PixelxCamera

Generated by Doxygen 1.8.13

Contents

1 PixelX Camera SDK **1**

2 Todo List **3**

3 Module Index **5**

3.1 Modules 5

4 Hierarchical Index **7**

4.1 Class Hierarchy 7

5 Class Index **9**

5.1 Class List 9

6 Module Documentation **11**

6.1 CCD initialization & deinitialization 11

6.1.1 Detailed Description 11

6.1.2 Function Documentation 11

6.1.2.1 Initialize() 11

6.1.2.2 ShutDown() 12

6.2 Exposure related functions 13

6.2.1 Detailed Description 13

6.2.2 Function Documentation 13

6.2.2.1 CancelWait() 14

6.2.2.2 GetExposureTime() 14

6.2.2.3 GetImage() 15

6.2.2.4 GetLDCMode() 15

6.2.2.5 GetNthImage() . . . 16
6.2.2.6 SetContinuousCapture() . . . 16
6.2.2.7 SetEraseCount() . . . 16
6.2.2.8 SetExposureInterval() . . . 17
6.2.2.9 SetExposureTime() . . . 17
6.2.2.10 SetPreAmpGain() . . . 18
6.2.2.11 SetReadoutSpeed() . . . 18
6.2.2.12 SetShutter() . . . 19
6.2.2.13 SetTriggerMode() . . . 19
6.2.2.14 StartExposure() . . . 19
6.2.2.15 StopExposure() . . . 20
6.2.2.16 WaitForAcquisition() . . . 20
6.2.2.17 WaitForAcquisitionTimeOut() . . . 20
6.3 Image saving functions . . . 22
6.3.1 Detailed Description . . . 22
6.3.2 Function Documentation . . . 22
6.3.2.1 ConfigAutoSave() . . . 22
6.3.2.2 SaveAsFITS() . . . 23
6.3.2.3 SaveAsFITSEx() . . . 23
6.3.2.4 SaveNthAsFITS() . . . 23
6.3.2.5 SetAutoSaveFITS() . . . 23
6.4 Miscellaneous functions . . . 25
6.4.1 Detailed Description . . . 26
6.5 Error codes . . . 27
6.5.1 Detailed Description . . . 27
6.5.2 Macro Definition Documentation . . . 27
6.5.2.1 PIXELX_ERRNO_MAP . . . 28
6.5.3 Enumeration Type Documentation . . . 28
6.5.3.1 pixelx_errno_t . . . 28
6.6 SerialNumber . . . 30
6.6.1 Detailed Description . . . 30

7 Class Documentation **31**

7.1 ADC_Channel Struct Reference . . . 31

7.2 CCD_ACQ_Set Struct Reference . . . 31

7.3 CMOS4040Camera Class Reference . . . 32

7.3.1 Member Function Documentation . . . 34

7.3.1.1 ROIEnable() . . . 34

7.3.1.2 SetContinuousCapture() . . . 34

7.4 CStarCamera Class Reference . . . 35

7.5 def_error.ErrorDef Class Reference . . . 36

7.6 FITSfile Struct Reference . . . 37

7.7 fitsfile Struct Reference . . . 39

7.8 FloatComp Struct Reference . . . 40

7.9 fx_known_device Struct Reference . . . 40

7.10 IFPACamera Class Reference . . . 40

7.11 ImgMeta Struct Reference . . . 42

7.12 iteratorCol Struct Reference . . . 43

7.13 KAF16803Camera Class Reference . . . 44

7.14 libusb_bos_descriptor Struct Reference . . . 45

7.14.1 Detailed Description . . . 45

7.14.2 Member Data Documentation . . . 46

7.14.2.1 bDescriptorType . . . 46

7.14.2.2 bLength . . . 46

7.14.2.3 bNumDeviceCaps . . . 46

7.14.2.4 dev_capability . . . 46

7.14.2.5 wTotalLength . . . 46

7.15 libusb_bos_dev_capability_descriptor Struct Reference . . . 46

7.15.1 Detailed Description . . . 47

7.15.2 Member Data Documentation . . . 47

7.15.2.1 bDescriptorType . . . 47

7.15.2.2 bDevCapabilityType . . . 47

7.15.2.3 bLength . . . 47
7.15.2.4 dev_capability_data . . . 47
7.16 libusb_config_descriptor Struct Reference . . . 48
7.16.1 Detailed Description . . . 48
7.16.2 Member Data Documentation . . . 49
7.16.2.1 bConfigurationValue . . . 49
7.16.2.2 bDescriptorType . . . 49
7.16.2.3 bLength . . . 49
7.16.2.4 bmAttributes . . . 49
7.16.2.5 bNumInterfaces . . . 49
7.16.2.6 extra . . . 49
7.16.2.7 extra_length . . . 49
7.16.2.8 iConfiguration . . . 50
7.16.2.9 interface . . . 50
7.16.2.10 MaxPower . . . 50
7.16.2.11 wTotalLength . . . 50
7.17 libusb_container_id_descriptor Struct Reference . . . 50
7.17.1 Detailed Description . . . 50
7.17.2 Member Data Documentation . . . 51
7.17.2.1 bDescriptorType . . . 51
7.17.2.2 bDevCapabilityType . . . 51
7.17.2.3 bLength . . . 51
7.17.2.4 bReserved . . . 51
7.17.2.5 ContainerID . . . 51
7.18 libusb_control_setup Struct Reference . . . 51
7.18.1 Detailed Description . . . 52
7.18.2 Member Data Documentation . . . 52
7.18.2.1 bmRequestType . . . 52
7.18.2.2 bRequest . . . 52
7.18.2.3 wIndex . . . 52

7.18.2.4 wLength . . . 52
7.18.2.5 wValue . . . 53
7.19 libusb_device_descriptor Struct Reference . . . 53
7.19.1 Detailed Description . . . 53
7.19.2 Member Data Documentation . . . 53
7.19.2.1 bcdDevice . . . 53
7.19.2.2 bcdUSB . . . 54
7.19.2.3 bDescriptorType . . . 54
7.19.2.4 bDeviceClass . . . 54
7.19.2.5 bDeviceProtocol . . . 54
7.19.2.6 bDeviceSubClass . . . 54
7.19.2.7 bLength . . . 54
7.19.2.8 bMaxPacketSize0 . . . 54
7.19.2.9 bNumConfigurations . . . 55
7.19.2.10 idProduct . . . 55
7.19.2.11 idVendor . . . 55
7.19.2.12 iManufacturer . . . 55
7.19.2.13 iProduct . . . 55
7.19.2.14 iSerialNumber . . . 55
7.20 libusb_endpoint_descriptor Struct Reference . . . 55
7.20.1 Detailed Description . . . 56
7.20.2 Member Data Documentation . . . 56
7.20.2.1 bDescriptorType . . . 56
7.20.2.2 bEndpointAddress . . . 56
7.20.2.3 bInterval . . . 56
7.20.2.4 bLength . . . 56
7.20.2.5 bmAttributes . . . 57
7.20.2.6 bRefresh . . . 57
7.20.2.7 bSynchAddress . . . 57
7.20.2.8 extra . . . 57

7.20.2.9 extra_length . . . 57
7.20.2.10 wMaxPacketSize . . . 57
7.21 libusb_interface Struct Reference . . . 58
7.21.1 Detailed Description . . . 58
7.21.2 Member Data Documentation . . . 58
7.21.2.1 altsetting . . . 58
7.21.2.2 num_altsetting . . . 59
7.22 libusb_interface_descriptor Struct Reference . . . 59
7.22.1 Detailed Description . . . 59
7.22.2 Member Data Documentation . . . 60
7.22.2.1 bAlternateSetting . . . 60
7.22.2.2 bDescriptorType . . . 60
7.22.2.3 bInterfaceClass . . . 60
7.22.2.4 bInterfaceNumber . . . 60
7.22.2.5 bInterfaceProtocol . . . 60
7.22.2.6 bInterfaceSubClass . . . 60
7.22.2.7 bLength . . . 60
7.22.2.8 bNumEndpoints . . . 61
7.22.2.9 endpoint . . . 61
7.22.2.10 extra . . . 61
7.22.2.11 extra_length . . . 61
7.22.2.12 iInterface . . . 61
7.23 libusb_iso_packet_descriptor Struct Reference . . . 61
7.23.1 Detailed Description . . . 62
7.23.2 Member Data Documentation . . . 62
7.23.2.1 actual_length . . . 62
7.23.2.2 length . . . 62
7.23.2.3 status . . . 62
7.24 libusb_pollfd Struct Reference . . . 62
7.24.1 Detailed Description . . . 62

7.24.2 Member Data Documentation . . . 63
7.24.2.1 events . . . 63
7.24.2.2 fd . . . 63
7.25 libusb_ss_endpoint_companion_descriptor Struct Reference . . . 63
7.25.1 Detailed Description . . . 63
7.25.2 Member Data Documentation . . . 63
7.25.2.1 bDescriptorType . . . 64
7.25.2.2 bLength . . . 64
7.25.2.3 bmAttributes . . . 64
7.25.2.4 bMaxBurst . . . 64
7.25.2.5 wBytesPerInterval . . . 64
7.26 libusb_ss_usb_device_capability_descriptor Struct Reference . . . 64
7.26.1 Detailed Description . . . 65
7.26.2 Member Data Documentation . . . 65
7.26.2.1 bDescriptorType . . . 65
7.26.2.2 bDevCapabilityType . . . 65
7.26.2.3 bFunctionalitySupport . . . 65
7.26.2.4 bLength . . . 65
7.26.2.5 bmAttributes . . . 65
7.26.2.6 bU1DevExitLat . . . 66
7.26.2.7 bU2DevExitLat . . . 66
7.26.2.8 wSpeedSupported . . . 66
7.27 libusb_transfer Struct Reference . . . 66
7.27.1 Detailed Description . . . 67
7.27.2 Member Data Documentation . . . 67
7.27.2.1 actual_length . . . 67
7.27.2.2 buffer . . . 67
7.27.2.3 callback . . . 67
7.27.2.4 dev_handle . . . 68
7.27.2.5 endpoint . . . 68

7.27.2.6 flags . . . 68
7.27.2.7 iso_packet_desc . . . 68
7.27.2.8 length . . . 68
7.27.2.9 num_iso_packets . . . 68
7.27.2.10 status . . . 68
7.27.2.11 timeout . . . 69
7.27.2.12 type . . . 69
7.27.2.13 user_data . . . 69
7.28 libusb_usb_2_0_extension_descriptor Struct Reference . . . 69
7.28.1 Detailed Description . . . 69
7.28.2 Member Data Documentation . . . 69
7.28.2.1 bDescriptorType . . . 70
7.28.2.2 bDevCapabilityType . . . 70
7.28.2.3 bLength . . . 70
7.28.2.4 bmAttributes . . . 70
7.29 libusb_version Struct Reference . . . 70
7.29.1 Detailed Description . . . 70
7.29.2 Member Data Documentation . . . 71
7.29.2.1 describe . . . 71
7.29.2.2 major . . . 71
7.29.2.3 micro . . . 71
7.29.2.4 minor . . . 71
7.29.2.5 nano . . . 71
7.29.2.6 rc . . . 71
7.30 LibusbCamera Class Reference . . . 72
7.31 MetaNode Struct Reference . . . 73
7.31.1 Detailed Description . . . 73
7.32 MiniGuideCamera Class Reference . . . 74
7.33 PixelFilter Struct Reference . . . 75
7.34 PIXELX_SERIAL Struct Reference . . . 76
7.35 PixelxCamera Class Reference . . . 76
7.35.1 Member Function Documentation . . . 81
7.35.1.1 PT100Temperature() . . . 81
7.36 tcolumn Struct Reference . . . 82
7.37 wtbarr Struct Reference . . . 82

Index **83**

Chapter 1

PixelX Camera SDK

Model supported

- CSTAR CCD
- CMOS4040

Configuration

(For runtime)

ENV:

- `PIXELX_IMG_HEAD_NOCNT`: If set, SDK assumes that image data from hardware does not send image counter info

For Maintain

1. install `git subrepo`, call `git subrepo status`, you will see how third party libraries being included. Also check CMakeLists.txt
2. For Windows, there is `typedef unsigned char TBYTE` in winnt.h (libusb includes it), which conflicts with `define TBYTE` in fitsio.h. We cannot `typedef` after `define`. You **SHOULD** a. include libusb then include cfitsio, b. avoid includes cfitsio in header (just cast pointer), c. undef TBYTE after include fitsio. (check cmos4040/ for details)

Installation under Debian10

1. Ready to work

1.1 Installation dependency

```
sudo apt install libusb-1.0-0-dev libccfits-dev
```

2. Acquisition of CCDSDK2

```
cd ~
git clone http://210.45.78.50:8888/lab/CCDSDK2.git
```

3. Installation and compilation

```
cd ~/CCDSDK2
mkdir build
cd build
cmake ..
make
sudo make install
```

Fits header keys standard format(PX4040)

keyword = value / comment

- EXPOTIME = 0.1 / [s] Exposure time
- TDCTIME = 43991938 / [10ns] Tdc time
- GPS-OBS = 2019-11-18 09:44:46.439919 / GPS UTC time of observation starup
- CMOSTEMP = -1.9 / Celsius of cmos
- DATE-OBS = 2019-11-18 09:44:46.391367 / Start UTC time of exposure
- DEC = value(etc: 38.0484) / [deg] J2000 DEC
- DEC-TAR = value(etc: 37.96226) / [deg] Target J2000 DEC
- EXPTIME = value(etc: 0.1) / [s] Expose time
- HOTTEMP = -5.187 / Celsius of TEC
- PICMODE = 0 / Dynamic Range Mode(0:LDR_LOW_GAIN 1:LDR_HIGH_GAIN 2:HDR 3:LDR_HIGH_LOW)
- RA = 318.69537 / [deg] J2000 RA
- RA-TAR = 318.50249 / [deg] Target J2000 RA

Chapter 2

Todo List

Member SetEraseCount (uint8_t number)

No explicit range

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

CCD initialization & deinitialization 11
Exposure related functions 13
Image saving functions 22
Miscellaneous functions 25
Error codes 27
SerialNumber 30

