Vision Soil Analyzer

1.0.0

Generated by Doxygen 1.8.9.1

Mon Oct 5 2015 21:06:41

Contents

1 Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

2 Hierarchical Index

2.1 Class Hierarchy

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

3 Class Index

3.1 Class List

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

4.1 File List

Here is a list of all files with brief descriptions:

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/GPIOReadException.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Hardware.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/MicroscopeNotFoundException.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/PWM.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/PWM.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/SoilCape.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/SoilCape.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/USB.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/USB.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ValueOutOfBoundsException.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/CommonOperations.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/FFT.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/FFT.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/GA.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/GA.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Mat_archive.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/MathException.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/NN.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/NN.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/predict_t_archive.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/psd.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/SoilMath.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/SoilMathTypes.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Sort.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Stats.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ChannelMismatchException.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ConversionNotSupportedException.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/EmptyImageException.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.cpp ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.h ??

/home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/MorphologicalFilter.cpp ??

5 Namespace Documentation

5.1 boost Namespace Reference

Namespaces

- serialization

5.2 boost::serialization Namespace Reference

Functions

- template<class Archive >
  void serialize (Archive &ar, cv::Mat &m, const unsigned int version __attribute__((unused)))

  serialize Serialize the openCV mat to disk
- template<class Archive >
  void serialize (Archive &ar, Predict_t &P, const unsigned int version __attribute__((unused)))

  serialize Serialize the openCV mat to disk
- template<class Archive >
  void serialize (Archive &ar, SoilAnalyzer::Lab_t &P, const unsigned int version __attribute__((unused)))

  serialize Serialize the openCV mat to disk

5.2.1 Function Documentation

5.2.1.1 template<class Archive > void boost::serialization::serialize (Archive & *ar,* SoilAnalyzer::Lab_t & *P,* const unsigned int version *__attribute__(unused)*) [inline]

serialize Serialize the openCV mat to disk

Definition at line 21 of file lab_t_archive.h.

References SoilAnalyzer::Lab_t::a, SoilAnalyzer::Lab_t::b, and SoilAnalyzer::Lab_t::L.

5.2.1.2 template<class Archive > void boost::serialization::serialize (Archive & *ar,* cv::Mat & *m,* const unsigned int version *__attribute__(unused)*) [inline]

serialize Serialize the openCV mat to disk

Definition at line 24 of file Mat_archive.h.

5.2.1.3 template<class Archive > void boost::serialization::serialize (Archive & *ar,* Predict_t & *P,* const unsigned int version *__attribute__(unused)*) [inline]

serialize Serialize the openCV mat to disk

Definition at line 25 of file predict_t_archive.h.

References Predict_struct::Accuracy, Predict_struct::Category, Predict_struct::OutputNeurons, and Predict_struct::RealValue.

5.3 Hardware Namespace Reference

Namespaces

- Exception

Classes

- class ADC
- class BBB
- class EC12P
- class eQEP
- class GPIO
- class Microscope
- class PWM
- class SoilCape
- class USB

Typedefs

- typedef int(∗ CallbackType) (int)

Functions

- void ∗ threadedPollADC (void ∗value)
- void ∗ colorLoop (void ∗value)
- void ∗ threadedPolleqep (void ∗value)
- void ∗ threadedPollGPIO (void ∗value)

5.3.1 Typedef Documentation

5.3.1.1 typedef int(∗ Hardware::CallbackType) (int)

CallbackType used to pass a function to a thread

Definition at line 37 of file BBB.h.

5.3.2 Function Documentation

5.3.2.1 void ∗ Hardware::colorLoop (void ∗ *value*)

The thread function that runs trough all the colors

Definition at line 91 of file EC12P.cpp.

References Hardware::EC12P::SetPixelColor(), Hardware::EC12P::sleepperiod, and Hardware::EC12P::threadRunning.

Here is the call graph for this function:

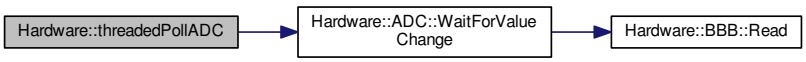
5.3.2.2 void * Hardware::threadedPollADC (void * *value*)

friendly function to start the threading

Definition at line 121 of file ADC.cpp.

References Hardware::BBB::callbackFunction, Hardware::BBB::threadRunning, and Hardware::ADC::WaitForValueChange().

Here is the call graph for this function:

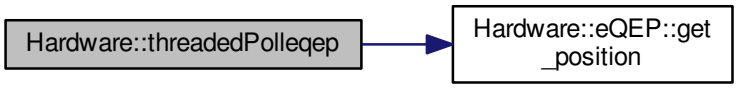


5.3.2.3 void * Hardware::threadedPolleqep (void * *value*)

Definition at line 242 of file eqep.cpp.

References Hardware::BBB::callbackFunction, Hardware::BBB::debounceTime, Hardware::eQEP::get_position(), and Hardware::BBB::threadRunning.

Here is the call graph for this function:



5.3.2.4 void * Hardware::threadedPollGPIO (void * *value*)

Definition at line 266 of file GPIO.cpp.

References Hardware::BBB::callbackFunction, Hardware::BBB::debounceTime, Hardware::BBB::threadRunning, and Hardware::GPIO::WaitForEdge().

Here is the call graph for this function:

5.4 Hardware::Exception Namespace Reference

Classes

- class ADCReadException
- class CouldNotGrabImageException
- class FailedToCreateGPIOPollingThreadException
- class FailedToCreateThreadException
- class GPIOReadException
- class MicroscopeException
- class ValueOutOfBoundsException

5.5 SoilAnalyzer Namespace Reference

Namespaces

- Exception

Classes

- class Analyzer
- struct Lab_t
- class Particle
- struct Point_t
- class Sample
- class SoilSettings

  The SoilSettings class.

5.6 SoilAnalyzer::Exception Namespace Reference

Classes

- class SoilAnalyzerException

5.7 SoilMath Namespace Reference

Genetic Algorithmes used for optimization problems.

Namespaces

- Exception

Classes

- class FFT

  Fast Fourier Transform class.
- class GA
- class NN

  The Neural Network class.
- class PSD
- class Sort

  The Sort template class.
- class Stats

  Stats class.

Functions

- uint16_t MinNotZero (uint16_t a, uint16_t b)
- uint16_t Max (uint16_t a, uint16_t b)
- uint16_t Max (uint16_t a, uint16_t b, uint16_t c, uint16_t d)
- uint16_t Min (uint16_t a, uint16_t b)
- uint16_t Min (uint16_t a, uint16_t b, uint16_t c, uint16_t d)
- static double quick_pow10 (int n)
- static double fastPow (double a, double b)
- static double quick_pow2 (int n)
- static long float2intRound (double d)
- static float calcVolume (float A)

  calcVolume according to ISO 9276-6
- static std::vector< float > makeOutput (uint8_t value, uint32_t noNeurons)
- static float calcDiameter (float A)

  calcDiameter according to ISO 9276-6

5.7.1 Detailed Description

Genetic Algorithmes used for optimization problems.

Use this class for optimization problems. It's currently optimized for Neural Network optimzation

5.7.2 Function Documentation

5.7.2.1 static float SoilMath::calcDiameter (float *A*) [inline],[static]

calcDiameter according to ISO 9276-6

Parameters

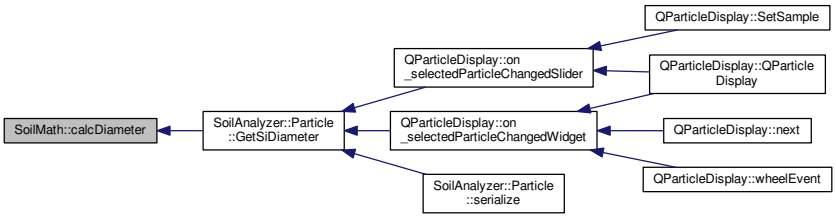
A	

Returns

Definition at line 115 of file CommonOperations.h.

Referenced by SoilAnalyzer::Particle::GetSiDiameter().

Here is the caller graph for this function:



5.7.2.2 static float SoilMath::calcVolume (float *A*) [inline], [static]

calcVolume according to ISO 9276-6

Parameters

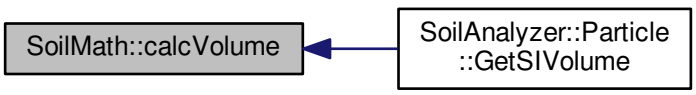
A	

Returns

Definition at line 100 of file CommonOperations.h.

Referenced by SoilAnalyzer::Particle::GetSIVolume().

Here is the caller graph for this function:

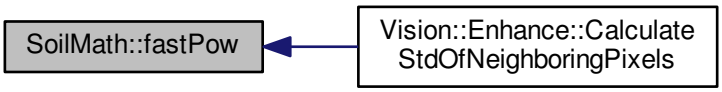


5.7.2.3 static double SoilMath::fastPow (double *a*, double *b*) [inline], [static]

Definition at line 49 of file CommonOperations.h.

Referenced by Vision::Enhance::CalculateStdOfNeighboringPixels().

Here is the caller graph for this function:



5.7.2.4 static long SoilMath::float2intRound (double *d*) [inline], [static]

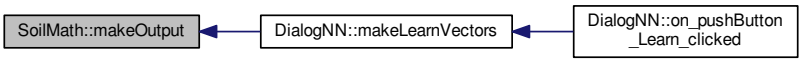
Definition at line 90 of file CommonOperations.h.

5.7.2.5 static std::vector<float> SoilMath::makeOutput (uint8_t *value*, uint32_t *noNeurons*) [inline], [static]

Definition at line 104 of file CommonOperations.h.

Referenced by DialogNN::makeLearnVectors().

Here is the caller graph for this function:



5.7.2.6 uint16_t SoilMath::Max (uint16_t *a*, uint16_t *b*) [inline]

Definition at line 25 of file CommonOperations.h.

Referenced by Max().

Here is the caller graph for this function:

5.7.2.7 uint16_t SoilMath::Max (uint16_t *a*, uint16_t *b*, uint16_t *c*, uint16_t *d*) [inline]

Definition at line 27 of file CommonOperations.h.

References Max().

Here is the call graph for this function:

5.7.2.8 uint16_t SoilMath::Min (uint16_t *a*, uint16_t *b*) [inline]

Definition at line 31 of file CommonOperations.h.

Referenced by Min().

Here is the caller graph for this function:

5.7.2.9 uint16_t SoilMath::Min (uint16_t *a*, uint16_t *b*, uint16_t *c*, uint16_t *d*) [inline]

Definition at line 33 of file CommonOperations.h.

References Min().

Here is the call graph for this function:

5.7.2.10 uint16_t SoilMath::MinNotZero (uint16_t *a*, uint16_t *b*) [inline]

Definition at line 17 of file CommonOperations.h.

5.7.2.11 static double SoilMath::quick_pow10 (int *n*) [inline], [static]

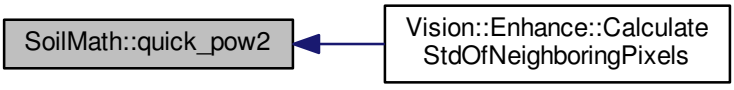
Definition at line 37 of file CommonOperations.h.

5.7.2.12 static double SoilMath::quick_pow2 (int *n*) [inline], [static]

Definition at line 59 of file CommonOperations.h.

Referenced by Vision::Enhance::CalculateStdOfNeighboringPixels().

