop_Col_ED_L
    unsigned char RECT_Scale_Col_M
    unsigned char RECT_Scale_Col_N
    unsigned char RECT_Scale_Row_M
    unsigned char RECT_Scale_Row_N
    float RECT_AvgErr
    unsigned short nLineBuffers
    float ReProjectMat [16]
  }
};
```

8.4.1 Member Data Documentation

8.4.1.1 CamDist1

```
float eSPCtrl_RectLogData::CamDist1[8]
```

Left Camera Distortion Matrix k1, k2, p1, p2, k3, k4, k5, k6 k1∼k6 : radial distort ; p1,p2 : tangential distort

8.4.1.2 CamDist2

```
float eSPCtrl_RectLogData::CamDist2[8]
```

Right Camera Distortion Matrix k1, k2, p1, p2, k3, k4, k5, k6 k1∼k6 : radial distort ; p1,p2 : tangential distort

8.4.1.3 CamMat1

```
float eSPCtrl_RectLogData::CamMat1[9]
```

Left Camera Matrix fx, 0, cx, 0, fy, cy, 0, 0, 1 fx,fy : focus ; cx,cy : principle point

8.4.1.4 CamMat2

```
float eSPCtrl_RectLogData::CamMat2[9]
```

Right Camera Matrix fx, 0, cx, 0, fy, cy, 0, 0, 1 fx,fy : focus ; cx,cy : principle point

8.4.1.5 InImgHeight

```
unsigned short eSPCtrl_RectLogData::InImgHeight
```

Input image height

8.4.1.6 InImgWidth

```
unsigned short eSPCtrl_RectLogData::InImgWidth
```

Input image width(SideBySide image)

8.4.1.7 LRotaMat

```
float eSPCtrl_RectLogData::LRotaMat[9]
```

3x3 rectification transform (rotation matrix) for the left camera. | [0] [1] [2] | |Xcl| | [3] [4] [5] | ∗ |Ycl| => cl = left camera coordinate | [6] [7] [8] | |Zcl|

8.4.1.8 NewCamMat1

```
float eSPCtrl_RectLogData::NewCamMat1[12]
```

3x4 projection matrix in the (rectified) coordinate systems for the left camera. fx' 0 cx' 0 0 fy' cy' 0 0 0 1 0 fx',fy' : rectified focus ; cx', cy; : rectified principle point

8.4.1.9 NewCamMat2

```
float eSPCtrl_RectLogData::NewCamMat2[12]
```

3x4 projection matrix in the (rectified) coordinate systems for the rightt camera. fx' 0 cx' TranMat[0]∗ 0 fy' cy' 0 0 0 1 0 fx',fy' : rectified focus ; cx', cy; : rectified principle point

8.4.1.10 nLineBuffers

```
unsigned short eSPCtrl_RectLogData::nLineBuffers
```

Linebuffer for Hardware limitation $<$ 60

8.4.1.11 OutImgHeight

```
unsigned short eSPCtrl_RectLogData::OutImgHeight
```

Output image height

8.4.1.12 OutImgWidth

```
unsigned short eSPCtrl_RectLogData::OutImgWidth
```

Output image width(SideBySide image)

8.4.1.13 RECT_AvgErr

```
float eSPCtrl_RectLogData::RECT_AvgErr
```

Reprojection error

8.4.1.14 RECT_Crop_Col_BG_L

```
unsigned short eSPCtrl_RectLogData::RECT_Crop_Col_BG_L
```

Rectidied image crop column begin

8.4.1.15 RECT_Crop_Col_ED_L

```
unsigned short eSPCtrl_RectLogData::RECT_Crop_Col_ED_L
```

Rectidied image crop column end

8.4.1.16 RECT_Crop_Row_BG

```
unsigned short eSPCtrl_RectLogData::RECT_Crop_Row_BG
```

Rectidied image crop row begin

8.4.1.17 RECT_Crop_Row_ED

```
unsigned short eSPCtrl_RectLogData::RECT_Crop_Row_ED
```

Rectidied image crop row end

8.4.1.18 RECT_CropEnable

```
int eSPCtrl_RectLogData::RECT_CropEnable
```

Rectified image crop

8.4.1.19 RECT_Scale_Col_M

```
unsigned char eSPCtrl_RectLogData::RECT_Scale_Col_M
```

Rectified image scale column factor M

8.4.1.20 RECT_Scale_Col_N

```
unsigned char eSPCtrl_RectLogData::RECT_Scale_Col_N
```

Rectified image scale column factor N Rectified image scale column ratio = Scale_Col_N/ Scale_Col_M

8.4.1.21 RECT_Scale_Row_M

```
unsigned char eSPCtrl_RectLogData::RECT_Scale_Row_M
```

Rectified image scale row factor M

8.4.1.22 RECT_Scale_Row_N

```
unsigned char eSPCtrl_RectLogData::RECT_Scale_Row_N
```

Rectified image scale row factor N

8.4.1.23 RECT_ScaleEnable

```
int eSPCtrl_RectLogData::RECT_ScaleEnable
```

Rectified image scale

8.4.1.24 RECT_ScaleHeight

```
unsigned short eSPCtrl_RectLogData::RECT_ScaleHeight
```

Input image height(Single image) ∗RECT_Scale_Row_N /RECT_Scale_Row_M

8.4.1.25 RECT_ScaleWidth

```
unsigned short eSPCtrl_RectLogData::RECT_ScaleWidth
```

Input image width(Single image) ∗RECT_Scale_Col_N /RECT_Scale_Col_M

8.4.1.26 RotaMat

```
float eSPCtrl_RectLogData::RotaMat[9]
```

Rotation matrix between the left and right camera coordinate systems. | [0] [1] [2] | |Xcr| | [3] [4] [5] | ∗ |Ycr| => cr = right camera coordinate | [6] [7] [8] | |Zcr|

8.4.1.27 RRotaMat

```
float eSPCtrl_RectLogData::RRotaMat[9]
```

3x3 rectification transform (rotation matrix) for the left camera. | [0] [1] [2] | |Xcr| | [3] [4] [5] | ∗ |Ycr| => cr = right camera coordinate | [6] [7] [8] | |Zcr|

8.4.1.28 TranMat

```
float eSPCtrl_RectLogData::TranMat[3]
```

Translation vector between the coordinate systems of the cameras. |[0]| |Xcr| |[1]| + |Ycr| => cr = right camera coordinate |[2]| |Zcr|

8.4.1.29 uByteArray

```
unsigned char eSPCtrl_RectLogData::uByteArray[1024]
```

union data defined as below struct { }

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

- eSPDI/eSPDI_def.h

8.5 GyroTag Struct Reference

Public Attributes

- short **x**
- short **y**
- short **z**

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

- eSPDI/eSPDI_def.h

8.6 packet_s Struct Reference

Public Attributes

- int **len**
- int **serial**
- bool **bisRGB**
- bool **bisReady**
- union {
  unsigned char **buffer_yuyv** [2 ∗2560 ∗2560]
  unsigned char **buffer_RGB** [3 ∗2560 ∗2560]
  };

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

- eSPDI/eSPDI_def.h

8.7 PointCloudInfo Struct Reference

Public Attributes

- float **centerX**
- float **centerY**
- float **focalLength**
- float **disparityToW** [2048]
- int **disparity_len**
- WORD **wDepthType**
- float **focalLength_K**
- float **baseline_K**
- float **diff_K**

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

- eSPDI/eSPDI_def.h

8.8 tagAPC_STREAM_INFO Struct Reference

Public Attributes

- int **nWidth**
- int **nHeight**
- BOOL **bFormatMJPG**

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

- eSPDI/eSPDI_def.h

8.9 tagDEVINFORMATION Struct Reference

Public Attributes

- unsigned short **wPID**
- unsigned short **wVID**
- char * **strDevName**
- unsigned short **nChipID**
- unsigned short **nDevType**

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

- eSPDI/eSPDI_def.h

8.10 tagDEVSEL Struct Reference

Public Attributes

- int **index**

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

- eSPDI/eSPDI_def.h

8.11 tagKEEP_DATA_CTRL Struct Reference

Public Attributes

- bool **bIsSerialNumberKeep**
- bool **bIsSensorPositionKeep**
- bool **bIsRectificationTableKeep**
- bool **bIsZDTableKeep**
- bool **bIsCalibrationLogKeep**

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

- eSPDI/eSPDI_def.h

8.12 tagZDTableInfo Struct Reference

Public Attributes

- int **nIndex**
- int **nDataType**

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

- eSPDI/eSPDI_def.h

Chapter 9

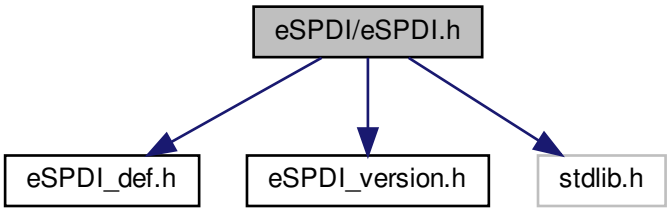
File Documentation

9.1 eSPDI/eSPDI.h File Reference

functions definitions