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ADC_Channel . . . 31
CCD_ACQ_Set . . . 31
def_error.ErrorDef . . . 36
FITSfile . . . 37
fitsfile . . . 39
FloatComp . . . 40
fx_known_device . . . 40
ImgMeta . . . 42
iteratorCol . . . 43
libusb_bos_descriptor . . . 45
libusb_bos_dev_capability_descriptor . . . 46
libusb_config_descriptor . . . 48
libusb_container_id_descriptor . . . 50
libusb_control_setup . . . 51
libusb_device_descriptor . . . 53
libusb_endpoint_descriptor . . . 55
libusb_interface . . . 58
libusb_interface_descriptor . . . 59
libusb_iso_packet_descriptor . . . 61
libusb_pollfd . . . 62
libusb_ss_endpoint_companion_descriptor . . . 63
libusb_ss_usb_device_capability_descriptor . . . 64
libusb_transfer . . . 66
libusb_usb_2_0_extension_descriptor . . . 69
libusb_version . . . 70
MetaNode . . . 73
PixelFilter . . . 75
PIXELX_SERIAL . . . 76
PixelxCamera . . . 76
    LibusbCamera . . . 72
        CMOS4040Camera . . . 32
        CStarCamera . . . 35
        IFPACamera . . . 40
        KAF16803Camera . . . 44
        MiniGuideCamera . . . 74
tcolumn . . . 82
wtbarr . . . 82

Chapter 5

Class Index

5.1 Class List

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

ADC_Channel . . . 31
CCD_ACQ_Set . . . 31
CMOS4040Camera . . . 32
CStarCamera . . . 35
def_error.ErrorDef . . . 36
FITSfile . . . 37
fitsfile . . . 39
FloatComp . . . 40
fx_known_device . . . 40
IFPACamera . . . 40
ImgMeta . . . 42
iteratorCol . . . 43
KAF16803Camera . . . 44
libusb_bos_descriptor . . . 45
libusb_bos_dev_capability_descriptor . . . 46
libusb_config_descriptor . . . 48
libusb_container_id_descriptor . . . 50
libusb_control_setup . . . 51
libusb_device_descriptor . . . 53
libusb_endpoint_descriptor . . . 55
libusb_interface . . . 58
libusb_interface_descriptor . . . 59
libusb_iso_packet_descriptor . . . 61
libusb_pollfd . . . 62
libusb_ss_endpoint_companion_descriptor . . . 63
libusb_ss_usb_device_capability_descriptor . . . 64
libusb_transfer . . . 66
libusb_usb_2_0_extension_descriptor . . . 69
libusb_version . . . 70
LibusbCamera . . . 72
MetaNode
< meta data node, will write to FITS . . . 73
MiniGuideCamera . . . 74
PixelFilter . . . 75
PIXELX_SERIAL . . . 76
PixelxCamera . . . 76
tcolumn . . . 82
wtbarr . . . 82

Chapter 6

Module Documentation

6.1 CCD initialization & deinitialization

This page describes how to initialize and deinitialize the CCD.

Functions

- CCD_API unsigned int Initialize (int debug_level, int which)
  The initialization function.
- CCD_API unsigned int ShutDown ()
  Deinitialize CCD.

6.1.1 Detailed Description

This page describes how to initialize and deinitialize the CCD.

6.1.2 Function Documentation

6.1.2.1 Initialize()

```
CCD_API unsigned int Initialize (
            int debug_level,
            int which )
```

The initialization function.

Parameters

debug_level	debug level which will send to low-level libusb's debug
which	which camera will used

Returns

- **CCD_SUCCESS** OK
- **CCD_SCAN_ERROR** CCD not found
- **CCD_ACCESS_ERROR** CCD connection error

6.1.2.2 ShutDown()

```
CCD_API unsigned int ShutDown ( )
```

Deinitialize CCD.

Returns

- **CCD_SUCCESS** OK

6.2 Exposure related functions

Functions

- CCD_API unsigned int SetExposureTime (float time)

  Set exposure time.
- CCD_API unsigned int GetExposureTime (float *time)

  Get exposure time.
- CCD_API unsigned int SetTriggerMode (uint8_t mode)

  Set Trigger Mode.
- CCD_API unsigned int SetExposureInterval (float time)

  Set exposure interval.
- CCD_API unsigned int SetReadoutSpeed (int idx)

  Set readout speed.
- CCD_API unsigned int GetNumReadoutSpeed ()

  Get number of available readout speed.
- CCD_API unsigned int **GetReadoutSpeed** (int index)
- CCD_API unsigned int SetEraseCount (uint8_t number)

  Set the erase count before exposure.
- CCD_API unsigned int StartExposure ()

  Start exposure.
- CCD_API unsigned int SetContinuousCapture (uint16_t pic_num)

  Set Capture Number for Continuous Capture.
- CCD_API unsigned int StopExposure ()

  Stop exposure.
- CCD_API unsigned int SetShutter (uint8_t mode)

  Set shutter mode.
- CCD_API unsigned int GetLDCMode (uint8_t *ldc_bool)

  Get LDC mode, as told.
- CCD_API unsigned int SetPreAmpGain (uint8_t gain)

  Set gain mode.
- CCD_API unsigned int CancelWait ()

  Cancel the WaitForAcquisition blocking.
- CCD_API unsigned int WaitForAcquisition ()

  Wait until the acquisition finished.
- CCD_API unsigned int WaitForAcquisitionTimeOut (int timeout_ms)

  Wait the acquisition with a timeout.
- CCD_API unsigned int GetImage (void *buf, int buf_size)

  Get the image data.
- CCD_API unsigned int GetNthImage (void *buf, int buf_size, int idx)

  Get the image data, for continuous capture.

6.2.1 Detailed Description

6.2.2 Function Documentation

6.2.2.1 CancelWait()

```
CCD_API unsigned int CancelWait ( )
```

Cancel the WaitForAcquisition blocking.

Returns

- **CCD_SUCCESS**

6.2.2.2 GetExposureTime()

```
CCD_API unsigned int GetExposureTime (
            float * time )
```

Get exposure time.

Get exposure interval.

Parameters

pointer	to time exposure time in seconds.

Returns

- **CCD_SUCCESS** OK
- **CCD_ARG_ERROR** Argument format error
- **CCD_NOT_INIT** CCD not initialized
- **CCD_ACCESS_ERROR** Internal communication error

Parameters

pointer	to exposure interval time in seconds.

Returns

- **CCD_SUCCESS** OK
- **CCD_ARG_ERROR** Argument format error
- **CCD_NOT_INIT** CCD not initialized
- **CCD_ACCESS_ERROR** Internal communication error

Get exposure time.

Parameters

pointer	to exposure interval time in seconds.

Returns

- **CCD_SUCCESS** OK
- **CCD_ARG_ERROR** Argument format error
- **CCD_NOT_INIT** CCD not initialized
- **CCD_ACCESS_ERROR** Internal communication error

6.2.2.3 GetImage()

```
CCD_API unsigned int GetImage (
            void * buf,
            int buf_size )
```

Get the image data.

Parameters

buf	the buffer data will store into. You need to allocate it manually.
buf_size	the size of buf in byte

Returns

- **CCD_SUCCESS**
- **CCD_NO_IMAGE**

6.2.2.4 GetLDCMode()

```
CCD_API unsigned int GetLDCMode (
            uint8_t * ldc_bool )
```

Get LDC mode, as told.

Parameters

ldc_bool,a	pointer toward the result, 1 as on, 0 as off;

Returns

- **CCD_SUCCESS** **
- ****CCD_NOT_INIT**
- **CCD_ARG_ERROR**
- **CCD_ACCESS_ERROR**

6.2.2.5 GetNthImage()

```
CCD_API unsigned int GetNthImage (
            void * buf,
            int buf_size,
            int idx )
```

Get the image data, for continuous capture.

Parameters

buf	the buffer data will store into. You also need to allocate it manually.
buf_size	the size of buf in byte
idx	the index of which image to get

Returns

- **CCD_SUCCESS**
- **CCD_NO_IMAGE**

6.2.2.6 SetContinuousCapture()

```
CCD_API unsigned int SetContinuousCapture (
            uint16_t pic_num )
```

Set Capture Number for Continuous Capture.

Parameters

pic_num	The exact number of pics to take

Returns

- **CCD_SUCCESS**
- **CCD_NOT_INIT**
- **CCD_ARG_ERROR**
- **CCD_ACCESS_ERROR**

6.2.2.7 SetEraseCount()

```
CCD_API unsigned int SetEraseCount (
            uint8_t number )
```

Set the erase count before exposure.

Parameters

number	The erase count

Returns

- **CCD_SUCCESS**
- **CCD_ARG_ERROR**
- **CCD_NOT_INIT**
- **CCD_ACCESS_ERROR**

Todo No explicit range

6.2.2.8 SetExposureInterval()

```
CCD_API unsigned int SetExposureInterval (
            float time )
```

Set exposure interval.

Parameters

exposure	interval time in seconds.

Returns

- **CCD_SUCCESS** OK
- **CCD_ARG_ERROR** Argument format error
- **CCD_NOT_INIT** CCD not initialized
- **CCD_ACCESS_ERROR** Internal communication error

6.2.2.9 SetExposureTime()

```
CCD_API unsigned int SetExposureTime (
            float time )
```

Set exposure time.

Parameters

time	exposure time in seconds.

Returns

- **CCD_SUCCESS** OK
- **CCD_ARG_ERROR** Argument format error
- **CCD_NOT_INIT** CCD not initialized
- **CCD_ACCESS_ERROR** Internal communication error

6.2.2.10 SetPreAmpGain()

```
CCD_API unsigned int SetPreAmpGain (
            uint8_t gain )
```

Set gain mode.

Parameters

gain	• 0 low gain • 1 high gain

6.2.2.11 SetReadoutSpeed()

```
CCD_API unsigned int SetReadoutSpeed (
            int idx )
```

Set readout speed.

Parameters

speed	The readout speed in index, which can be retrived by GetNumReadoutSpeed

Returns

- **CCD_SUCCESS**
- **CCD_ARG_ERROR**
- **CCD_NOT_INIT**
- **CCD_ACCESS_ERROR**
- **CCD_OUTOFRANGE** Argument value is out of range.

6.2.2.12 SetShutter()

```
CCD_API unsigned int SetShutter (
            uint8_t mode )
```

Set shutter mode.

Parameters

mode	• 0 Open permanently • 1 Closed permanently • 2 Auto

Returns

- **CCD_SUCCESS**
- **CCD_NOT_INIT**
- **CCD_ARG_ERROR**
- **CCD_ACCESS_ERROR**

6.2.2.13 SetTriggerMode()

```
CCD_API unsigned int SetTriggerMode (
            uint8_t mode )
```

Set Trigger Mode.

Parameters

00	for trigger by software, 01 for ext trigger, 02 for ext pps triger, 03 for trigger by time

Returns

- **CCD_SUCCESS** OK
- **CCD_ARG_ERROR** Argument format error
- **CCD_NOT_INIT** CCD not initialized
- **CCD_ACCESS_ERROR** Internal communication error

6.2.2.14 StartExposure()

```
CCD_API unsigned int StartExposure ( )
```

Start exposure.

Returns

- **CCD_SUCCESS**
- **CCD_NOT_INIT**
- **CCD_ARG_ERROR**
- **CCD_ACCESS_ERROR**

6.2.2.15 StopExposure()

```
CCD_API unsigned int StopExposure ( )
```

Stop exposure.

Call this function when you need to break an exposure

Returns

- **CCD_

6.2.2.16 WaitForAcquisition()

```
CCD_API unsigned int WaitForAcquisition ( )
```

Wait until the acquisition finished.

Returns

- **CCD_SUCCESS**
- **CCD_NOT_INIT**
- **CCD_NO_IMAGE**

6.2.2.17 WaitForAcquisitionTimeOut()

```
CCD_API unsigned int WaitForAcquisitionTimeOut (
            int timeout_ms )
```

Wait the acquisition with a timeout.

Parameters

timeout_ms	timeout in ms

Returns

- **CCD_SUCCESS**
- **CCD_NOT_INIT**
- **CCD_NO_IMAGE**

6.3 Image saving functions

Classes

- struct MetaNode

  < *meta data node, will write to FITS*

Typedefs

- typedef struct MetaNode MetaNode

  < *meta data node, will write to FITS*

Functions

- CCD_API unsigned int SaveAsFITS (const char *file)

  Save the image as FITS file.
- CCD_API unsigned int SaveAsFITSEx (const char *file, uint16_t meta_num, MetaNode *meta_node)

  Save the image as FITS file with extra metadata.
- CCD_API unsigned int SaveNthAsFITS (const char *file, int idx)

  Save the image as FITS file, for continuous capture.
- CCD_API unsigned int SetAutoSaveFITS (int autosave, const char *file_path_name, MetaNode *meta_node, int meta_num)

  Set autosave file path and name.
- CCD_API unsigned int ConfigAutoSave (const char *cmd,...)

  Config options about SDK image data autosave.

6.3.1 Detailed Description

6.3.2 Function Documentation

6.3.2.1 ConfigAutoSave()

```
CCD_API unsigned int ConfigAutoSave (
            const char * cmd,
             ... )
```

Config options about SDK image data autosave.

Parameters

cmd	[in] string, option name

6.3.2.2 SaveAsFITS()

```
CCD_API unsigned int SaveAsFITS (
            const char * file )
```

Save the image as FITS file.

Parameters

file	a string, which is the name of file

6.3.2.3 SaveAsFITSEx()

```
CCD_API unsigned int SaveAsFITSEx (
            const char * file,
            uint16_t meta_num,
            MetaNode * meta_node )
```

Save the image as FITS file with extra metadata.

Parameters

file	[in] string, name of file
meta_num	[in] int, number of meta_node
meta_node	[in] meta node with `struct MetaNode`

6.3.2.4 SaveNthAsFITS()

```
CCD_API unsigned int SaveNthAsFITS (
            const char * file,
            int idx )
```

Save the image as FITS file, for continuous capture.

Parameters

file	a string, which is the name of file
idx,idx	represent the index of the captured file

6.3.2.5 SetAutoSaveFITS()

```
CCD_API unsigned int SetAutoSaveFITS (
            int autosave,
```

```
const char * file_path_name,
MetaNode * meta_node,
int meta_num )
```

Set autosave file path and name.