Here is the caller graph for this function:



5.8 SoilMath::Exception Namespace Reference

Classes

- class MathException

5.9 Ui Namespace Reference

5.10 Vision Namespace Reference

Namespaces

- Exception

Classes

- class Conversion
- class Enhance
- class ImageProcessing
- class MorphologicalFilter
- class Segment

5.11 Vision::Exception Namespace Reference

Classes

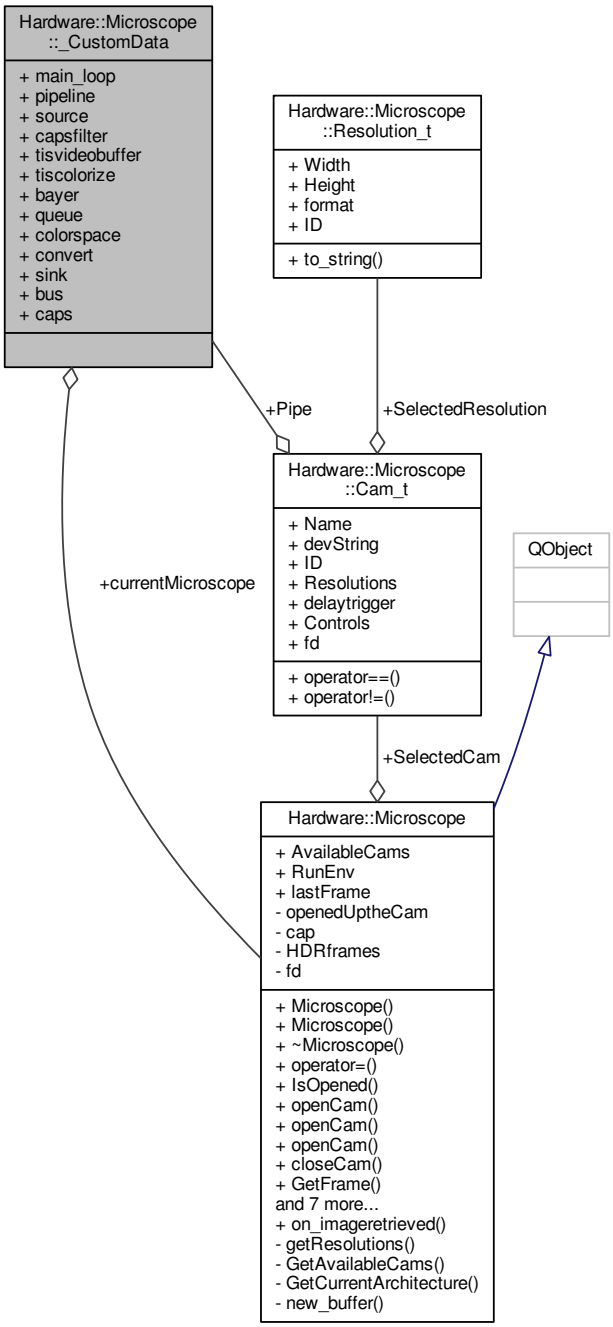
- class ChannelMismatchException
- class ConversionNotSupportedException
- class EmtpyImageException
- class PixelValueOutOfBoundException
- class WrongKernelSizeException

6 Class Documentation

6.1 Hardware::Microscope::_CustomData Struct Reference

`#include <Microscope.h>`

Collaboration diagram for Hardware::Microscope::_CustomData:



Public Attributes

- GMainLoop * main_loop
- GstElement * pipeline
- GstElement * source
- GstElement * capsfilter
- GstElement * tisvideobuffer
- GstElement * tiscolorize
- GstElement * bayer
- GstElement * queue
- GstElement * colorspace
- GstElement * convert
- GstElement * sink
- GstBus * bus
- GstCaps * caps
- Hardware::Microscope * currentMicroscope

6.1.1 Detailed Description

Definition at line 105 of file Microscope.h.

6.1.2 Member Data Documentation

6.1.2.1 GstElement* Hardware::Microscope::_CustomData::bayer

Definition at line 112 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.2 GstBus* Hardware::Microscope::_CustomData::bus

Definition at line 117 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.3 GstCaps* Hardware::Microscope::_CustomData::caps

Definition at line 118 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.4 GstElement* Hardware::Microscope::_CustomData::capsfilter

Definition at line 109 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.5 GstElement* Hardware::Microscope::_CustomData::colorspace

Definition at line 114 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.6 GstElement* Hardware::Microscope::_CustomData::convert

Definition at line 115 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.7 Hardware::Microscope* Hardware::Microscope::_CustomData::currentMicroscope

Definition at line 119 of file Microscope.h.

Referenced by Hardware::Microscope::new_buffer(), and Hardware::Microscope::openCam().

6.1.2.8 GMainLoop* Hardware::Microscope::_CustomData::main_loop

Definition at line 106 of file Microscope.h.

6.1.2.9 GstElement* Hardware::Microscope::_CustomData::pipeline

Definition at line 107 of file Microscope.h.

Referenced by Hardware::Microscope::closeCam(), Hardware::Microscope::GetFrame(), and Hardware::Microscope::openCam().

6.1.2.10 GstElement* Hardware::Microscope::_CustomData::queue

Definition at line 113 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.11 GstElement* Hardware::Microscope::_CustomData::sink

Definition at line 116 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.12 GstElement* Hardware::Microscope::_CustomData::source

Definition at line 108 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.13 GstElement* Hardware::Microscope::_CustomData::tiscolorize

Definition at line 111 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

6.1.2.14 GstElement* Hardware::Microscope::_CustomData::tisvideobuffer

Definition at line 110 of file Microscope.h.

Referenced by Hardware::Microscope::openCam().

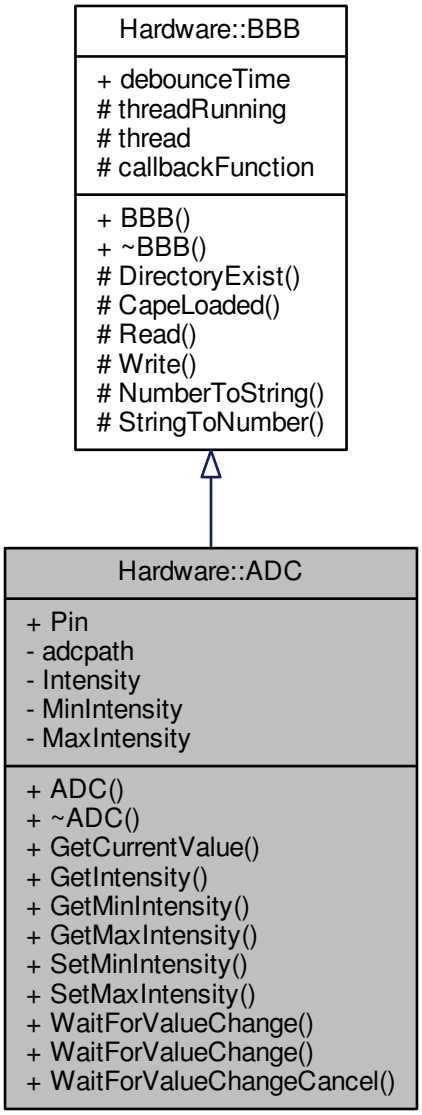
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h

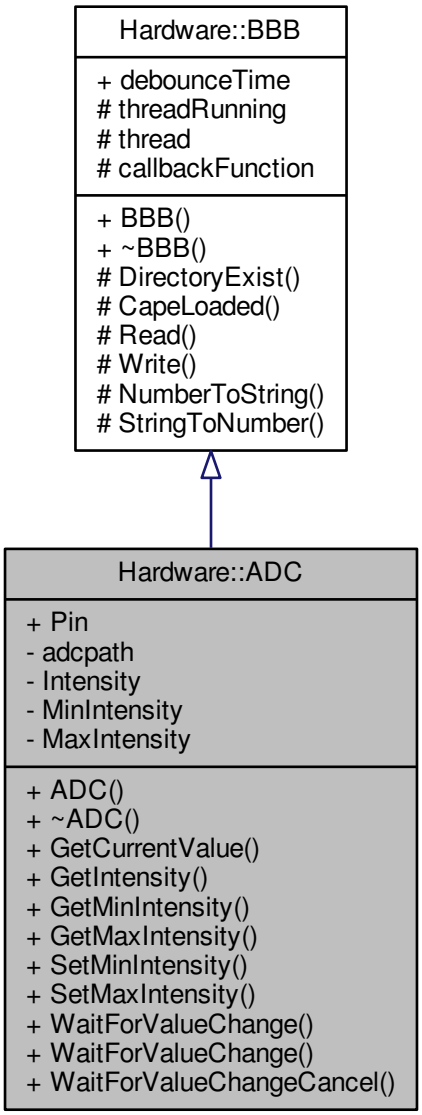
6.2 Hardware::ADC Class Reference

`#include <ADC.h>`

Inheritance diagram for Hardware::ADC:



Collaboration diagram for Hardware::ADC:



Public Types

- enum ADCPin {
  ADC0, ADC1, ADC2, ADC3,
  ADC4, ADC5, ADC6, ADC7 }

Public Member Functions

- ADC (ADCPin pin)
- ~ADC ()
- int GetCurrentValue ()
- float GetIntensity ()
- int GetMinIntensity ()
- int GetMaxIntensity ()
- void SetMinIntensity ()
- void SetMaxIntensity ()
- int WaitForValueChange ()
- int WaitForValueChange (CallbackType callback)
- void WaitForValueChangeCancel ()

Public Attributes

- ADCPin Pin

Private Attributes

- string adcpath
- float Intensity
- int MinIntensity
- int MaxIntensity

Friends

- void * threadedPollADC (void *value)

Additional Inherited Members

6.2.1 Detailed Description

Definition at line 51 of file ADC.h.

6.2.2 Member Enumeration Documentation

6.2.2.1 enum Hardware::ADC::ADCPin

Enumerator to indicate the analogue pin

Enumerator

ADC0 AIN0 pin

ADC1 AIN1 pin

ADC2 AIN2 pin

ADC3 AIN3 pin

ADC4 AIN4 pin

ADC5 AIN5 pin

ADC6 AIN6 pin

ADC7 AIN7 pin

Definition at line 54 of file ADC.h.

6.2.3 Constructor & Destructor Documentation

6.2.3.1 ADC::ADC (ADCPin *pin*)

Constructor

Parameters

pin	and ADCPin type indicating which analogue pin to use

Definition at line 14 of file ADC.cpp.

References ADC0, ADC0_PATH, ADC1, ADC1_PATH, ADC2, ADC2_PATH, ADC3, ADC3_PATH, ADC4, ADC4_PATH, ADC5, ADC5_PATH, ADC6, ADC6_PATH, ADC7, ADC7_PATH, adcpath, MaxIntensity, MinIntensity, and Pin.

6.2.3.2 ADC::~ADC ( )

De-constructor

Definition at line 48 of file ADC.cpp.

6.2.4 Member Function Documentation

6.2.4.1 int ADC::GetCurrentValue ( )

Reads the current voltage in the pin

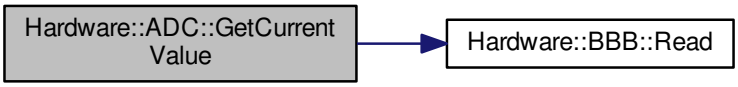
Returns

an integer between 0 and 4096

Definition at line 53 of file ADC.cpp.

References adcpath, Intensity, MaxIntensity, MinIntensity, and Hardware::BBB::Read().

Here is the call graph for this function:



6.2.4.2 float Hardware::ADC::GetIntensity ( ) [inline]

Definition at line 71 of file ADC.h.

6.2.4.3 int Hardware::ADC::GetMaxIntensity ( ) [inline]

Definition at line 73 of file ADC.h.

6.2.4.4 int Hardware::ADC::GetMinIntensity ( ) [inline]

Definition at line 72 of file ADC.h.

6.2.4.5 void ADC::SetMaxIntensity ( )

Definition at line 65 of file ADC.cpp.

References adcpath, MaxIntensity, and Hardware::BBB::Read().

Here is the call graph for this function:

6.2.4.6 void ADC::SetMinIntensity ( )

Set the current voltage at the pin as the minimum voltage

Definition at line 61 of file ADC.cpp.

References adcpath, MinIntensity, and Hardware::BBB::Read().

Here is the call graph for this function:

6.2.4.7 int ADC::WaitForValueChange ( )

Polling of the analogue pin

Returns

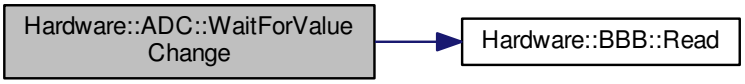
the current value

Definition at line 88 of file ADC.cpp.

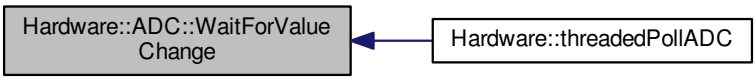
References adcpath, and Hardware::BBB::Read().

Referenced by Hardware::threadedPollADC().

Here is the call graph for this function:



Here is the caller graph for this function:



6.2.4.8 int ADC::WaitForValueChange (CallbackType *callback*)

Threading enabled polling of the analogue pin

Parameters

callback	the function which should be called when polling indicates a change CallbackType

Returns

0

Definition at line 74 of file ADC.cpp.

References Hardware::BBB::callbackFunction, Hardware::BBB::thread, threadedPollADC, and Hardware::BBB::threadRunning.

6.2.4.9 void Hardware::ADC::WaitForValueChangeCancel ( ) [inline]

Definition at line 80 of file ADC.h.

6.2.5 Friends And Related Function Documentation

6.2.5.1 void* threadedPollADC (void * *value*) [friend]

friend polling function

friendly function to start the threading

Definition at line 121 of file ADC.cpp.

