WIC SDK

1.0.6

Generated by Doxygen 1.8.11

Contents

Chapter 1

WIC SDK Documentation - version for Linux

1.1 Introduction

Workswell WIC SDK is set of libraries for controlling and operating with WIC devices.

Workswell Infrared Camera „WIC“ is designed and manufactured for easy and user-friendly integration to all machine vision applications as well as security projects. All Workswell Infrared Cameras use the newest FLIR Long Wave Infrared Detector Technology.

1.2 Requirements

To use WIC SDK libraries, you need to have an Linux operating system with architecture x86-x64 (64bit), i386 (32bit), armhf (ARM hard-float) or armsf(ARM soft-float). Please, check if you have installed correct version for your system architecture.

The library was tested with several linux distributions:

- Ubuntu 14.04 (gcc version 4.8.5)
- Ubuntu 16.04 (gcc version 5.4.0)
- Raspbian 8-jessie (gcc version 4.9.2)

Additionally, root (sudo) privileges are needed. For running the libraries, these Ubuntu packages are required:

- build-essential
- libjpeg-dev

1.3 Installation

Installation can be started from console. Root privileges are needed. Please, read the descriptions during installation.

All files are installed in /**opt** folder.

- The WIC SDK is found in folder: /**opt**/**workswell**/**wic_sdk**
- The Pleora eBUS is found in folder: /**opt**/**pleora**/**ebus_sdk**/**xxx-architecture-xxx**

Here, necessary libraries in **lib** directory and headers in **include** directory are located.

For WIC with USB 3.0 connection, you have to install eBUSd deamon (if you haven't done so). The script can be found in Pleora bin folder:

- /opt/pleora/ebus_sdk/xxx-architecture-xxx/bin/install_daemon.sh

1.4 Libraries description

The library consists out of three basic classes:

- CameraCenter()
  - Class creates list of available WIC devices
- Camera()
  - Class controls the WIC device (connection and image data)
- CameraSerialSettings()
  - Class sets up the WIC camera

The general use of WIC SDK library is presented in WIC_SDK_Sample.cpp. Sample project can be found in:

- /opt/workswell/wic_sdk/sample
  - In subfolder *build* is makefile
  - In subfolder *src* is source file WIC_SDK_Sample.cpp

1.4.1 Compilation

To compile your code, add these **include paths** to you project makefile:

- /opt/workswell/wic_sdk/include
- /opt/pleora/ebus_sdk/Ubuntu-14.04-x86_64/include

These **library paths** needs to be added:

- /opt/workswell/wic_sdk/lib
- /opt/pleora/ebus_sdk/xxx-architecture-xxx/lib

These **libraries** needs to be added to your makefile:

- WIC_SDK
- jpeg
- PvBase
- PvDevice
- PvBuffer
- PvGenICam
- PvTransmitter
- PvVirtualDevice
- PvAppUtils
- PvPersistence
- PvSerial
- PvStream

You need to define these symbols:

- _UNIX_
- _LINUX_

We use the c++11 standard, so another option needs to be added: -std=c++11

Since the libraries are not in standard system folder, you have to add the reference to your LD_LIBRARY_PATH. We have prepared a script that do so. Please refer to next section Running the applications.

1.4.2 Running the applications

You need to set up the Linux system environment like library paths etc. To do so, we have prepared simple script. You can add this script to your .bashrc file in home directory to be loaded each time console starts.

The script is located here:

- /opt/workswell/wic_sdk/set_env_variables

To add the script to your .bashrc file, you can run command:

- echo ". /opt/workswell/wic_sdk/set_env_variables" >> ~/.bashrc

1.4.3 Temperature calculations and radiometry information

This library is mainly intended for temperature calculation. As such, we have provided several function for setting and calculating temperatures from WIC thermal camera. It is mainly:

- Camera::CalculateTemperatureC()
- CameraSerialSettings::SetEmissivity()
- CameraSerialSettings::SetReflectedTemperatureC()
- CameraSerialSettings::SetAtmospericTemperatureC()
- CameraSerialSettings::SetDistance()
- CameraSerialSettings::SetHumidity()

Radimetry cameras: by "radiometry" we mean the property of some WIC cameras to return data that are linear in temperature. This leads to one positive: recalculation of RAW pixel values to temperature is much more CPU efficient.

1.5 About

This documentation was generated by Doxygen from program comments. It suppose to give information about the library code and how to use it.

The code was written by Workswell s.r.o. Please, contact us in case of bugs or questions on email address info@workswell.cz.

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Class Documentation

4.1 Authentificator Class Reference

Public Member Functions

- **Authentificator** (std::string)
- void UpdateLicenses ()
- std::vector< std::string > GetSerialNumbers ()
- std::vector< std::string > GetCalibrationData ()
- std::vector< std::string > GetLicenseVersions ()
- std::vector< std::string > GetTimeRestrictions ()
- std::vector< std::string > GetSoftwareShortcuts ()
- std::vector< std::string > GetLicenseCreators ()
- std::vector< std::string > GetLicenseDates ()
- std::vector< std::string > GetLicenseArticles ()

4.1.1 Member Function Documentation

4.1.1.1 std::vector< std::string > Authentificator::GetCalibrationData ( )

Getter for calibration data

Returns

vector of strings of calibration data

4.1.1.2 std::vector< std::string > Authentificator::GetLicenseArticles ( )

Getter for license article data

Returns

vector of strings of license article data

4.1.1.3 std::vector< std::string > Authentificator::GetLicenseCreators ( )

Getter for license date data

Returns

vector of strings of license date data

4.1.1.4 std::vector< std::string > Authentificator::GetLicenseDates ( )

Getter for calibration data

Returns

vector of strings of calibration data

4.1.1.5 std::vector< std::string > Authentificator::GetLicenseVersions ( )

Getter for license versions

Returns

vector of strings of license versions

4.1.1.6 std::vector< std::string > Authentificator::GetSerialNumbers ( )

Getter for serial numbers

Returns

vector with string serial numbers

4.1.1.7 std::vector< std::string > Authentificator::GetSoftwareShortcuts ( )

Getter for software shortcuts data

Returns

vector of strings of software shortcuts data

4.1.1.8 std::vector< std::string > Authentificator::GetTimeRestrictions ( )

Getter for time restrictions

Returns

vector of strings of time restrictions

4.1.1.9 void Authentificator::UpdateLicenses ( )

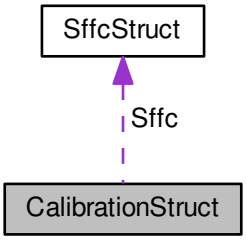
Search selected folder for license files, decrypt each file and extract serial numbers and calibration data. Saves them to serialNumbers and calibrationData vectors

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

- documentation/SDK_WIC2/Authentificator.h
- documentation/SDK_WIC2/Authentificator.cpp

4.2 CalibrationStruct Struct Reference

Collaboration diagram for CalibrationStruct:



Public Attributes

- bool **isTwoLineCalib** = false
- bool **is1500Available** = false
- int **blackBodies**
- std::string **date**
- std::string **lens**
- double **coefficients** [3][60][5]
- double **lineChangeTemperature** [2][60]
- SffcStruct **Sffc**
- CameraSerialSettings::CameraSpeed **speed**

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

- documentation/SDK_WIC2/CameraSerialSettings.cpp

4.3 Camera Class Reference

Public Member Functions

- int Connect ()
- void StartAcquisition ()
- uint8_t * RetreiveBuffer ()
- void ReleaseBuffer ()
- void StopAcquisition ()
- void Disconnect ()
- bool IsConnected ()
- bool IsAcquiring ()
- string GetStatus ()
- double CalculateTemperatureC (uint16_t raw)

  Function calculates the temperature of RAW pixel value. The function works for current camera setting, regardless if its radiometric or non-radiometric. The camera must be set for CMOS 14bit RAW values (default setting)
- double CalculateTemperatureK (uint16_t raw)

  Function calculates the temperature of RAW pixel value. The function works for current camera setting, regardless if its radiometric or non-radiometric. The camera must be set for CMOS 14bit RAW values (default setting)
- CameraSerialSettings * GetSettings ()

Friends

- class **CameraCenter**

4.3.1 Member Function Documentation

4.3.1.1 double Camera::CalculateTemperatureC (uint16_t *raw*)

Function calculates the temperature of RAW pixel value. The function works for current camera setting, regardless if its radiometric or non-radiometric. The camera must be set for CMOS 14bit RAW values (default setting)

Parameters

raw	Raw pixel value

Returns

Temperature in degrees Celsius

4.3.1.2 double Camera::CalculateTemperatureK (uint16_t *raw*)

Function calculates the temperature of RAW pixel value. The function works for current camera setting, regardless if its radiometric or non-radiometric. The camera must be set for CMOS 14bit RAW values (default setting)

Parameters

raw	Raw pixel value

Returns

Temperature in Kelvin

4.3.1.3 int Camera::Connect ( )

Connect establishes connection to the selected camera.