Parameters

autosave	[in] int, 1: autosave, 0: dont save
file_path_name	[in] string, output file location (contain directory name)
meta_node	[in] meta node with `struct MetaNode`
meta_num	[in] int, number of meta_node

6.4 Miscellaneous functions

Functions

- CCD_API unsigned int **SetLDCMode** ()
- CCD_API unsigned int **UnsetLDCMode** ()
- CCD_API unsigned int **LDCModeSet** (uint8_t onoff)
- CCD_API unsigned int **GetModelInfo** (uint8_t ∗, uint8_t ∗, uint8_t ∗)
- CCD_API unsigned int **CoolerOn** ()
- CCD_API unsigned int **CoolerOff** ()
- CCD_API unsigned int **CoolerSet** (uint8_t onoff)
- CCD_API unsigned int **SetCoolerTemp** (float temp)
- CCD_API unsigned int **PauseCool** ()
- CCD_API unsigned int **RecoverCool** ()
- CCD_API unsigned int **GetTemperature** (const char ∗temp_name, float ∗temp)
- CCD_API unsigned int **GetCurrent** (float ∗iboard, float ∗i5_5v, float ∗i24v, float ∗itec)
- CCD_API unsigned int **GetVoltageByChannel** (int vc, float ∗vol)
- CCD_API unsigned int **GetCoolerRange** (int ∗cooler_min, int ∗cooler_max)
- CCD_API unsigned int **GetDetector** (int ∗xpixels, int ∗ypixels)
- CCD_API unsigned int **GetBitDepth** (int ∗depth)
- CCD_API unsigned int **GetAcquisitionStatus** ()
- CCD_API unsigned int **ControllerFan** (uint8_t speed)
- CCD_API unsigned int **PowerFan** (uint8_t speed)
- CCD_API unsigned int **FanStatusSet** (uint8_t onoff)
- CCD_API unsigned int **VideoMode** (uint16_t photoNum)
- CCD_API unsigned int **ROIEnable** (int16_t rowStartNum, int16_t rowKeepNum, int16_t columnStartNum, int16_t columnKeepNum)
- CCD_API unsigned int **ROIDisable** ()
- CCD_API unsigned int **BINEnable** (uint16_t binReadRow, uint16_t binReadColumn)
- CCD_API unsigned int **BINDisable** ()
- CCD_API unsigned int **SetCurrImageSize** (bool imageDefault, int sizex, int sizey)
- CCD_API unsigned int **SpecialImageCircle** ()
- CCD_API unsigned int **PhotoMode** (uint8_t mode)
- CCD_API unsigned int **TransferCycle** (uint16_t cycle)
- CCD_API unsigned int **SetDriftMode** (uint8_t mode)
- CCD_API unsigned int **SetDriftSpeed** (float nline)
- CCD_API unsigned int **SetDCDS** (uint8_t T1, uint8_t T2, uint16_t T3, uint8_t T4)
- CCD_API unsigned int **SetDCDS_2** (uint8_t T1, uint8_t T2, uint16_t T3, uint8_t T4)
- CCD_API unsigned int **GetWaitingTime** (uint16_t readoutSpeed, float ∗waitingTime)
- CCD_API unsigned int **GetModel** (uint8_t ∗Num)
- CCD_API unsigned int **GetSerialNum** (uint8_t ∗Num)
- CCD_API unsigned int **GetLogicVer** (uint8_t ∗Num)
- CCD_API unsigned int **AnalogPowerOn** ()
- CCD_API unsigned int **AnalogPowerOff** ()
- CCD_API unsigned int **SetForcedTraining** ()
- CCD_API unsigned int **GetForcedTraining** (uint8_t ∗ft_bool)
- CCD_API unsigned int **SetIntegrateCycle** (uint16_t integrate_cycle)
- CCD_API unsigned int **GetEffectiveArea** (int ∗pstartX, int ∗pstartY, int ∗pendX, int ∗pendY)
- CCD_API unsigned int **ListCameras** (int ∗cam_num)
- CCD_API unsigned int **GetClockVoltage** (uint8_t mode, uint8_t channel, float ∗voltage)
- CCD_API unsigned int **PICMode** (uint8_t mode)
- CCD_API unsigned int **getPICMode** (uint8_t ∗mode)
- CCD_API unsigned int **setPID** (uint8_t p, uint8_t i, uint8_t d, uint8_t T)
- CCD_API unsigned int **SetBlackLevel** (uint16_t top_bl, uint16_t bottom_bl)

- CCD_API unsigned int **GetBlackLevel** (uint16_t *top_bl, uint16_t *bottom_bl)
- CCD_API unsigned int **SetExposureStartTime** (uint64_t sec, uint32_t nsec)
- CCD_API unsigned int **RecordExposureStartTime** (uint64_t sec, uint32_t nsec)
- CCD_API unsigned int **SetGain** (float gain_top, float gain_bot)
- CCD_API unsigned int **GetGain** (float *high_gain, float *low_gain)
- CCD_API unsigned int **GetGPSTime** (int *, int *, int *)
- CCD_API unsigned int **GetTDCTime** (uint32_t *)

6.4.1 Detailed Description

6.5 Error codes

Error codes.

Macros

- #define **PIXELX_ERRNO_MAP**(XX)

Enumerations

- enum pixelx_errno_t {
  PIXELX_SUCCESS = 20001, PIXELX_SCAN_ERROR = 20002, PIXELX_DEVICE_NOT_FOUND = 20003,
  PIXELX_FORMAT_ERROR = 20004,
  PIXELX_DEVICE_BUSY = 20005, PIXELX_ARG_ERROR = 20006, PIXELX_ACCESS_ERROR = 20007,
  PIXELX_NOT_INIT = 20008,
  PIXELX_SOFTWARE_BUG = 20009, PIXELX_OUTOFRANGE = 20010, PIXELX_TEMPERATURE_ERR↩
  OR = 20011, PIXELX_TEMPERATURE_STABILIZED = 20012,
  PIXELX_TEMPERATURE_NOT_REACHED = 20013, PIXELX_TEMPERATURE_OFF = 20014, PIXELX_↩
  NO_IMAGE = 20015, PIXELX_TIMEOUT = 20016,
  PIXELX_ACQUIRING = 20017, PIXELX_IMAGE_ALREADY_TRANSFERED = 20018, PIXELX_CONFIG↩
  _NOT_FOUND = 20019, PIXELX_NOT_IMPLEMENTED = 20020,
  PIXELX_IO_ERROR = 20021, PIXELX_ACCESS_ABORT = 20022 }

Functions

- CCD_API const char * CamErrorName (int errcode)

  Map the given error code to the name.
- CCD_API const char * CamErrorStrerror (int errcode)

  Return error detail message for the given error code.

6.5.1 Detailed Description

Error codes.

6.5.2 Macro Definition Documentation

6.5.2.1 PIXELX_ERRNO_MAP

```
#define PIXELX_ERRNO_MAP(
               XX )
```

Value:

```
XX(SUCCESS, "Success")  \
    XX(SCAN_ERROR, "Scan device error") \
    XX(DEVICE_NOT_FOUND, "Device not found")    \
    XX(FORMAT_ERROR, "File format error")   \
    XX(DEVICE_BUSY, "Device is in use") \
    XX(ARG_ERROR, "Argument out of range")  \
    XX(ACCESS_ERROR, "Connection error")    \
    XX(NOT_INIT, "CCD not initialized") \
    XX(SOFTWARE_BUG, "Software BUG")    \
    XX(OUTOFRANGE, "Parameter out of range")    \
    XX(TEMPERATURE_ERROR, "CCD temperature is not stabilized")  \
    XX(TEMPERATURE_STABILIZED, "CCD temperature is stabilized") \
    XX(TEMPERATURE_NOT_REACHED, "CCD temperature is not reached")   \
    XX(TEMPERATURE_OFF, "CCD cooler off")   \
    XX(NO_IMAGE, "No image data returned")  \
    XX(TIMEOUT, "Timeout during waiting for image") \
    XX(ACQUIRING, "Acquiring")  \
    XX(IMAGE_ALREADY_TRANSFERED, "File received already")   \
    XX(CONFIG_NOT_FOUND, "Config for the model not found")  \
    XX(NOT_IMPLEMENTED, "The function is unsupported")  \
    XX(IO_ERROR, "I/O failed")  \
    XX(ACCESS_ABORT, "Access failed")
```

6.5.3 Enumeration Type Documentation

6.5.3.1 pixelx_errno_t

```
enum pixelx_errno_t
```

Error enum generated by `def_error.py`

Enumerator

PIXELX_SUCCESS	Success
PIXELX_SCAN_ERROR	Scan device error
PIXELX_DEVICE_NOT_FOUND	Device not found
PIXELX_FORMAT_ERROR	File format error
PIXELX_DEVICE_BUSY	Device is in use
PIXELX_ARG_ERROR	Argument out of range
PIXELX_ACCESS_ERROR	Connection error
PIXELX_NOT_INIT	CCD not initialized
PIXELX_SOFTWARE_BUG	Software BUG
PIXELX_OUTOFRANGE	Parameter out of range
PIXELX_TEMPERATURE_ERROR	CCD temperature is not stabilized
PIXELX_TEMPERATURE_STABILIZED	CCD temperature is stabilized
PIXELX_TEMPERATURE_NOT_REACHED	CCD temperature is not reached
PIXELX_TEMPERATURE_OFF	CCD cooler off
PIXELX_NO_IMAGE	No image data returned
PIXELX_TIMEOUT	Timeout during waiting for image

Enumerator

PIXELX_ACQUIRING	Acquiring
PIXELX_IMAGE_ALREADY_TRANSFERED	File received already
PIXELX_CONFIG_NOT_FOUND	Config for the model not found
PIXELX_NOT_IMPLEMENTED	The function is unsupported
PIXELX_IO_ERROR	I/O failed
PIXELX_ACCESS_ABORT	Access failed

6.6 SerialNumber

Classes

- struct PIXELX_SERIAL

Functions

- CCD_API unsigned int **SetSerialNumber** (PIXELX_SERIAL *pse)
- CCD_API unsigned int **GetSerialNumber** (PIXELX_SERIAL *pse)

6.6.1 Detailed Description

Chapter 7

Class Documentation

7.1 ADC_Channel Struct Reference

Public Attributes

- std::string **name**
- int **is_clock**
- int **channel**

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

- /home/jerryjia/CCDSDK2/_private.h

7.2 CCD_ACQ_Set Struct Reference

Public Attributes

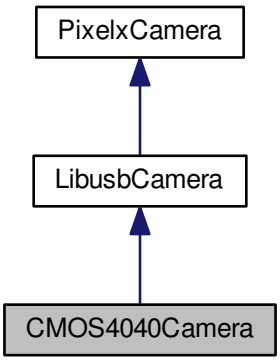
- std::string **setName**
- int16_t **coolerTarget**
- uint64_t **expoTime**
- uint16_t **readspd**
- uint16_t **gain**
- uint16_t **integrateCycle**
- uint16_t **transferCycle**
- int16_t **binParams** [2]
- int16_t **roiParams** [4]
- uint16_t **video**

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

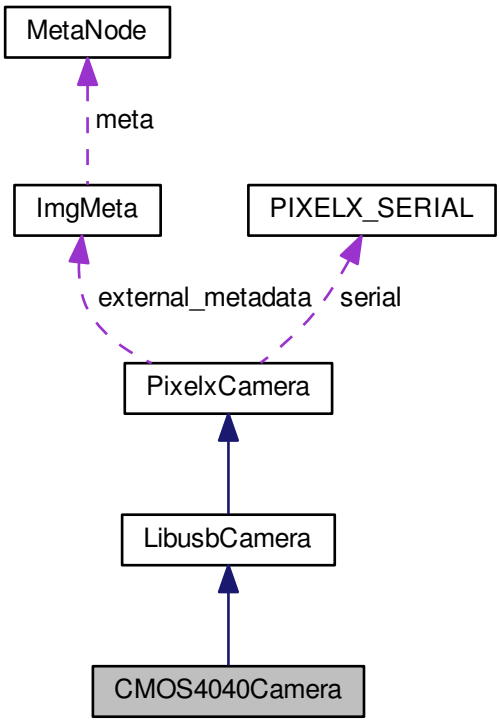
- /home/jerryjia/CCDSDK2/_private.h

7.3 CMOS4040Camera Class Reference

Inheritance diagram for CMOS4040Camera:



Collaboration diagram for CMOS4040Camera:



Public Member Functions

- void **addExMetaToFITS** (fitsfile *fptr, struct ImgMeta meta, int *status)
- unsigned int **GetTemperature** (const char *temp_name, float *temp) override
- unsigned int **SetExposureTime** (float time) override
- unsigned int **SetCoolerTemp** (float temp) override
- unsigned int ROIEnable (int16_t rowStartNum, int16_t rowKeepNum, int16_t columnStartNum, int16_t columnKeepNum) override
- unsigned int **ROIDisable** () override
- unsigned int **GetDetector** (int *xpixels, int *ypixels) override
- unsigned int **GetNthImage** (void *buf, int buf_size, int idx) override
- unsigned int SetContinuousCapture (uint16_t rowCaptureNum) override
- unsigned int **PICMode** (uint8_t mode) override
- void **CopyToImageBuffer** (uint8_t *buf, int imageWidth, int imageHeight, uint8_t pixelDepth) override
- void **CopyToImageBuffer_withDest** (uint8_t *buf, int imageWidth, int imageHeight, uint8_t pixelDepth, uint8_t *dest, uint8_t pic_index)
- virtual unsigned int **SetAutoSaveDir** (const char *dir) override
- virtual unsigned int **SetAutoSave** (int if_auto_save) override
- virtual void image_thread_func () override

  Thread function for reading image. image_data, image_size, image_mutex and transfer_cond should be used.
- void **video_thread_func** ()
- virtual unsigned int **StopExposure** () override
- virtual unsigned int **StartExposure** () override
- unsigned int **FanStatusSet** (uint8_t onoff) override
- unsigned int **ControllerFan** (uint8_t speed) override
- unsigned int **ShutDown** () override
- unsigned int **CoolerOn** () override
- unsigned int **CoolerOff** () override
- unsigned int **GetCurrent** (float *iboard, float *i5_5v, float *i24v, float *itec) override
- unsigned int **GetVoltageByChannel** (int vc, float *vol) override
- unsigned int **GetCurrentByChannel** (int vc, float *cur) override
- unsigned int **GetGPSTime** (int &sec, int &min, int &hour) override
- unsigned int **GetGPSEpochTime** (time_t &t) override
- unsigned int **GetGPSEpochTime** (struct tm &expotime)
- unsigned int **GetTDCTime** (uint32_t &tdc_count) override
- unsigned int **GetPPSRise** (uint32_t &pps_rise)
- virtual unsigned **SaveNthAsFITS** (const char *file, int idx, struct ImgMeta *pmeta=nullptr) override
- virtual unsigned **SaveAsFITS** (const char *file, struct ImgMeta meta={}) override
- unsigned int **SaveAsFitsWithSpecifiedMeta** (uint8_t *, long, long, const char *, std::map< std::string, std::vector< std::string >> metas)
- unsigned int **SetTriggerMode** (uint8_t mode) override
- virtual unsigned int **SetExposureStartTime** (uint64_t sec, uint32_t nsec) override
- void **setConfigFile** () override
- void **setConfigurations** () override
- unsigned int **VideoMode** (uint16_t) override
- unsigned int **BINEnable** (uint16_t, uint16_t) override
- unsigned int **BINDisable** () override
- unsigned int **software_2X2BIN** (uint8_t *dest, uint8_t *from, int width, int length)
- unsigned int **Dummy** ()
- unsigned int **DownloadFpgaBitStream** (const char *filename)
- unsigned int **combine_a_download_package** (const uint16_t, const uint16_t, const uint8_t *, size_t, uint8_t *, size_t buf_len, size_t *act)

Protected Attributes

- std::string **save_prefix**
- std::string **save_dir**
- std::vector< std::string > **last_save_file_list**

Additional Inherited Members

7.3.1 Member Function Documentation

7.3.1.1 ROIEnable()

```
unsigned int CMOS4040Camera::ROIEnable (
            int16_t rowStartNum,
            int16_t rowKeepNum,
            int16_t columnStartNum,
            int16_t columnKeepNum )  [override], [virtual]
```

Set the Capture Region of CMOS4040 camera

Parameters

rowStartNum	the start position of capturing, 0-4095
rowEndNum	the End position of capturing, 0-4096, must be more than rowStartNumtwo uint16 parameters while be splite into LSB8bit and MSB8bit forming into the real cmd

An example StartRow = 100 0x0064 , EndRow = 2000 0x07D0 CMD: 0x84C0 0x00D0 0x2007 0x4064 0x6000

Reimplemented from PixelxCamera.