Referenced by WaitForValueChange().

6.2.6 Member Data Documentation

6.2.6.1 string Hardware::ADC::adcpath [private]

Path to analogue write file

Definition at line 83 of file ADC.h.

Referenced by ADC(), GetCurrentValue(), SetMaxIntensity(), SetMinIntensity(), and WaitForValueChange().

6.2.6.2 float Hardware::ADC::Intensity [private]

Current intensity expressed as percentage

Definition at line 84 of file ADC.h.

Referenced by GetCurrentValue().

6.2.6.3 int Hardware::ADC::MaxIntensity [private]

Voltage level which represent 100 percentage

Definition at line 86 of file ADC.h.

Referenced by ADC(), GetCurrentValue(), and SetMaxIntensity().

6.2.6.4 int Hardware::ADC::MinIntensity [private]

Voltage level which represent 0 percentage

Definition at line 85 of file ADC.h.

Referenced by ADC(), GetCurrentValue(), and SetMinIntensity().

6.2.6.5 ADCPin Hardware::ADC::Pin

current pin

Definition at line 65 of file ADC.h.

Referenced by ADC().

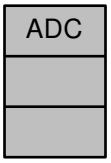
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ADC.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ADC.cpp

6.3 ADC Class Reference

`#include <ADC.h>`

Collaboration diagram for ADC:



6.3.1 Detailed Description

Interaction with the beaglebone analogue pins

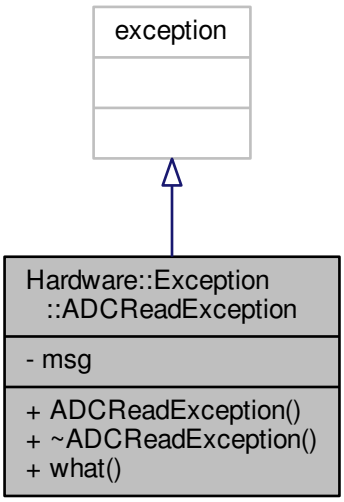
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ADC.h

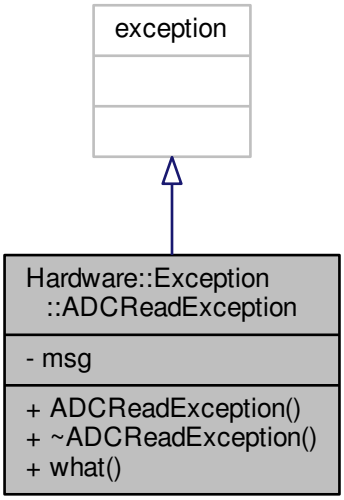
6.4 Hardware::Exception::ADCReadException Class Reference

`#include <ADCReadException.h>`

Inheritance diagram for Hardware::Exception::ADCReadException:



Collaboration diagram for Hardware::Exception::ADCReadException:



Public Member Functions

- ADCReadException (string m="Can't read ADC data!")
- ~ADCReadException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.4.1 Detailed Description

Definition at line 16 of file ADCReadException.h.

6.4.2 Constructor & Destructor Documentation

6.4.2.1 Hardware::Exception::ADCReadException::ADCReadException (string *m* = "Can't read ADC data!") [inline]

Definition at line 18 of file ADCReadException.h.

6.4.2.2 Hardware::Exception::ADCReadException::~ADCReadException ( ) [inline]

Definition at line 19 of file ADCReadException.h.

6.4.3 Member Function Documentation

6.4.3.1 const char* Hardware::Exception::ADCReadException::what ( ) const [inline]

Definition at line 20 of file ADCReadException.h.

6.4.4 Member Data Documentation

6.4.4.1 string Hardware::Exception::ADCReadException::msg [private]

Definition at line 20 of file ADCReadException.h.

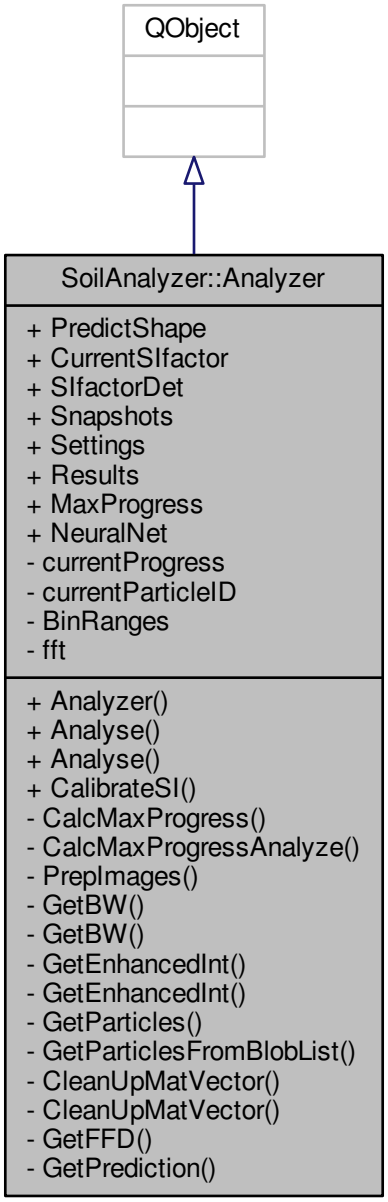
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ADCReadException.h

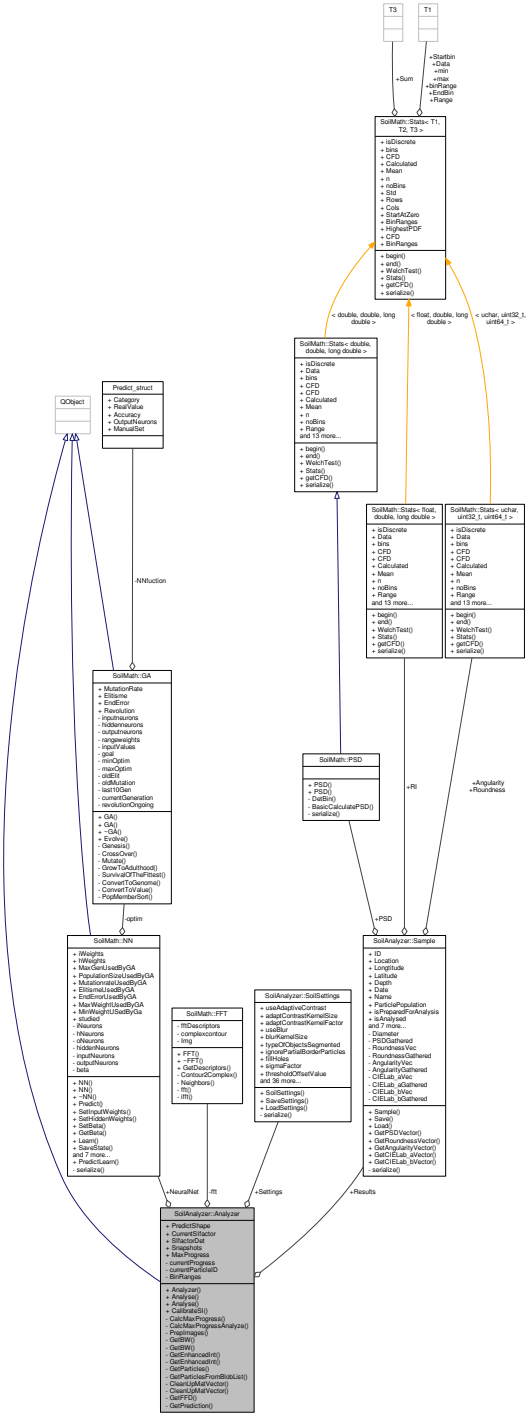
6.5 SoilAnalyzer::Analyzer Class Reference

`#include <analyzer.h>`

Inheritance diagram for SoilAnalyzer::Analyzer:



Collaboration diagram for SoilAnalyzer::Analyzer:



Classes

- struct Image_t

Public Types

- typedef std::vector< Image_t > Images_t

Signals

- void on_progressUpdate (int value)
- void on_maxProgressUpdate (int value)
- void on_AnalysisFinished ()

Public Member Functions

- Analyzer (Images_t *snapshots, Sample *results, SoilSettings *settings)
  Analyzer::Analyzer.
- void Analyse ()
  Analyzer::Analyse.
- void Analyse (Images_t *snapshots, Sample *results, SoilSettings *settings)
- float CalibrateSI (float SI, cv::Mat &img)

Public Attributes

- bool PredictShape = true
- float CurrentSIfactor = 0.0111915
- bool SIfactorDet = false
- Images_t * Snapshots = nullptr
- SoilSettings * Settings = nullptr
- Sample * Results
- uint32_t MaxProgress = STARTING_ESTIMATE_PROGRESS
- SoilMath::NN NeuralNet

Private Member Functions

- void CalcMaxProgress ()
  Analyzer::CalcMaxProgress.
- void CalcMaxProgressAnalyze ()
- void PrepImages ()
  Analyzer::PrepImages.
- void GetBW (std::vector< cv::Mat > &images, std::vector< cv::Mat > &BWvector)
  Analyzer::GetBW.
- void GetBW (cv::Mat &img, cv::Mat &BW)
  Analyzer::GetBW.
- void GetEnhancedInt (Images_t *snapshots, std::vector< cv::Mat > &intensityVector)
- void GetEnhancedInt (cv::Mat &img, cv::Mat &intensity)
- void GetParticles (std::vector< cv::Mat > &BW, Images_t *snapshots, Particle::ParticleVector_t &partPopulation)
  Analyzer::GetParticles.
- void GetParticlesFromBlobList (Vision::Segment::BlobList_t &bloblist, Image_t *snapshot, Particle::ParticleVector_t &partPopulation)
  Analyzer::GetParticlesFromBlobList.
- void CleanUpMatVector (std::vector< cv::Mat > &mv)
- void CleanUpMatVector (Images_t *mv)
  Analyzer::CleanUpMatVector.
- void GetFFD (Particle::ParticleVector_t &particalPopulation)
  Analyzer::GetFFD.
- void GetPrediction (Particle::ParticleVector_t &particlePopulation)
  Analyzer::GetPrediction.

Private Attributes

- uint32_t currentProgress = 0
- uint32_t currentParticleID = 0
- double BinRanges [15]
- SoilMath::FFT fft

6.5.1 Detailed Description

Definition at line 32 of file analyzer.h.

6.5.2 Member Typedef Documentation

6.5.2.1 typedef std::vector<Image_t> SoilAnalyzer::Analyzer::Images_t

Definition at line 45 of file analyzer.h.

6.5.3 Constructor & Destructor Documentation

6.5.3.1 SoilAnalyzer::Analyzer::Analyzer (Images_t * *snapshots*, Sample * *results*, SoilSettings * *settings* = `nullptr`)

Analyzer::Analyzer.

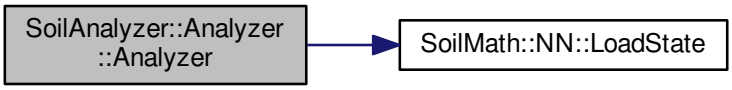
Parameters

snapshots	
results	
settings	

Definition at line 18 of file analyzer.cpp.

References SoilMath::NN::LoadState(), NeuralNet, SoilAnalyzer::SoilSettings::NNlocation, Results, Settings, and Snapshots.

Here is the call graph for this function:



6.5.4 Member Function Documentation

6.5.4.1 void SoilAnalyzer::Analyzer::Analyse ( )

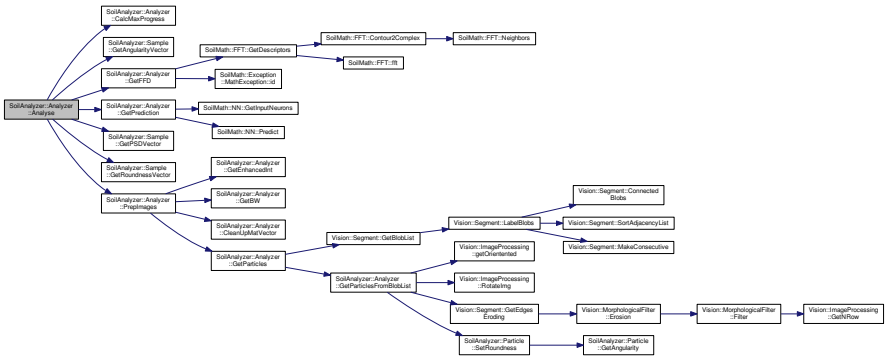
Analyzer::Analyse.