Returns

success

4.3.1.4 void Camera::Disconnect ( )

Disconnect disconnects the connected camera

4.3.1.5 CameraSerialSettings* Camera::GetSettings ( ) [inline]

Getter for Camera settings

Returns

Current setting for WIC camera

4.3.1.6 string Camera::GetStatus ( )

GetStatus gets camera status

Returns

camera status - Connected/Disconnected/Acquiring

4.3.1.7 bool Camera::IsAcquiring ( )

IsAcquiring gets acquisition status of the camera

Returns

true if the camera is acquiring images

4.3.1.8 bool Camera::IsConnected ( )

IsConnected gets connection status of the camera

Returns

true if the camera is connected

4.3.1.9 void Camera::ReleaseBuffer ( )

ReleaseBuffer releases the current buffer

4.3.1.10 uint8_t * Camera::RetreiveBuffer ( )

RetreiveBuffer gets actual data buffer from the camera

Returns

pointer to the data buffer

4.3.1.11 void Camera::StartAcquisition ( )

StartAcquisition starts the streaming from the camera

4.3.1.12 void Camera::StopAcquisition ( )

StopAcquisition stops streaming from the connected camera

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

- documentation/SDK_WIC2/Camera.h
- documentation/SDK_WIC2/Camera.cpp

4.4 CameraCenter Class Reference

Public Member Functions

- **CameraCenter** (string path)
- vector< Camera * > getCameras ()
- void RefindCameras ()

4.4.1 Member Function Documentation

4.4.1.1 vector<Camera*> CameraCenter::getCameras ( ) [inline]

Function returns list of available WIC thermocameras.

Returns

vector<Camera*> cameras

4.4.1.2 void CameraCenter::RefindCameras ( )

Function refinds all connected cameras.

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

- documentation/SDK_WIC2/CameraCenter.h
- documentation/SDK_WIC2/CameraCenter.cpp

4.5 CameraSerialSettings Class Reference

Public Types

- enum ArgumentType { **Int16**, **Int32**, **ByteArray** }
- enum Resolution {
  Resolution_640, **Resolution_336**, **Resolution_324**, **Resolution_168**,
  Resolution_162, **Resolution_160** }
- enum RadiometryCommand { **ResolutionCommand**, **ModeCommand** }
- enum ResponseStatuses {
  CAM_OK, ResponseStatuses::CAM_RANGE_ERROR, **CAM_CHECKSUM_ERROR**, **CAM_UNDEFINED_PROCESS_ERROR**,
  CAM_UNDEFINED_FUNCTION_ERROR, **CAM_TIMEOUT_ERROR**, **CAM_BYTE_COUNT_ERROR**, **CAM_FEATURE_NOT_ENABLED**,
  WRITE_ONLY, **READ_ONLY**, **UNKNOWN_ERROR** }
- enum FunctionCodes : uint8_t {
  NO_OP = 0x00, **SET_DEFAULT** = 0x01, **CAMERA_RESET** = 0x02, **RESTORE_FACTORY_DEFAULTS** = 0x03,
  SERIAL_NUMBER = 0x04, **GET_REVISION** = 0x05, **BAUD_RATE** = 0x07, **GAIN_MODE** = 0x0A,
  FFC_MODE_SELECT = 0x0B, **DO_FFC** = 0x0C, **FFC_PERIOD** = 0x0D, **FFC_TEMP_DELTA** = 0x0E,
  VIDEO_MODE = 0x0F, **VIDEO_PALETTE** = 0x10, **VIDEO_ORIENTATION** = 0x11, **DIGITAL_OUTPUT_MODE** = 0x12,
  AGC_TYPE = 0x13, **CONTRAST** = 0x14, **BRIGHTNESS** = 0x15, **BRIGHTNESS_BIAS** = 0x18,
  SPOT_METER_MODE = 0x1F, **READ_SENSOR** = 0x20, **EXTERNAL_SYNC** = 0x21, **ISOTHERM** = 0x22,
  ISOTHERM_THRESHOLDS = 0x23, **TEST_PATTERN** = 0x25, **VIDEO_COLOR_MODE** = 0x26, **GET_SPOT_METER** = 0x2A,
  SPOT_DISPLAY = 0x2B, **DDE_GAIN** = 0x2C, **FFC_WARN_TIME** = 0x3C, **AGC_FILTER** = 0x3E,
  PLATEAU_LEVEL = 0x3F, **SHUTTER_TEMP** = 0x4D, **AGC_MIDPOINT** = 0x55, **CAMERA_PART** = 0x66,
  MAX_AGC_GAIN = 0x6A, **VIDEO_STANDARD** = 0x72, **DDE_THRESHOLD** = 0xE2, **SPATIAL_THRESHOLD** = 0xE3,
  LENS_RESPONSE_PARAMS = 0xE5, **RADIOMETRY** = 0x8E, **SHUTTER_POSITION** = 0x79 }
- enum CameraSpeed { **_9Hz**, **_30Hz**, **_60Hz** }
- enum RadiometricParameters {
  NONE, **RAD_EMISSIVITY**, RadiometricParameters::RAD_TBKG_X100, RadiometricParameters::RAD_TRANSMISSION_WIN,
  RAD_TREFL_X100, RadiometricParameters::RAD_TATM_X100 }
- enum DigitalOutputModes {
  XPMode, **LVDSMode**, **CMOSBitDepth**, **LVDSBitDepth**,
  NONE }
- enum RangeModes { **Low**, **Middle**, **High** }
- enum FFCModes { **Manual**, **Auto**, **External** }
- enum TestPatterns {
  Off, **Ramp14b**, **BigVertical**, **HorizontalShade**,
  FactoryUse, **ColorBars**, **RampWithSteps** }
- enum XPBusModes { **Disabled**, **BT656**, **CMOS** }

- enum LVDSModes { **Disabled**, **Enabled** }
- enum DigitalOutputDepth { **Bits14b**, **Bits8b**, **Bit8bBayer**, **Bit16bYCbCr** }
- enum SensorTemp { **Sensor**, **Housing** }
- enum Palettes {
  WhiteHot = 0, **BlackHot**, **Fusion**, **RainBow**,
  Globow, **Ironbow1**, **Ironbow2**, **Sepia**,
  Color1, **Color2**, **Icefire**, **Rain**,
  RedHot, **GreenHot** }
- enum AGCTypes {
  PlateauHistogram, **OnceBright**, **AutoBright**, **Manual**,
  NotDefined, **LinearAGC** }
- enum ThresholdUnits { **degC**, **percentage** }
- enum VideoColorModes { **Monochrome**, **Color** }
- enum VideoStandards { **NTSC30Hz** = 0x00, **PAL25Hz** = 0x01, **NTSC60Hz** = 0x03, **PAL50Hz** = 0x04 }
- enum SpotDisplayModes { **Off**, **Numeric**, **Thermometer**, **Both** }
- enum SpotMeterModes { **Off**, **Fahrenheit**, **Celsius** }
- enum ExternalSyncModes { **Disabled**, **Master**, **Slave** }