```
#include "eSPDI_def.h"
#include "eSPDI_version.h"
#include <stdlib.h>
```

Include dependency graph for eSPDI.h:



Functions

- int APC_Init (void **ppHandleEYSD, bool bIsLogEnabled)

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

  find out all EYSD USB devices by PID, VID and ChipID, also remember device types
- void APC_Release (void **ppHandleEYSD)

  release resource that APC_Init had allocated
- int APC_RefreshDevice (void *pHandleEYSD)

  refresh all EYSD UVC devices
- int APC_SwitchBaseline (int index)

  Swich the baseline index.

- bool APC_IsMLBaseLine (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

  Check the device is multiple baseline device.
- int APC_DoFusion (unsigned char **pDepthBufList, double *pDepthMerge, unsigned char *pDepthMerge↩ Flag, int nDWidth, int nDHeight, double fFocus, double *pBaseline, double *pWRNear, double *pWRFar, double *pWRFusion, int nMergeNum, bool bdepth2Byte11bit, int method)

  Do Fusion Merge.
- int APC_GetDeviceNumber (void *pHandleEYSD)

  get EYSD USB device numbers
- int APC_GetDeviceInfo (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, DEVINFORMATION *pdevinfo)

  get informations of EYSD UVC devices, see DEVINFORMATION
- int APC_GetDeviceInfoMBL_15cm (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, DEVINFORMATION *pdevinfo)

  get informations of EYSD UVC devices, see DEVINFORMATION
- int APC_SelectDevice (void *pHandleEYSD, int dev_index)

  do not support currently
- bool APC_IsInterleaveDevice (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

  check module support interleave function or not
- int APC_EnableInterleave (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, bool enable)

  enable or disable interleave function
- int APC_SetControlCounterMode (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char nValue)

  enable or disable interleave function
- int APC_GetControlCounterMode (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *n↩ Value)

  enable or disable interleave function
- int APC_GetSensorRegister (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nId, unsigned short address, unsigned short *pValue, int flag, SENSORMODE_INFO SensorMode)
- int APC_SetSensorRegister (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nId, unsigned short address, unsigned short nValue, int flag, SENSORMODE_INFO SensorMode)

  set sensor register value
- int APC_GetFWRegister (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short *pValue, int flag)

  get firmware register value
- int APC_SetFWRegister (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short nValue, int flag)

  set firmware register value
- int APC_GetHWRegister (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short *pValue, int flag)

  get hardware register value
- int APC_SetHWRegister (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned short nValue, int flag)

  set hardware register
- int APC_GetMultiBytesHWRegister (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned char *Data, int size, int flag)

  set hardware register
- int APC_SetMultiBytesHWRegister (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short address, unsigned char *Data, int size, int flag)

  set hardware register
- int APC_GetBusInfo (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, char *pszBusInfo, int *pActual↩ Length)

  get the firmware version of device, the version is a string
- int APC_GetFwVersion (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, char *pszFwVersion, int n↩ BufferSize, int *pActualLength)

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

- int APC_GetPidVid (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short *pPidBuf, unsigned short *pVidBuf)

  get PID(product ID) and VID(vendor ID) of device
- int APC_SetPidVid (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short *pPidBuf, unsigned short *pVidBuf)

  set PID and VID to device
- int APC_GetSerialNumber (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *pData, int nbufferSize, int *pLen)

  get device serial number
- int APC_SetSerialNumber (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *pData, int nLen)

  set serial number to device
- int APC_GetYOffset (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)

  get Y offset (file ID 30+) value
- int APC_GetRectifyTable (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index)

  get rectify values (file ID 40+) from flash
- int APC_GetZDTable (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, PZDTABLEINFO pZDTableInfo)

  get disparity and Z values from flash
- int APC_GetLogData (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, int *pActualLength, int index, CALIBRATION_LOG_TYPE type)

  get log data from flash
- int APC_GetUserData (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, BYTE *buffer, int BufferLength, USERDATA_SECTION_INDEX usi)

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

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

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

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

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

  set user data to flash
- int APC_ReadFlashData (void *pHandleEYSD, 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 APC_WriteFlashData (void *pHandleEYSD, 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 APC_GetDevicePortType (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, USB_PORT_TYPE *pUSB_Port_Type)

Get Device USB-port-type.

- int APC_GetDeviceResolutionList (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nMaxCount, APC_STREAM_INFO *pStreamInfo0, int nMaxCvoidount1, APC_STREAM_INFO *pStreamInfo1)

  get the device resolution list
- int APC_Setup_v4l2_requestbuffers (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int cnt)

  Setup v4l2 request buffers, default = 4.
- int APC_OpenDevice (void *pHandleEYSD, 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 EYSD camera device by V4L2(https://en.wikipedia.org/wiki/Video4Linux), can open color and depth at one time call, do functions as below,
- int APC_OpenDevice2 (void *pHandleEYSD, 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 EYSD camera device by V4L2(https://en.wikipedia.org/wiki/Video4Linux), can open color and depth at one time call, do functions as below,
- int APC_OpenDeviceMBL (void *pHandleEYSD, 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 EYSD camera device by V4L2(https://en.wikipedia.org/wiki/Video4Linux), can open color and depth at one time call, do functions as below,
- int APC_CloseDeviceMBL (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

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

  close device and free resource
- int APC_CloseDeviceEx (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

  close device and free resource for warm reset
- int APC_GetImage (void *pHandleEYSD, 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 APC_GetColorImage (void *pHandleEYSD, 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 APC_GetColorImageWithTimestamp (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, BYTE *pBuf, unsigned long int *pImageSize, int *pSerial, int nDepthDataType, int64_t *pcur_tv_sec, int64_t *pcur_tv_usec)

  get color image by issuing V4L2's IOCTL to get frame data
- int APC_GetDepthImage (void *pHandleEYSD, 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 APC_GetDepthImageWithTimestamp (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, BYTE *pBuf, unsigned long int *pImageSize, int *pSerial, int nDepthDataType, int64_t *pcur_tv_sec, int64_t *pcur_tv_usec)

  get color image by issuing V4L2's IOCTL to get frame data
- int APC_SetupBlock (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, bool enable)

  get color or depth pin image by issuing V4L2's IOCTL to get frame data
- int APC_Get_Color_30_mm_depth (void *pHandleEYSD, 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 APC_Get_60_mm_depth (void *pHandleEYSD, 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 APC_Get_150_mm_depth (void *pHandleEYSD, 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 APC_Get2Image (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, BYTE *pColorImgBuf, BYTE *p↩ DepthImgBuf, 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 APC_GetImage for detailed description
- int **APC_Get2ImageWithTimestamp** (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, BYTE *pColor↩ ImgBuf, BYTE *pDepthImgBuf, unsigned long int *pColorImageSize, unsigned long int *pDepthImageSize, int *pColorSerial, int *pDepthSerial, int nDepthDataType, int64_t *pcur_tv_sec, int64_t *pcur_tv_usec)
- int APC_GetExposureTime (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, float *pf↩ ExpTimeMS)

  get exposure time of ISP setting in millisecond the target sensor type was set in APC_SetSensorTypeName()
- int APC_SetExposureTime (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, float f↩ ExpTimeMS)

  set exposure time of ISP sensor setting the target sensor type was set in APC_SetSensorTypeName()
- int APC_GetGlobalGain (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, float *pf↩ GlobalGain)

  get global gain of ISP setting the target sensor type was set in APC_SetSensorTypeName()
- int APC_SetGlobalGain (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, float fGlobal↩ Gain)

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

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

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

  set color gain of ISP
- bool APC_GetThermalFD (void *pHandleEYSD, int *p_FD)

  get file description of thermal device
- int APC_GetAccMeterValue (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int *pX, int *pY, int *pZ)

  get acc meter value
- int APC_EnableAE (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

  enable auto exposure(AE) function of ISP
- int APC_DisableAE (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

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

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

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

  get auto exposure(AE) is enabled or disable
- int APC_GetAWBStatus (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, PAWB_STATUS pAWBStatus)

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

  get GPIO values
- int APC_SetGPIOValue (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE nValue)

  set GPIO values

- int APC_SetGPIOCtrl (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nGPIOIndex, BYTE nValue)

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

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

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

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

  set processing unit property value by v4l2_control to set processing unit(PU) property value
- int APC_GetCTRangeAndStep (void *pHandleEYSD, 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 APC_GetPURangeAndStep (void *pHandleEYSD, 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 APC_SetDepthDataType (void *pHandleEYSD, 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 APC_GetDepthDataType (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short *pValue)

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

  set infrared radiation(IR) value of PUMA type IC
- int APC_GetCurrentIRValue (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short *pValue)

  get infrared radiation(IR) value of PUMA type IC
- int APC_GetIRMinValue (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short *pValue)

  get minimum IR value of camera module
- int APC_SetIRMaxValue (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short nValue)

  get maximum IR value of camera module
- int APC_GetIRMaxValue (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short *pValue)

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

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

  to check IR is turn on or off
- int APC_GetRectifyLogData (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, eSPCtrl_RectLogData *p↩ Data, int index)

  get rectify log data from flash, just for AXES1 device type
- int APC_GetRectifyMatLogData (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, eSPCtrl_RectLogData *pData, int index)

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

  Not support now.
- int APC_PostInitial (void *pHandleEYSD)

  Not support now.

- int APC_PostEnd (void *pHandleEYSD)

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

  Not support now.
- int APC_PostSetParam (void *pHandleEYSD, int Idx, int Val)

  Not support now.
- int APC_PostGetParam (void *pHandleEYSD, int Idx, int *pVal)

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

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

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

  do software post process on a depth buffer
- int APC_FlyingDepthCancellation_D8 (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *pdepthD8, int width, int height)

  Flying Pixcel Depth Cancellation, just for EX8029.
- int APC_FlyingDepthCancellation_D11 (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *pdepthD11, int width, int height)

  Flying Pixcel Depth Cancellation.
- int APC_Convert_Depth_Y_To_Buffer (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *depth_y, unsigned char *rgb, unsigned int width, unsigned int height, bool color, unsigned short nDepth↩DataType)

  Convert Depth to RGB color or gray.
- int APC_Convert_Depth_Y_To_Buffer_offset (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *depth_y, unsigned char *rgb, unsigned int width, unsigned int height, bool color, unsigned short n↩DepthDataType, int offset)

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

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

  Get EYSD UAC Name.
- int APC_InitialUAC (char *deviceName)

  UAC inital function.
- int APC_WriteWaveHeader (int fd)

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

  Modified Wave Header.
- int APC_GetUACData (unsigned char *buffer, int length)

  UAC inital function.
- int APC_ReleaseUAC (void)

  UAC inital function.
- int APC_InitialFlexibleGyro (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

  gyro sensor inital function
- int APC_ReleaseFlexibleGyro (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

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

  getting gyro data function

- int APC_GetFlexibleGyroLength (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short *GyroLen)

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

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

  gyro sensor inital function
- int APC_ReleaseHidGyro (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

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

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

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

  getting gyro data function
- int APC_GenerateLutFile (void *pHandleEYSD, 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 APC_SaveLutData (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, const char *filename)

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

  Read LUT parameters into the specified buffer.
- int APC_EncryptMP4 (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, const char *filename)

  encrypt a H.264 video
- int APC_DecryptMP4 (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, const char *filename)

  decrypt a H.264 video was generated by APC_EncryptMP4()
- int APC_InjectExtraDataToMp4 (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, const char *filename, const char *data, int dataLen)

  APC_InjectExtraDataToMp4.
- int APC_RetrieveExtraDataFromMp4 (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, const char *filename, char *data, int *dataLen)

  APC_RetrieveExtraDataFromMp4.
- int APC_EncryptString (const char *src, char *dst)

  APC_EncryptString.
- int APC_DecryptString (const char *src, char *dst)

  APC_DecryptString.
- int APC_EncryptString (const char *src1, const char *src2, char *dst)

  APC_EncryptString.
- int APC_DecryptString (const char *src, char *dst1, char *dst2)

  APC_DecryptString.
- int APC_GetAutoExposureMode (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned short *mode)

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

  Setup Auto Exposure Mode.
- int APC_RotateImg90 (APCImageType::Value imgType, int width, int height, unsigned char *src, unsigned char *dst, int len, bool clockwise)

  Rotate the image to 90 degree.
- int APC_RotateImg180 (APCImageType::Value imgType, int width, int height, unsigned char *src, unsigned char *dst, int len)

  Rotate the image to 180 degree.

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

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

  Make the image to Mirro.
- int APC_RGB2BMP (char *filename, int width, int height, unsigned char *data)

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

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

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

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

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

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

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

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

  Get Packet from SRB.
- int APC_DepthMerge (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char **pDepthBufList, float *pDepthMergeOut, unsigned char *pDepthMergeFlag, int nDWidth, int nDHeight, float fFocus, float *pBaseline, float *pWRNear, float *pWRFar, float *pWRFusion, int nMergeNum)

  do depth merge
- int APC_GetPointCloud (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *ImgColor, int CW, int CH, unsigned char *ImgDepth, int DW, int DH, PointCloudInfo *pPointCloudInfo, unsigned char *p↩ PointCloudRGB, float *pPointCloudXYZ, float Near, float Far)

  get point cloud
- int APC_ColorFormat_to_RGB24 (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *Img↩ Dst, unsigned char *ImgSrc, int SrcSize, int width, int height, APCImageType::Value type)

  get hardware post processing status
- int APC_RotateImg90 (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, APCImageType::Value imgType, int width, int height, unsigned char *src, unsigned char *dstBuf, int len, bool clockwise)

  Make the image to rotate.
- int APC_RotateImg180 (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, APCImageType::Value imgType, int width, int height, unsigned char *src, unsigned char *dst, int len)

  Rotate the image to 180 degree.
- int APC_ImgMirro (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, APCImageType::Value imgType, int width, int height, unsigned char *src, unsigned char *dstBuf)

  Make the image to Mirro.
- int APC_SubSample (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char **SubSample, unsigned char *depthBuf, int bytesPerPixel, int width, int height, int &new_width, int &new_height, int mode=0, int factor=3)

  APC_SubSample.
- int APC_HoleFill (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *depthBuf, int bytes↩ PerPixel, int kernel_size, int width, int height, int level, bool horizontal)

  APC_HoleFill.
- int APC_TemporalFilter (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *depthBuf, int bytesPerPixel, int width, int height, float alpha, int history)

APC_TemporalFilter.

- int APC_EdgePreServingFilter (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *depthBuf, int type, int width, int height, int level, float sigma, float lumda)

  APC_EdgePreServingFilter.
- int APC_ApplyFilters (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, unsigned char *depthBuf, unsigned char *subDisparity, int bytesPerPixel, int width, int height, int sub_w, int sub_h, int threshold=64)

  APC_ApplyFilters.
- int APC_ResetFilters (void *pHandleEYSD, PDEVSELINFO pDevSelInfo)

  APC_ResetFilters.
- int APC_EnableGPUAcceleration (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, bool enable)

  APC_EnableGPUAcceleration.
- int APC_TableToData (void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int width, int height, int TableSize, unsigned short *Table, unsigned short *Src, unsigned short *Dst)

  transfer Src to Dst by Table
- int APC_InitPostProcess (void **ppPostProcessHandle, unsigned int nWidth, unsigned int nHeight, APC↩ ImageType::Value imageType)

  APC_InitPostProcess.
- int APC_PostProcess (void *pPostProcessHandle, unsigned char *pDepthData)

  APC_PostProcess.
- int APC_ReleasePostProcess (void *pPostProcessHandle)

  APC_ReleasePostProcess.

9.1.1 Detailed Description

functions definitions

Copyright:

This file copyright (C) 2021 by eYs3D Microelectronics, Co.

An unpublished work. All rights reserved.

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

9.1.2 Function Documentation

9.1.2.1 APC_ApplyFilters()