Definition at line 65 of file analyzer.cpp.

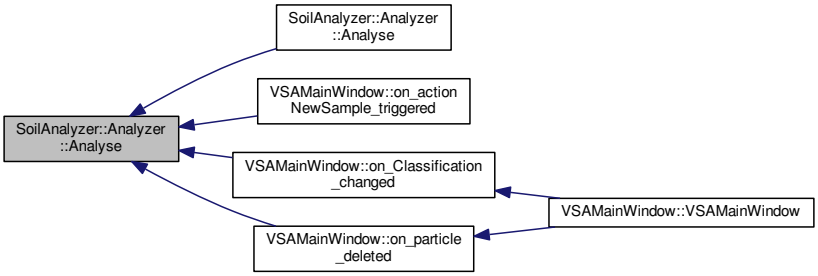
References SoilAnalyzer::Sample::Angularity, BinRanges, CalcMaxProgress(), currentProgress, SoilAnalyzer::Sample::GetAngularityVector(), GetFFD(), GetPrediction(), SoilAnalyzer::Sample::GetPSDVector(), SoilAnalyzer::Sample::GetRoundnessVector(), SoilAnalyzer::Sample::IsLoadedFromDisk, SoilAnalyzer::Sample::isPreparedForAnalysis, on_AnalysisFinished(), on_progressUpdate(), SoilAnalyzer::Sample::ParticlePopulation, PredictShape, SoilAnalyzer::SoilSettings::PredictTheShape, PrepImages(), SoilAnalyzer::Sample::PSD, Results, SoilAnalyzer::Sample::Roundness, and Settings.

Referenced by Analyse(), VSAMainWindow::on_actionNewSample_triggered(), VSAMainWindow::on_Classification_changed(), and VSAMainWindow::on_particle_deleted().

Here is the call graph for this function:



Here is the caller graph for this function:

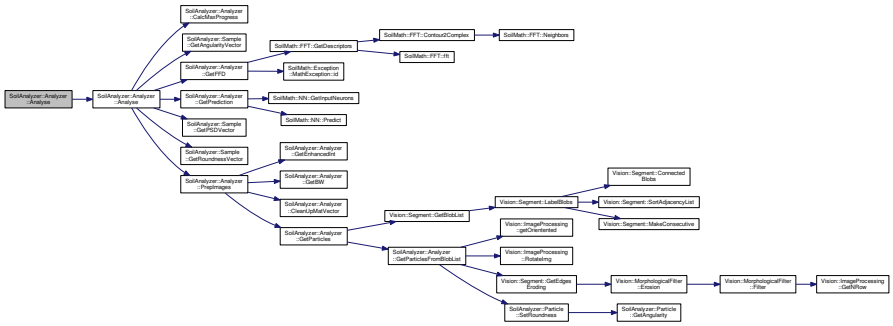


6.5.4.2 void SoilAnalyzer::Analyzer::Analyse (Images_t * *snapshots,* Sample * *results,* SoilSettings * *settings*)

Definition at line 54 of file analyzer.cpp.

References Analyse(), Results, Settings, and Snapshots.

Here is the call graph for this function:



6.5.4.3 void SoilAnalyzer::Analyzer::CalcMaxProgress ( ) [private]

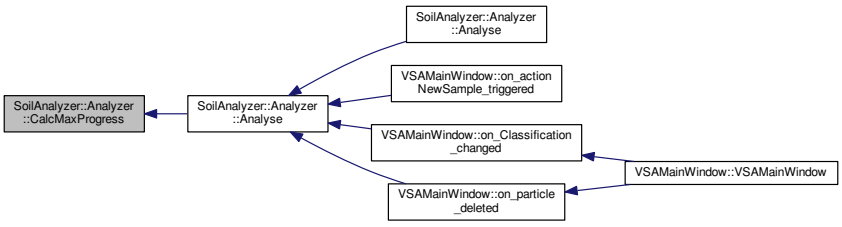
Analyzer::CalcMaxProgress.

Definition at line 112 of file analyzer.cpp.

References SoilAnalyzer::SoilSettings::fillHoles, SoilAnalyzer::SoilSettings::ignorePartialBorderParticles, MaxProgress, SoilAnalyzer::SoilSettings::morphFilterType, Vision::MorphologicalFilter::NONE, on_maxProgressUpdate(), Settings, Snapshots, SoilAnalyzer::SoilSettings::useAdaptiveContrast, and SoilAnalyzer::SoilSettings::useBlur.

Referenced by Analyse().

Here is the caller graph for this function:



6.5.4.4 void SoilAnalyzer::Analyzer::CalcMaxProgressAnalyze ( ) [private]

Definition at line 136 of file analyzer.cpp.

References MaxProgress, on_maxProgressUpdate(), SoilAnalyzer::Sample::ParticlePopulation, Results, and STARTING_ESTIMATE_PROGRESS.

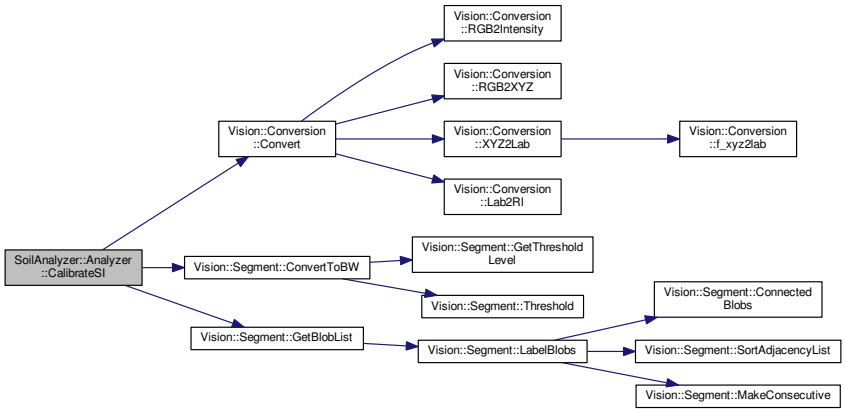
6.5.4.5 float SoilAnalyzer::Analyzer::CalibrateSI (float *SI*, cv::Mat & *img*)

Definition at line 388 of file analyzer.cpp.

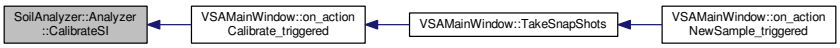
References Vision::Segment::BlobList, Vision::Conversion::Convert(), Vision::Segment::ConvertToBW(), CurrentSIfactor, Vision::Segment::Dark, Vision::Segment::GetBlobList(), Vision::Conversion::Intensity, Vision::ImageProcessing::ProcessedImg, and Vision::Conversion::RGB.

Referenced by VSAMainWindow::on_actionCalibrate_triggered().

Here is the call graph for this function:



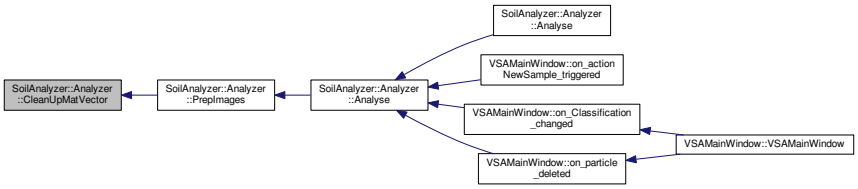
Here is the caller graph for this function:



6.5.4.6 void SoilAnalyzer::Analyzer::CleanUpMatVector (std::vector< cv::Mat > & *mv*) [private]

Referenced by PrepImages().

Here is the caller graph for this function:



6.5.4.7 void SoilAnalyzer::Analyzer::CleanUpMatVector (Images_t * *mv*) [private]

Analyzer::CleanUpMatVector.

Parameters

mv	

Definition at line 101 of file analyzer.cpp.

6.5.4.8 void SoilAnalyzer::Analyzer::GetBW (std::vector< cv::Mat > & *images*, std::vector< cv::Mat > & *BWvector*) [private]

Analyzer::GetBW.

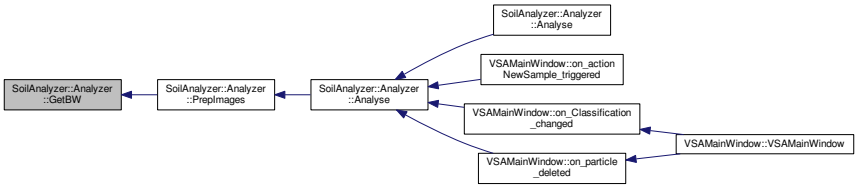
Parameters

images	
BWvector	

Definition at line 222 of file analyzer.cpp.

Referenced by PrepImages().

Here is the caller graph for this function:



6.5.4.9 void SoilAnalyzer::Analyzer::GetBW (cv::Mat & *img*, cv::Mat & *BW*) [private]

Analyzer::GetBW.

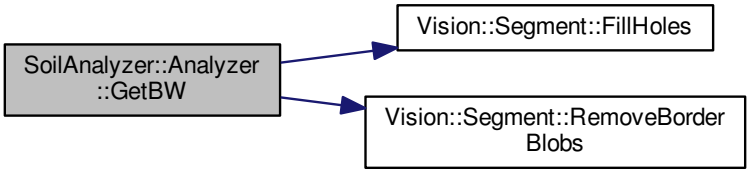
Parameters

img	
BW	

Definition at line 236 of file analyzer.cpp.

References Vision::MorphologicalFilter::CLOSE, currentProgress, Vision::MorphologicalFilter::DILATE, Vision::MorphologicalFilter::EROD↩ E, SoilAnalyzer::SoilSettings::fillHoles, Vision::Segment::FillHoles(), SoilAnalyzer::SoilSettings::filterMaskSize, SoilAnalyzer::SoilSettings↩ ::ignorePartialBorderParticles, SoilAnalyzer::SoilSettings::morphFilterType, Vision::MorphologicalFilter::NONE, on_progressUpdate(), Vision::↩ MorphologicalFilter::OPEN, Vision::ImageProcessing::ProcessedImg, Vision::Segment::RemoveBorderBlobs(), Settings, SHOW_DEBUG_IMG, Vision::Segment::sigma, SoilAnalyzer::SoilSettings::sigmaFactor, SoilAnalyzer::SoilSettings::thresholdOffsetValue, and SoilAnalyzer::Soil↩ Settings::typeOfObjectsSegmented.

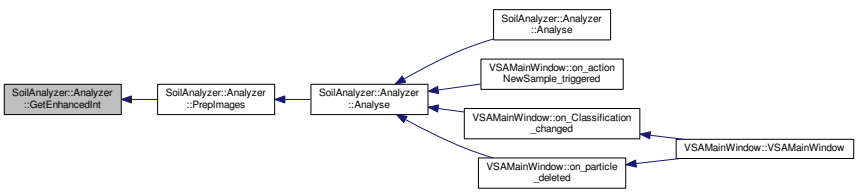
Here is the call graph for this function:



6.5.4.10 void SoilAnalyzer::Analyzer::GetEnhancedInt (Images_t * *snapshots*, std::vector< cv::Mat > & *intensityVector*) [private]

Referenced by PrepImages().

Here is the caller graph for this function:



6.5.4.11 void SoilAnalyzer::Analyzer::GetEnhancedInt (cv::Mat & *img*, cv::Mat & *intensity*) [private]

6.5.4.12 void SoilAnalyzer::Analyzer::GetFFD (Particle::ParticleVector_t & *particalPopulation*) [private]

Analyzer::GetFFD.

Parameters

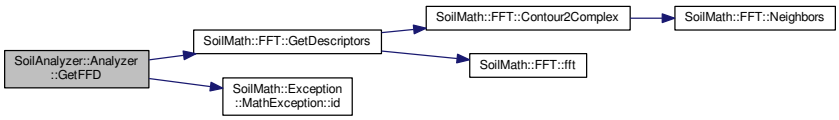
particalPopulation	

Definition at line 350 of file analyzer.cpp.

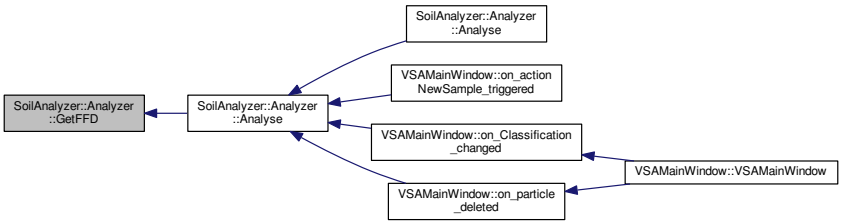
References currentProgress, EXCEPTION_NO_CONTOUR_FOUND_NR, fft, SoilMath::FFT::GetDescriptors(), SoilMath::Exception::Math↩Exception::id(), and on_progressUpdate().