Public Member Functions

- **CameraSerialSettings** (PvDevice *device, PvDeviceSerial port)
- int SetDefault (void)

  This function sets the WIC device to default setting. The default mode is CMOS 14bit RAW data. This setting is optimal (and mostly necessary) for temperature measurement functions. In this mode, each pixel of camera image is represented by 2 bytes (uint16_t). If you want to use test patterns, palettes and other interpreted data, the bit depth could differ.
- int DoNothing ()
- void DoFFC ()
- void DoSetDefaults (void)
- void DoCameraReset (void)
- void DoRestoreFactoryDefaults (void)
- PvDevice * GetPvDevice (void)

  Getter for PvDevice. This describes the WIC connected device.
- void SetPvDevice (PvDevice *dev)

  Setter for PvDevice. This describes the WIC connected device.
- PvDeviceSerial GetPvDeviceSerial (void)

  Getter for Serial port of PvDevice in use.
- void SetPvDeviceSerial (PvDeviceSerial devSerial)

  Setter for PvDeviceSerial.
- RangeModes GetRangeMode (void)

  Function checks current range mode. Range mode describes the quality and range of pixel data measurement. RangeModes::Low = temperature between approximately -25 - +150 degree Celsius. RangeModes::Middle = temperature between approximately -40 - +500 degree Celsius. RangeModes::High = temperature between approximately +400 - +1500 degree Celsius.
- void SetRangeMode (RangeModes range)

  Function sets the camera range mode. Range mode describes the quality and range of pixel data measurement. RangeModes::Low = temperature between approximately -25 - +150 degree Celsius. RangeModes::Middle = temperature between approximately -40 - +500 degree Celsius. RangeModes::High = temperature between approximately +400 - +1500 degree Celsius. High temperature filter has to be placed in front of camera when RangeModes::High is set.
- bool GetHighRangeLensAvailable (void)

  Function checks whether high temperature mode is available for current lens. This parameter depends on selected lens. For proper measurement it is necessary to place a high temperature filter in front of the camera.
- FFCModes GetFFCMode (void)

Function checks the camera FFC mode in current use. Flat Field Correction (FFC) is used for calibrating the camera. It can make the image and the measurement better. You can perform the FFC manually with DoFFC() function.

- void SetFFCMode (FFCModes mode)

  Function checks the camera FFC mode in current use. Flat Field Correction (FFC) is used for calibrating the camera. It can make the image and the measurement better.
- XPBusModes GetXPBusMode (void)

  Function checks the XP Bus mode in current use. XP Bus mode is the protocol by which the digital data is transfered. It should be set to CMOS, which is also the default value.
- void SetXPBusMode (XPBusModes mode)

  Function sets the XP Bus mode for WIC device. Warning, do not use, if you are sure what you are doing. Default should be CMOS.
- LVDSModes GetLVDSMode (void)

  Function gets the current LVDS mode. Warning, the WIC device is set to use CMOS mode.
- void SetLVDSMode (LVDSModes mode)

  Function sets the LVDS mode. Warning, the WIC device is set to use CMOS mode.
- DigitalOutputDepth GetCMOSBitDepth (void)

  Function gets the CMOS bit depth. The default bit depth should be 14 bit. That is the most precise data for measurement and image from camera. If you use 8 bit, the function for temperature calculation will not work. 8 bit depth (or 16bit YCbCr) could be useful for interpreted image with palettes or test pattern.
- void SetCMOSBitDepth (DigitalOutputDepth depth)

  Function sets the CMOS bit depth.
- DigitalOutputDepth GetLVDSBitDepth (void)

  Function gets the LVDS bit depth.
- void SetLVDSBitDepth (DigitalOutputDepth depth)

  Function sets the LVDS bit depth.
- TestPatterns GetTestPattern (void)

  Gets the current camera image test pattern.
- void SetTestPattern (TestPatterns pattern)

  Sets camera image test pattern.
- ExternalSyncModes GetExternalSync (void)

  Gets the current synchronization mode for FFC.
- void SetExternalSync (ExternalSyncModes mode)

  Sets the synchronization mode for FFC.
- int GetFFCPeriod (void)

  Function gets the current FFC period in number of camera frames. If automatic mode of FFC is selected, the camera will perform FFC each X frames. You can set the number of frames when the correction is performed.
- void SetFFCPeriod (uint16_t period)

  Function sets the FFC period in number of camera frames. If automatic mode of FFC is selected, the camera will perform FFC each X frames. You can set the number of frames when the correction is performed. An argument value of 0 signals that elapsed time will not be used to trigger FFC.
- uint16_t GetFFCTempDelta (void)

  Gets the temperature difference used to trigger automatic FFC. The specified value is converted to Celsius degrees by dividing by 10 then adding 0.1. For Set example, a value of 10 corresponds to a delta temperature of 1.1C degrees.
- void SetFFCTempDelta (uint16_t delta)

  Sets the temperature difference used to trigger automatic FFC The specified value is converted to Celsius degrees by dividing by 10 then adding 0.1. For Set example, a value of 10 corresponds to a delta temperature of 1.1C degree.
- bool GetInvertVideo (void)

  Reverse video horizontally.
- void SetInvertVideo (bool invert)

  Reverse video horizontally.
- bool GetRevertVideo (void)

  Reverse video vertically.
- void SetRevertVideo (bool revert)

Reverse video vertically.

- Palettes GetPalette (void)

  Function gets the current palette in use. The palette are designed for image data which should be not used for temperature measurement.
- void SetPalette (Palettes pal)

  Function sets palette. The palette are designed for image data which should be not used for temperature measurement.
- bool GetAnalogVideo (void)

  Sets WIC to analog video output.
- void SetAnalogVideo (bool video)

  Sets WIC to analog video output.
- bool GetIsRadiometric (void)

  Function returns true, if WIC camera is capable of radiometry.
- bool GetRadiometryMode (void)

  Function returns true, if radiometry is activated.
- void SetRadiometryMode (bool value)

  Functions sets or disables the radiometry mode.
- AGCTypes GetAGCType (void)

  Gets the AGC algorithm.
- void SetAGCType (AGCTypes type)

  Sets the AGC algorithm.
- uint16_t GetContrast (void)

  Gets the contrast value used by oncebright, auto-bright, and manual AGC algorithms. Value 0 to 255.
- void SetContrast (uint16_t contrast)
- uint16_t GetBrightness (void)

  Gets or sets the AGC brightness value used by the manual and auto-bright AGC algorithms.
- void SetBrightness (uint16_t brightness)

  Gets or sets the AGC brightness value used by the manual and auto-bright AGC algorithms.
- int16_t GetBrightnessBias (void)

  Gets the brightness bias value used by the once-bright AGC algorithm.
- void SetBrightnessBias (int16_t brightnessBias)

  Sets the brightness bias value used by the once-bright AGC algorithm.
- bool GetIsotherm (void)

  Gets the isotherm mode (on/off).
- void SetIsotherm (bool isotherm)

  Sets the isotherm mode (on/off).
- ThresholdUnits GetIsothermThresholdUnit (void)

  Gets the isotherm thresholds in percent (e.g. 97 decimal = 97%) or in degress Celsius C (e.g., 97 decimal = 97C).
- void SetIsothermThresholdUnit (ThresholdUnits unit)

  Sets the isotherm thresholds in percent (e.g. 97 decimal = 97%) or in degress Celsius C (e.g., 97 decimal = 97C). Percent is relative to a value of 160C when in high-gain mode and 600C when in low-gain mode. For example, a value of 97% equates to 155C in high-gain mode, 582C in low-gain mode.
- int16_t GetIsothermThresholdLower (void)