```
int APC_ApplyFilters (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * depthBuf,
            unsigned char * subDisparity,
            int bytesPerPixel,
            int width,
            int height,
            int sub_w,
            int sub_h,
            int threshold = 64 )
```

APC_ApplyFilters.

Parameters

void	∗pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char∗ depthBuf depth buffer pointer
unsigned	char∗ subDisparity [TODO]
int	bytesPerPixel byte number of one pixel
int	width depth width
int	height depth height
int	sub_w [TODO]
int	sub_h [TODO]
int	threshold [TODO]

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.2 APC_CloseCmdFiFo()

```
int APC_CloseCmdFiFo (
            int FileDescrption )
```

Cmd FiFo Close function.

Parameters

int	FileDescrption File Description

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.3 APC_CloseDevice()

```
int APC_CloseDevice (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

close device and free resource

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.4 APC_CloseDeviceEx()

```
int APC_CloseDeviceEx (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

close device and free resource for warm reset

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.5 APC_CloseDeviceMBL()

```
int APC_CloseDeviceMBL (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

close Multiple Base Linedevice and free resource

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.6 APC_ColorFormat_to_RGB24()

```
int APC_ColorFormat_to_RGB24 (
            void * pHandleEYSD,
```

```
            PDEVSELINFO pDevSelInfo,
            unsigned char * ImgDst,
            unsigned char * ImgSrc,
            int SrcSize,
            int width,
            int height,
            APCImageType::Value type )
```

get hardware post processing status

Parameters

void	∗pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char ∗ImgDst output image buffer
unsigned	char ∗ImgSrc input image buffer
int	SrcSize sizeof of source image
int	width input image width
int	height input image height
APCImageType::Value	type input image-format

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.7 APC_Convert_Depth_Y_To_Buffer()

```
int APC_Convert_Depth_Y_To_Buffer (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.8 APC_Convert_Depth_Y_To_Buffer_offset()

```
int APC_Convert_Depth_Y_To_Buffer_offset (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.9 APC_CreateSwPostProc()

```
int APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.10 APC_DecryptMP4()

```
int APC_DecryptMP4 (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            const char * filename )
```

decrypt a H.264 video was generated by APC_EncryptMP4()

Parameters

void∗	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char ∗filename the input video file for decryption

Returns

success: APC_OK, others:see eSPDI_def.h

9.1.2.11 APC_DecryptString() [1/2]

```
int APC_DecryptString (
            const char * src,
            char * dst )
```

APC_DecryptString.

Parameters

const	char∗ src input string
char∗	dst output string (decrypted)

Returns

success: APC_OK, others:see eSPDI_def.h

9.1.2.12 APC_DecryptString() [2/2]

```
int APC_DecryptString (
            const char * src,
```

```
            char * dst1,
            char * dst2 )
```

APC_DecryptString.

Parameters

const	char* src input string
*char**	dst1 output string #1 (decrypted)
*char**	dst2 output string #2 (decrypted)

Returns

success: APC_OK, others:see eSPDI_def.h

9.1.2.13 APC_DepthMerge()

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

do depth merge

Parameters

void	*pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char** pDepthBufList [TODO]
float	*pDepthMergeOut [TODO]
unsigned	char *pDepthMergeFlag [TODO]
int	nDWidth [TODO]
int	nDHeight [TODO]
float	fFocus [TODO]
float	* pBaseline [TODO]
float	* pWRNear [TODO]
float	* pWRFar [TODO]
float	* pWRFusion [TODO]
int	nMergeNum [TODO]

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.14 APC_DisableAE()

```
int APC_DisableAE (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

disable auto exposure(AE) function of ISP

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.15 APC_DisableAWB()

```
int APC_DisableAWB (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

disable auto white balance of ISP

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.16 APC_DoFusion()

```
int APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.17 APC_DoSwPostProc()

```
int APC_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

Parameters

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: APC_OK, others: see eSPDI_def.h

9.1.2.18 APC_EdgePreServingFilter()

```
int APC_EdgePreServingFilter (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * depthBuf,
            int type,
            int width,
            int height,
            int level,
            float sigma,
            float lumda )
```

APC_EdgePreServingFilter.

Parameters

void	*pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char* depthBuf depth buffer pointer
int	bytesPerPixel byte number of one pixel
int	width depth width
int	height depth height
int	level [TODO]
float	sigma [TODO]
float	lumda [TODO]

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.19 APC_EnableAE()

```
int APC_EnableAE (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

enable auto exposure(AE) function of ISP

Parameters

void	*pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.20 APC_EnableAWB()

```
int APC_EnableAWB (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

enable auto white balance function of ISP

Parameters

void	*pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.21 APC_EnableGPUAcceleration()

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

APC_EnableGPUAcceleration.

Parameters

void	∗pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
bool	enable enable it or not

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.22 APC_EnableInterleave()

```
int APC_EnableInterleave (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            bool enable )
```

enable or disable interleave function

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.23 APC_EnableSensorIF()

```
int APC_EnableSensorIF (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            bool bIsEnable )
```

enable or disable sensor IF

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
bool	bIsEnable true is enable, false is disable

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.24 APC_EncryptMP4()

```
int APC_EncryptMP4 (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            const char * filename )
```

encrypt a H.264 video

Parameters

void∗	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char ∗filename the input video file for encryption

Returns

success: APC_OK, others:see eSPDI_def.h

9.1.2.25 APC_EncryptString() [1/2]

```
int APC_EncryptString (
            const char * src,
            char * dst )
```

APC_EncryptString.

Parameters

const	char∗ src input string
char∗	dst output string (encrypted)

Returns

success: APC_OK, others:see eSPDI_def.h

9.1.2.26 APC_EncryptString() [2/2]

```
int APC_EncryptString (
            const char * src1,
```

```
            const char * src2,
            char * dst )
```

APC_EncryptString.

Parameters

const	char* src1 input string #1
const	char* src2 input string #2
*char**	dst output string (encrypted)

Returns

success: APC_OK, others:see eSPDI_def.h

9.1.2.27 APC_FindDevice()

```
int APC_FindDevice (
            void * pHandleEYSD )
```

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

Parameters

void	*pHandleEYSD handle

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.28 APC_FlyingDepthCancellation_D11()

```
int APC_FlyingDepthCancellation_D11 (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pdepthD11,
            int width,
            int height )
```

Flying Pixcel Depth Cancellation.

Parameters

void	*pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char *pdepthD11 point toinput depth buffer
int	width depth width
int	height depth height

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.29 APC_FlyingDepthCancellation_D8()

```
int APC_FlyingDepthCancellation_D8 (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pdepthD8,
            int width,
            int height )
```

Flying Pixcel Depth Cancellation, just for EX8029.

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char ∗pdepthD8 point toinput depth buffer
int	width depth width
int	height depth height

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.30 APC_GenerateLutFile()

```
int APC_GenerateLutFile (
            void * pHandleEYSD,
            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∗	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char∗ filename output LUT file name

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.31 APC_Get2Image()

```
int APC_Get2Image (
            void * pHandleEYSD,
            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 APC_GetImage for detailed description

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.32 APC_Get_150_mm_depth()

```
int APC_Get_150_mm_depth (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.33 APC_Get_60_mm_depth()

```
int APC_Get_60_mm_depth (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.34 APC_Get_Color_30_mm_depth()

```
int APC_Get_Color_30_mm_depth (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.35 APC_GetAccMeterValue()

```
int APC_GetAccMeterValue (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int * pX,
            int * pY,
            int * pZ )
```

get acc meter value

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
int	∗pX X posiztion
int	∗pY Y posiztion
int	∗pZ Z posiztion

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.36 APC_GetAEStatus()

```
int APC_GetAEStatus (
            void * pHandleEYSD,
```

```
            PDEVSELINFO pDevSelInfo,
            PAE_STATUS pAEStatus )
```

get auto exposure(AE) is enabled or disable

Parameters

void	*pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
PAE_STATUS	pAEStatus see enum definition as to AE_STATUS in eSPDI_def.h

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.37 APC_GetAutoExposureMode()

```
int APC_GetAutoExposureMode (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short * mode )
```

Get Auto Exposure Mode.