Referenced by Analyse().

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.4.13 void SoilAnalyzer::Analyzer::GetParticles (std::vector< cv::Mat > & *BW*, Images_t * *snapshots*, Particle::ParticleVector_t & *partPopulation*) [private]

Analyzer::GetParticles.

Parameters

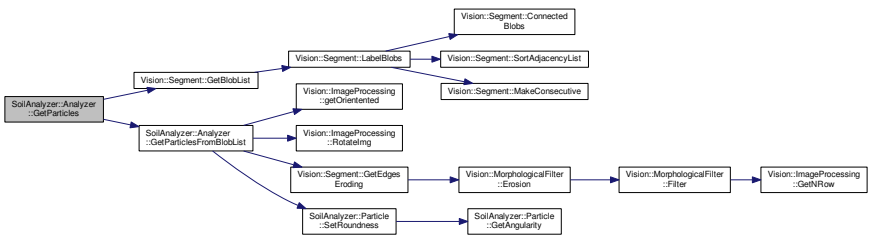
BW	
snapshots	
partPopulation	

Definition at line 303 of file analyzer.cpp.

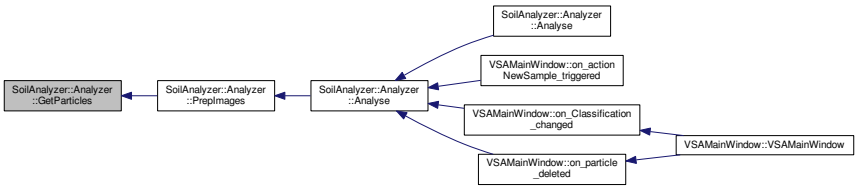
References Vision::Segment::BlobList, currentProgress, Vision::Segment::GetBlobList(), GetParticlesFromBlobList(), and on_progressUpdate().

Referenced by PrepImages().

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.4.14 void SoilAnalyzer::Analyzer::GetParticlesFromBlobList (Vision::Segment::BlobList_t & *bloblist*, Image_t * *snapshot*, Particle::ParticleVector_t & *partPopulation*) [private]

Analyzer::GetParticlesFromBlobList.

Parameters

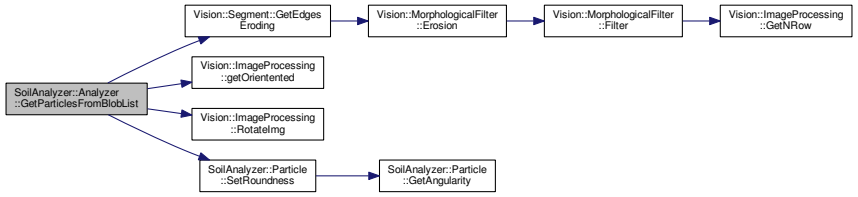
bloblist	
snapshot	
edge	
partPopulation	

Definition at line 322 of file analyzer.cpp.

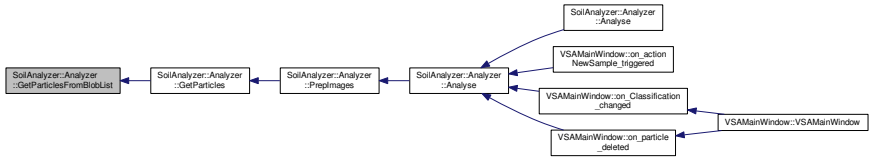
References SoilAnalyzer::Particle::BW, currentParticleID, SoilAnalyzer::Particle::Eccentricty, SoilAnalyzer::Particle::Edge, SoilAnalyzer::↩ Analyzer::Image_t::FrontLight, Vision::Segment::GetEdgesEroding(), Vision::ImageProcessing::getOrientented(), SoilAnalyzer::Particle::ID, SoilAnalyzer::Particle::isPreparedForAnalysis, SoilAnalyzer::Particle::PixelArea, Vision::ImageProcessing::ProcessedImg, SoilAnalyzer::↩ Particle::RGB, Vision::ImageProcessing::RotateImg(), SoilAnalyzer::Particle::SetRoundness(), SoilAnalyzer::Analyzer::Image_t::SIPixelFactor, and SoilAnalyzer::Particle::SIPixelFactor.

Referenced by GetParticles().

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.4.15 void SoilAnalyzer::Analyzer::GetPrediction (Particle::ParticleVector_t & *particlePopulation*) [private]

Analyzer::GetPrediction.

Parameters

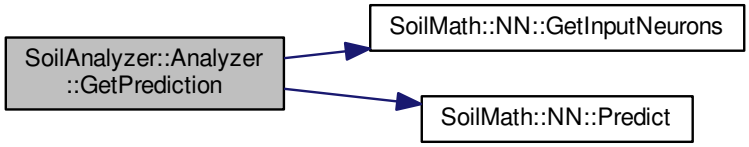
particlePopulation	

Definition at line 373 of file analyzer.cpp.

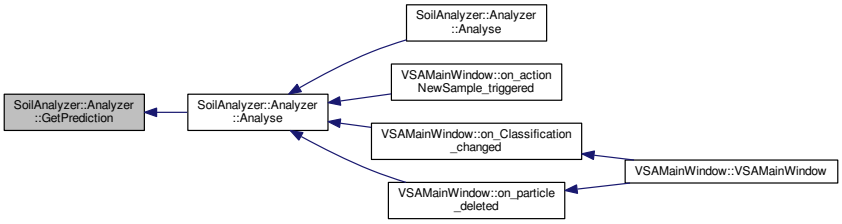
References SoilMath::NN::GetInputNeurons(), NeuralNet, and SoilMath::NN::Predict().

Referenced by Analyse().

Here is the call graph for this function:



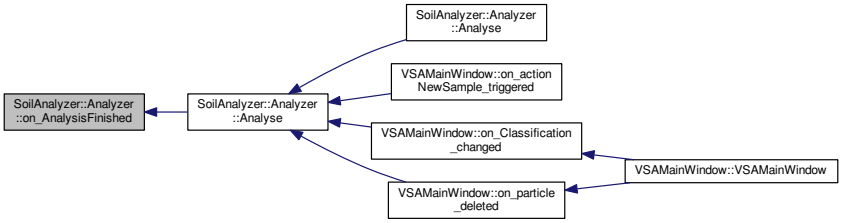
Here is the caller graph for this function:



6.5.4.16 void SoilAnalyzer::Analyzer::on_AnalysisFinished ( ) [signal]

Referenced by Analyse().

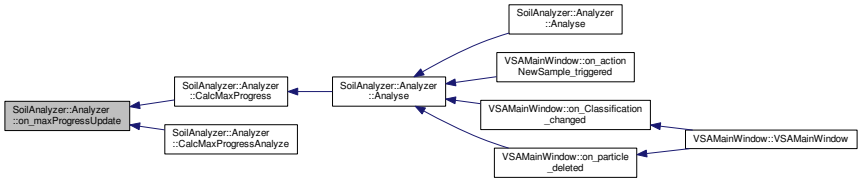
Here is the caller graph for this function:



6.5.4.17 void SoilAnalyzer::Analyzer::on_maxProgressUpdate (int *value*) [signal]

Referenced by CalcMaxProgress(), and CalcMaxProgressAnalyze().

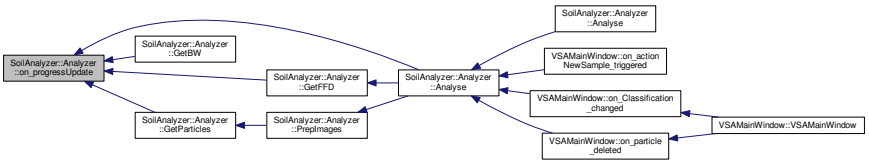
Here is the caller graph for this function:



6.5.4.18 void SoilAnalyzer::Analyzer::on_progressUpdate (int *value*) [signal]

Referenced by Analyse(), GetBW(), GetFFD(), and GetParticles().

Here is the caller graph for this function:



6.5.4.19 void SoilAnalyzer::Analyzer::PrepImages ( ) [private]

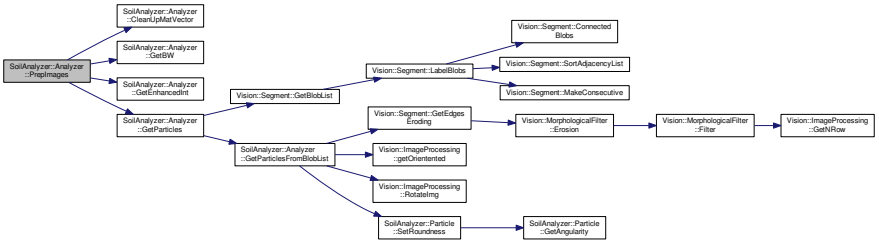
Analyzer::PrepImages.

Definition at line 33 of file analyzer.cpp.

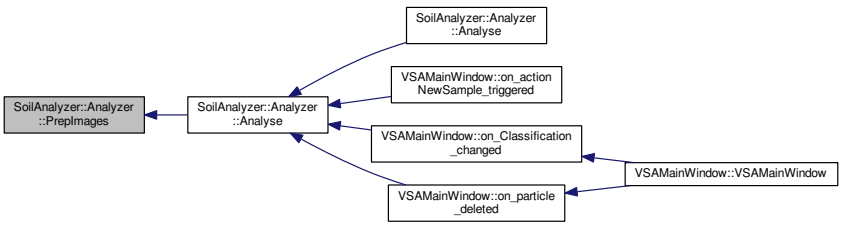
References CleanUpMatVector(), EXCEPTION_NO_SNAPSHOTS, EXCEPTION_NO_SNAPSHOTS_NR, GetBW(), GetEnhancedInt(), Get↩ Particles(), SoilAnalyzer::Sample::isPreparedForAnalysis, SoilAnalyzer::Sample::ParticlePopulation, Results, and Snapshots.

Referenced by Analyse().

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.5 Member Data Documentation

6.5.5.1 double SoilAnalyzer::Analyzer::BinRanges[15] [private]

Initial value:

```
{0.0,  0.038, 0.045, 0.063, 0.075, 0.09, 0.125, 0.18,
                     0.25, 0.355, 0.5,   0.71,  1.0,   1.4,  2.0}
```

Definition at line 69 of file analyzer.h.

Referenced by Analyse().

6.5.5.2 uint32_t SoilAnalyzer::Analyzer::currentParticleID = 0 [private]

Definition at line 68 of file analyzer.h.

Referenced by GetParticlesFromBlobList().

6.5.5.3 uint32_t SoilAnalyzer::Analyzer::currentProgress = 0 [private]

Definition at line 67 of file analyzer.h.

Referenced by Analyse(), GetBW(), GetFFD(), and GetParticles().

6.5.5.4 float SoilAnalyzer::Analyzer::CurrentSIfactor = 0.0111915

Definition at line 37 of file analyzer.h.

Referenced by CalibrateSI(), and VSAMainWindow::TakeSnapShots().

6.5.5.5 SoilMath::FFT SoilAnalyzer::Analyzer::fft [private]

Definition at line 72 of file analyzer.h.

Referenced by GetFFD().

6.5.5.6 uint32_t SoilAnalyzer::Analyzer::MaxProgress = STARTING_ESTIMATE_PROGRESS

Definition at line 57 of file analyzer.h.

Referenced by CalcMaxProgress(), CalcMaxProgressAnalyze(), and VSAMainWindow::VSAMainWindow().

6.5.5.7 SoilMath::NN SoilAnalyzer::Analyzer::NeuralNet

Definition at line 59 of file analyzer.h.

Referenced by Analyzer(), GetPrediction(), VSAMainWindow::on_actionNeuralNet_triggered(), and VSAMainWindow::VSAMainWindow().

6.5.5.8 bool SoilAnalyzer::Analyzer::PredictShape = true

Definition at line 36 of file analyzer.h.

Referenced by Analyse(), and VSAMainWindow::on_actionUseLearning_toggled().

6.5.5.9 Sample* SoilAnalyzer::Analyzer::Results

Definition at line 49 of file analyzer.h.

Referenced by Analyse(), Analyzer(), CalcMaxProgressAnalyze(), VSAMainWindow::on_actionLoadSample_triggered(), and PrepImages().

6.5.5.10 SoilSettings* SoilAnalyzer::Analyzer::Settings = nullptr

Definition at line 47 of file analyzer.h.

Referenced by Analyse(), Analyzer(), CalcMaxProgress(), and GetBW().