7.3.1.2 SetContinuousCapture()

```
unsigned int CMOS4040Camera::SetContinuousCapture (
            uint16_t rowCaptureNum )  [override], [virtual]
```

Set the Continously Capture Mode by Rows num

Parameters

rowCaptureNum	the num of rows to captur

Returns

return PIXELX_SUCCESS if ok

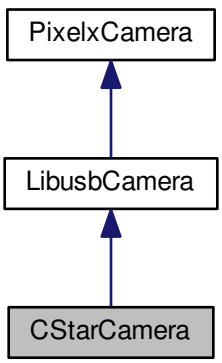
Reimplemented from PixelxCamera.

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

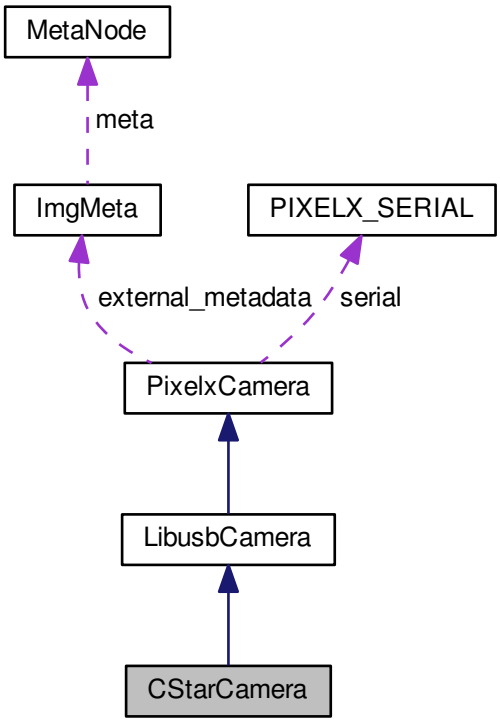
- /home/jerryjia/CCDSDK2/cmos4040/cmos4040.h
- /home/jerryjia/CCDSDK2/cmos4040/cmos4040.cpp

7.4 CStarCamera Class Reference

Inheritance diagram for CStarCamera:



Collaboration diagram for CStarCamera:



Public Member Functions

- void **setConfigurations** () override
- void **setConfigFile** () override

Additional Inherited Members

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

- /home/jerryjia/CCDSDK2/cstar/cstar.h
- /home/jerryjia/CCDSDK2/cstar/cstar.cpp

7.5 def_error.ErrorDef Class Reference

Public Member Functions

- def **update** ()

Static Public Member Functions

- def **define** (key, desc, detail=None)

Static Public Attributes

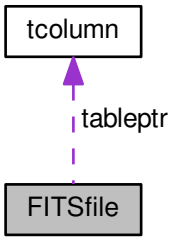
- int **CNT** = 20000
- string **PREFIX** = 'PIXELX'
- string **TYPEDEF** = ''
- string **ERRMAP** = ''
- list **GEN_TYPEDEF** = ["typedef enum {",]
- list **GEN_TYPEDEF_END** = ["} pixelx_errno_t;",]
- list **GEN_XX** = [ ]

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

- /home/jerryjia/CCDSDK2/def_error.py

7.6 FITSfile Struct Reference

Collaboration diagram for FITSfile:



Public Attributes

- int **filehandle**
- int **driver**
- int **open_count**
- char * **filename**
- int **validcode**
- int **only_one**
- LONGLONG **filesize**
- LONGLONG **logfilesize**
- int **lasthdu**
- LONGLONG **bytepos**
- LONGLONG **io_pos**

- int **curbuf**
- int **curhdu**
- int **hdutype**
- int **writemode**
- int **maxhdu**
- int **MAXHDU**
- LONGLONG * **headstart**
- LONGLONG **headend**
- LONGLONG **ENDpos**
- LONGLONG **nextkey**
- LONGLONG **datastart**
- int **imgdim**
- LONGLONG **imgnaxis** [99]
- int **tfield**
- int **startcol**
- LONGLONG **origrows**
- LONGLONG **numrows**
- LONGLONG **rowlength**
- tcolumn * **tableptr**
- LONGLONG **heapstart**
- LONGLONG **heapsize**
- int **request_compress_type**
- long **request_tilesize** [MAX_COMPRESS_DIM]
- float **request_quantize_level**
- int **request_quantize_method**
- int **request_dither_seed**
- int **request_lossy_int_compress**
- int **request_huge_hdu**
- float **request_hcomp_scale**
- int **request_hcomp_smooth**
- int **compress_type**
- long **tilesize** [MAX_COMPRESS_DIM]
- float **quantize_level**
- int **quantize_method**
- int **dither_seed**
- int **compressimg**
- char **zcmptype** [12]
- int **zbitpix**
- int **zndim**
- long **znaxis** [MAX_COMPRESS_DIM]
- long **maxtilelen**
- long **maxelem**
- int **cn_compressed**
- int **cn_uncompressed**
- int **cn_gzip_data**
- int **cn_zscale**
- int **cn_zzero**
- int **cn_zblank**
- double **zscale**
- double **zzero**
- double **cn_bscale**
- double **cn_bzero**
- double **cn_actual_bzero**
- int **zblank**
- int **rice_blocksize**

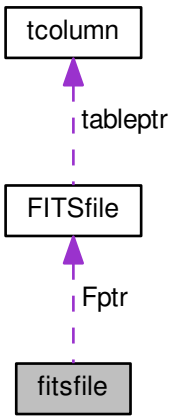
- int **rice_bytepix**
- float **hcomp_scale**
- int **hcomp_smooth**
- int * **tilerow**
- long * **tiledatasize**
- int * **tiletype**
- void ** **tiledata**
- void ** **tilenullarray**
- int * **tileanynull**
- char * **iobuffer**
- long **bufrecnum** [NIOBUF]
- int **dirty** [NIOBUF]
- int **ageindex** [NIOBUF]

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

- /home/jerryjia/CCDSDK2/lib/include/fitsio.h

7.7 fitsfile Struct Reference

Collaboration diagram for fitsfile:



Public Attributes

- int **HDUposition**
- FITSfile * **Fptr**

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

- /home/jerryjia/CCDSDK2/lib/include/fitsio.h

7.8 FloatComp Struct Reference

Public Member Functions

- bool **operator()** (const float &lhs, const float &rhs) const

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

- /home/jerryjia/CCDSDK2/PixelxCamera.cpp

7.9 fx_known_device Struct Reference

Public Attributes

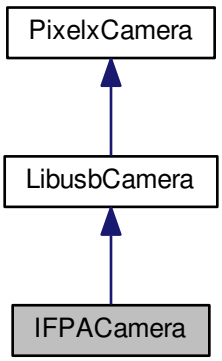
- uint16_t **vid**
- uint16_t **pid**
- int **type**
- const char * **designation**

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

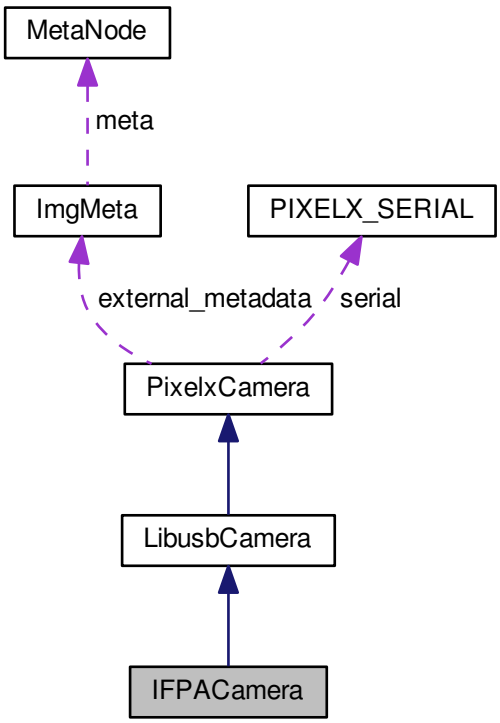
- /home/jerryjia/CCDSDK2/ezusb.h

7.10 IFPACamera Class Reference

Inheritance diagram for IFPACamera:



Collaboration diagram for IFPACamera:



Public Member Functions

- virtual void image_thread_func () override

  Thread function for reading image. image_data, image_size, image_mutex and transfer_cond should be used.
- void **setConfigFile** () override
- void **setConfigurations** () override

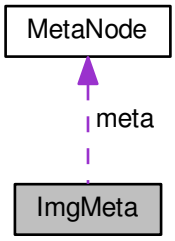
Additional Inherited Members

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

- /home/jerryjia/CCDSDK2/IFPA/ifpa.h
- /home/jerryjia/CCDSDK2/IFPA/ifpa.cpp

7.11 ImgMeta Struct Reference

Collaboration diagram for ImgMeta:



Public Attributes

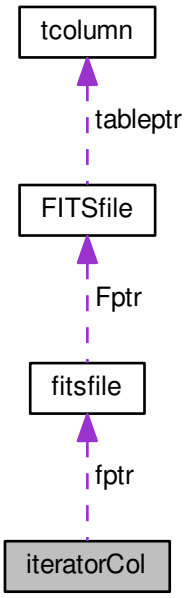
- int **meta_num**
- MetaNode * **meta**

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

- /home/jerryjia/CCDSDK2/_private.h

7.12 iteratorCol Struct Reference

Collaboration diagram for iteratorCol:



Public Attributes

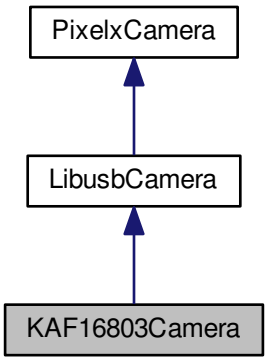
- fitsfile * **fptr**
- int **colnum**
- char **colname** [70]
- int **datatype**
- int **iotype**
- void * **array**
- long **repeat**
- long **tlmin**
- long **tlmax**
- char **tunit** [70]
- char **tdisp** [70]

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

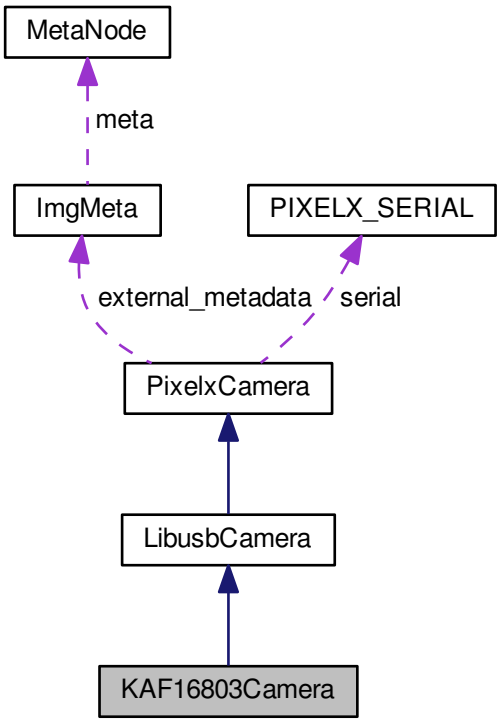
- /home/jerryjia/CCDSDK2/lib/include/fitsio.h

7.13 KAF16803Camera Class Reference

Inheritance diagram for KAF16803Camera:



Collaboration diagram for KAF16803Camera:



Public Member Functions

- void **setConfigurations** () override
- void **setConfigFile** () override

Additional Inherited Members

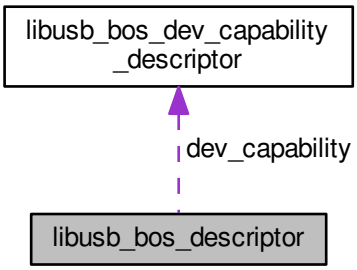
The documentation for this class was generated from the following files:

- /home/jerryjia/CCDSDK2/KAF16803/kaf16803.h
- /home/jerryjia/CCDSDK2/KAF16803/kaf16803.cpp

7.14 libusb_bos_descriptor Struct Reference

`#include <libusb.h>`

Collaboration diagram for libusb_bos_descriptor:



Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint16_t wTotalLength
- uint8_t bNumDeviceCaps
- struct libusb_bos_dev_capability_descriptor * dev_capability [0]

7.14.1 Detailed Description

A structure representing the Binary Device Object Store (BOS) descriptor. This descriptor is documented in section 9.6.2 of the USB 3.0 specification. All multiple-byte fields are represented in host-endian format.

7.14.2 Member Data Documentation

7.14.2.1 bDescriptorType

```
uint8_t libusb_bos_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_BOS LIBUSB_DT_BOS in this context.

7.14.2.2 bLength

```
uint8_t libusb_bos_descriptor::bLength
```

Size of this descriptor (in bytes)

7.14.2.3 bNumDeviceCaps

```
uint8_t libusb_bos_descriptor::bNumDeviceCaps
```

The number of separate device capability descriptors in the BOS

7.14.2.4 dev_capability

```
struct libusb_bos_dev_capability_descriptor* libusb_bos_descriptor::dev_capability[0]
```

bNumDeviceCap Device Capability Descriptors

7.14.2.5 wTotalLength

```
uint16_t libusb_bos_descriptor::wTotalLength
```

Length of this descriptor and all of its sub descriptors

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.15 libusb_bos_dev_capability_descriptor Struct Reference

```
#include <libusb.h>
```

Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint8_t bDevCapabilityType
- uint8_t dev_capability_data [0]

7.15.1 Detailed Description

A generic representation of a BOS Device Capability descriptor. It is advised to check bDevCapabilityType and call the matching libusb_get_*_descriptor function to get a structure fully matching the type.

7.15.2 Member Data Documentation

7.15.2.1 bDescriptorType

```
uint8_t libusb_bos_dev_capability_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_DEVICE_CAPABILITY LIBUSB_DT_DEVICE_CAPABILITY in this context.