Parameters

*void**	pHandleEYSD handle.
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: APC_OK, others:eSPDI_def.h

9.1.2.38 APC_GetAWBStatus()

```
int APC_GetAWBStatus (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            PAWB_STATUS pAWBStatus )
```

get auto white balance(AWB) is enabled or disable

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.39 APC_GetBusInfo()

```
int APC_GetBusInfo (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            char * pszBusInfo,
            int * pActualLength )
```

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

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
char	∗pszBusInfo Bus information string
int	∗pActualLength the actual length of Bus info in byte

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.40 APC_GetColorGain()

```
int APC_GetColorGain (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float * pfGainR,
            float * pfGainG,
            float * pfGainB )
```

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

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.41 APC_GetColorImage()

```
int APC_GetColorImage (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.42 APC_GetColorImageWithTimestamp()

```
int APC_GetColorImageWithTimestamp (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            BYTE * pBuf,
```

```
            unsigned long int * pImageSize,
            int * pSerial,
            int nDepthDataType,
            int64_t * pcur_tv_sec,
            int64_t * pcur_tv_usec )
```

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

Parameters

void	∗pHandleEYSD handle
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.
int64_t	∗pcur_tv_sec seconds in 'v4l2_buffer' timestamp of this image data
int64_t	∗pcur_tv_usec microseconds in 'v4l2_buffer' timestamp of this image data

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.43 APC_GetControlCounterMode()

```
int APC_GetControlCounterMode (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * nValue )
```

enable or disable interleave function

Parameters

pHandleEYSD	the pointer to the initilized EYSD 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: APC_OK, others: see eSPDI_def.h

9.1.2.44 APC_GetCTPropVal()

```
int APC_GetCTPropVal (
            void * pHandleEYSD,
```

```
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.45 APC_GetCTRangeAndStep()

```
int APC_GetCTRangeAndStep (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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

Parameters

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 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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.46 APC_GetCurrentIRValue()

```
int APC_GetCurrentIRValue (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

get infrared radiation(IR) value of PUMA type IC

Parameters

void	*pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short *pValue current 1 byte IR value setting

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.47 APC_GetDepthDataType()

```
int APC_GetDepthDataType (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

get current depth data type setting

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
WORD	∗pValue pointer of current depth data type in device

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.48 APC_GetDepthImage()

```
int APC_GetDepthImage (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.49 APC_GetDepthImageWithTimestamp()

```
int APC_GetDepthImageWithTimestamp (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            BYTE * pBuf,
            unsigned long int * pImageSize,
            int * pSerial,
            int nDepthDataType,
            int64_t * pcur_tv_sec,
            int64_t * pcur_tv_usec )
```

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

Parameters

void	∗pHandleEYSD handle
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.
int64_t	∗pcur_tv_sec seconds in 'v4l2_buffer' timestamp of this image data
int64_t	∗pcur_tv_usec microseconds in 'v4l2_buffer' timestamp of this image data

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.50 APC_GetDeviceInfo()

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

get informations of EYSD UVC devices, see DEVINFORMATION

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
DEVINFORMATION∗	pdevinfo pointer of device information

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.51 APC_GetDeviceInfoMBL_15cm()

```
int APC_GetDeviceInfoMBL_15cm (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            DEVINFORMATION * pdevinfo )
```

get informations of EYSD UVC devices, see DEVINFORMATION

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
DEVINFORMATION∗	pdevinfo pointer of device information

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.52 APC_GetDeviceNumber()

```
int APC_GetDeviceNumber (
            void * pHandleEYSD )
```

get EYSD USB device numbers

Parameters

void	∗pHandleEYSD handle

Returns

number of EYSD device

9.1.2.53 APC_GetDeviceResolutionList()

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

get the device resolution list

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
int	nMaxCount0 max count of endpoint1 resolutions
APC_STREAM_INFO	∗pStreamInfo0 resolution infos of endpoint1
int	nMaxCount1 max count of endpoint2 resolutions
APC_STREAM_INFO	∗pStreamInfo1 resolutions infos of endpoint2

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.54 APC_GetExposureTime()

```
int APC_GetExposureTime (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float * pfExpTimeMS )
```

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

Parameters

void	∗pHandleEYSD pHandleEYSD
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: APC_OK, others: see eSPDI_def.h

9.1.2.55 APC_GetFlexibleGyroData()

```
int APC_GetFlexibleGyroData (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int length,
            unsigned char * pGyroData )
```

getting gyro data function

Parameters

void∗	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
int	length Gyro Data Length
unsigned	char ∗pGyroData pointer of Gyro Data.

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.56 APC_GetFlexibleGyroLength()

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

getting length of gyro data function.

Parameters

void∗	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short∗ GyroLen pointer of Gyro Data Lenhth.

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.57 APC_GetFWRegister()

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

get firmware register value

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.58 APC_GetFwVersion()

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

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

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.59 APC_GetGlobalGain()

```
int APC_GetGlobalGain (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float * pfGlobalGain )
```

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

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.60 APC_GetHidGyro()

```
int APC_GetHidGyro (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pBuffer,
            int length )
```

getting gyro data function

Parameters

void∗	pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.61 APC_GetHWRegister()

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

get hardware register value

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.62 APC_GetImage()

```
int APC_GetImage (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.63 APC_GetImageInterrupt()

```
int APC_GetImageInterrupt (
            void  )
```

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

Returns

success: 0, others: not got interrupt

9.1.2.64 APC_GetInfoHidGyro()

```
int APC_GetInfoHidGyro (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pCmdBuf,
            int cmdlength,
            unsigned char * pResponseBuf,
            int * resplength )
```

getting gyro data function

Parameters

void∗	pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.65 APC_GetIRMaxValue()

```
int APC_GetIRMaxValue (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

get maximum IR value of camera module

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.66 APC_GetIRMinValue()

```
int APC_GetIRMinValue (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

get minimum IR value of camera module

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.67 APC_GetIRMode()

```
int APC_GetIRMode (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short * pValue )
```

to check IR is turn on or off

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.68 APC_GetLogData()

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

get log data from flash

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.69 APC_GetLutData()

```
int APC_GetLutData (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int nSize )
```

Read LUT parameters into the specified buffer.

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.70 APC_GetMultiBytesHWRegister()

```
int APC_GetMultiBytesHWRegister (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short address,
            unsigned char * Data,
            int size,
            int flag )
```

set hardware register

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short address register address
unsigned	char ∗Data multiple-bytes regigster value to set
int	size multiple-bytes regigster size
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: APC_OK, others: see eSPDI_def.h

9.1.2.71 APC_GetPidVid()

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

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

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.72 APC_GetPointCloud()

```
int APC_GetPointCloud (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * ImgColor,
            int CW,
            int CH,
            unsigned char * ImgDepth,
            int DW,
            int DH,
            PointCloudInfo * pPointCloudInfo,
            unsigned char * pPointCloudRGB,
            float * pPointCloudXYZ,
            float Near,
            float Far )
```

get point cloud

Parameters

void	∗pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index

Parameters

unsigned	char *ImgColor RGB-buffer
int	CW ImgColor width
int	CH ImgColor height
unsigned	char *ImgDepth depth-buffer
int	DW ImgDepth width
int	DH ImgDepth height
PointCloudInfo	*pPointCloudInfo point-cloud information
unsigned	char *pPointCloudRGB point-cloud RGB value
float	*pPointCloudXYZ point-cloud XYZ value
float	Near filter range near dist.
float	Far filter range far dist.

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.73 APC_GetPUPropVal()

```
int APC_GetPUPropVal (
            void * pHandleEYSD,
            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	*pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.74 APC_GetPURangeAndStep()

```
int APC_GetPURangeAndStep (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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 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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.75 APC_GetRectifyLogData()

```
int APC_GetRectifyLogData (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            eSPCtrl_RectLogData * pData,
            int index )
```

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

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.76 APC_GetRectifyMatLogData()

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

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

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.77 APC_GetRectifyTable()

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

get rectify values (file ID 40+) from flash

Parameters

void	∗pHandleEYSD handle
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:APC_OK, others: see eSPDI_def.h

9.1.2.78 APC_GetSensorRegister()

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

get value from sensor register

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.79 APC_GetSerialNumber()

```
int APC_GetSerialNumber (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pData,
            int nbufferSize,
            int * pLen )
```

get device serial number

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.80 APC_GetSRB()

```
int APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.81 APC_GetThermalFD()

```
int APC_GetThermalFD (
            void * pHandleEYSD,
            int * p_FD )
```

get file description of thermal device

Parameters

void	∗pHandleEYSD handle
int	∗p_FD file description of thermal device

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.82 APC_GetUACData()

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

UAC inital function.

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.83 APC_getUACNAME()

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

Get EYSD UAC Name.

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.84 APC_GetUserData()

```
int APC_GetUserData (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            USERDATA_SECTION_INDEX usi )
```

get user data from flash

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.85 APC_GetYOffset()

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

get Y offset (file ID 30+) value

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.86 APC_GetZDTable()

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

get disparity and Z values from flash

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.87 APC_HoleFill()

```
int APC_HoleFill (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * depthBuf,
            int bytesPerPixel,
            int kernel_size,
            int width,
            int height,
            int level,
            bool horizontal )
```

APC_HoleFill.