6.5.5.11 bool SoilAnalyzer::Analyzer::SIfactorDet = false

Definition at line 38 of file analyzer.h.

Referenced by VSAMainWindow::TakeSnapShots().

6.5.5.12 Images_t* SoilAnalyzer::Analyzer::Snapshots = nullptr

Definition at line 46 of file analyzer.h.

Referenced by Analyse(), Analyzer(), CalcMaxProgress(), and PrepImages().

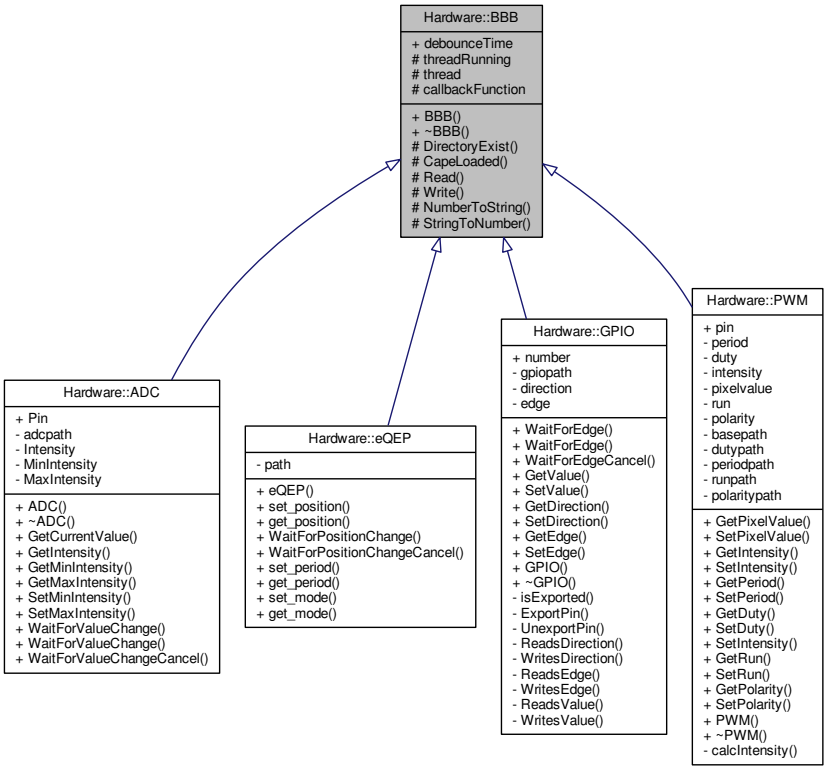
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/analyzer.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/analyzer.cpp

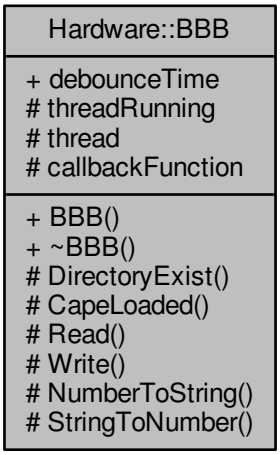
6.6 Hardware::BBB Class Reference

`#include <BBB.h>`

Inheritance diagram for Hardware::BBB:



Collaboration diagram for Hardware::BBB:



Public Member Functions

- BBB ()
- ~BBB ()

Public Attributes

- int debounceTime

Protected Member Functions

- bool DirectoryExist (const string &path)
- bool CapeLoaded (const string &shield)
- string Read (const string &path)
- void Write (const string &path, const string &value)
- template<typename T >
  string NumberToString (T Number)
- template<typename T >
  T StringToNumber (string Text)

Protected Attributes

- bool threadRunning
- pthread_t thread
- CallbackType callbackFunction

6.6.1 Detailed Description

Definition at line 40 of file BBB.h.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 BBB::BBB ( )

Constructor

Definition at line 12 of file BBB.cpp.

References callbackFunction, debounceTime, thread, and threadRunning.

6.6.2.2 BBB::~BBB ( )

De-constructor

Definition at line 20 of file BBB.cpp.

6.6.3 Member Function Documentation

6.6.3.1 bool BBB::CapeLoaded (const string & *shield*) [protected]

Checks if a cape is loaded in the file /sys/devices/bone_capemgr.9/slots

Parameters

shield	a const search string which is a (part) of the shield name

Returns

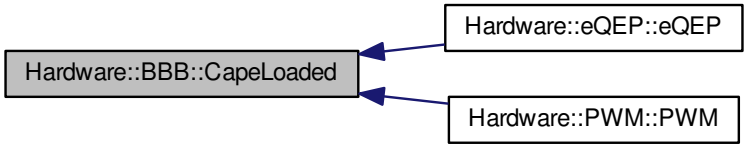
true if the search string is found otherwise false

Definition at line 67 of file BBB.cpp.

References SLOTS.

Referenced by Hardware::eQEP::eQEP(), and Hardware::PWM::PWM().

Here is the caller graph for this function:



6.6.3.2 bool BBB::DirectoryExist (const string & *path*) [protected]

Checks if a directory exist

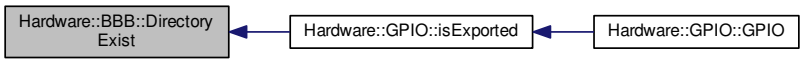
Returns

true if the directory exists and false if not

Definition at line 55 of file BBB.cpp.

Referenced by Hardware::GPIO::isExported().

Here is the caller graph for this function:



6.6.3.3 template<typename T > string Hardware::BBB::NumberToString (T *Number*) [inline], [protected]

Converts a number to a string

Parameters

Number	as typename

Returns

the number as a string

Definition at line 62 of file BBB.h.

6.6.3.4 string BBB::Read (const string & *path*) [protected]

Reads the first line from a file

Parameters

path	constant string pointing towards the file

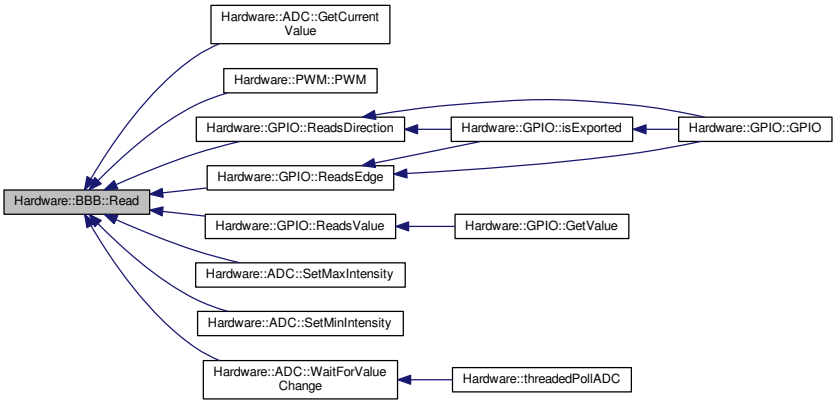
Returns

this first line

Definition at line 26 of file BBB.cpp.

Referenced by Hardware::ADC::GetCurrentValue(), Hardware::PWM::PWM(), Hardware::GPIO::ReadsDirection(), Hardware::GPIO::ReadsEdge(), Hardware::GPIO::ReadsValue(), Hardware::ADC::SetMaxIntensity(), Hardware::ADC::SetMinIntensity(), and Hardware::ADC::WaitForValueChange().

Here is the caller graph for this function:



6.6.3.5 template<typename T> T Hardware::BBB::StringToNumber (string *Text*) [inline], [protected]

Converts a string to a number

Parameters

Text	the string that needs to be converted

Returns

the number as typename

Definition at line 72 of file BBB.h.

6.6.3.6 void BBB::Write (const string & *path*, const string & *value*) [protected]

Writes a value to a file

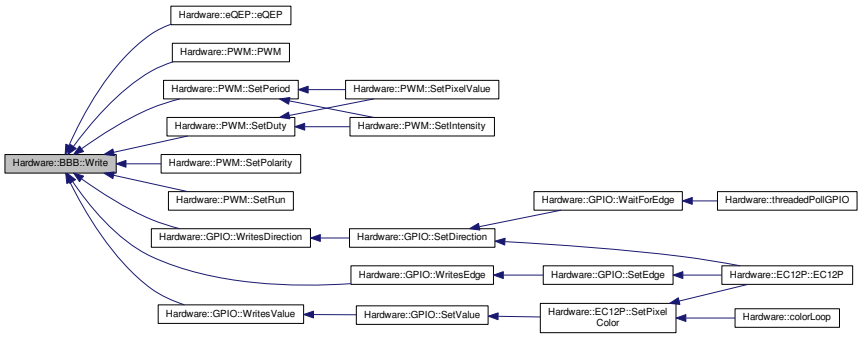
Parameters

path	a constant string pointing towards the file
value	a constant string which should be written in the file

Definition at line 42 of file BBB.cpp.

Referenced by Hardware::eQEP::eQEP(), Hardware::PWM::PWM(), Hardware::PWM::SetDuty(), Hardware::PWM::SetPeriod(), Hardware::↩ PWM::SetPolarity(), Hardware::PWM::SetRun(), Hardware::GPIO::WritesDirection(), Hardware::GPIO::WritesEdge(), and Hardware::GPIO::↩ WritesValue().

Here is the caller graph for this function:



6.6.4 Member Data Documentation

6.6.4.1 CallbackType Hardware::BBB::callbackFunction [protected]

the callbakcfunction

Definition at line 50 of file BBB.h.

Referenced by BBB(), Hardware::threadedPollADC(), Hardware::threadedPolleqep(), Hardware::threadedPollGPIO(), Hardware::GPIO::Wait↩ ForEdge(), Hardware::eQEP::WaitForPositionChange(), and Hardware::ADC::WaitForValueChange().

6.6.4.2 int Hardware::BBB::debounceTime

debounce time for a button in milliseconds

Definition at line 42 of file BBB.h.

Referenced by BBB(), Hardware::threadedPolleqep(), and Hardware::threadedPollGPIO().

6.6.4.3 pthread_t Hardware::BBB::thread [protected]

The thread

Definition at line 49 of file BBB.h.

Referenced by BBB(), Hardware::GPIO::WaitForEdge(), Hardware::eQEP::WaitForPositionChange(), and Hardware::ADC::WaitForValue↩ Change().

6.6.4.4 bool Hardware::BBB::threadRunning [protected]

used to stop the thread

Definition at line 48 of file BBB.h.

Referenced by BBB(), Hardware::threadedPollADC(), Hardware::threadedPolleqep(), Hardware::threadedPollGPIO(), Hardware::GPIO::WaitForEdge(), Hardware::eQEP::WaitForPositionChange(), Hardware::eQEP::WaitForPositionChangeCancel(), and Hardware::ADC::WaitForValueChange().

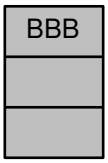
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/BBB.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/BBB.cpp

6.7 BBB Class Reference

`#include <BBB.h>`

Collaboration diagram for BBB:



6.7.1 Detailed Description

The core BeagleBone Black class used for all hardware related classes. Consisting of universal used method, functions and variables. File operations, polling and threading

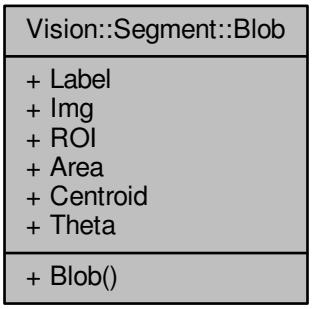
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/BBB.h

6.8 Vision::Segment::Blob Struct Reference

`#include <Segment.h>`

Collaboration diagram for Vision::Segment::Blob:



Public Member Functions

- Blob (uint16_t label, uint32_t area)

Public Attributes

- uint16_t Label
- cv::Mat Img
- cv::Rect ROI
- uint32_t Area
- cv::Point_< double > Centroid
- double Theta

6.8.1 Detailed Description

Individual blob

Definition at line 42 of file Segment.h.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Vision::Segment::Blob::Blob (uint16_t *label*, uint32_t *area*) [inline]

Definition at line 51 of file Segment.h.

6.8.3 Member Data Documentation

6.8.3.1 uint32_t Vision::Segment::Blob::Area

Calculated stats of the blob

Definition at line 48 of file Segment.h.

6.8.3.2 cv::Point_<double> Vision::Segment::Blob::Centroid

Definition at line 49 of file Segment.h.

6.8.3.3 cv::Mat Vision::Segment::Blob::Img

BW image of the blob all the pixel belonging to the blob are set to 1 others are 0

Definition at line 44 of file Segment.h.

6.8.3.4 uint16_t Vision::Segment::Blob::Label

ID of the blob

Definition at line 43 of file Segment.h.