7.15.2.2 bDevCapabilityType

```
uint8_t libusb_bos_dev_capability_descriptor::bDevCapabilityType
```

Device Capability type

7.15.2.3 bLength

```
uint8_t libusb_bos_dev_capability_descriptor::bLength
```

Size of this descriptor (in bytes)

7.15.2.4 dev_capability_data

```
uint8_t libusb_bos_dev_capability_descriptor::dev_capability_data[0]
```

Device Capability data (bLength - 3 bytes)

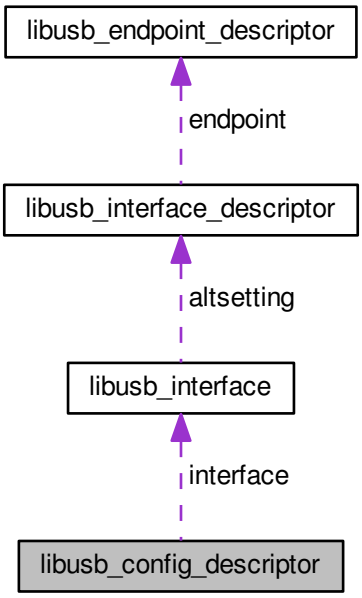
The documentation for this struct was generated from the following file:

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.16 libusb_config_descriptor Struct Reference

`#include <libusb.h>`

Collaboration diagram for libusb_config_descriptor:



Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint16_t wTotalLength
- uint8_t bNumInterfaces
- uint8_t bConfigurationValue
- uint8_t iConfiguration
- uint8_t bmAttributes
- uint8_t MaxPower
- const struct libusb_interface * interface
- const unsigned char * extra
- int extra_length

7.16.1 Detailed Description

A structure representing the standard USB configuration descriptor. This descriptor is documented in section 9.6.3 of the USB 3.0 specification. All multiple-byte fields are represented in host-endian format.

7.16.2 Member Data Documentation

7.16.2.1 bConfigurationValue

```
uint8_t libusb_config_descriptor::bConfigurationValue
```

Identifier value for this configuration

7.16.2.2 bDescriptorType

```
uint8_t libusb_config_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_CONFIG LIBUSB_DT_CONFIG in this context.

7.16.2.3 bLength

```
uint8_t libusb_config_descriptor::bLength
```

Size of this descriptor (in bytes)

7.16.2.4 bmAttributes

```
uint8_t libusb_config_descriptor::bmAttributes
```

Configuration characteristics

7.16.2.5 bNumInterfaces

```
uint8_t libusb_config_descriptor::bNumInterfaces
```

Number of interfaces supported by this configuration

7.16.2.6 extra

```
const unsigned char* libusb_config_descriptor::extra
```

Extra descriptors. If libusb encounters unknown configuration descriptors, it will store them here, should you wish to parse them.

7.16.2.7 extra_length

```
int libusb_config_descriptor::extra_length
```

Length of the extra descriptors, in bytes.

7.16.2.8 iConfiguration

```
uint8_t libusb_config_descriptor::iConfiguration
```

Index of string descriptor describing this configuration

7.16.2.9 interface

```
const struct libusb_interface* libusb_config_descriptor::interface
```

Array of interfaces supported by this configuration. The length of this array is determined by the bNumInterfaces field.

7.16.2.10 MaxPower

```
uint8_t libusb_config_descriptor::MaxPower
```

Maximum power consumption of the USB device from this bus in this configuration when the device is fully operation. Expressed in units of 2 mA when the device is operating in high-speed mode and in units of 8 mA when the device is operating in super-speed mode.

7.16.2.11 wTotalLength

```
uint16_t libusb_config_descriptor::wTotalLength
```

Total length of data returned for this configuration

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.17 libusb_container_id_descriptor Struct Reference

```
#include <libusb.h>
```

Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint8_t bDevCapabilityType
- uint8_t bReserved
- uint8_t ContainerID [16]

7.17.1 Detailed Description

A structure representing the Container ID descriptor. This descriptor is documented in section 9.6.2.3 of the USB 3.0 specification. All multiple-byte fields, except UUIDs, are represented in host-endian format.

7.17.2 Member Data Documentation

7.17.2.1 bDescriptorType

```
uint8_t libusb_container_id_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_DEVICE_CAPABILITY LIBUSB_DT_DEVI↩ CE_CAPABILITY in this context.

7.17.2.2 bDevCapabilityType

```
uint8_t libusb_container_id_descriptor::bDevCapabilityType
```

Capability type. Will have value libusb_capability_type::LIBUSB_BT_CONTAINER_ID LIBUSB_BT_CONTAINE↩ R_ID in this context.

7.17.2.3 bLength

```
uint8_t libusb_container_id_descriptor::bLength
```

Size of this descriptor (in bytes)

7.17.2.4 bReserved

```
uint8_t libusb_container_id_descriptor::bReserved
```

Reserved field

7.17.2.5 ContainerID

```
uint8_t libusb_container_id_descriptor::ContainerID[16]
```

128 bit UUID

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.18 libusb_control_setup Struct Reference

```
#include <libusb.h>
```

Public Attributes

- uint8_t bmRequestType
- uint8_t bRequest
- uint16_t wValue
- uint16_t wIndex
- uint16_t wLength

7.18.1 Detailed Description

Setup packet for control transfers.

7.18.2 Member Data Documentation

7.18.2.1 bmRequestType

```
uint8_t libusb_control_setup::bmRequestType
```

Request type. Bits 0:4 determine recipient, see libusb_request_recipient. Bits 5:6 determine type, see libusb_↩ request_type. Bit 7 determines data transfer direction, see libusb_endpoint_direction.

7.18.2.2 bRequest

```
uint8_t libusb_control_setup::bRequest
```

Request. If the type bits of bmRequestType are equal to LIBUSB_REQUEST_TYPE_STANDARD then this field refers to libusb_standard_request. For other cases, use of this field is application-specific.

7.18.2.3 wIndex

```
uint16_t libusb_control_setup::wIndex
```

Index. Varies according to request, typically used to pass an index or offset

7.18.2.4 wLength

```
uint16_t libusb_control_setup::wLength
```

Number of bytes to transfer

7.18.2.5 wValue

```
uint16_t libusb_control_setup::wValue
```

Value. Varies according to request

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.19 libusb_device_descriptor Struct Reference

```
#include <libusb.h>
```

Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint16_t bcdUSB
- uint8_t bDeviceClass
- uint8_t bDeviceSubClass
- uint8_t bDeviceProtocol
- uint8_t bMaxPacketSize0
- uint16_t idVendor
- uint16_t idProduct
- uint16_t bcdDevice
- uint8_t iManufacturer
- uint8_t iProduct
- uint8_t iSerialNumber
- uint8_t bNumConfigurations

7.19.1 Detailed Description

A structure representing the standard USB device descriptor. This descriptor is documented in section 9.6.1 of the USB 3.0 specification. All multiple-byte fields are represented in host-endian format.

7.19.2 Member Data Documentation

7.19.2.1 bcdDevice

```
uint16_t libusb_device_descriptor::bcdDevice
```

Device release number in binary-coded decimal

7.19.2.2 bcdUSB

```
uint16_t libusb_device_descriptor::bcdUSB
```

USB specification release number in binary-coded decimal. A value of 0x0200 indicates USB 2.0, 0x0110 indicates USB 1.1, etc.

7.19.2.3 bDescriptorType

```
uint8_t libusb_device_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_DEVICE LIBUSB_DT_DEVICE in this context.

7.19.2.4 bDeviceClass

```
uint8_t libusb_device_descriptor::bDeviceClass
```

USB-IF class code for the device. See libusb_class_code.

7.19.2.5 bDeviceProtocol

```
uint8_t libusb_device_descriptor::bDeviceProtocol
```

USB-IF protocol code for the device, qualified by the bDeviceClass and bDeviceSubClass values

7.19.2.6 bDeviceSubClass

```
uint8_t libusb_device_descriptor::bDeviceSubClass
```

USB-IF subclass code for the device, qualified by the bDeviceClass value

7.19.2.7 bLength

```
uint8_t libusb_device_descriptor::bLength
```

Size of this descriptor (in bytes)

7.19.2.8 bMaxPacketSize0

```
uint8_t libusb_device_descriptor::bMaxPacketSize0
```

Maximum packet size for endpoint 0

7.19.2.9 bNumConfigurations

```
uint8_t libusb_device_descriptor::bNumConfigurations
```

Number of possible configurations

7.19.2.10 idProduct

```
uint16_t libusb_device_descriptor::idProduct
```

USB-IF product ID

7.19.2.11 idVendor

```
uint16_t libusb_device_descriptor::idVendor
```

USB-IF vendor ID

7.19.2.12 iManufacturer

```
uint8_t libusb_device_descriptor::iManufacturer
```

Index of string descriptor describing manufacturer

7.19.2.13 iProduct

```
uint8_t libusb_device_descriptor::iProduct
```

Index of string descriptor describing product

7.19.2.14 iSerialNumber

```
uint8_t libusb_device_descriptor::iSerialNumber
```

Index of string descriptor containing device serial number

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.20 libusb_endpoint_descriptor Struct Reference

```
#include <libusb.h>
```

Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint8_t bEndpointAddress
- uint8_t bmAttributes
- uint16_t wMaxPacketSize
- uint8_t bInterval
- uint8_t bRefresh
- uint8_t bSynchAddress
- const unsigned char * extra
- int extra_length

7.20.1 Detailed Description

A structure representing the standard USB endpoint descriptor. This descriptor is documented in section 9.6.6 of the USB 3.0 specification. All multiple-byte fields are represented in host-endian format.

7.20.2 Member Data Documentation

7.20.2.1 bDescriptorType

```
uint8_t libusb_endpoint_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_ENDPOINT LIBUSB_DT_ENDPOINT in this context.

7.20.2.2 bEndpointAddress

```
uint8_t libusb_endpoint_descriptor::bEndpointAddress
```

The address of the endpoint described by this descriptor. Bits 0:3 are the endpoint number. Bits 4:6 are reserved. Bit 7 indicates direction, see libusb_endpoint_direction.

7.20.2.3 bInterval

```
uint8_t libusb_endpoint_descriptor::bInterval
```

Interval for polling endpoint for data transfers.

7.20.2.4 bLength

```
uint8_t libusb_endpoint_descriptor::bLength
```

Size of this descriptor (in bytes)

7.20.2.5 bmAttributes

```
uint8_t libusb_endpoint_descriptor::bmAttributes
```

Attributes which apply to the endpoint when it is configured using the bConfigurationValue. Bits 0:1 determine the transfer type and correspond to libusb_transfer_type. Bits 2:3 are only used for isochronous endpoints and correspond to libusb_iso_sync_type. Bits 4:5 are also only used for isochronous endpoints and correspond to libusb_iso_usage_type. Bits 6:7 are reserved.

7.20.2.6 bRefresh

```
uint8_t libusb_endpoint_descriptor::bRefresh
```

For audio devices only: the rate at which synchronization feedback is provided.

7.20.2.7 bSynchAddress

```
uint8_t libusb_endpoint_descriptor::bSynchAddress
```

For audio devices only: the address if the synch endpoint

7.20.2.8 extra

```
const unsigned char* libusb_endpoint_descriptor::extra
```

Extra descriptors. If libusb encounters unknown endpoint descriptors, it will store them here, should you wish to parse them.

7.20.2.9 extra_length

```
int libusb_endpoint_descriptor::extra_length
```

Length of the extra descriptors, in bytes.

7.20.2.10 wMaxPacketSize

```
uint16_t libusb_endpoint_descriptor::wMaxPacketSize
```

Maximum packet size this endpoint is capable of sending/receiving.

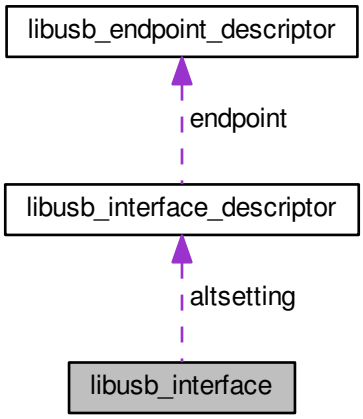
The documentation for this struct was generated from the following file:

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.21 libusb_interface Struct Reference

`#include <libusb.h>`

Collaboration diagram for libusb_interface:



Public Attributes

- const struct libusb_interface_descriptor * altsetting
- int num_altsetting

7.21.1 Detailed Description

A collection of alternate settings for a particular USB interface.

7.21.2 Member Data Documentation

7.21.2.1 altsetting

`const struct libusb_interface_descriptor* libusb_interface::altsetting`

Array of interface descriptors. The length of this array is determined by the num_altsetting field.

7.21.2.2 num_altsetting

```
int libusb_interface::num_altsetting
```

The number of alternate settings that belong to this interface

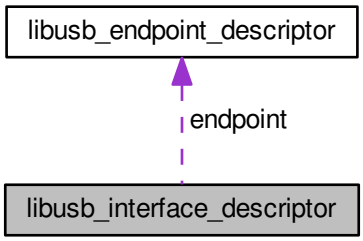
The documentation for this struct was generated from the following file:

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.22 libusb_interface_descriptor Struct Reference

```
#include <libusb.h>
```

Collaboration diagram for libusb_interface_descriptor:



Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint8_t bInterfaceNumber
- uint8_t bAlternateSetting
- uint8_t bNumEndpoints
- uint8_t bInterfaceClass
- uint8_t bInterfaceSubClass
- uint8_t bInterfaceProtocol
- uint8_t iInterface
- const struct libusb_endpoint_descriptor * endpoint
- const unsigned char * extra
- int extra_length

7.22.1 Detailed Description

A structure representing the standard USB interface descriptor. This descriptor is documented in section 9.6.5 of the USB 3.0 specification. All multiple-byte fields are represented in host-endian format.