Parameters

void	∗pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char∗ depthBuf depth buffer pointer
int	bytesPerPixel byte number of one pixel
int	kernel_size [TODO]
int	width depth width
int	height depth height
int	level [TODO]
bool	horizontal [TODO]

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.88 APC_HoleFilled()

```
int APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.89 APC_ImgMirro() [1/2]

```
int APC_ImgMirro (
            APCImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dstBuf )
```

Make the image to Mirro.

Parameters

APCImageType::Value	imgType Image Type
int	width image width
int	height image height
unsigned	char *src image source
unsigned	char *dstBuf image desteration

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.90 APC_ImgMirro() [2/2]

```
int APC_ImgMirro (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            APCImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dstBuf )
```

Make the image to Mirro.

Parameters

void	∗pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
APCImageType::Value	imgType Image Type
int	width image width
int	height image height
unsigned	char ∗src image source
unsigned	char ∗dstBuf image desteration

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.91 APC_Init()

```
int APC_Init (
            void ** ppHandleEYSD,
            bool bIsLogEnabled )
```

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

Parameters

∗∗*ppHandleEYSD*	a pointer of pointer to access CEYSD class
bIsLogEnabled	generate log or not

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.92 APC_InitialCmdFiFo()

```
int APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.93 APC_InitialFlexibleGyro()

```
int APC_InitialFlexibleGyro (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

gyro sensor inital function

Parameters

*void**	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.94 APC_InitialHidGyro()

```
int APC_InitialHidGyro (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

gyro sensor inital function

Parameters

void∗	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.95 APC_InitialUAC()

```
int APC_InitialUAC (
            char * deviceName )
```

UAC inital function.

Parameters

char	∗deviceName Point to device Name.

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.96 APC_InitPostProcess()

```
APC_InitPostProcess (
            void ** ppPostProcessHandle,
            unsigned int nWidth,
            unsigned int nHeight,
            APCImageType::Value imageType )
```

APC_InitPostProcess.

Parameters

void	∗∗ppPostProcessHandle [TODO]
unsigned	int nWidth [TODO]
unsigned	int nHeight [TODO]
APCImageType::Value	imageType [TODO]

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.97 APC_InitSRB()

```
int APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.98 APC_InjectExtraDataToMp4()

```
int APC_InjectExtraDataToMp4 (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            const char * filename,
            const char * data,
            int dataLen )
```

APC_InjectExtraDataToMp4.

Parameters

*void**	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char *filename input video file name
const	char *data video data
const	int dataLen video data length

Returns

success: APC_OK, others:see eSPDI_def.h

9.1.2.99 APC_IsInterleaveDevice()

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

check module support interleave function or not

Parameters

pHandleEYSD	the pointer to the initilized EYSD SDK instance
pDevSelInfo	pointer of device select index

Returns

true: support interleave, false: not support

9.1.2.100 APC_IsMLBaseLine()

```
bool APC_IsMLBaseLine (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

Check the device is multiple baseline device.

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

true: multiplies baseline device, false: normally device.

9.1.2.101 APC_OpenDevice()

```
int APC_OpenDevice (
            void * pHandleEYSD,
            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 EYSD 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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.102 APC_OpenDevice2()

```
int APC_OpenDevice2 (
            void * pHandleEYSD,
            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 EYSD 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	∗pHandleEYSD handle
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 APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.103 APC_OpenDeviceMBL()

```
int APC_OpenDeviceMBL (
            void * pHandleEYSD,
            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 EYSD 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	∗pHandleEYSD handle
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 APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.104 APC_PostProcess()

```
APC_PostProcess (
            void * pPostProcessHandle,
            unsigned char * pDepthData )
```

APC_PostProcess.

Parameters

void	∗ppPostProcessHandle [TODO]
unsigned	char ∗pDepthData [TODO]

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.105 APC_PutSRB()

```
int APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.106 APC_ReadCmdFiFo()

```
APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.107 APC_ReadFlashData()

```
int APC_ReadFlashData (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.108 APC_RefreshDevice()

```
int APC_RefreshDevice (
            void * pHandleEYSD )
```

refresh all EYSD UVC devices

Parameters

void	∗pHandleEYSD handle

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.109 APC_Release()

```
void APC_Release (
            void ** ppHandleEYSD )
```

release resource that APC_Init had allocated

Parameters

void	∗∗ppHandleEYSD array of CEYSD class handlers

Returns

none

9.1.2.110 APC_ReleaseFlexibleGyro()

```
int APC_ReleaseFlexibleGyro (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

gyro sensor release function

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.111 APC_ReleaseHidGyro()

```
int APC_ReleaseHidGyro (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

gyro sensor release function

Parameters

void∗	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.112 APC_ReleasePostProcess()

```
APC_ReleasePostProcess (
            void * pPostProcessHandle )
```

APC_ReleasePostProcess.

Parameters

void	∗ppPostProcessHandle [TODO]

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.113 APC_ReleaseSwPostProc()

```
int APC_ReleaseSwPostProc (
            void ** handle )
```

release a software post process class

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.114 APC_ReleaseUAC()

```
int APC_ReleaseUAC (
            void  )
```

UAC inital function.

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.115 APC_ResetFilters()

```
int APC_ResetFilters (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo )
```

APC_ResetFilters.

Parameters

void	*pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.116 APC_ResizeImgToHalf()

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

Resize the image to half.

Parameters

APCImageType::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: APC_OK, others: see eSPDI_def.h

9.1.2.117 APC_RetrieveExtraDataFromMp4()

```
int APC_RetrieveExtraDataFromMp4 (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            const char * filename,
            char * data,
            int * dataLen )
```

APC_RetrieveExtraDataFromMp4.

Parameters

*void**	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char *filename input video file name
const	char *data video data
const	int dataLen video data length

Returns

success: APC_OK, others:see eSPDI_def.h

9.1.2.118 APC_RGB2BMP()

```
int APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.119 APC_RotateImg180() [1/2]

```
int APC_RotateImg180 (
            APCImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dstBuf,
            int len )
```

Rotate the image to 180 degree.

Parameters

APCImageType::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: APC_OK, others: see eSPDI_def.h

9.1.2.120 APC_RotateImg180() [2/2]

```
int APC_RotateImg180 (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            APCImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dstBuf,
            int len )
```

Rotate the image to 180 degree.

Parameters

void	∗ pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
APCImageType::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: APC_OK, others: see eSPDI_def.h

9.1.2.121 APC_RotateImg90() [1/2]

```
int APC_RotateImg90 (
            APCImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dstBuf,
            int len,
            bool clockwise )
```

Rotate the image to 90 degree.

Parameters

APCImageType::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: APC_OK, others: see eSPDI_def.h

9.1.2.122 APC_RotateImg90() [2/2]

```
int APC_RotateImg90 (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            APCImageType::Value imgType,
            int width,
            int height,
            unsigned char * src,
            unsigned char * dstBuf,
            int len,
            bool clockwise )
```

Make the image to rotate.

Parameters

void	* pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
APCImageType::Value	imgType Image Type
int	width image width
int	height image height
unsigned	char *src image source
unsigned	char *dstBuf image desteration
bool	clockwise clockwise rotate or not

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.123 APC_SaveLutData()

```
int APC_SaveLutData (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            const char * filename )
```

Save LUT parameters in the specified file.

Parameters

*void**	pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
const	char* filename output LUT file name

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.124 APC_SelectDevice()

```
int APC_SelectDevice (
            void * pHandleEYSD,
            int dev_index )
```

do not support currently

Returns

APC_NotSupport

9.1.2.125 APC_SetAutoExposureMode()

```
int APC_SetAutoExposureMode (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short mode )
```

Setup Auto Exposure Mode.

Parameters

*void**	pHandleEYSD handle.
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: APC_OK, others:eSPDI_def.h

9.1.2.126 APC_SetColorGain()

```
int APC_SetColorGain (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float fGainR,
```

```
            float fGainG,
            float fGainB )
```

set color gain of ISP

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.127 APC_SetControlCounterMode()

```
int APC_SetControlCounterMode (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char nValue )
```

enable or disable interleave function

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.128 APC_SetCTPropVal()

```
int APC_SetCTPropVal (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int nId,
            long int nValue )
```

set camera terminal property values By v4l2_control to set

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.129 APC_SetCurrentIRValue()

```
t APC_SetCurrentIRValue (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short nValue )
```

set infrared radiation(IR) value of PUMA type IC

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short nValue 1 byte IR value to set

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.130 APC_SetDepthDataType()

```
APC_SetDepthDataType (
            void * pHandleEYSD,
            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	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short nValue depth data type you want to set, see APC_DEPTH_DATA_xxx in eSPDI_def.h

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.131 APC_SetExposureTime()

```
int APC_SetExposureTime (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float fExpTimeMS )
```

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

APC_SetExposureTime(void *pHandleEYSD, PDEVSELINFO pDevSelInfo, int nSensorMode, float fExpTimeMS)

Parameters

void	*pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.132 APC_SetFWRegister()

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

set firmware register value

Parameters

void	*pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.133 APC_SetGlobalGain()

```
int APC_SetGlobalGain (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int nSensorMode,
            float fGlobalGain )
```

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

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.134 APC_SetHWRegister()

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

set hardware register

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.135 APC_SetIRMaxValue()

```
int APC_SetIRMaxValue (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short nValue )
```

get maximum IR value of camera module

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short nValue the IR maximum setting value

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.136 APC_SetIRMode()

```
APC_SetIRMode (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short nValue )
```

enable or disable IRs

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.137 APC_SetLogData()

```
int APC_SetLogData (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
```

```
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            int index )
```

set log data to flash

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.138 APC_SetMultiBytesHWRegister()

```
int APC_SetMultiBytesHWRegister (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned short address,
            unsigned char * Data,
            int size,
            int flag )
```

set hardware register

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	short address register address
unsigned	char ∗Data multiple-bytes regigster value to set
int	size multiple-bytes regigster size
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: APC_OK, others: see eSPDI_def.h

9.1.2.139 APC_SetPidVid()

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

set PID and VID to device

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.140 APC_SetPUPropVal()

```
int APC_SetPUPropVal (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int nId,
            long int nValue )
```

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

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.141 APC_SetRectifyTable()

```
int APC_SetRectifyTable (
            void * pHandleEYSD,
```

```
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            int index )
```

set rectify values to flash

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.142 APC_SetSensorRegister()

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

set sensor register value

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.143 APC_SetSensorTypeName()

```
int APC_SetSensorTypeName (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            SENSOR_TYPE_NAME stn )
```

set the sensor type you want to work on

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
SENSOR_TYPE_NAME	stn which sensor you want to work on

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.144 APC_SetSerialNumber()

```
int APC_SetSerialNumber (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pData,
            int nLen )
```

set serial number to device

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.145 APC_Setup_v4l2_requestbuffers()

```
int APC_Setup_v4l2_requestbuffers (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int cnt )
```

Setup v4l2 request buffers, default = 4.