6.8.3.5 cv::Rect Vision::Segment::Blob::ROI

Coordinates for the blob in the original picture as a cv::Rect

Definition at line 46 of file Segment.h.

6.8.3.6 double Vision::Segment::Blob::Theta

Definition at line 50 of file Segment.h.

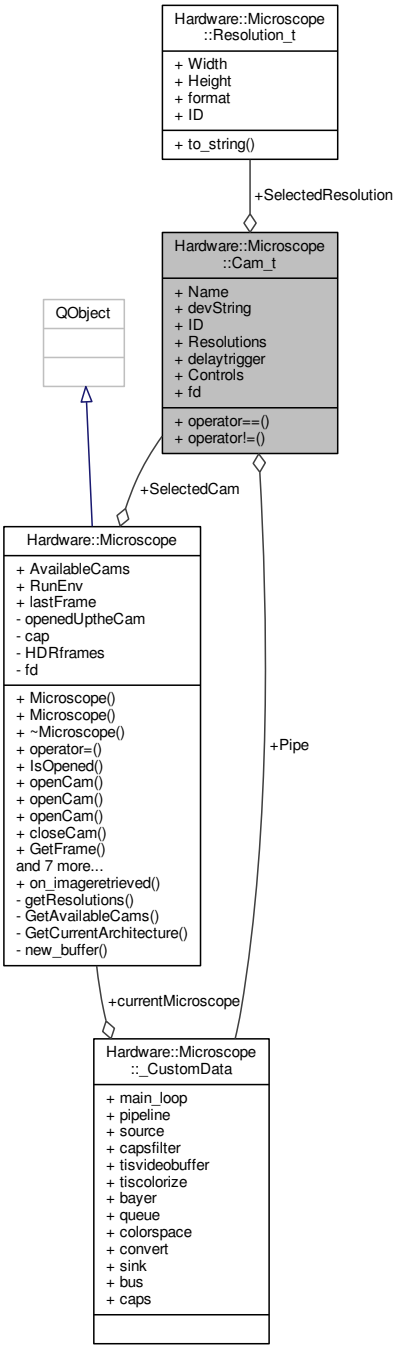
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.h

6.9 Hardware::Microscope::Cam_t Struct Reference

`#include <Microscope.h>`

Collaboration diagram for Hardware::Microscope::Cam_t:



Public Member Functions

- bool operator== (Cam_t const &rhs)
- bool operator!= (Cam_t const &rhs)

Public Attributes

- std::string Name
- std::string devString
- uint32_t ID
- std::vector< Resolution_t > Resolutions
- uint32_t delaytrigger = 1
- Resolution_t * SelectedResolution = nullptr
- Controls_t Controls
- CustomData Pipe
- int fd

6.9.1 Detailed Description

Definition at line 122 of file Microscope.h.

6.9.2 Member Function Documentation

6.9.2.1 bool Hardware::Microscope::Cam_t::operator!= (Cam_t const & *rhs*) [inline]

Definition at line 139 of file Microscope.h.

References ID, and Name.

6.9.2.2 bool Hardware::Microscope::Cam_t::operator== (Cam_t const & *rhs*) [inline]

Definition at line 132 of file Microscope.h.

References ID, and Name.

6.9.3 Member Data Documentation

6.9.3.1 Controls_t Hardware::Microscope::Cam_t::Controls

Definition at line 129 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), Hardware::Microscope::GetControl(), Hardware::Microscope::openCam(), and DialogSettings::SetCamControl().

6.9.3.2 uint32_t Hardware::Microscope::Cam_t::delaytrigger = 1

Definition at line 127 of file Microscope.h.

6.9.3.3 std::string Hardware::Microscope::Cam_t::devString

Definition at line 124 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), Hardware::Microscope::openCam(), and Hardware::Microscope::SetControl().

6.9.3.4 int Hardware::Microscope::Cam_t::fd

Definition at line 131 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), Hardware::Microscope::getResolutions(), and Hardware::Microscope::SetControl().

6.9.3.5 uint32_t Hardware::Microscope::Cam_t::ID

Definition at line 125 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), operator!=(), and operator==().

6.9.3.6 std::string Hardware::Microscope::Cam_t::Name

Definition at line 123 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), Hardware::Microscope::openCam(), operator!=(), and operator==().

6.9.3.7 CustomData Hardware::Microscope::Cam_t::Pipe

Definition at line 130 of file Microscope.h.

Referenced by Hardware::Microscope::closeCam(), Hardware::Microscope::GetFrame(), and Hardware::Microscope::openCam().

6.9.3.8 std::vector<Resolution_t> Hardware::Microscope::Cam_t::Resolutions

Definition at line 126 of file Microscope.h.

Referenced by Hardware::Microscope::getResolutions().

6.9.3.9 Resolution_t* Hardware::Microscope::Cam_t::SelectedResolution = nullptr

Definition at line 128 of file Microscope.h.

Referenced by Hardware::Microscope::new_buffer(), and Hardware::Microscope::openCam().

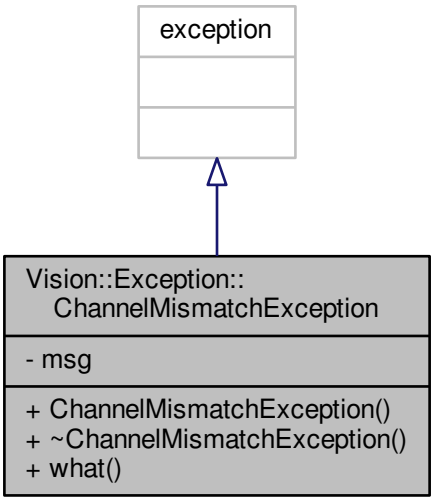
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h

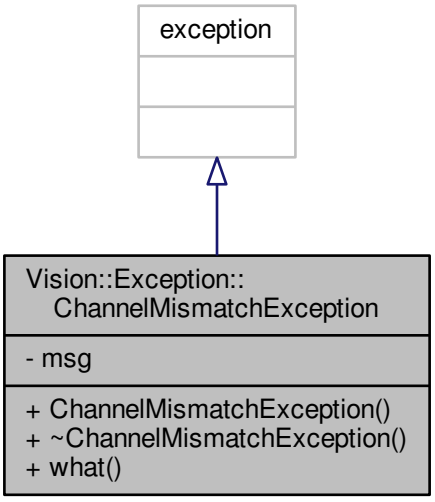
6.10 Vision::Exception::ChannelMismatchException Class Reference

`#include <ChannelMismatchException.h>`

Inheritance diagram for Vision::Exception::ChannelMismatchException:



Collaboration diagram for Vision::Exception::ChannelMismatchException:



Public Member Functions

- ChannelMismatchException (string m="Extracted channel out of bounds exception!")
- ~ChannelMismatchException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.10.1 Detailed Description

Definition at line 21 of file ChannelMismatchException.h.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 Vision::Exception::ChannelMismatchException::ChannelMismatchException (string *m* = "Extracted channel out of bounds exception!") [inline]

Definition at line 23 of file ChannelMismatchException.h.

6.10.2.2 Vision::Exception::ChannelMismatchException::~ChannelMismatchException ( ) [inline]

Definition at line 26 of file ChannelMismatchException.h.

6.10.3 Member Function Documentation

6.10.3.1 const char* Vision::Exception::ChannelMismatchException::what ( ) const [inline]

Definition at line 27 of file ChannelMismatchException.h.

6.10.4 Member Data Documentation

6.10.4.1 string Vision::Exception::ChannelMismatchException::msg [private]

Definition at line 27 of file ChannelMismatchException.h.

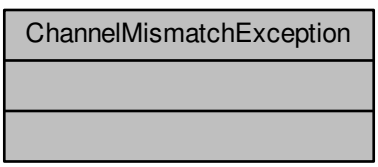
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ChannelMismatchException.h

6.11 ChannelMismatchException Class Reference

`#include <ChannelMismatchException.h>`

Collaboration diagram for ChannelMismatchException:



6.11.1 Detailed Description

Exception class which is thrown when Extracted channel out of bounds exception

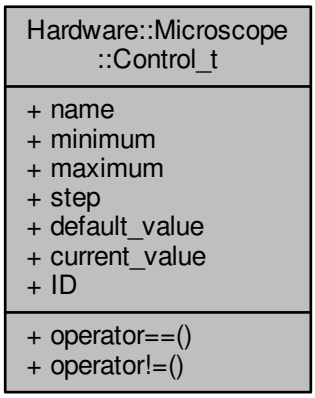
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ChannelMismatchException.h

6.12 Hardware::Microscope::Control_t Struct Reference

`#include <Microscope.h>`

Collaboration diagram for Hardware::Microscope::Control_t:



Public Member Functions

- bool operator== (Control_t &rhs)
- bool operator!= (Control_t &rhs)

Public Attributes

- std::string name
- int minimum
- int maximum
- int step
- int default_value
- int current_value
- uint32_t ID = V4L2_CID_BASE

6.12.1 Detailed Description

Definition at line 79 of file Microscope.h.

6.12.2 Member Function Documentation

6.12.2.1 bool Hardware::Microscope::Control_t::operator!= (Control_t & *rhs*) [inline]

Definition at line 94 of file Microscope.h.

References name.

6.12.2.2 bool Hardware::Microscope::Control_t::operator== (Control_t & *rhs*) [inline]

Definition at line 87 of file Microscope.h.

References name.

6.12.3 Member Data Documentation

6.12.3.1 int Hardware::Microscope::Control_t::current_value

Definition at line 85 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), Hardware::Microscope::GetHDRFrame(), and Hardware::Microscope::SetControl().

6.12.3.2 int Hardware::Microscope::Control_t::default_value

Definition at line 84 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams().

6.12.3.3 uint32_t Hardware::Microscope::Control_t::ID = V4L2_CID_BASE

Definition at line 86 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), and Hardware::Microscope::SetControl().

6.12.3.4 int Hardware::Microscope::Control_t::maximum

Definition at line 82 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), and Hardware::Microscope::GetHDRFrame().

6.12.3.5 int Hardware::Microscope::Control_t::minimum

Definition at line 81 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), and Hardware::Microscope::GetHDRFrame().

6.12.3.6 std::string Hardware::Microscope::Control_t::name

Definition at line 80 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams(), operator!=(), and operator==().

6.12.3.7 int Hardware::Microscope::Control_t::step

Definition at line 83 of file Microscope.h.

Referenced by Hardware::Microscope::GetAvailableCams().

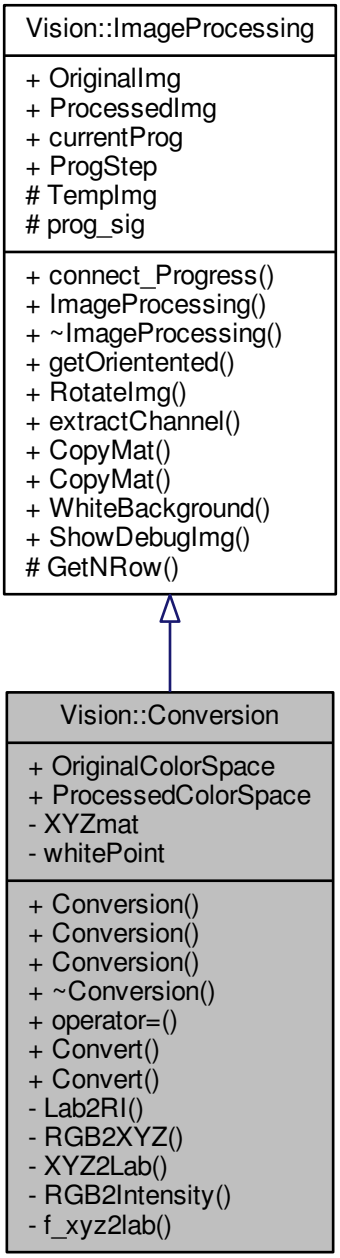
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h