  Gets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)
- void SetIsothermThresholdLower (int16_t treshold)

  Sets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)
- int16_t GetIsothermThresholdMiddle (void)

  Gets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)
- void SetIsothermThresholdMiddle (int16_t treshold)

Sets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)

- int16_t GetIsothermThresholdUpper (void)

  Gets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)
- void SetIsothermThresholdUpper (int16_t treshold)

  Sets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)
- VideoColorModes GetVideoColorMode (void)

  Gets the current video color mode.
- void SetVideoColorMode (VideoColorModes mode)

  Sets the video color mode.
- SpotDisplayModes GetSpotDisplayMode (void)

  Gets the spot display mode.
- void SetSpotDisplayMode (SpotDisplayModes mode)

  Sets the spot display mode. mode Spot display mode to be set.
- SpotMeterModes GetSpotMeterMode (void)

  Gets the spot meter mode.
- void SetSpotMeterMode (SpotMeterModes mode)

  Sets the spot meter mode.
- uint16_t GetDDEGain (void)

  Gets the gain value for DDE in manual mode.
- void SetDDEGain (uint16_t gain)

  Sets the gain value for DDE in manual mode.
- uint16_t GetFFCWarnTime (void)

  Gets FFC warn time.
- void SetFFCWarnTime (uint16_t time)

  Sets FFC warn time.
- uint16_t GetAGCFilter (void)

  Gets the AGC filter value.
- void SetAGCFilter (uint16_t filter)

  Sets the AGC filter value.
- uint16_t GetPlateauLevel (void)

  Gets the plateau level for the Plateau AGC algorithm.
- void SetPlateauLevel (uint16_t level)

  Sets the plateau level for the Plateau AGC algorithm.
- uint16_t GetAGCMidpoint (void)

  Gets the AGC midpoint offset, a parameter used by the Plateau-Equalization and Linear AGC algorithms.
- void SetAGCMidpoint (uint16_t point)

  Gets the AGC midpoint offset, a parameter used by the Plateau-Equalization and Linear AGC algorithms.
- uint16_t GetMaxACGGain (void)

  Gets the max-gain parameter for Plateau AGC.
- void SetMaxACGGain (uint16_t gain)

  Sets the max-gain parameter for Plateau AGC.
- uint16_t GetSpatialTreshold (void)

  Gets the spatial threshold of the DDE filter and the DDE mode (auto or manual).
- void SetSpatialTreshold (uint16_t threshold)

  Sets the spatial threshold of the DDE filter and the DDE mode (auto or manual)
- int GetCameraSerialNumber (void)

  Function returns the camera serial number.
- int GetSensorSerialNumber (void)

Function returns the sensor serial number.

- std::string GetCameraPartNumber (void)

  Function returns the camera part number.
- std::string GetPartNumber (void)

  Function returns the WIC part number.
- bool GetIsSupported (void)

  Function checks, if current PvDevice object is supported.
- int GetResolutionX (void)

  Function returns the camera resolution in pixels.
- int GetResolutionY (void)

  Function returns the camera resolution in pixel.
- std::string GetManufacturer (void)

  Function returns the camera device manufacture.
- std::string GetModel (void)

  Function returns WIC model name.
- int GetFWMajorVersion (void)
- int GetFWMinorVersion (void)
- int GetSWMajorVersion (void)
- int GetSWMinorVersion (void)
- double GetShutterTemperature (void)

  Function returns current camera shutter temperature.
- double GetSensorTemperature (void)

  Function returns camera core sensor temperature.
- double GetHousingTemperature (void)

  Function returns current camera housing temperature.
- double GetEmissivity (void)

  Returns current emissivity. This values is used for temperature calculation.
- void SetEmissivity (double value)

  Sets emissivity. This values is used for temperature calculation.
- double GetReflectedTemperatureK (void)

  Returns the reflected temperature value in Kelvins. This is used for temperature calculation.
- void SetReflectedTemperatureK (double value)

  Sets the reflected temperature value in Kelvins. This is used for temperature calculation.
- double GetReflectedTemperatureC (void)

  Returns the reflected temperature value in degrees Celsius. This is used for temperature calculation.
- void SetReflectedTemperatureC (double value)

  Sets the reflected temperature value in degrees Celsius. This is used for temperature calculation.
- double GetAtmosphericTemperatureK (void)

  Returns the atmospheric temperature value in Kelvins. This is used for temperature calculation.
- void SetAtmosphericTemperatureK (double value)

  Sets the atmospheric temperature value in Kelvins. This is used for temperature calculation.
- double GetAtmospericTemperatureC (void)

  Returns the atmospheric temperature value in degrees Celsius. This is used for temperature calculation.
- void SetAtmospericTemperatureC (double value)

  Sets the atmospheric temperature value in degrees Celsius. This is used for temperature calculation.
- double GetDistance (void)

  Returns the current distance of measured object in meters. This is used for temperature calculation.
- void SetDistance (double distance)

  Sets the distance of measured object in meters. This is used for temperature calculation.
- double GetHumidity (void)

  Gets the current humidity in 0 to 1 value (50% = 0.5). This is used for temperature calculation.

- void SetHumidity (double humidity)

  Sets the humidity in 0 to 1 value (50% = 0.5). This is used for temperature calculation.
- bool GetDebugMessages ()

  Function gets current debug mode Debug mode writes special messages used for debugging or development.
- void SetDebugMessages (bool set)

  Function sets debug mode Debug mode writes special messages used for debugging or development.
- std::vector< std::string > GetAvailibleLenses (void)

  Function reruns vector with available lenses setting. Each lens can have different temperature calibration. For the temperature measurement to be precise, the lens should be set correctly.
- void SetLens (std::string lens)

  Function sets selected lens, if available. Each lens has different calibration data, thus it needs to be set correctly to get the best and most accurate temperature results. This operation can take several minutes. The available lenses can be acquired with the GetAvailibleLenses() function. The progress can be checked with function isSetLensReady().
- bool isSetLensReady (void)
- std::string GetCurrentLense (void)

  Function gets which lens is currently used.
- CameraSpeed GetCameraSpeed (void)

  Function gets camera speed.

Friends

- class **Camera**

4.5.1 Member Enumeration Documentation

4.5.1.1 enum CameraSerialSettings::AGCTypes [strong]

Enum for analog video gain control.

4.5.1.2 enum CameraSerialSettings::ArgumentType [strong]

Enum of available argument types for methods SetParameter() and GetParameter().

4.5.1.3 enum CameraSerialSettings::CameraSpeed [strong]

Enum for speed of camera.

4.5.1.4 enum CameraSerialSettings::DigitalOutputDepth [strong]

Enum for setting the digital output depth. Default is 14 bit.

4.5.1.5 enum CameraSerialSettings::DigitalOutputModes [strong]

Enum for camera digital output mode setting. Default is digital CMOS.

4.5.1.6 enum CameraSerialSettings::ExternalSyncModes [strong]

Enum for external FFC mode selection

4.5.1.7 enum CameraSerialSettings::FFCModes [strong]

Enum for Flat Field Correction setting. Default is Auto.

4.5.1.8 enum CameraSerialSettings::FunctionCodes : uint8_t [strong]

Enum for all possible camera commands.

4.5.1.9 enum CameraSerialSettings::LVDSModes [strong]

Enum of LVDS Modes.

4.5.1.10 enum CameraSerialSettings::Palettes [strong]

Enum for selecting usage of palettes.

4.5.1.11 enum CameraSerialSettings::RadiometricParameters [strong]

Enum for radiometric parameter setting.