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
int	cnt Should be >= 0

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.146 APC_SetupBlock()

```
int APC_SetupBlock (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            bool enable )
```

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

Parameters

void	∗pHandleEYSD handle
PDEVSELINFO	pDevSelInfo pointer of device select index
bool	enable Enable the Blocking mode or not)

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.147 APC_SetupHidGyro()

```
int APC_SetupHidGyro (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * pCmdBuf,
            int cmdlength )
```

getting gyro data function

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.148 APC_SetUserData()

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

set user data to flash

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.149 APC_SetYOffset()

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

set Y offset values

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.150 APC_SetZDTable()

```
int APC_SetZDTable (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            BYTE * buffer,
            int BufferLength,
            int * pActualLength,
            PZDTABLEINFO pZDTableInfo )
```

set disparity and Z values to flash

Parameters

void	∗pHandleEYSD handle
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: APC_OK, others: see eSPDI_def.h

9.1.2.151 APC_SubSample()

```
int APC_SubSample (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char ** SubSample,
```

```
            unsigned char * depthBuf,
            int bytesPerPixel,
            int width,
            int height,
            int & new_width,
            int & new_height,
            int mode = 0,
            int factor = 3 )
```

APC_SubSample.

Parameters

void	∗pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char ∗∗SubSample [TODO]
unsigned	char ∗depthBuf depth buffer pointer
int	bytesPerPixel byte number of one pixel
int	width depth width
int	height depth height
int&	new_width new depth width
int&	new_height new depth height
int	mode [TODO]
int	factor [TODO]

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.152 APC_SwitchBaseline()

```
int APC_SwitchBaseline (
            int index )
```

Swich the baseline index.

Parameters

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

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.153 APC_TableToData()

```
int APC_TableToData (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            int width,
            int height,
            int TableSize,
            unsigned short * Table,
            unsigned short * Src,
            unsigned short * Dst )
```

transfer Src to Dst by Table

Parameters

void	∗pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
int	width input image width
int	height input image height
int	TableSize input Table size in bytes
unsigned	short ∗Table input Table buffer
unsigned	short ∗Src input Src buffer
unsigned	short ∗Dst output Dst buffer

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.154 APC_TemporalFilter()

```
int APC_TemporalFilter (
            void * pHandleEYSD,
            PDEVSELINFO pDevSelInfo,
            unsigned char * depthBuf,
            int bytesPerPixel,
            int width,
            int height,
            float alpha,
            int history )
```

APC_TemporalFilter.

Parameters

void	∗pHandleEYSD the pointer to the initilized EYSD SDK instance
PDEVSELINFO	pDevSelInfo pointer of device select index
unsigned	char∗ depthBuf depth buffer pointer
int	bytesPerPixel byte number of one pixel
int	width depth width
int	height depth height
float	alpha [TODO]
int	history [TODO]

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.155 APC_WriteCmdFiFo()

```
int APC_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: APC_OK, others: see eSPDI_def.h

9.1.2.156 APC_WriteFlashData()

```
int APC_WriteFlashData (
            void * pHandleEYSD,
            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	*pHandleEYSD 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: APC_OK, others: see eSPDI_def.h

9.1.2.157 APC_WriteWaveEnd()

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

Modified Wave Header.

Parameters

int	fd wave file descript.

Returns

success: APC_OK, others: see eSPDI_def.h

9.1.2.158 APC_WriteWaveHeader()

```
int APC_WriteWaveHeader (
            int fd )
```

Write Wave Header.

Parameters

int	fd wave file descript.

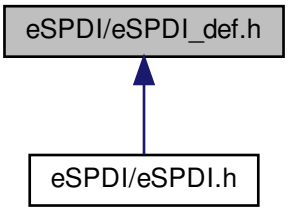
Returns

success: APC_OK, others: see eSPDI_def.h

9.2 eSPDI/eSPDI_def.h File Reference

error/data type definitions

This graph shows which files directly or indirectly include this file:



Classes

- struct packet_s
- struct tagDEVINFORMATION
- struct tagDEVSEL
- struct tagAPC_STREAM_INFO
- struct tagZDTableInfo
- struct tagKEEP_DATA_CTRL
- struct eSPCtrl_RectLogData
- struct GyroTag
- struct AccelerationTag
- struct CompassTag
- struct APCImageType
- struct PointCloudInfo

Macros

- #define **APC_OK** 0
- #define **APC_NoDevice** -1
- #define **APC_NullPtr** -2
- #define **APC_ErrBufLen** -3
- #define **APC_Init_Fail** -4
- #define **APC_NoZDTable** -5
- #define **APC_READFLASHFAIL** -6
- #define **APC_WRITEFLASHFAIL** -7
- #define **APC_VERIFY_DATA_FAIL** -8
- #define **APC_KEEP_DATA_FAIL** -9
- #define **APC_RECT_DATA_LEN_FAIL** -10
- #define **APC_RECT_DATA_PARSING_FAIL** -11
- #define **APC_RET_BAD_PARAM** -12
- #define **APC_RET_OPEN_FILE_FAIL** -13
- #define **APC_NO_CALIBRATION_LOG** -14
- #define **APC_POSTPROCESS_INIT_FAIL** -15
- #define **APC_POSTPROCESS_NOT_INIT** -16
- #define **APC_POSTPROCESS_FRAME_FAIL** -17
- #define **APC_NotSupport** -18

- #define **APC_GET_RES_LIST_FAIL** -19
- #define **APC_READ_REG_FAIL** -20
- #define **APC_WRITE_REG_FAIL** -21
- #define **APC_SET_FPS_FAIL** -22
- #define **APC_VIDEO_RENDER_FAIL** -23
- #define **APC_OPEN_DEVICE_FAIL** -24
- #define **APC_FIND_DEVICE_FAIL** -25
- #define **APC_GET_IMAGE_FAIL** -26
- #define **APC_NOT_SUPPORT_RES** -27
- #define **APC_CALLBACK_REGISTER_FAIL** -28
- #define **APC_CLOSE_DEVICE_FAIL** -29
- #define **APC_GET_CALIBRATIONLOG_FAIL** -30
- #define **APC_SET_CALIBRATIONLOG_FAIL** -31
- #define **APC_DEVICE_NOT_SUPPORT** -32
- #define **APC_DEVICE_BUSY** -33
- #define **APC_DEVICE_TIMEOUT** -34
- #define **APC_IO_SELECT_EINTR** -35
- #define **APC_IO_SELECT_ERROR** -36
- #define **APC_ILLEGAL_ANGLE** -40
- #define **APC_ILLEGAL_STEP** -41
- #define **APC_ILLEGAL_TIMEPERSTEP** -42
- #define **APC_MOTOR_RUNNING** -43
- #define **APC_GETSENSORREG_FAIL** -44
- #define **APC_SETSENSORREG_FAIL** -45
- #define **APC_READ_X_AXIS_FAIL** -46
- #define **APC_READ_Y_AXIS_FAIL** -47
- #define **APC_READ_Z_AXIS_FAIL** -48
- #define **APC_READ_PRESS_DATA_FAIL** -49
- #define **APC_READ_TEMPERATURE_FAIL** -50
- #define **APC_RETURNHOME_RUNNING** -51
- #define **APC_MOTOTSTOP_BY_HOME_INDEX** -52
- #define **APC_MOTOTSTOP_BY_PROTECT_SCHEME** -53
- #define **APC_MOTOTSTOP_BY_NORMAL** -54
- #define **APC_ILLEGAL_FIRMWARE_VERSION** -55
- #define **APC_ILLEGAL_STEPPERTIME** -56
- #define **APC_GET_PU_PROP_VAL_FAIL** -60
- #define **APC_SET_PU_PROP_VAL_FAIL** -61
- #define **APC_GET_CT_PROP_VAL_FAIL** -62
- #define **APC_SET_CT_PROP_VAL_FAIL** -63
- #define **APC_GET_CT_PROP_RANGE_STEP_FAIL** -64
- #define **APC_GET_PU_PROP_RANGE_STEP_FAIL** -65
- #define **APC_INVALID_USERDATA** -70
- #define **APC_MAP_LUT_FAIL** -71
- #define **APC_APPEND_TO_FILE_FRONT_FAIL** -72
- #define **APC_TOO_MANY_DEVICE** -80
- #define **APC_ACCESS_MP4_EXTRA_DATA_FAIL** -81
- #define **BIT_SET**(a, b) ((a) |= (1<<(b)))
- #define **BIT_CLEAR**(a, b) ((a) &= ~(1<<(b)))
- #define **BIT_FLIP**(a, b) ((a) ^= (1<<(b)))
- #define **BIT_CHECK**(a, b) ((a) & (1<<(b)))
- #define **FG_Address_1Byte** 0x01
- #define **FG_Address_2Byte** 0x02
- #define **FG_Value_1Byte** 0x10
- #define **FG_Value_2Byte** 0x20
- #define **EVENT_BUFFER_SHM_COLOR** "/shm_ring_buffer_color"