7.22.2 Member Data Documentation

7.22.2.1 bAlternateSetting

```
uint8_t libusb_interface_descriptor::bAlternateSetting
```

Value used to select this alternate setting for this interface

7.22.2.2 bDescriptorType

```
uint8_t libusb_interface_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_INTERFACE LIBUSB_DT_INTERFACE in this context.

7.22.2.3 bInterfaceClass

```
uint8_t libusb_interface_descriptor::bInterfaceClass
```

USB-IF class code for this interface. See libusb_class_code.

7.22.2.4 bInterfaceNumber

```
uint8_t libusb_interface_descriptor::bInterfaceNumber
```

Number of this interface

7.22.2.5 bInterfaceProtocol

```
uint8_t libusb_interface_descriptor::bInterfaceProtocol
```

USB-IF protocol code for this interface, qualified by the bInterfaceClass and bInterfaceSubClass values

7.22.2.6 bInterfaceSubClass

```
uint8_t libusb_interface_descriptor::bInterfaceSubClass
```

USB-IF subclass code for this interface, qualified by the bInterfaceClass value

7.22.2.7 bLength

```
uint8_t libusb_interface_descriptor::bLength
```

Size of this descriptor (in bytes)

7.22.2.8 bNumEndpoints

```
uint8_t libusb_interface_descriptor::bNumEndpoints
```

Number of endpoints used by this interface (excluding the control endpoint).

7.22.2.9 endpoint

```
const struct libusb_endpoint_descriptor* libusb_interface_descriptor::endpoint
```

Array of endpoint descriptors. This length of this array is determined by the bNumEndpoints field.

7.22.2.10 extra

```
const unsigned char* libusb_interface_descriptor::extra
```

Extra descriptors. If libusb encounters unknown interface descriptors, it will store them here, should you wish to parse them.

7.22.2.11 extra_length

```
int libusb_interface_descriptor::extra_length
```

Length of the extra descriptors, in bytes.

7.22.2.12 iInterface

```
uint8_t libusb_interface_descriptor::iInterface
```

Index of string descriptor describing this interface

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.23 libusb_iso_packet_descriptor Struct Reference

```
#include <libusb.h>
```

Public Attributes

- unsigned int length
- unsigned int actual_length
- enum libusb_transfer_status status

7.23.1 Detailed Description

Isochronous packet descriptor.

7.23.2 Member Data Documentation

7.23.2.1 actual_length

```
unsigned int libusb_iso_packet_descriptor::actual_length
```

Amount of data that was actually transferred

7.23.2.2 length

```
unsigned int libusb_iso_packet_descriptor::length
```

Length of data to request in this packet

7.23.2.3 status

```
enum libusb_transfer_status libusb_iso_packet_descriptor::status
```

Status code for this packet

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.24 libusb_pollfd Struct Reference

```
#include <libusb.h>
```

Public Attributes

- int fd
- short events

7.24.1 Detailed Description

File descriptor for polling

7.24.2 Member Data Documentation

7.24.2.1 events

```
short libusb_pollfd::events
```

Event flags to poll for from <poll.h>. POLLIN indicates that you should monitor this file descriptor for becoming ready to read from, and POLLOUT indicates that you should monitor this file descriptor for nonblocking write readiness.

7.24.2.2 fd

```
int libusb_pollfd::fd
```

Numeric file descriptor

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.25 libusb_ss_endpoint_companion_descriptor Struct Reference

```
#include <libusb.h>
```

Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint8_t bMaxBurst
- uint8_t bmAttributes
- uint16_t wBytesPerInterval

7.25.1 Detailed Description

A structure representing the superspeed endpoint companion descriptor. This descriptor is documented in section 9.6.7 of the USB 3.0 specification. All multiple-byte fields are represented in host-endian format.

7.25.2 Member Data Documentation

7.25.2.1 bDescriptorType

```
uint8_t libusb_ss_endpoint_companion_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_SS_ENDPOINT_COMPANION in this context.

7.25.2.2 bLength

```
uint8_t libusb_ss_endpoint_companion_descriptor::bLength
```

Size of this descriptor (in bytes)

7.25.2.3 bmAttributes

```
uint8_t libusb_ss_endpoint_companion_descriptor::bmAttributes
```

In bulk EP: bits 4:0 represents the maximum number of streams the EP supports. In isochronous EP: bits 1:0 represents the Mult - a zero based value that determines the maximum number of packets within a service interval

7.25.2.4 bMaxBurst

```
uint8_t libusb_ss_endpoint_companion_descriptor::bMaxBurst
```

The maximum number of packets the endpoint can send or recieve as part of a burst.

7.25.2.5 wBytesPerInterval

```
uint16_t libusb_ss_endpoint_companion_descriptor::wBytesPerInterval
```

The total number of bytes this EP will transfer every service interval. valid only for periodic EPs.

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.26 libusb_ss_usb_device_capability_descriptor Struct Reference

```
#include <libusb.h>
```

Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint8_t bDevCapabilityType
- uint8_t bmAttributes
- uint16_t wSpeedSupported
- uint8_t bFunctionalitySupport
- uint8_t bU1DevExitLat
- uint16_t bU2DevExitLat

7.26.1 Detailed Description

A structure representing the SuperSpeed USB Device Capability descriptor This descriptor is documented in section 9.6.2.2 of the USB 3.0 specification. All multiple-byte fields are represented in host-endian format.

7.26.2 Member Data Documentation

7.26.2.1 bDescriptorType

```
uint8_t libusb_ss_usb_device_capability_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_DEVICE_CAPABILITY LIBUSB_DT_DEVI↩ CE_CAPABILITY in this context.

7.26.2.2 bDevCapabilityType

```
uint8_t libusb_ss_usb_device_capability_descriptor::bDevCapabilityType
```

Capability type. Will have value libusb_capability_type::LIBUSB_BT_SS_USB_DEVICE_CAPABILITY LIBUSB_↩ BT_SS_USB_DEVICE_CAPABILITY in this context.

7.26.2.3 bFunctionalitySupport

```
uint8_t libusb_ss_usb_device_capability_descriptor::bFunctionalitySupport
```

The lowest speed at which all the functionality supported by the device is available to the user. For example if the device supports all its functionality when connected at full speed and above then it sets this value to 1.

7.26.2.4 bLength

```
uint8_t libusb_ss_usb_device_capability_descriptor::bLength
```

Size of this descriptor (in bytes)

7.26.2.5 bmAttributes

```
uint8_t libusb_ss_usb_device_capability_descriptor::bmAttributes
```

Bitmap encoding of supported device level features. A value of one in a bit location indicates a feature is supported; a value of zero indicates it is not supported. See libusb_ss_usb_device_capability_attributes.

7.26.2.6 bU1DevExitLat

```
uint8_t libusb_ss_usb_device_capability_descriptor::bU1DevExitLat
```

U1 Device Exit Latency.

7.26.2.7 bU2DevExitLat

```
uint16_t libusb_ss_usb_device_capability_descriptor::bU2DevExitLat
```

U2 Device Exit Latency.

7.26.2.8 wSpeedSupported

```
uint16_t libusb_ss_usb_device_capability_descriptor::wSpeedSupported
```

Bitmap encoding of the speed supported by this device when operating in SuperSpeed mode. See libusb_↩ supported_speed.

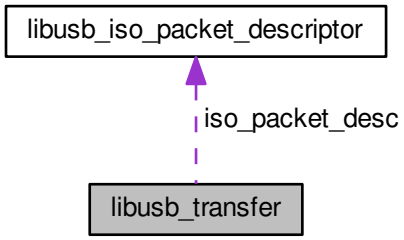
The documentation for this struct was generated from the following file:

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.27 libusb_transfer Struct Reference

```
#include <libusb.h>
```

Collaboration diagram for libusb_transfer:



Public Attributes

- libusb_device_handle * dev_handle
- uint8_t flags
- unsigned char endpoint
- unsigned char type
- unsigned int timeout
- enum libusb_transfer_status status
- int length
- int actual_length
- libusb_transfer_cb_fn callback
- void * user_data
- unsigned char * buffer
- int num_iso_packets
- struct libusb_iso_packet_descriptor iso_packet_desc [0]

7.27.1 Detailed Description

The generic USB transfer structure. The user populates this structure and then submits it in order to request a transfer. After the transfer has completed, the library populates the transfer with the results and passes it back to the user.

7.27.2 Member Data Documentation

7.27.2.1 actual_length

```
int libusb_transfer::actual_length
```

Actual length of data that was transferred. Read-only, and only for use within transfer callback function. Not valid for isochronous endpoint transfers.

7.27.2.2 buffer

```
unsigned char* libusb_transfer::buffer
```

Data buffer

7.27.2.3 callback

```
libusb_transfer_cb_fn libusb_transfer::callback
```

Callback function. This will be invoked when the transfer completes, fails, or is cancelled.

7.27.2.4 dev_handle

```
libusb_device_handle* libusb_transfer::dev_handle
```

Handle of the device that this transfer will be submitted to

7.27.2.5 endpoint

```
unsigned char libusb_transfer::endpoint
```

Address of the endpoint where this transfer will be sent.

7.27.2.6 flags

```
uint8_t libusb_transfer::flags
```

A bitwise OR combination of libusb_transfer_flags.

7.27.2.7 iso_packet_desc

```
struct libusb_iso_packet_descriptor libusb_transfer::iso_packet_desc[0]
```

Isochronous packet descriptors, for isochronous transfers only.

7.27.2.8 length

```
int libusb_transfer::length
```

Length of the data buffer

7.27.2.9 num_iso_packets

```
int libusb_transfer::num_iso_packets
```

Number of isochronous packets. Only used for I/O with isochronous endpoints.

7.27.2.10 status

```
enum libusb_transfer_status libusb_transfer::status
```

The status of the transfer. Read-only, and only for use within transfer callback function.

If this is an isochronous transfer, this field may read COMPLETED even if there were errors in the frames. Use the status field in each packet to determine if errors occurred.

7.27.2.11 timeout

```
unsigned int libusb_transfer::timeout
```

Timeout for this transfer in millseconds. A value of 0 indicates no timeout.

7.27.2.12 type

```
unsigned char libusb_transfer::type
```

Type of the endpoint from libusb_transfer_type

7.27.2.13 user_data

```
void* libusb_transfer::user_data
```

User context data to pass to the callback function.

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.28 libusb_usb_2_0_extension_descriptor Struct Reference

```
#include <libusb.h>
```

Public Attributes

- uint8_t bLength
- uint8_t bDescriptorType
- uint8_t bDevCapabilityType
- uint32_t bmAttributes

7.28.1 Detailed Description

A structure representing the USB 2.0 Extension descriptor This descriptor is documented in section 9.6.2.1 of the USB 3.0 specification. All multiple-byte fields are represented in host-endian format.

7.28.2 Member Data Documentation

7.28.2.1 bDescriptorType

```
uint8_t libusb_usb_2_0_extension_descriptor::bDescriptorType
```

Descriptor type. Will have value libusb_descriptor_type::LIBUSB_DT_DEVICE_CAPABILITY LIBUSB_DT_DEVI↩ CE_CAPABILITY in this context.

7.28.2.2 bDevCapabilityType

```
uint8_t libusb_usb_2_0_extension_descriptor::bDevCapabilityType
```

Capability type. Will have value libusb_capability_type::LIBUSB_BT_USB_2_0_EXTENSION LIBUSB_BT_USB↩ _2_0_EXTENSION in this context.

7.28.2.3 bLength

```
uint8_t libusb_usb_2_0_extension_descriptor::bLength
```

Size of this descriptor (in bytes)

7.28.2.4 bmAttributes

```
uint32_t libusb_usb_2_0_extension_descriptor::bmAttributes
```

Bitmap encoding of supported device level features. A value of one in a bit location indicates a feature is supported; a value of zero indicates it is not supported. See libusb_usb_2_0_extension_attributes.

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

- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.29 libusb_version Struct Reference

```
#include <libusb.h>
```

Public Attributes

- const uint16_t major
- const uint16_t minor
- const uint16_t micro
- const uint16_t nano
- const char * rc
- const char * describe

7.29.1 Detailed Description

Structure providing the version of the libusb runtime

7.29.2 Member Data Documentation

7.29.2.1 describe

```
const char* libusb_version::describe
```

For ABI compatibility only.

7.29.2.2 major

```
const uint16_t libusb_version::major
```

Library major version.

7.29.2.3 micro

```
const uint16_t libusb_version::micro
```

Library micro version.

7.29.2.4 minor

```
const uint16_t libusb_version::minor
```

Library minor version.

7.29.2.5 nano

```
const uint16_t libusb_version::nano
```

Library nano version.

7.29.2.6 rc

```
const char* libusb_version::rc
```

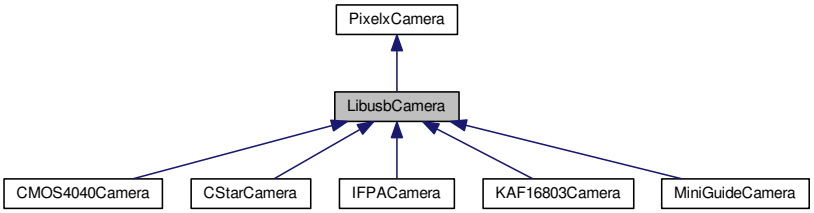
Library release candidate suffix string, e.g. "-rc4".

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

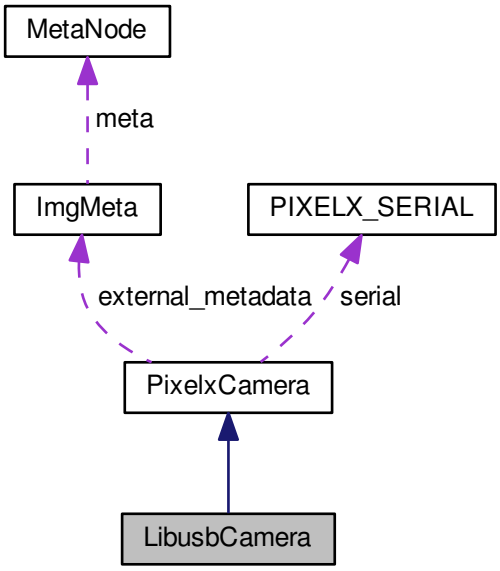
- /home/jerryjia/CCDSDK2/lib/include/libusb-1.0/libusb.h

7.30 LibusbCamera Class Reference

Inheritance diagram for LibusbCamera:



Collaboration diagram for LibusbCamera:



Public Member Functions

- virtual unsigned int **ShutDown** () override
- virtual unsigned int **SendCmd** (uint16_t cmd_id, size_t nargs, const uint8_t ∗args) override
- virtual unsigned int **RecvCmd** (uint8_t ∗buffer, size_t size, size_t ∗actual) override
- virtual unsigned int **GetTemperature** (const char ∗temp_name, float ∗temp) override
- virtual void image_thread_func () override

  Thread function for reading image. image_data, image_size, image_mutex and transfer_cond should be used.

Protected Member Functions

- virtual unsigned int **InitDeviceSpecific** () override
- unsigned int **InitUSBHandle** ()

Protected Attributes

- libusb_device_handle * **dev_handle** = NULL

Additional Inherited Members

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

- /home/jerryjia/CCDSDK2/LibusbCamera.h
- /home/jerryjia/CCDSDK2/LibusbCamera.cpp

7.31 MetaNode Struct Reference

< meta data node, will write to FITS