Enumerator

RAD_TBKG_X100 Emissivity

RAD_TRANSMISSION_WIN Background temperature

RAD_TATM_X100 Reflected temperature Atmospheric temperature

4.5.1.12 enum CameraSerialSettings::RadiometryCommand [strong]

Enum for selecting radiometry command for methods SetParameter() and GetParameter().

4.5.1.13 enum CameraSerialSettings::RangeModes [strong]

Enum for WIC camera range mode setting. Default is Low.

4.5.1.14 enum CameraSerialSettings::Resolution [strong]

Enum of possible WIC camera resolutions.

4.5.1.15 enum CameraSerialSettings::ResponseStatuses [strong]

WIC camera response statuses. When command to camera is send, this is the available answers.

Enumerator

CAM_RANGE_ERROR All fine

4.5.1.16 enum CameraSerialSettings::SensorTemp [strong]

Enum for selecting the sensor measurement for methods SetParameter() and GetParameter().

4.5.1.17 enum CameraSerialSettings::SpotDisplayModes [strong]

Enum for analog video in-picture indicators.

4.5.1.18 enum CameraSerialSettings::SpotMeterModes [strong]

Enum for the unit of SpotDisplayModes

4.5.1.19 enum CameraSerialSettings::TestPatterns [strong]

Enum of available camera test patterns.

4.5.1.20 enum CameraSerialSettings::ThresholdUnits [strong]

Enum for selecting isotherm treshold units.

4.5.1.21 enum CameraSerialSettings::VideoColorModes [strong]

Enum for selecting video color mode.

4.5.1.22 enum CameraSerialSettings::VideoStandards [strong]

Enum for selecting standard of video mode.

4.5.1.23 enum CameraSerialSettings::XPBusModes [strong]

Enum of available camera XP BUS Modes. Default is CMOS.

4.5.2 Member Function Documentation

4.5.2.1 void CameraSerialSettings::DoCameraReset (void)

Resets camera (warning, communication will be lost).

4.5.2.2 void CameraSerialSettings::DoFFC ( )

Makes WIC camera to perform Flat Field Correction (FFC). It can make the image clearer and temperature measurement more precise.

4.5.2.3 int CameraSerialSettings::DoNothing ( )

Test function for camera communication. If communication is fine, returns 0.

Returns

0 = success, -1 = error in communication

4.5.2.4 void CameraSerialSettings::DoRestoreFactoryDefaults (void)

Resets WIC camera to factory defaults.

4.5.2.5 void CameraSerialSettings::DoSetDefaults (void)

Set camera to default settings (Only camera, not WIC device).

4.5.2.6 uint16_t CameraSerialSettings::GetAGCFilter (void)

Gets the AGC filter value.

Returns

Current AGC filter value

4.5.2.7 uint16_t CameraSerialSettings::GetAGCMidpoint (void)

Gets the AGC midpoint offset, a parameter used by the Plateau-Equalization and Linear AGC algorithms.

Returns

Midpoint value

4.5.2.8 CameraSerialSettings::AGCTypes CameraSerialSettings::GetAGCType (void)

Gets the AGC algorithm.

Returns

Current AGC algorithm in use

4.5.2.9 bool CameraSerialSettings::GetAnalogVideo (void)

Sets WIC to analog video output.

Returns

True = analog video enabled

4.5.2.10 double CameraSerialSettings::GetAtmospericTemperatureC (void)

Returns the atmospheric temperature value in degrees Celsius. This is used for temperature calculation.

Returns

Atmospheric temperature [Celsius]

4.5.2.11 double CameraSerialSettings::GetAtmosphericTemperatureK (void)

Returns the atmospheric temperature value in Kelvins. This is used for temperature calculation.

Returns

Atmospheric temperature [Kelvin]

4.5.2.12 std::vector< std::string > CameraSerialSettings::GetAvailibleLenses (void)

Function reruns vector with available lenses setting. Each lens can have different temperature calibration. For the temperature measurement to be precise, the lens should be set correctly.

Returns

vector with available lenses setting (e.g. "019mm")

4.5.2.13 uint16_t CameraSerialSettings::GetBrightness (void)

Gets or sets the AGC brightness value used by the manual and auto-bright AGC algorithms.

Returns

Range: 0 to 16383

4.5.2.14 int16_t CameraSerialSettings::GetBrightnessBias (void)

Gets the brightness bias value used by the once-bright AGC algorithm.

Returns

Range: -16384 to 16383

4.5.2.15 std::string CameraSerialSettings::GetCameraPartNumber (void)

Function returns the camera part number.

Returns

Camera part number as string

4.5.2.16 int CameraSerialSettings::GetCameraSerialNumber (void)

Function returns the camera serial number.

Returns

Camera serial number

4.5.2.17 CameraSerialSettings::CameraSpeed CameraSerialSettings::GetCameraSpeed (void)

Function gets camera speed.

Returns

speed of camera i.e 9Hz, 30Hz, 60Hz

4.5.2.18 CameraSerialSettings::DigitalOutputDepth CameraSerialSettings::GetCMOSBitDepth (void)

Function gets the CMOS bit depth. The default bit depth should be 14 bit. That is the most precise data for measurement and image from camera. If you use 8 bit, the function for temperature calculation will not work. 8 bit depth (or 16bit YCbCr) could be useful for interpreted image with palettes or test pattern.

Returns

Current CMOS bit depth

4.5.2.19 uint16_t CameraSerialSettings::GetContrast (void)

Gets the contrast value used by oncebright, auto-bright, and manual AGC algorithms. Value 0 to 255.

Returns

Contrast value

4.5.2.20 std::string CameraSerialSettings::GetCurrentLense (void)

Function gets which lens is currently used.

Returns

lens String with lens name (e.g. "019mm") or empty String when no lens is currently set

4.5.2.21 uint16_t CameraSerialSettings::GetDDEGain (void)

Gets the gain value for DDE in manual mode.

Returns

DDE gain currently used

4.5.2.22 bool CameraSerialSettings::GetDebugMessages ( )

Function gets current debug mode Debug mode writes special messages used for debugging or development.

Returns

True = debug mode active

4.5.2.23 double CameraSerialSettings::GetDistance (void)

Returns the current distance of measured object in meters. This is used for temperature calculation.

Returns

Distance to measured object [m]

4.5.2.24 double CameraSerialSettings::GetEmissivity (void)

Returns current emissivity. This values is used for temperature calculation.

Returns

Emissivity value (0.5 - 1)

4.5.2.25 CameraSerialSettings::ExternalSyncModes CameraSerialSettings::GetExternalSync (void)

Gets the current synchronization mode for FFC.

Returns

Current FFC synchronization mode

4.5.2.26 CameraSerialSettings::FFCModes CameraSerialSettings::GetFFCMode (void)

Function checks the camera FFC mode in current use. Flat Field Correction (FFC) is used for calibrating the camera. It can make the image and the measurement better. You can perform the FFC manually with DoFFC() function.

Returns

Current FFC mode in use

4.5.2.27 int CameraSerialSettings::GetFFCPeriod (void)

Function gets the current FFC period in number of camera frames. If automatic mode of FFC is selected, the camera will perform FFC each X frames. You can set the number of frames when the correction is performed.

Returns

Number of frames between FFC

4.5.2.28 uint16_t CameraSerialSettings::GetFFCTempDelta (void)

Gets the temperature difference used to trigger automatic FFC. The specified value is converted to Celsius degrees by dividing by 10 then adding 0.1. For Set example, a value of 10 corresponds to a delta temperature of 1.1C degrees.

Returns

Temperature difference - Range: 0 to 1000 (0.1C to 100.1C degrees)

4.5.2.29 uint16_t CameraSerialSettings::GetFFCWarnTime (void)

Gets FFC warn time.