- #define **EVENT_BUFFER_SHM_DEPTH** "/shm_ring_buffer_depth"
- #define **EVENT_BUFFER_SHM** "/shm_ring_buffer"
- #define **CMD_FIFO_PATH** "/tmp/cmdfifo"
- #define **ZD_PATH** "/tmp/zd_addr"
- #define **RECTIFY_LOG_PATH** "/tmp/rectifylog_addr"
- #define **SRB_LENGTH** 10
- #define **CHIPID_ADDR** 0xf014
- #define **SERIAL_2BIT_ADDR** 0xf0fe
- #define **APC_DEPTH_DATA_OFF_RAW** 0 /∗ raw (depth off, only raw color) ∗/
- #define **APC_DEPTH_DATA_DEFAULT** 0 /∗ raw (depth off, only raw color) ∗/
- #define **APC_DEPTH_DATA_8_BITS** 1 /∗ rectify, 1 byte per pixel ∗/
- #define **APC_DEPTH_DATA_14_BITS** 2 /∗ rectify, 2 byte per pixel ∗/
- #define **APC_DEPTH_DATA_8_BITS_x80** 3 /∗ rectify, 2 byte per pixel but using 1 byte only ∗/
- #define **APC_DEPTH_DATA_11_BITS** 4 /∗ rectify, 2 byte per pixel but using 11 bit only ∗/
- #define **APC_DEPTH_DATA_OFF_RECTIFY** 5 /∗ rectify (depth off, only rectify color) ∗/
- #define **APC_DEPTH_DATA_8_BITS_RAW** 6 /∗ raw ∗/
- #define **APC_DEPTH_DATA_14_BITS_RAW** 7 /∗ raw ∗/
- #define **APC_DEPTH_DATA_8_BITS_x80_RAW** 8 /∗ raw ∗/
- #define **APC_DEPTH_DATA_11_BITS_RAW** 9 /∗ raw ∗/
- #define **APC_DEPTH_DATA_14_BITS_COMBINED_RECTIFY** 11
- #define **APC_DEPTH_DATA_11_BITS_COMBINED_RECTIFY** 13
- #define **APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET** 16
- #define **APC_DEPTH_DATA_ILM_OFF_RAW** APC_DEPTH_DATA_OFF_RAW + APC_DEPTH_DATA_↩ INTERLEAVE_MODE_OFFSET /∗ raw (depth off, only raw color) ∗/
- #define **APC_DEPTH_DATA_ILM_DEFAULT** APC_DEPTH_DATA_DEFAULT + APC_DEPTH_DATA_I↩ NTERLEAVE_MODE_OFFSET /∗ raw (depth off, only raw color) ∗/
- #define **APC_DEPTH_DATA_ILM_8_BITS** APC_DEPTH_DATA_8_BITS + APC_DEPTH_DATA_INTER↩ LEAVE_MODE_OFFSET /∗ rectify, 1 byte per pixel ∗/
- #define **APC_DEPTH_DATA_ILM_14_BITS** APC_DEPTH_DATA_14_BITS + APC_DEPTH_DATA_INT↩ ERLEAVE_MODE_OFFSET /∗ rectify, 2 byte per pixel ∗/
- #define **APC_DEPTH_DATA_ILM_8_BITS_x80** APC_DEPTH_DATA_8_BITS_x80 + APC_DEPTH_DAT↩ A_INTERLEAVE_MODE_OFFSET /∗ rectify, 2 byte per pixel but using 1 byte only ∗/
- #define **APC_DEPTH_DATA_ILM_11_BITS** APC_DEPTH_DATA_11_BITS + APC_DEPTH_DATA_INT↩ ERLEAVE_MODE_OFFSET /∗ rectify, 2 byte per pixel but using 11 bit only ∗/
- #define **APC_DEPTH_DATA_ILM_OFF_RECTIFY** APC_DEPTH_DATA_OFF_RECTIFY + APC_DEPTH↩ _DATA_INTERLEAVE_MODE_OFFSET /∗ rectify (depth off, only rectify color) ∗/
- #define **APC_DEPTH_DATA_ILM_8_BITS_RAW** APC_DEPTH_DATA_8_BITS_RAW + APC_DEPTH_D↩ ATA_INTERLEAVE_MODE_OFFSET /∗ raw ∗/
- #define **APC_DEPTH_DATA_ILM_14_BITS_RAW** APC_DEPTH_DATA_14_BITS_RAW + APC_DEPTH↩ _DATA_INTERLEAVE_MODE_OFFSET /∗ raw ∗/
- #define **APC_DEPTH_DATA_ILM_8_BITS_x80_RAW** APC_DEPTH_DATA_8_BITS_x80_RAW + APC_↩ DEPTH_DATA_INTERLEAVE_MODE_OFFSET /∗ raw ∗/
- #define **APC_DEPTH_DATA_ILM_11_BITS_RAW** APC_DEPTH_DATA_11_BITS_RAW + APC_DEPTH↩ _DATA_INTERLEAVE_MODE_OFFSET /∗ raw ∗/
- #define **APC_DEPTH_DATA_ILM_14_BITS_COMBINED_RECTIFY** APC_DEPTH_DATA_14_BITS_CO↩ MBINED_RECTIFY + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET
- #define **APC_DEPTH_DATA_ILM_11_BITS_COMBINED_RECTIFY** APC_DEPTH_DATA_11_BITS_CO↩ MBINED_RECTIFY + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET
- #define **APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET** 32
- #define **APC_DEPTH_DATA_SCALE_DOWN_OFF_RAW** (APC_DEPTH_DATA_OFF_RAW + APC_DE↩ PTH_DATA_SCALE_DOWN_MODE_OFFSET)/∗ raw (depth off, only raw color) ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_DEFAULT** (APC_DEPTH_DATA_DEFAULT + APC_DEP↩ TH_DATA_SCALE_DOWN_MODE_OFFSET) /∗ raw (depth off, only raw color) ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_8_BITS** (APC_DEPTH_DATA_8_BITS + APC_DEPTH_D↩ ATA_SCALE_DOWN_MODE_OFFSET)/∗ rectify, 1 byte per pixel ∗/

- #define **APC_DEPTH_DATA_SCALE_DOWN_14_BITS** (APC_DEPTH_DATA_14_BITS + APC_DEPTH↩ _DATA_SCALE_DOWN_MODE_OFFSET) /∗ rectify, 2 byte per pixel ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80** (APC_DEPTH_DATA_8_BITS_x80 + APC_↩ DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /∗ rectify, 2 byte per pixel but using 1 byte only ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_11_BITS** (APC_DEPTH_DATA_11_BITS + APC_DEPTH↩ _DATA_SCALE_DOWN_MODE_OFFSET)/∗ rectify, 2 byte per pixel but using 11 bit only ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_OFF_RECTIFY** (APC_DEPTH_DATA_OFF_RECTIFY + APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /∗ Rule 0.4b Reserved unused in any firmware∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_8_BITS_RAW** (APC_DEPTH_DATA_8_BITS_RAW + AP↩ C_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /∗ raw ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_14_BITS_RAW** (APC_DEPTH_DATA_14_BITS_RAW + APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /∗ raw ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_8_BITS_x80_RAW** (APC_DEPTH_DATA_8_BITS_x80_↩ RAW + APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /∗ raw ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_11_BITS_RAW** (APC_DEPTH_DATA_11_BITS_RAW + APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /∗ raw ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_14_BITS_COMBINED_RECTIFY** (APC_DEPTH_DATA_↩ 14_BITS_COMBINED_RECTIFY + APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /∗ Rule 0.4b Reserved unused in any firmware∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_11_BITS_COMBINED_RECTIFY** (APC_DEPTH_DATA_↩ 11_BITS_COMBINED_RECTIFY + APC_DEPTH_DATA_SCALE_DOWN_MODE_OFFSET) /∗ Rule 0.4b Reserved unused in any firmware∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_OFF_RAW** (APC_DEPTH_DATA_SCALE_DOWN_↩ OFF_RAW + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ raw (depth off, only raw color) ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_DEFAULT** (APC_DEPTH_DATA_SCALE_DOWN_D↩ EFAULT + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ raw (depth off, only raw color) ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS** (APC_DEPTH_DATA_SCALE_DOWN_8_B↩ ITS + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ rectify, 1 byte per pixel ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS** (APC_DEPTH_DATA_SCALE_DOWN_14↩ _BITS + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ rectify, 2 byte per pixel ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_x80** (APC_DEPTH_DATA_SCALE_DOWN↩ _8_BITS_x80 + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ rectify, 2 byte per pixel but using 1 byte only ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS** (APC_DEPTH_DATA_SCALE_DOWN_11↩ _BITS + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ rectify, 2 byte per pixel but using 11 bit only ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_OFF_RECTIFY** (APC_DEPTH_DATA_SCALE_DO↩ WN_OFF_RECTIFY + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ rectify (depth off, only rectify color) ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_RAW** (APC_DEPTH_DATA_SCALE_DOW↩ N_8_BITS_RAW + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ raw ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS_RAW** (APC_DEPTH_DATA_SCALE_DO↩ WN_14_BITS_RAW + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ raw ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_8_BITS_x80_RAW** (APC_DEPTH_DATA_SCALE_↩ DOWN_8_BITS_x80_RAW + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ raw ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS_RAW** (APC_DEPTH_DATA_SCALE_DO↩ WN_11_BITS_RAW + APC_DEPTH_DATA_INTERLEAVE_MODE_OFFSET) /∗ raw ∗/
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_14_BITS_COMBINED_RECTIFY** (APC_DEPTH_D↩ ATA_SCALE_DOWN_14_BITS_COMBINED_RECTIFY + APC_DEPTH_DATA_INTERLEAVE_MODE_O↩ FFSET)
- #define **APC_DEPTH_DATA_SCALE_DOWN_ILM_11_BITS_COMBINED_RECTIFY** (APC_DEPTH_D↩ ATA_SCALE_DOWN_11_BITS_COMBINED_RECTIFY + APC_DEPTH_DATA_INTERLEAVE_MODE_O↩ FFSET)
- #define **APC_READ_FLASH_TOTAL_SIZE** 128
- #define **APC_READ_FLASH_FW_PLUGIN_SIZE** 104
- #define **APC_WRITE_FLASH_TOTAL_SIZE** 128

- #define **APC_Y_OFFSET_FILE_ID_0** 30
- #define **APC_Y_OFFSET_FILE_SIZE** 256
- #define **APC_RECTIFY_FILE_ID_0** 40
- #define **APC_RECTIFY_FILE_SIZE** 1024
- #define **APC_ZD_TABLE_FILE_ID_0** 50
- #define **APC_ZD_TABLE_FILE_SIZE_8_BITS** 512
- #define **APC_ZD_TABLE_FILE_SIZE_11_BITS** 4096
- #define **APC_CALIB_LOG_FILE_ID_0** 240
- #define **APC_CALIB_LOG_FILE_SIZE** 4096
- #define **APC_USER_DATA_FILE_ID_0** 200
- #define **APC_USER_DATA_FILE_SIZE_0** 1024
- #define **APC_USER_DATA_FILE_SIZE_1** 4096
- #define **APC_PID_8029** 0x0568
- #define **APC_PID_8030** APC_PID_8029
- #define **APC_PID_8039** APC_PID_8029
- #define **APC_PID_8031** 0x0117
- #define **APC_PID_8032** 0x0118
- #define **APC_PID_8036** 0x0120
- #define **APC_PID_8037** 0x0121
- #define **APC_PID_8038** 0x0124
- #define **APC_PID_8038_M0** APC_PID_8038
- #define **APC_PID_8038_M1** 0x0147
- #define **APC_PID_8040W** 0x0130
- #define **APC_PID_8040S** 0x0131
- #define **APC_PID_8040S_K** 0x0149
- #define **APC_PID_8041** 0x0126
- #define **APC_PID_8042** 0x0127
- #define **APC_PID_8043** 0x0128
- #define **APC_PID_8044** 0x0129
- #define **APC_PID_8045K** 0x0134
- #define **APC_PID_8046K** 0x0135
- #define **APC_PID_8051** 0x0136
- #define **APC_PID_8052** 0x0137
- #define **APC_PID_8053** 0x0138
- #define **APC_PID_8054** 0x0139
- #define **APC_PID_8054_K** 0x0143
- #define **APC_PID_8059** 0x0146
- #define **APC_PID_8060** 0x0152
- #define **APC_PID_8060_K** 0x0150
- #define **APC_PID_8060_T** 0x0151
- #define **APC_PID_AMBER** 0x0112
- #define **APC_PID_SALLY** 0x0158
- #define **APC_PID_HYPATIA** 0x0160
- #define **APC_PID_8062** 0x0162
- #define **APC_PID_GRAP** 0x0179
- #define **APC_PID_GRAP_K** 0x0183
- #define **APC_PID_GRAP_SLAVE** 0x0279
- #define **APC_PID_GRAP_SLAVE_K** 0x0283
- #define **APC_PID_SANDRA** 0x0167
- #define **APC_PID_NORA** 0x0168
- #define **APC_PID_GRAP_THERMAL** 0xf9f9
- #define **APC_PID_GRAP_THERMAL2** 0xf8f8
- #define **APC_VID_GRAP_THERMAL** 0x04b4
- #define **APC_VID_2170** 0x0110
- #define **CT_PROPERTY_ID_AUTO_EXPOSURE_MODE_CTRL** 0