```
#include <CCDSDK.h>
```

Public Attributes

- int val_size
  size of val
- char key [24]
  key name
- char type [16]
  support INT, LONG, STRING, DOUBLE, etc (type in fitsio.h)
- char **desc** [64]
- void * **val**

7.31.1 Detailed Description

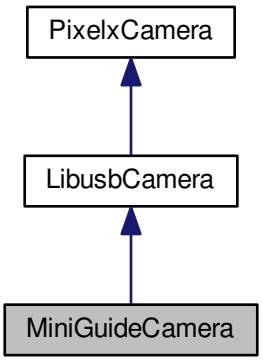
< meta data node, will write to FITS

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

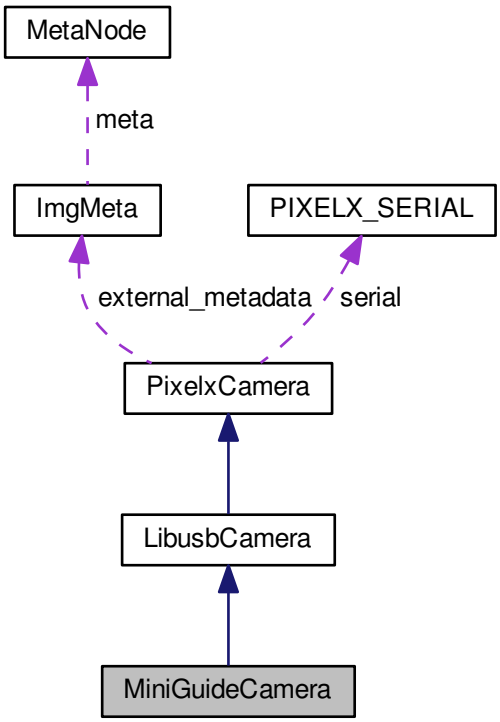
- /home/jerryjia/CCDSDK2/CCDSDK.h

7.32 MiniGuideCamera Class Reference

Inheritance diagram for MiniGuideCamera:



Collaboration diagram for MiniGuideCamera:



Public Member Functions

- unsigned int **Initialize** (libusb_device_handle *handle)
- void **setConfigurations** () override
- void **setConfigFile** () override

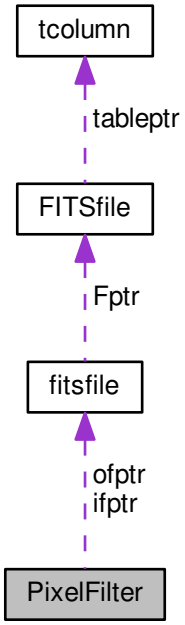
Additional Inherited Members

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

- /home/jerryjia/CCDSDK2/mini_guide/mini_guide.h
- /home/jerryjia/CCDSDK2/mini_guide/mini_guide.cpp

7.33 PixelFilter Struct Reference

Collaboration diagram for PixelFilter:



Public Attributes

- int **count**
- char ** **path**
- char ** **tag**
- fitsfile ** **ifptr**

- char * **expression**
- int **bitpix**
- long **blank**
- fitsfile * **ofptr**
- char **keyword** [FLEN_KEYWORD]
- char **comment** [FLEN_COMMENT]

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

- /home/jerryjia/CCDSDK2/lib/include/fitsio.h

7.34 PIXELX_SERIAL Struct Reference

Public Attributes

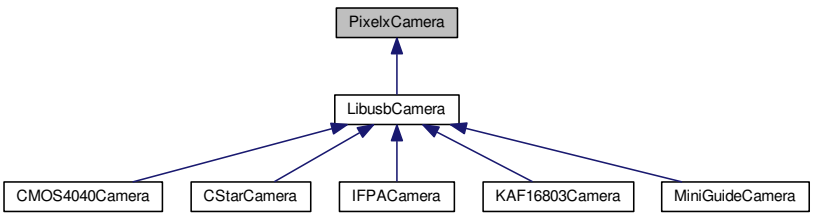
- uint8_t **Model_Enum**
- uint8_t **MB_ID**
- uint8_t **Year**
- uint8_t **Month**
- uint8_t **Day**
- uint8_t **Hour**
- uint8_t **version**
- uint8_t **version_aa**
- uint16_t **id**
- uint16_t **checksum**

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

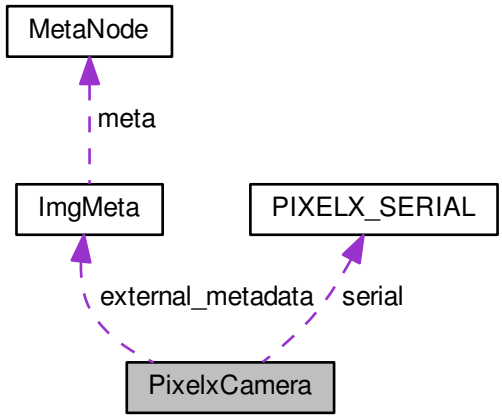
- /home/jerryjia/CCDSDK2/CCDSDK.h

7.35 PixelxCamera Class Reference

Inheritance diagram for PixelxCamera:



Collaboration diagram for PixelxCamera:



Public Types

- using **VidPid** = std::tuple< uint16_t, uint16_t >

Public Member Functions

- virtual unsigned int **Initialize** (int debug_level=0, VidPid vidpid={}) final
- virtual unsigned int **ShutDown** ()
- virtual unsigned int **SetExposureTime** (float time)
- virtual unsigned int **GetExposureTime** (float *expo_time)
- virtual unsigned int **SetExposureInterval** (float intr)
- virtual unsigned int **GetExposureInterval** (float *intr)
- virtual unsigned int **SetReadoutSpeed** (int idx)
- virtual unsigned int **GetNumReadoutSpeed** ()
- virtual unsigned int **GetReadoutSpeed** (int index)
- virtual unsigned int **SetEraseCount** (uint8_t number)
- virtual unsigned int **StartExposure** ()
- virtual unsigned int **StopExposure** ()
- virtual unsigned int **SetShutter** (uint8_t mode)
- virtual unsigned int **SetPreAmpGain** (uint8_t gain)
- virtual unsigned int **SetGain** (float high_gain, float low_gain)
- virtual unsigned int **GetGain** (float *high_gain, float *low_gain)
- virtual unsigned int **CancelWait** ()
- virtual unsigned int **WaitForAcquisition** ()
- virtual unsigned int **WaitForAcquisitionTimeOut** (int timeout_ms)
- virtual unsigned int **GetImage** (void *buf, int buf_size)
- virtual unsigned int **GetNthImage** (void *buf, int buf_size, int idx)
- virtual unsigned int **SaveAsFITS** (const char *file, struct ImgMeta meta={})
- virtual unsigned int **SetAutoSaveDir** (const char *dir)
- virtual unsigned int **SetAutoSave** (int if_auto_save)

- virtual unsigned int **SaveNthAsFITS** (const char *file, int idx, struct ImgMeta *pMeta=nullptr)
- virtual unsigned int **GetModelInfo** (uint8_t *, uint8_t *, uint8_t *)
- virtual unsigned int **GetDeviceSerial** (uint8_t *, size_t)
- virtual unsigned int **GetDeviceLogicVer** (uint8_t *, uint8_t *, uint8_t *, uint16_t *)
- virtual unsigned int **CoolerOn** ()
- virtual unsigned int **CoolerOff** ()
- virtual unsigned int **SetCoolerTemp** (float temp)
- virtual unsigned int **SetForcedTraining** ()
- virtual unsigned int **GetForcedTraining** (uint8_t *ft_bool)
- virtual unsigned int **PauseCool** ()
- virtual unsigned int **RecoverCool** ()
- virtual unsigned int **GetTemperature** (const char *temp_name, float *temp)
- virtual unsigned int **GetCoolerRange** (int *cooler_min, int *cooler_max)
- virtual unsigned int **GetDetector** (int *xpixels, int *ypixels)
- virtual unsigned int **GetBitDepth** (int *depth)
- virtual unsigned int **GetAcquisitionStatus** ()
- virtual unsigned int **ControllerFan** (uint8_t speed)
- virtual unsigned int **PowerFan** (uint8_t speed)
- virtual unsigned int **FanStatusSet** (uint8_t speed)
- virtual unsigned int **VideoMode** (uint16_t photoNum)
- virtual unsigned int **ROIEnable** (int16_t rowStartNum, int16_t rowKeepNum, int16_t columnStartNum, int16_t columnKeepNum)
- virtual unsigned int **ROIDisable** ()
- virtual unsigned int **BINEnable** (uint16_t binReadRow, uint16_t binReadColumn)
- virtual unsigned int **BINDisable** ()
- virtual unsigned int **SetCurrImageSize** (bool imageDefault, int sizex, int sizey)
- virtual unsigned int **SpecialImageCircle** ()
- virtual unsigned int **PhotoMode** (uint8_t mode)
- virtual unsigned int **TransferCycle** (uint16_t cycle)
- virtual unsigned int **SetDriftMode** (uint8_t mode)
- virtual unsigned int **SetDriftSpeed** (float nline)
- virtual unsigned int **SetDCDS** (uint8_t T1, uint8_t T2, uint16_t T3, uint8_t T4)
- virtual unsigned int **SetDCDS_2** (uint8_t T1, uint8_t T2, uint16_t T3, uint8_t T4)
- virtual unsigned int **GetWaitingTime** (uint16_t readoutSpeed, float *waitingTime)
- virtual unsigned int **GetModel** (uint8_t *Num)
- virtual unsigned int **GetSerialNum** (uint8_t *Num)
- virtual unsigned int **GetLogicVer** (uint8_t *Num)
- virtual unsigned int **AnalogPowerOn** ()
- virtual unsigned int **AnalogPowerOff** ()
- virtual unsigned int **SetBlackLevel** (uint16_t top_bl, uint16_t bot_bl)
- virtual unsigned int **GetBlackLevel** (uint16_t *top_bl, uint16_t *bot_bl)
- virtual unsigned int **SetIntegrateCycle** (uint16_t integrate_cycle)
- virtual unsigned int **GetEffectiveArea** (int *pstartX, int *pstartY, int *pendX, int *pendY)
- virtual unsigned int **SetClockVoltage** (uint8_t channel, float voltage)
- virtual unsigned int **GetNumberSetClockVoltage** (size_t *number)
- virtual unsigned int **GetNameSetClockVoltage** (uint8_t channel, char *buf, size_t size)
- virtual unsigned int **GetClockVoltage** (uint8_t mode, uint8_t channel, float *voltage)
- virtual unsigned int **GetNumberGetClockVoltage** (size_t *number)
- virtual unsigned int **GetClockVoltageParam** (uint8_t channel, ADC_Channel *param)
- virtual unsigned int **GetAnalogVoltage** (uint8_t channel, float *voltage)
- virtual unsigned int **GetNumberGetAnalogVoltage** (size_t *number)
- virtual unsigned int **GetNameGetAnalogVoltage** (uint8_t channel, char *buf, size_t size)
- virtual unsigned int **GetNumberGetAcqSet** (size_t *number)
- virtual unsigned int **GetAcqSetParam** (uint8_t number, CCD_ACQ_Set *param)
- virtual unsigned int **GetTECIV** (float *current, float *voltage)

- virtual unsigned int **SetRCMode** (uint8_t mode)
- virtual unsigned int **ShutterReset** ()
- virtual unsigned int **SetLDCMode** ()
- virtual unsigned int **UnsetLDCMode** ()
- virtual unsigned int **GetLDCMode** (uint8_t *ldc_bool)
- virtual unsigned int **SetContinuousCapture** (uint16_t captureNum)
- virtual std::map< std::string, std::vector< std::string > > **GetLastImageMetaRaw** ()
- virtual unsigned int **GetLastImageMetaStruct** (struct ImgMeta *)
- virtual unsigned int **PICMode** (uint8_t)
- virtual unsigned int **getPICMode** (uint8_t *mode)
- virtual unsigned int **setPID** (uint8_t p, uint8_t i, uint8_t d, uint8_t T)
- virtual unsigned int **GetCurrent** (float *iboard, float *i5_5v, float *i24v, float *itec)
- virtual unsigned int **GetGPSTime** (int &sec, int &min, int &hr)
- virtual unsigned int **GetGPSEpochTime** (time_t &t)
- virtual unsigned int **GetTDCTime** (uint32_t &tdc_count)
- virtual unsigned int **GetVoltageByChannel** (int vc, float *vol)
- virtual unsigned int **GetCurrentByChannel** (int vc, float *cur)
- virtual unsigned int **SetTriggerMode** (uint8_t mode)
- virtual unsigned int **SetExposureStartTime** (uint64_t sec, uint32_t usec)
- virtual unsigned int **RecordExposureStartTime** (uint64_t sec, uint32_t usec)
- virtual unsigned int **EnableExternalTrigger** ()
- virtual unsigned int **SetSerialNumber** (PIXELX_SERIAL *pse)
- virtual unsigned int **GetSerialNumber** (PIXELX_SERIAL *pse)
- virtual unsigned int **SendCmd** (uint16_t cmd_id, size_t nargs, const uint8_t *args)=0
- virtual unsigned int **RecvCmd** (uint8_t *buffer, size_t size, size_t *actual)=0
- virtual unsigned int **SendRecv** (uint16_t cmd_id, size_t nargs, void *args, size_t recv_buf_size, void *recv↩ _buf, size_t *actual)

Static Public Member Functions

- static double PT100Temperature (uint32_t temp_adc)

  Convert PT100 ADC value to temperature.
- static int **SetupLogger** ()

Public Attributes

- PIXELX_SERIAL **serial**

Static Public Attributes

- static std::shared_ptr< spdlog::logger > **logger** = nullptr

Protected Types

- enum **TRIGGER_MODE** { **INTERNAL**, **EXTERNAL**, **TIMER** }
- enum {
  CHANNEL_E, **CHANNEL_F**, **CHANNEL_G**, **CHANNEL_H**,
  CHANNEL_EH, **CHANNEL_GF**, **CHANNEL_EFGH** }

Protected Member Functions

- virtual unsigned int **InitDeviceSpecific** ()=0
- virtual unsigned int **CheckIfNotImplemented** (uint8_t *msg_buf, size_t msg_buf_size)
- virtual void **CopyToImageBuffer** (uint8_t *buf, int imageWidth, int imageHeight, uint8_t pixelDepth)
- void **saveRawData** (uint8_t *buf, int length)
- virtual void **setConfigurations** ()=0
- virtual unsigned int **setConfigurationsFromFile** (const char *file)
- virtual void **setConfigFile** ()=0
- virtual void image_thread_func ()=0

  Thread function for reading image. image_data, image_size, image_mutex and transfer_cond should be used.

Protected Attributes

- uint16_t **vid** = 0
- uint16_t **pid** = 0
- int **video_enable**
- uint64_t **exp_sec**
- uint32_t **exp_usec**
- enum TRIGGER_MODE **trigger_mode** = INTERNAL
- float **interval**
- std::condition_variable **transfer_cond**
- std::mutex **image_mutex**
- std::mutex **cmd_mutex**
- float **exposure_time**
- image_status_t image_status

  indicating if image is ready
- bool image_thread_run = false

  for stopping image thread
- void * **image_data** = nullptr
- size_t **image_size**
- int curr_image_width

  current image width
- int curr_image_height

  current image height
- float cooler_target = 0

  cooler target temperature
- int image_num = 0

  number of images waiting to be read
- int **effective_area_startx**
- int **effective_area_starty**
- int **effective_area_endx**
- int **effective_area_endy**
- enum PixelxCamera:: { ... } **photoMode**
- uint8_t **depth**
- size_t **bytepix**
- int **sizex**
- int **sizey**
- std::vector< int > **gains**
- int **cooler_min**
- int **cooler_max**
- uint64_t **expo_min**
- uint64_t **expo_max**

- std::vector< uint16_t > **readspd**
- std::vector< uint8_t > **device_serial_number**
- uint8_t **model_number**
- uint8_t **logic_version**
- uint8_t **serial_number**
- int config_data_autosave = 0

  config flag for autosave
- std::string config_data_autosave_location

  name with directory for data output;
- ImgMeta external_metadata

  metadata to be written, sent by external caller
- std::vector< uint8_t * > **image_frames** = std::vector<uint8_t *>(0, nullptr)
- std::vector< int > **clock_dac_factor**
- std::vector< std::string > **clock_dac_channels**
- std::vector< int > **clock_dac_number**
- std::vector< float > **clock_adc_factor**
- std::vector< float > **clock_adc_constant**
- std::vector< ADC_Channel > **clock_adc_channels**
- std::vector< int > **analog_adc_factor**
- std::vector< std::string > **analog_adc_channels**
- std::string **configFile**
- std::vector< CCD_ACQ_Set > **acq_set**

7.35.1 Member Function Documentation

7.35.1.1 PT100Temperature()