Returns

FFC warn time (frames)

4.5.2.30 int CameraSerialSettings::GetFWMajorVersion (void)

Returns

Camera firmware major version

4.5.2.31 int CameraSerialSettings::GetFWMinorVersion (void)

Returns

Camera firmware minor version

4.5.2.32 bool CameraSerialSettings::GetHighRangeLensAvailable (void)

Function checks whether high temperature mode is available for current lens. This parameter depends on selected lens. For proper measurement it is necessary to place a high temperature filter in front of the camera.

Returns

True if High temperature mode is available

4.5.2.33 double CameraSerialSettings::GetHousingTemperature (void)

Function returns current camera housing temperature.

Returns

Camera housing temperature in degrees Celsius

4.5.2.34 double CameraSerialSettings::GetHumidity (void)

Gets the current humidity in 0 to 1 value (50% = 0.5). This is used for temperature calculation.

Returns

Humidity [0 - 1]

4.5.2.35 bool CameraSerialSettings::GetInvertVideo (void)

Reverse video horizontally.

Returns

True if video is reversed

4.5.2.36 bool CameraSerialSettings::GetIsotherm (void)

Gets the isotherm mode (on/off).

Returns

True = isotherm mode active

4.5.2.37 int16_t CameraSerialSettings::GetIsothermThresholdLower (void)

Gets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)

Returns

Isotherm threshold

4.5.2.38 int16_t CameraSerialSettings::GetIsothermThresholdMiddle (void)

Gets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)

Parameters

threshold	

4.5.2.39 CameraSerialSettings::ThresholdUnits CameraSerialSettings::GetIsothermThresholdUnit (void)

Gets the isotherm thresholds in percent (e.g. 97 decimal = 97%) or in degress Celsius C (e.g., 97 decimal = 97C).

Returns

Current units

4.5.2.40 int16_t CameraSerialSettings::GetIsothermThresholdUpper (void)

Gets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)

Parameters

threshold	

4.5.2.41 bool CameraSerialSettings::GetIsRadiometric (void)

Function returns true, if WIC camera is capable of radiometry.

Returns

True = radiometric camera

4.5.2.42 bool CameraSerialSettings::GetIsSupported (void)

Function checks, if current PvDevice object is supported.

Returns

True = is supported

4.5.2.43 CameraSerialSettings::DigitalOutputDepth CameraSerialSettings::GetLVDSBitDepth (void)

Function gets the LVDS bit depth.

Returns

Current LVDS bit depth

4.5.2.44 CameraSerialSettings::LVDSModes CameraSerialSettings::GetLVDSMode (void)

Function gets the current LVDS mode. Warning, the WIC device is set to use CMOS mode.

Returns

Current LVDS mode

4.5.2.45 std::string CameraSerialSettings::GetManufacturer (void)

Function returns the camera device manufacture.

Returns

Camera manufacture in string

4.5.2.46 uint16_t CameraSerialSettings::GetMaxACGGain (void)

Gets the max-gain parameter for Plateau AGC.

Returns

Current maximal gain

4.5.2.47 std::string CameraSerialSettings::GetModel (void)

Function returns WIC model name.

Returns

WIC model name in string

4.5.2.48 CameraSerialSettings::Palettes CameraSerialSettings::GetPalette (void)

Function gets the current palette in use. The palette are designed for image data which should be not used for temperature measurement.

Returns

Current palette

4.5.2.49 std::string CameraSerialSettings::GetPartNumber (void)

Function returns the WIC part number.

Returns

WIC part number in string

4.5.2.50 uint16_t CameraSerialSettings::GetPlateauLevel (void)

Gets the plateau level for the Plateau AGC algorithm.

Returns

Plateau level value

4.5.2.51 PvDevice * CameraSerialSettings::GetPvDevice (void)

Getter for PvDevice. This describes the WIC connected device.

Returns

PvDevice that is currently in use

4.5.2.52 PvDeviceSerial CameraSerialSettings::GetPvDeviceSerial (void)

Getter for Serial port of PvDevice in use.

Returns

PvDeviceSerial that is currently in use

4.5.2.53 bool CameraSerialSettings::GetRadiometryMode (void)

Function returns true, if radiometry is activated.

Returns

True = radiometry is active

4.5.2.54 CameraSerialSettings::RangeModes CameraSerialSettings::GetRangeMode (void)

Function checks current range mode. Range mode describes the quality and range of pixel data measurement. RangeModes::Low = temperature between approximately -25 - +150 degree Celsius. RangeModes::Middle = temperature between approximately -40 - +500 degree Celsius. RangeModes::High = temperature between approximately +400 - +1500 degree Celsius.

Returns

Current range mode

4.5.2.55 double CameraSerialSettings::GetReflectedTemperatureC (void)

Returns the reflected temperature value in degrees Celsius. This is used for temperature calculation.

Returns

value Reflected temperature [Celsius]

4.5.2.56 double CameraSerialSettings::GetReflectedTemperatureK (void)

Returns the reflected temperature value in Kelvins. This is used for temperature calculation.

Returns

Reflected temperature [Kelvin]

4.5.2.57 int CameraSerialSettings::GetResolutionX (void)

Function returns the camera resolution in pixels.

Returns

X resolution, image width

4.5.2.58 int CameraSerialSettings::GetResolutionY (void)

Function returns the camera resolution in pixel.

Returns

Y resolution, image height

4.5.2.59 bool CameraSerialSettings::GetRevertVideo (void)

Reverse video vertically.

Returns

True if video is reversed

4.5.2.60 int CameraSerialSettings::GetSensorSerialNumber (void)

Function returns the sensor serial number.

Returns

Sensor serial number

4.5.2.61 double CameraSerialSettings::GetSensorTemperature (void)

Function returns camera core sensor temperature.

Returns

Camera sensor temperature in degrees Celsius

4.5.2.62 double CameraSerialSettings::GetShutterTemperature (void)

Function returns current camera shutter temperature.

Returns

Shutter temperature in degrees Celsius

4.5.2.63 uint16_t CameraSerialSettings::GetSpatialTreshold (void)

Gets the spatial threshold of the DDE filter and the DDE mode (auto or manual).

Returns

Current spatial treshold

4.5.2.64 CameraSerialSettings::SpotDisplayModes CameraSerialSettings::GetSpotDisplayMode (void)

Gets the spot display mode.

Returns

Current spot display mode

4.5.2.65 int CameraSerialSettings::GetSWMajorVersion (void)

Returns

Camera software major version

4.5.2.66 int CameraSerialSettings::GetSWMinorVersion (void)

Returns

Camera software minor version

4.5.2.67 CameraSerialSettings::TestPatterns CameraSerialSettings::GetTestPattern (void)

Gets the current camera image test pattern.

Returns

Current test pattern

4.5.2.68 CameraSerialSettings::VideoColorModes CameraSerialSettings::GetVideoColorMode (void)

Gets the current video color mode.

Returns

Current video color mode

4.5.2.69 CameraSerialSettings::XPBusModes CameraSerialSettings::GetXPBusMode (void)

Function checks the XP Bus mode in current use. XP Bus mode is the protocol by which the digital data is transfered. It should be set to CMOS, which is also the default value.

Returns

Current XP Bus mode

4.5.2.70 bool CameraSerialSettings::isSetLensReady (void)

Checks if function GetAvailibleLenses() finished and if the lens is ready. If the return value is false, you can not communicate with WIC camera.

Returns

True if ready, False if not.

4.5.2.71 void CameraSerialSettings::SetAGCFilter (uint16_t *filter*)

Sets the AGC filter value.

Parameters

filter	Range: 0 to 255

4.5.2.72 void CameraSerialSettings::SetAGCMidpoint (uint16_t *point*)

Gets the AGC midpoint offset, a parameter used by the Plateau-Equalization and Linear AGC algorithms.

Parameters

point	Range: 0 to 255

4.5.2.73 void CameraSerialSettings::SetAGCType (AGCTypes *type*)

Sets the AGC algorithm.

Parameters

type	AGC algorithm to be set

4.5.2.74 void CameraSerialSettings::SetAnalogVideo (bool *video*)

Sets WIC to analog video output.