- #define **CT_PROPERTY_ID_AUTO_EXPOSURE_PRIORITY_CTRL** 1
- #define **CT_PROPERTY_ID_EXPOSURE_TIME_ABSOLUTE_CTRL** 2
- #define **CT_PROPERTY_ID_EXPOSURE_TIME_RELATIVE_CTRL** 3
- #define **CT_PROPERTY_ID_FOCUS_ABSOLUTE_CTRL** 4
- #define **CT_PROPERTY_ID_FOCUS_RELATIVE_CTRL** 5
- #define **CT_PROPERTY_ID_FOCUS_AUTO_CTRL** 6
- #define **CT_PROPERTY_ID_IRIS_ABSOLUTE_CTRL** 7
- #define **CT_PROPERTY_ID_IRIS_RELATIVE_CTRL** 8
- #define **CT_PROPERTY_ID_ZOOM_ABSOLUTE_CTRL** 9
- #define **CT_PROPERTY_ID_ZOOM_RELATIVE_CTRL** 10
- #define **CT_PROPERTY_ID_PAN_ABSOLUTE_CTRL** 11
- #define **CT_PROPERTY_ID_PAN_RELATIVE_CTRL** 12
- #define **CT_PROPERTY_ID_TILT_ABSOLUTE_CTRL** 13
- #define **CT_PROPERTY_ID_TILT_RELATIVE_CTRL** 14
- #define **CT_PROPERTY_ID_PRIVACY_CTRL** 15
- #define **PU_PROPERTY_ID_BACKLIGHT_COMPENSATION_CTRL** 0
- #define **PU_PROPERTY_ID_BRIGHTNESS_CTRL** 1
- #define **PU_PROPERTY_ID_CONTRAST_CTRL** 2
- #define **PU_PROPERTY_ID_GAIN_CTRL** 3
- #define **PU_PROPERTY_ID_POWER_LINE_FREQUENCY_CTRL** 4
- #define **PU_PROPERTY_ID_HUE_CTRL** 5
- #define **PU_PROPERTY_ID_HUE_AUTO_CTRL** 6
- #define **PU_PROPERTY_ID_SATURATION_CTRL** 7
- #define **PU_PROPERTY_ID_SHARPNESS_CTRL** 8
- #define **PU_PROPERTY_ID_GAMMA_CTRL** 9
- #define **PU_PROPERTY_ID_WHITE_BALANCE_CTRL** 10
- #define **PU_PROPERTY_ID_WHITE_BALANCE_AUTO_CTRL** 11
- #define **POSTPAR_HR_MODE** 5
- #define **POSTPAR_HR_CURVE_0** 6
- #define **POSTPAR_HR_CURVE_1** 7
- #define **POSTPAR_HR_CURVE_2** 8
- #define **POSTPAR_HR_CURVE_3** 9
- #define **POSTPAR_HR_CURVE_4** 10
- #define **POSTPAR_HR_CURVE_5** 11
- #define **POSTPAR_HR_CURVE_6** 12
- #define **POSTPAR_HR_CURVE_7** 13
- #define **POSTPAR_HR_CURVE_8** 14
- #define **POSTPAR_HF_MODE** 17
- #define **POSTPAR_DC_MODE** 20
- #define **POSTPAR_DC_CNT_THD** 21
- #define **POSTPAR_DC_GRAD_THD** 22
- #define **POSTPAR_SEG_MODE** 23
- #define **POSTPAR_SEG_THD_SUB** 24
- #define **POSTPAR_SEG_THD_SLP** 25
- #define **POSTPAR_SEG_THD_MAX** 26
- #define **POSTPAR_SEG_THD_MIN** 27
- #define **POSTPAR_SEG_FILL_MODE** 28
- #define **POSTPAR_HF2_MODE** 31
- #define **POSTPAR_GRAD_MODE** 34
- #define **POSTPAR_TEMP0_MODE** 37
- #define **POSTPAR_TEMP0_THD** 38
- #define **POSTPAR_TEMP1_MODE** 41
- #define **POSTPAR_TEMP1_LEVEL** 42
- #define **POSTPAR_TEMP1_THD** 43
- #define **POSTPAR_FC_MODE** 46

- #define **POSTPAR_FC_EDGE_THD** 47
- #define **POSTPAR_FC_AREA_THD** 48
- #define **POSTPAR_MF_MODE** 51
- #define **POSTPAR_ZM_MODE** 52
- #define **POSTPAR_RF_MODE** 53
- #define **POSTPAR_RF_LEVEL** 54

Typedefs

- typedef unsigned char **BYTE**
- typedef signed int **BOOL**
- typedef unsigned short **WORD**
- typedef struct packet_s **srb_packet_s**
- typedef struct tagDEVINFORMATION **DEVINFORMATION**
- typedef struct tagDEVINFORMATION * **PDEVINFORMATION**
- typedef struct tagDEVSEL **DEVSELINFO**
- typedef struct tagDEVSEL * **PDEVSELINFO**
- typedef struct tagAPC_STREAM_INFO **APC_STREAM_INFO**
- typedef struct tagAPC_STREAM_INFO * **PAPC_STREAM_INFO**
- typedef struct tagZDTableInfo **ZDTABLEINFO**
- typedef struct tagZDTableInfo * **PZDTABLEINFO**
- typedef struct tagKEEP_DATA_CTRL **KEEP_DATA_CTRL**
- typedef enum AE_STATUS * **PAE_STATUS**
- typedef enum AWB_STATUS * **PAWB_STATUS**
- typedef struct eSPCtrl_RectLogData **eSPCtrl_RectLogData**
- typedef struct GyroTag **GYRO_ANGULAR_RATE_DATA**
- typedef struct AccelerationTag **ACCELERATION_DATA**
- typedef struct CompassTag **COMPASS_DATA**

Enumerations

- enum **SENSORMODE_INFO** {
  SENSOR_A = 0, **SENSOR_B**, **SENSOR_BOTH**, **SENSOR_C**,
  SENSOR_D }
- enum **DEVICE_TYPE** { **OTHERS** = 0, **AXES1**, **PUMA**, **KIWI** }
- enum **FLASH_DATA_TYPE** {
  Total = 0, **FW_PLUGIN**, **BOOTLOADER_ONLY**, **FW_ONLY**,
  PLUGIN_ONLY }
- enum **USERDATA_SECTION_INDEX** {
  USERDATA_SECTION_0 = 0, **USERDATA_SECTION_1**, **USERDATA_SECTION_2**, **USERDATA_SEC↩TION_3**,
  USERDATA_SECTION_4, **USERDATA_SECTION_5**, **USERDATA_SECTION_6**, **USERDATA_SECTIO↩N_7**,
  USERDATA_SECTION_8, **USERDATA_SECTION_9** }
- enum **CALIBRATION_LOG_TYPE** {
  ALL_LOG = 0, **SERIAL_NUMBER**, **PRJFILE_LOG**, **STAGE_TIME_RESULT_LOG**,
  SENSOR_OFFSET, **AUTO_ADJUST_LOG**, **RECTIFY_LOG**, **ZD_LOG**,
  DEPTHMAP_KOG }
- enum **CONTROL_MODE** { **IMAGE_SN_NONSYNC** = 0, **IMAGE_SN_SYNC** }
- enum **DEPTH_TRANSFER_CTRL** { **DEPTH_IMG_NON_TRANSFER**, **DEPTH_IMG_GRAY_TRANSFER**, **DEPTH_IMG_COLORFUL_TRANSFER** }

- enum **SENSOR_TYPE_NAME** {
  APC_SENSOR_TYPE_H22 = 0, **APC_SENSOR_TYPE_H65** = 1, **APC_SENSOR_TYPE_OV7740** = 2, **A↩
  PC_SENSOR_TYPE_AR0134** = 3,
  APC_SENSOR_TYPE_AR0135 = 4, **APC_SENSOR_TYPE_AR0144** = 5, **APC_SENSOR_TYPE_AR0330** = 6, **APC_SENSOR_TYPE_AR0522** = 7,
  APC_SENSOR_TYPE_AR1335 = 8, **APC_SENSOR_TYPE_OV9714** = 9, **APC_SENSOR_TYPE_OV9282** = 10, **APC_SENSOR_TYPE_H68** = 11,
  APC_SENSOR_TYPE_OV2740 = 12, **APC_SENSOR_TYPE_UNKOWN** = 0xffff }
- enum **AE_STATUS** { **AE_ENABLE** = 0, **AE_DISABLE** }
- enum **AWB_STATUS** { **AWB_ENABLE** = 0, **AWB_DISABLE** }
- enum **USB_PORT_TYPE** { **USB_PORT_TYPE_2_0** = 2, **USB_PORT_TYPE_3_0**, **USB_PORT_TYPE_U↩
  NKNOW** }
- enum **SENSITIVITY_LEVEL_L3G** { **DPS_245** = 0, **DPS_500**, **DPS_2000** }
- enum **SENSITIVITY_LEVEL_LSM** {
  _2G = 0, **_4G**, **_6G**, **_8G**,
  _16G }
- enum **OUTPUT_DATA_RATE** {
  One_Shot = 0, **_1_HZ_1_HZ**, **_7_HZ_1_HZ**, **_12_5_HZ_1HZ**,
  _25_HZ_1_HZ, **_7_HZ_7_HZ**, **_12_5_HZ_12_5_HZ**, **_25_HZ_25_HZ** }
- enum **POWER_STATE** { **POWER_ON** = 0, **POWER_OFF** }
- enum **BRIGHTNESS_LEVEL** {
  LEVEL_0 = 0, **LEVEL_1**, **LEVEL_2**, **LEVEL_3**,
  LEVEL_4, **LEVEL_5**, **LEVEL_6**, **LEVEL_7**,
  LEVEL_8, **LEVEL_9**, **LEVEL_10**, **LEVEL_11**,
  LEVEL_12, **LEVEL_13**, **LEVEL_14**, **LEVEL_15** }

9.2.1 Detailed Description

error/data type definitions

Copyright:

Index