```
double PixelxCamera::PT100Temperature (
            uint32_t temp_adc )  [static]
```

Convert PT100 ADC value to temperature.

XXX Maybe we should take it as a temperature sensor module which may be replaced in the future.

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

- /home/jerryjia/CCDSDK2/PixelxCamera.h
- /home/jerryjia/CCDSDK2/PixelxCamera.cpp

7.36 tcolumn Struct Reference

Public Attributes

- char **ttype** [70]
- LONGLONG **tbcol**
- int **tdatatype**
- LONGLONG **trepeat**
- double **tscale**
- double **tzero**
- LONGLONG **tnull**
- char **strnull** [20]
- char **tform** [10]
- long **twidth**

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

- /home/jerryjia/CCDSDK2/lib/include/fitsio.h

7.37 wtbarr Struct Reference

Public Attributes

- int **i**
- int **m**
- int **kind**
- char **extnam** [72]
- int **extver**
- int **extlev**
- char **ttype** [72]
- long **row**
- int **ndim**
- int ∗ **dimlen**
- double ∗∗ **arrayp**

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

- /home/jerryjia/CCDSDK2/lib/include/fitsio.h

Index

ADC_Channel, 31
actual_length
    libusb_iso_packet_descriptor, 62
    libusb_transfer, 67
altsetting
    libusb_interface, 58

bAlternateSetting
    libusb_interface_descriptor, 60
bConfigurationValue
    libusb_config_descriptor, 49
bDescriptorType
    libusb_bos_descriptor, 46
    libusb_bos_dev_capability_descriptor, 47
    libusb_config_descriptor, 49
    libusb_container_id_descriptor, 51
    libusb_device_descriptor, 54
    libusb_endpoint_descriptor, 56
    libusb_interface_descriptor, 60
    libusb_ss_endpoint_companion_descriptor, 63
    libusb_ss_usb_device_capability_descriptor, 65
    libusb_usb_2_0_extension_descriptor, 69
bDevCapabilityType
    libusb_bos_dev_capability_descriptor, 47
    libusb_container_id_descriptor, 51
    libusb_ss_usb_device_capability_descriptor, 65
    libusb_usb_2_0_extension_descriptor, 70
bDeviceClass
    libusb_device_descriptor, 54
bDeviceProtocol
    libusb_device_descriptor, 54
bDeviceSubClass
    libusb_device_descriptor, 54
bEndpointAddress
    libusb_endpoint_descriptor, 56
bFunctionalitySupport
    libusb_ss_usb_device_capability_descriptor, 65
bInterfaceClass
    libusb_interface_descriptor, 60
bInterfaceNumber
    libusb_interface_descriptor, 60
bInterfaceProtocol
    libusb_interface_descriptor, 60
bInterfaceSubClass
    libusb_interface_descriptor, 60
bInterval
    libusb_endpoint_descriptor, 56
bLength
    libusb_bos_descriptor, 46
    libusb_bos_dev_capability_descriptor, 47
    libusb_config_descriptor, 49
    libusb_container_id_descriptor, 51
    libusb_device_descriptor, 54
    libusb_endpoint_descriptor, 56
    libusb_interface_descriptor, 60
    libusb_ss_endpoint_companion_descriptor, 64
    libusb_ss_usb_device_capability_descriptor, 65
    libusb_usb_2_0_extension_descriptor, 70
bMaxBurst
    libusb_ss_endpoint_companion_descriptor, 64
bMaxPacketSize0
    libusb_device_descriptor, 54
bNumConfigurations
    libusb_device_descriptor, 54
bNumDeviceCaps
    libusb_bos_descriptor, 46
bNumEndpoints
    libusb_interface_descriptor, 60
bNumInterfaces
    libusb_config_descriptor, 49
bRefresh
    libusb_endpoint_descriptor, 57
bRequest
    libusb_control_setup, 52
bReserved
    libusb_container_id_descriptor, 51
bSynchAddress
    libusb_endpoint_descriptor, 57
bU1DevExitLat
    libusb_ss_usb_device_capability_descriptor, 65
bU2DevExitLat
    libusb_ss_usb_device_capability_descriptor, 66
bcdDevice
    libusb_device_descriptor, 53
bcdUSB
    libusb_device_descriptor, 53
bmAttributes
    libusb_config_descriptor, 49
    libusb_endpoint_descriptor, 56
    libusb_ss_endpoint_companion_descriptor, 64
    libusb_ss_usb_device_capability_descriptor, 65
    libusb_usb_2_0_extension_descriptor, 70
bmRequestType
    libusb_control_setup, 52
buffer
    libusb_transfer, 67

CCD initialization & deinitialization, 11
    Initialize, 11
    ShutDown, 12

CCD_ACQ_Set, 31
CMOS4040Camera, 32
    ROIEnable, 34
    SetContinuousCapture, 34
CStarCamera, 35
callback
    libusb_transfer, 67
CancelWait
    Exposure related functions, 13
ConfigAutoSave
    Image saving functions, 22
ContainerID
    libusb_container_id_descriptor, 51

def_error.ErrorDef, 36
describe
    libusb_version, 71
dev_capability
    libusb_bos_descriptor, 46
dev_capability_data
    libusb_bos_dev_capability_descriptor, 47
dev_handle
    libusb_transfer, 67

endpoint
    libusb_interface_descriptor, 61
    libusb_transfer, 68
Error codes, 27
    PIXELX_ERRNO_MAP, 27
    pixelx_errno_t, 28
events
    libusb_pollfd, 63
Exposure related functions, 13
    CancelWait, 13
    GetExposureTime, 14
    GetImage, 15
    GetLDCMode, 15
    GetNthImage, 15
    SetContinuousCapture, 16
    SetEraseCount, 16
    SetExposureInterval, 17
    SetExposureTime, 17
    SetPreAmpGain, 18
    SetReadoutSpeed, 18
    SetShutter, 18
    SetTriggerMode, 19
    StartExposure, 19
    StopExposure, 20
    WaitForAcquisition, 20
    WaitForAcquisitionTimeOut, 20
extra
    libusb_config_descriptor, 49
    libusb_endpoint_descriptor, 57
    libusb_interface_descriptor, 61
extra_length
    libusb_config_descriptor, 49
    libusb_endpoint_descriptor, 57
    libusb_interface_descriptor, 61

FITSfile, 37
fd
    libusb_pollfd, 63
fitsfile, 39
flags
    libusb_transfer, 68
FloatComp, 40
fx_known_device, 40

GetExposureTime
    Exposure related functions, 14
GetImage
    Exposure related functions, 15
GetLDCMode
    Exposure related functions, 15
GetNthImage
    Exposure related functions, 15

iConfiguration
    libusb_config_descriptor, 49
IFPACamera, 40
iInterface
    libusb_interface_descriptor, 61
iManufacturer
    libusb_device_descriptor, 55
iProduct
    libusb_device_descriptor, 55
iSerialNumber
    libusb_device_descriptor, 55
idProduct
    libusb_device_descriptor, 55
idVendor
    libusb_device_descriptor, 55
Image saving functions, 22
    ConfigAutoSave, 22
    SaveAsFITSEx, 23
    SaveAsFITS, 22
    SaveNthAsFITS, 23
    SetAutoSaveFITS, 23
ImgMeta, 42
Initialize
    CCD initialization & deinitialization, 11
interface
    libusb_config_descriptor, 50
iso_packet_desc
    libusb_transfer, 68
iteratorCol, 43

KAF16803Camera, 44

length
    libusb_iso_packet_descriptor, 62
    libusb_transfer, 68
libusb_bos_descriptor, 45
    bDescriptorType, 46
    bLength, 46
    bNumDeviceCaps, 46
    dev_capability, 46
    wTotalLength, 46

libusb_bos_dev_capability_descriptor, 46
bDescriptorType, 47
bDevCapabilityType, 47
bLength, 47
dev_capability_data, 47
libusb_config_descriptor, 48
bConfigurationValue, 49
bDescriptorType, 49
bLength, 49
bNumInterfaces, 49
bmAttributes, 49
extra, 49
extra_length, 49
iConfiguration, 49
interface, 50
MaxPower, 50
wTotalLength, 50
libusb_container_id_descriptor, 50
bDescriptorType, 51
bDevCapabilityType, 51
bLength, 51
bReserved, 51
ContainerID, 51
libusb_control_setup, 51
bRequest, 52
bmRequestType, 52
wIndex, 52
wLength, 52
wValue, 52
libusb_device_descriptor, 53
bDescriptorType, 54
bDeviceClass, 54
bDeviceProtocol, 54
bDeviceSubClass, 54
bLength, 54
bMaxPacketSize0, 54
bNumConfigurations, 54
bcdDevice, 53
bcdUSB, 53
iManufacturer, 55
iProduct, 55
iSerialNumber, 55
idProduct, 55
idVendor, 55
libusb_endpoint_descriptor, 55
bDescriptorType, 56
bEndpointAddress, 56
bInterval, 56
bLength, 56
bRefresh, 57
bSynchAddress, 57
bmAttributes, 56
extra, 57
extra_length, 57
wMaxPacketSize, 57
libusb_interface, 58
altsetting, 58
num_altsetting, 58
libusb_interface_descriptor, 59
bAlternateSetting, 60
bDescriptorType, 60
bInterfaceClass, 60
bInterfaceNumber, 60
bInterfaceProtocol, 60
bInterfaceSubClass, 60
bLength, 60
bNumEndpoints, 60
endpoint, 61
extra, 61
extra_length, 61
iInterface, 61
libusb_iso_packet_descriptor, 61
actual_length, 62
length, 62
status, 62
libusb_pollfd, 62
events, 63
fd, 63
libusb_ss_endpoint_companion_descriptor, 63
bDescriptorType, 63
bLength, 64
bMaxBurst, 64
bmAttributes, 64
wBytesPerInterval, 64
libusb_ss_usb_device_capability_descriptor, 64
bDescriptorType, 65
bDevCapabilityType, 65
bFunctionalitySupport, 65
bLength, 65
bU1DevExitLat, 65
bU2DevExitLat, 66
bmAttributes, 65
wSpeedSupported, 66
libusb_transfer, 66
actual_length, 67
buffer, 67
callback, 67
dev_handle, 67
endpoint, 68
flags, 68
iso_packet_desc, 68
length, 68
num_iso_packets, 68
status, 68
timeout, 68
type, 69
user_data, 69
libusb_usb_2_0_extension_descriptor, 69
bDescriptorType, 69
bDevCapabilityType, 70
bLength, 70
bmAttributes, 70
libusb_version, 70
describe, 71
major, 71
micro, 71

minor, 71
nano, 71
rc, 71
LibusbCamera, 72

major
libusb_version, 71
MaxPower
libusb_config_descriptor, 50
MetaNode, 73
micro
libusb_version, 71
MiniGuideCamera, 74
minor
libusb_version, 71
Miscellaneous functions, 25

nano
libusb_version, 71
num_altsetting
libusb_interface, 58
num_iso_packets
libusb_transfer, 68

PIXELX_ERRNO_MAP
Error codes, 27
PIXELX_SERIAL, 76
PT100Temperature
PixelxCamera, 81
PixelFilter, 75
pixelx_errno_t
Error codes, 28
PixelxCamera, 76
PT100Temperature, 81

ROIEnable
CMOS4040Camera, 34
rc
libusb_version, 71

SaveAsFITSEx
Image saving functions, 23
SaveAsFITS
Image saving functions, 22
SaveNthAsFITS
Image saving functions, 23
SerialNumber, 30
SetAutoSaveFITS
Image saving functions, 23
SetContinuousCapture
CMOS4040Camera, 34
Exposure related functions, 16
SetEraseCount
Exposure related functions, 16
SetExposureInterval
Exposure related functions, 17
SetExposureTime
Exposure related functions, 17
SetPreAmpGain
Exposure related functions, 18
SetReadoutSpeed
Exposure related functions, 18
SetShutter
Exposure related functions, 18
SetTriggerMode
Exposure related functions, 19
ShutDown
CCD initialization & deinitialization, 12
StartExposure
Exposure related functions, 19
status
libusb_iso_packet_descriptor, 62
libusb_transfer, 68
StopExposure
Exposure related functions, 20

tcolumn, 82
timeout
libusb_transfer, 68
type
libusb_transfer, 69

user_data
libusb_transfer, 69

wBytesPerInterval
libusb_ss_endpoint_companion_descriptor, 64
wIndex
libusb_control_setup, 52
wLength
libusb_control_setup, 52
wMaxPacketSize
libusb_endpoint_descriptor, 57
wSpeedSupported
libusb_ss_usb_device_capability_descriptor, 66
wTotalLength
libusb_bos_descriptor, 46
libusb_config_descriptor, 50
wValue
libusb_control_setup, 52
WaitForAcquisition
Exposure related functions, 20
WaitForAcquisitionTimeOut
Exposure related functions, 20
wtbarr, 82