Parameters

video	Sets the analog video output

4.5.2.75 void CameraSerialSettings::SetAtmospericTemperatureC (double *value*)

Sets the atmospheric temperature value in degrees Celsius. This is used for temperature calculation.

Parameters

value	Atmospheric temperature [Celsius]

4.5.2.76 void CameraSerialSettings::SetAtmosphericTemperatureK (double *value*)

Sets the atmospheric temperature value in Kelvins. This is used for temperature calculation.

Parameters

value	Atmospheric temperature [Kelvin]

4.5.2.77 void CameraSerialSettings::SetBrightness (uint16_t *brightness*)

Gets or sets the AGC brightness value used by the manual and auto-bright AGC algorithms.

Parameters

brightness	Range: 0 to 16383

4.5.2.78 void CameraSerialSettings::SetBrightnessBias (int16_t *brightnessBias*)

Sets the brightness bias value used by the once-bright AGC algorithm.

Parameters

brightnessBias	Range: -16384 to 16383

4.5.2.79 void CameraSerialSettings::SetCMOSBitDepth (DigitalOutputDepth *depth*)

Function sets the CMOS bit depth.

Parameters

depth	CMOS bit depth to be set

4.5.2.80 void CameraSerialSettings::SetContrast (uint16_t *contrast*)

Sets the contrast value used by oncebright, auto-bright, and manual AGC algorithms.

Parameters

contrast	Value 0 to 255.

4.5.2.81 void CameraSerialSettings::SetDDEGain (uint16_t *gain*)

Sets the gain value for DDE in manual mode.

Parameters

gain	Range changed from 255 to 65535

4.5.2.82 void CameraSerialSettings::SetDebugMessages (bool *set*)

Function sets debug mode Debug mode writes special messages used for debugging or development.

Parameters

set	True = debug mode active

4.5.2.83 int CameraSerialSettings::SetDefault (void)

This function sets the WIC device to default setting. The default mode is CMOS 14bit RAW data. This setting is optimal (and mostly necessary) for temperature measurement functions. In this mode, each pixel of camera image is represented by 2 bytes (uint16_t). If you want to use test patterns, palettes and other interpreted data, the bit depth could differ.

Returns

0 = success, -1 = error

4.5.2.84 void CameraSerialSettings::SetDistance (double *distance*)

Sets the distance of measured object in meters. This is used for temperature calculation.

Parameters

distance	Distance to measured object [m]

4.5.2.85 void CameraSerialSettings::SetEmissivity (double *value*)

Sets emissivity. This values is used for temperature calculation.

Parameters

value	Emissivity value (0.5 - 1)

4.5.2.86 void CameraSerialSettings::SetExternalSync (ExternalSyncModes *mode*)

Sets the synchronization mode for FFC.

Parameters

mode	FFC synchronization mode to be set

4.5.2.87 void CameraSerialSettings::SetFFCMode (FFCModes *mode*)

Function checks the camera FFC mode in current use. Flat Field Correction (FFC) is used for calibrating the camera. It can make the image and the measurement better.

Parameters

mode	FFC mode to be set

4.5.2.88 void CameraSerialSettings::SetFFCPeriod (uint16_t *period*)

Function sets the FFC period in number of camera frames. If automatic mode of FFC is selected, the camera will perform FFC each X frames. You can set the number of frames when the correction is performed. An argument value of 0 signals that elapsed time will not be used to trigger FFC.

Parameters

period	Number of frames between FFC (0 to 30,000)

4.5.2.89 void CameraSerialSettings::SetFFCTempDelta (uint16_t *delta*)

Sets the temperature difference used to trigger automatic FFC The specified value is converted to Celsius degrees by dividing by 10 then adding 0.1. For Set example, a value of 10 corresponds to a delta temperature of 1.1C degree.

Parameters

delta	Temperature difference - Range: 0 to 1000 (0.1C to 100.1C degrees)

4.5.2.90 void CameraSerialSettings::SetFFCWarnTime (uint16_t *time*)

Sets FFC warn time.

Parameters

time	Range: 0 to 600 (frames)

4.5.2.91 void CameraSerialSettings::SetHumidity (double *humidity*)

Sets the humidity in 0 to 1 value (50% = 0.5). This is used for temperature calculation.

Parameters

humidity	[0 - 1]

4.5.2.92 void CameraSerialSettings::SetInvertVideo (bool *invert*)

Reverse video horizontally.

Parameters

invert	True if video should be reversed

4.5.2.93 void CameraSerialSettings::SetIsotherm (bool *isotherm*)

Sets the isotherm mode (on/off).

Parameters

isotherm	True = isotherm mode active

4.5.2.94 void CameraSerialSettings::SetIsothermThresholdLower (int16_t *treshold*)

Sets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)

Parameters

threshold	

4.5.2.95 void CameraSerialSettings::SetIsothermThresholdMiddle (int16_t *treshold*)

Sets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)

Parameters

threshold	

4.5.2.96 void CameraSerialSettings::SetIsothermThresholdUnit (ThresholdUnits *unit*)

Sets the isotherm thresholds in percent (e.g. 97 decimal = 97%) or in degress Celsius C (e.g., 97 decimal = 97C). Percent is relative to a value of 160C when in high-gain mode and 600C when in low-gain mode. For example, a value of 97% equates to 155C in high-gain mode, 582C in low-gain mode.

Parameters

unit	Unit to be set

4.5.2.97 void CameraSerialSettings::SetIsothermThresholdUpper (int16_t *treshold*)

Sets the isotherm threshold in selected units ThresholdUnits. Thresholds must be in proper order: (Lower <= Middle <= Upper)

Parameters

threshold	

4.5.2.98 void CameraSerialSettings::SetLens (std::string *lens*)

Function sets selected lens, if available. Each lens has different calibration data, thus it needs to be set correctly to get the best and most accurate temperature results. This operation can take several minutes. The available lenses can be acquired with the GetAvailibleLenses() function. The progress can be checked with function isSetLens↩ Ready().

WARNING! Do not power off or disconnect the camera during this time! You can not communicate (get or sets properties) with WIC during this time!

Parameters

lens	String with lens name (e.g. "019mm")

4.5.2.99 void CameraSerialSettings::SetLVDSBitDepth (DigitalOutputDepth *depth*)

Function sets the LVDS bit depth.

Parameters

depth	LVDS bit depth to be set

4.5.2.100 void CameraSerialSettings::SetLVDSMode (LVDSModes *mode*)

Function sets the LVDS mode. Warning, the WIC device is set to use CMOS mode.

Parameters

mode	LVDS mode to be set

4.5.2.101 void CameraSerialSettings::SetMaxACGGain (uint16_t *gain*)

Sets the max-gain parameter for Plateau AGC.

Parameters

gain	Range: 0 to 255

4.5.2.102 void CameraSerialSettings::SetPalette (Palettes *pal*)

Function sets palette. The palette are designed for image data which should be not used for temperature measurement.

Parameters

pal	Palette to be set

4.5.2.103 void CameraSerialSettings::SetPlateauLevel (uint16_t *level*)

Sets the plateau level for the Plateau AGC algorithm.

Parameters

level	Range: 0 to 4095

4.5.2.104 void CameraSerialSettings::SetPvDevice (PvDevice * *dev*)

Setter for PvDevice. This describes the WIC connected device.

Parameters

dev	PvDevice that should be use

4.5.2.105 void CameraSerialSettings::SetPvDeviceSerial (PvDeviceSerial *devSerial*)

Setter for PvDeviceSerial.

Parameters

devSerial	PvDeviceSerial that should be use

4.5.2.106 void CameraSerialSettings::SetRadiometryMode (bool *value*)

Functions sets or disables the radiometry mode.

Parameters

value	True = radiometry should be active

4.5.2.107 void CameraSerialSettings::SetRangeMode (RangeModes *range*)

Function sets the camera range mode. Range mode describes the quality and range of pixel data measurement. RangeModes::Low = temperature between approximately -25 - +150 degree Celsius. RangeModes::Middle = temperature between approximately -40 - +500 degree Celsius. RangeModes::High = temperature between approximately +400 - +1500 degree Celsius. High temperature filter has to be placed in front of camera when Range↩ Modes::High is set.

Parameters

range	Range mode to be set

4.5.2.108 void CameraSerialSettings::SetReflectedTemperatureC (double *value*)

Sets the reflected temperature value in degrees Celsius. This is used for temperature calculation.

Parameters

value	Reflected temperature [Celsius]

4.5.2.109 void CameraSerialSettings::SetReflectedTemperatureK (double *value*)

Sets the reflected temperature value in Kelvins. This is used for temperature calculation.

Parameters

value	Reflected temperature [Kelvin]

4.5.2.110 void CameraSerialSettings::SetRevertVideo (bool *revert*)

Reverse video vertically.

Parameters

revert	True if video should be reversed

4.5.2.111 void CameraSerialSettings::SetSpatialTreshold (uint16_t *threshold*)

Sets the spatial threshold of the DDE filter and the DDE mode (auto or manual)

Parameters

threshold	Auto Threshold range is -20 to 100 (-20 to -1 blurs the image, 1 to 100 sharpens the image)

4.5.2.112 void CameraSerialSettings::SetTestPattern (TestPatterns *pattern*)

Sets camera image test pattern.

Parameters

pattern	Test pattern to be set.

4.5.2.113 void CameraSerialSettings::SetVideoColorMode (VideoColorModes *mode*)

Sets the video color mode.

Parameters

mode	Video color mode to be set

4.5.2.114 void CameraSerialSettings::SetXPBusMode (XPBusModes *mode*)

Function sets the XP Bus mode for WIC device. Warning, do not use, if you are sure what you are doing. Default should be CMOS.

Parameters

mode	XP Bus mode to be set

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

- documentation/SDK_WIC2/CameraSerialSettings.h
- documentation/SDK_WIC2/CameraSerialSettings.cpp

4.6 SffcStruct Struct Reference

Public Attributes

- int **size** = 0
- uint8_t **LowRangeData** [640 ∗512 ∗4]
- uint8_t **MiddleRangeData** [640 ∗512 ∗4]
- uint8_t **Data1500** [640 ∗512 ∗4]

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

- documentation/SDK_WIC2/CameraSerialSettings.cpp

Chapter 5

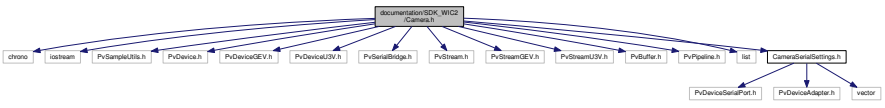
File Documentation

5.1 documentation/SDK_WIC2/Camera.h File Reference

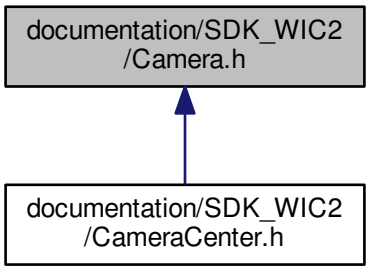
Class for WIC thermocamera control.

```
#include <chrono>
#include <iostream>
#include <PvSampleUtils.h>
#include <PvDevice.h>
#include <PvDeviceGEV.h>
#include <PvDeviceU3V.h>
#include <PvSerialBridge.h>
#include <PvStream.h>
#include <PvStreamGEV.h>
#include <PvStreamU3V.h>
#include <PvBuffer.h>
#include <PvPipeline.h>
#include <list>
#include "CameraSerialSettings.h"
```

Include dependency graph for Camera.h:



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



Classes

- class Camera

5.1.1 Detailed Description

Class for WIC thermocamera control.

Author

Jan Jerabek (jan.jerabek@workswell.cz), Jan Moravec (jan.moravec@workswell.cz)

Copyright

(c) 2016, Workswell s.r.o., All rights reserved.

Date

March, 2016 This class controls WIC cameras. The default use is described in WIC_SDK_Sample.cpp. Basically it is: Connect(), StartAcquisition(), RetreiveBuffer(), ReleaseBuffer(), StopAcquisition() and lastly Disconnect().

Use CameraCenter.cpp class for finding available cameras.

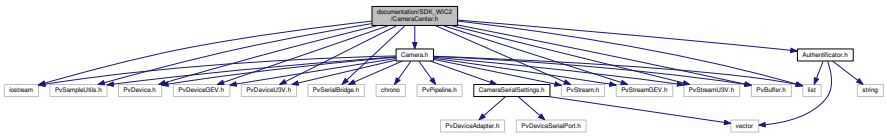
Use the getter GetSettings() for thermocamera settings control.

5.2 documentation/SDK_WIC2/CameraCenter.h File Reference

Class for searching and creating list of available WIC devices.

```
#include "Camera.h"
#include "Authentificator.h"
#include <iostream>
#include <PvSampleUtils.h>
#include <PvDevice.h>
#include <PvDeviceGEV.h>
#include <PvDeviceU3V.h>
#include <PvSerialBridge.h>
#include <PvStream.h>
#include <PvStreamGEV.h>
#include <PvStreamU3V.h>
#include <PvBuffer.h>
#include <list>
```

Include dependency graph for CameraCenter.h:



Classes

- class CameraCenter

5.2.1 Detailed Description

Class for searching and creating list of available WIC devices.

Author

Jan Jerabek (jan.jerabek@workswell.cz), Jan Moravec (jan.moravec@workswell.cz)

Copyright

Date

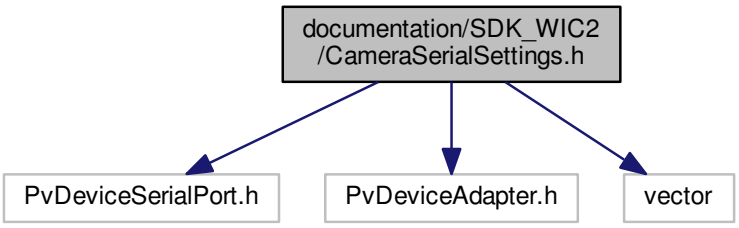
March, 2016 This is the center class for operating with WIC devices. Please, refer to WIC_SDK_Sample.cpp for example use.

5.3 documentation/SDK_WIC2/CameraSerialSettings.h File Reference

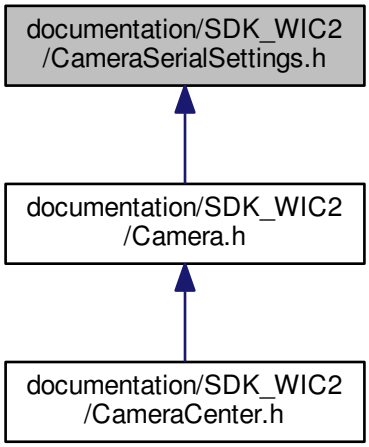
Class for WIC thermocamera setting.

```
#include <PvDeviceSerialPort.h>
#include <PvDeviceAdapter.h>
#include <vector>
```

Include dependency graph for CameraSerialSettings.h:



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



Classes

- class CameraSerialSettings

5.3.1 Detailed Description

Class for WIC thermocamera setting.

Author

Jan Jerabek (jan.jerabek@workswell.cz), Jan Moravec (jan.moravec@workswell.cz)

Copyright

Date

March, 2016 Class uses eBUS SDK from Pleora and implements all necessary function for WIC thermocamera control. When new Camera.cpp instance is created, the method SetDefault() prepares the WIC device for standard use. Basic camera setting is presented in WIC_SDK_Sample.cpp.

Index