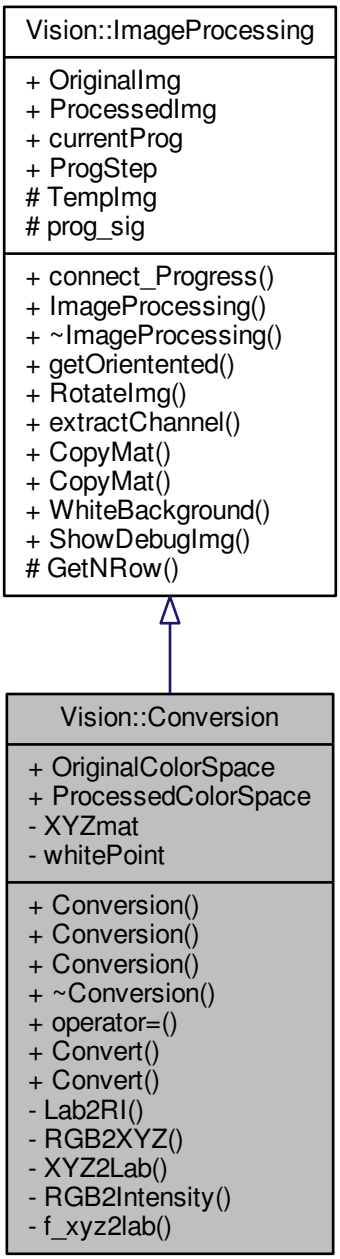
6.13 Vision::Conversion Class Reference

`#include <Conversion.h>`

Inheritance diagram for Vision::Conversion:



Collaboration diagram for Vision::Conversion:



Public Types

- enum ColorSpace {
  CIE_lab, CIE_XYZ, RI, RGB,
  Intensity, None }

Public Member Functions

- Conversion ()
- Conversion (const Mat &src)
- Conversion (const Conversion &rhs)
- ~Conversion ()
- Conversion & operator= (Conversion rhs)
- void Convert (ColorSpace convertFrom, ColorSpace convertTo, bool chain=false)
- void Convert (const Mat &src, Mat &dst, ColorSpace convertFrom, ColorSpace convertTo, bool chain=false)

Public Attributes

- ColorSpace OriginalColorSpace
- ColorSpace ProcessedColorSpace

Private Member Functions

- void Lab2RI (float *O, float *P, int nData)
- void RGB2XYZ (uchar *O, float *P, int nData)
- void XYZ2Lab (float *O, float *P, int nData)
- void RGB2Intensity (uchar *O, uchar *P, int nData)
- float f_xyz2lab (float t)

Private Attributes

- float XYZmat [3][3]
- float whitePoint [3]

Additional Inherited Members

6.13.1 Detailed Description

Definition at line 13 of file Conversion.h.

6.13.2 Member Enumeration Documentation

6.13.2.1 enum Vision::Conversion::ColorSpace

Enumerator which indicates the colorspace used

Enumerator

CIE_lab CIE La*b* colorspace

CIE_XYZ CIE XYZ colorspace

RI Redness Index colorspace

RGB RGB colorspace

Intensity Grayscale colorspace

None none

Definition at line 16 of file Conversion.h.

6.13.3 Constructor & Destructor Documentation

6.13.3.1 Conversion::Conversion ( )

Constructor of the class

Definition at line 14 of file Conversion.cpp.

References None, OriginalColorSpace, and ProcessedColorSpace.

6.13.3.2 Conversion::Conversion (const Mat & *src*)

Constructor of the class

Parameters

src	a cv::Mat object which is the source image

Definition at line 22 of file Conversion.cpp.

References None, OriginalColorSpace, Vision::ImageProcessing::OriginalImg, and ProcessedColorSpace.

6.13.3.3 Conversion::Conversion (const Conversion & *rhs*)

Copy constructor

Definition at line 29 of file Conversion.cpp.

References OriginalColorSpace, Vision::ImageProcessing::OriginalImg, ProcessedColorSpace, Vision::ImageProcessing::ProcessedImg, and Vision::ImageProcessing::TempImg.

6.13.3.4 Conversion::~Conversion ( )

De-constructor of the class

Definition at line 38 of file Conversion.cpp.

6.13.4 Member Function Documentation

6.13.4.1 void Conversion::Convert (ColorSpace *convertFrom*, ColorSpace *convertTo*, bool *chain* = `false`)

Convert the source image from one colorspace to a destination colorspace posibilities are:

- RGB 2 Intensity
- RGB 2 XYZ
- RGB 2 Lab
- RGB 2 Redness Index
- XYZ 2 Lab
- XYZ 2 Redness Index
- Lab 2 Redness Index

Parameters

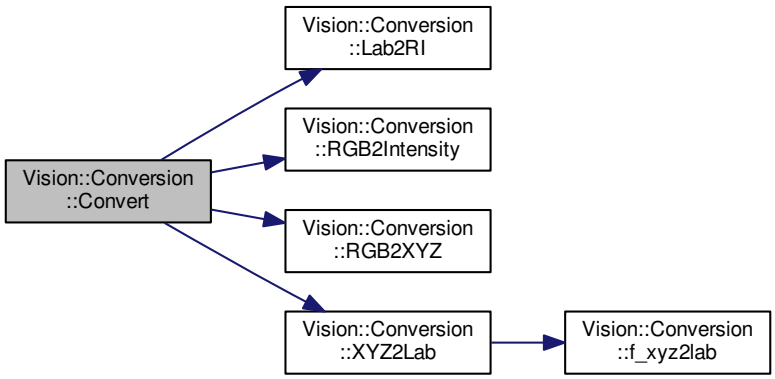
convertFrom	the starting colorspace
convertTo	the destination colorspace
chain	use the results from the previous operation default value = false;

Definition at line 86 of file Conversion.cpp.

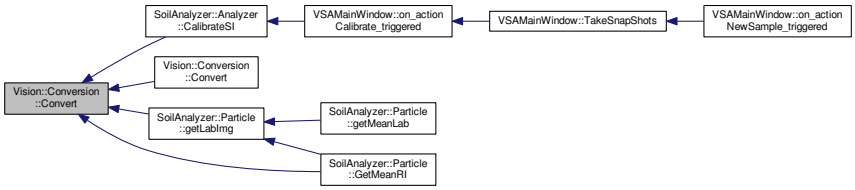
References CHAIN_PROCESS, CIE_lab, CIE_XYZ, Vision::ImageProcessing::currentProg, EMPTY_CHECK, Intensity, Lab2RI(), OriginalColorSpace, Vision::ImageProcessing::OriginalImg, ProcessedColorSpace, Vision::ImageProcessing::ProcessedImg, Vision::ImageProcessing::prog_sig, Vision::ImageProcessing::ProgStep, RGB, RGB2Intensity(), RGB2XYZ(), RI, and XYZ2Lab().

Referenced by SoilAnalyzer::Analyzer::CalibrateSI(), Convert(), SoilAnalyzer::Particle::getLabImg(), and SoilAnalyzer::Particle::GetMeanRI().

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.4.2 void Conversion::Convert (const Mat & *src*, Mat & *dst*, ColorSpace *convertFrom*, ColorSpace *convertTo*, bool *chain* = `false`)

Convert the source image from one colorspace to a destination colorspace

- RGB 2 Intensity
- RGB 2 XYZ
- RGB 2 Lab
- RGB 2 Redness Index
- XYZ 2 Lab
- XYZ 2 Redness Index
- Lab 2 Redness Index

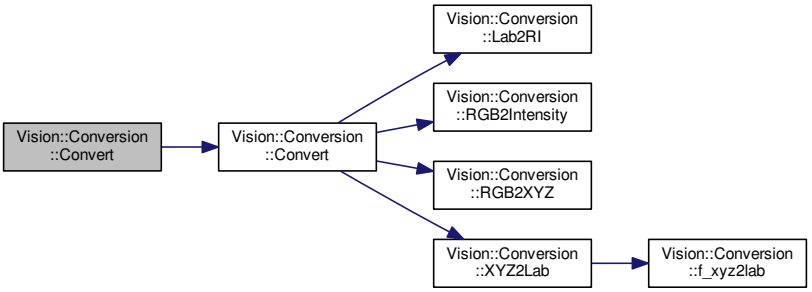
Parameters

src	a cv::Mat object which is the source image
dst	a cv::Mat object which is the destination image
convertFrom	the starting colorspace
convertTo	the destination colorspace
chain	use the results from the previous operation default value = false;

Definition at line 66 of file Conversion.cpp.

References Convert(), Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

Here is the call graph for this function:

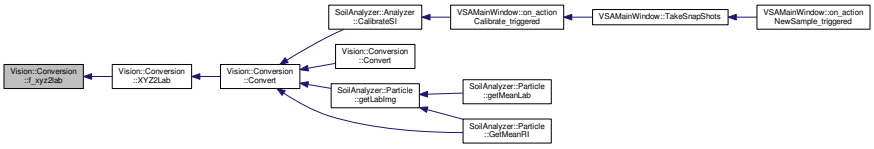


6.13.4.3 float Conversion::f_xyz2lab (float *t*) `[inline],[private]`

Definition at line 244 of file Conversion.cpp.

Referenced by XYZ2Lab().

Here is the caller graph for this function:



6.13.4.4 void Conversion::Lab2RI (float * *O*, float * *P*, int *nData*) [private]

Conversion from CIE La*b* to Redness Index

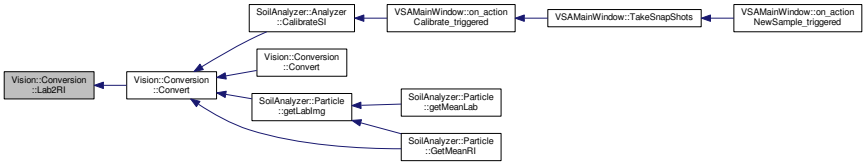
Parameters

O	a uchar pointer to the source image
P	a uchar pointer to the destination image
nData	an int indicating the total number of pixels

Definition at line 256 of file Conversion.cpp.

Referenced by Convert().

Here is the caller graph for this function:



6.13.4.5 Conversion & Conversion::operator= (Conversion *rhs*)

Assignment operator

Definition at line 41 of file Conversion.cpp.

References OriginalColorSpace, Vision::ImageProcessing::OriginalImg, ProcessedColorSpace, Vision::ImageProcessing::ProcessedImg, and Vision::ImageProcessing::TempImg.

6.13.4.6 void Conversion::RGB2Intensity (uchar * *O*, uchar * *P*, int *nData*) [private]

Conversion from RGB to Intensity

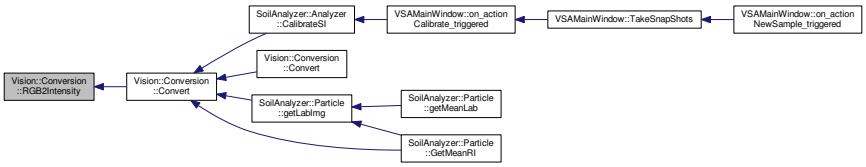
Parameters

O	a uchar pointer to the source image
P	a uchar pointer to the destination image
nData	an int indicating the total number of pixels

Definition at line 190 of file Conversion.cpp.

Referenced by Convert().

Here is the caller graph for this function:



6.13.4.7 void Conversion::RGB2XYZ (uchar * *O*, float * *P*, int *nData*) [private]

Conversion from RGB to CIE XYZ

Parameters

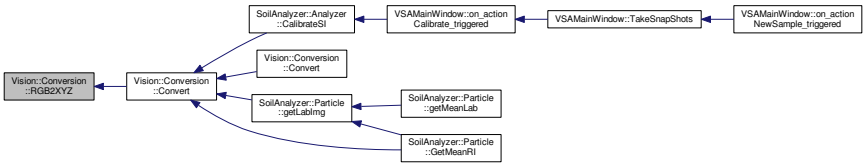
O	a uchar pointer to the source image
P	a uchar pointer to the destination image
nData	an int indicating the total number of pixels

Definition at line 207 of file Conversion.cpp.

References Vision::ImageProcessing::OriginalImg, and XYZmat.

Referenced by Convert().

Here is the caller graph for this function:



6.13.4.8 void Conversion::XYZ2Lab (float * *O*, float * *P*, int *nData*) [private]

Conversion from CIE XYZ to CIE La*b*

Parameters

O	a uchar pointer to the source image
P	a uchar pointer to the destination image
nData	an int indicating the total number of pixels

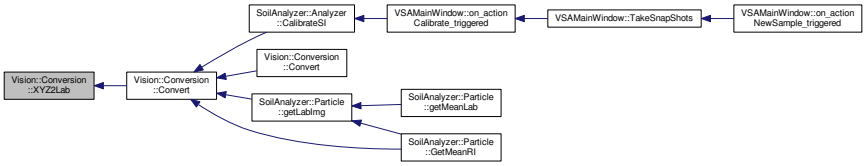
Definition at line 225 of file Conversion.cpp.

References f_xyz2lab(), and whitePoint.

Referenced by Convert().

Here is the call graph for this function:

Here is the caller graph for this function:



6.13.5 Member Data Documentation

6.13.5.1 ColorSpace Vision::Conversion::OriginalColorSpace

The original colorspace

Definition at line 24 of file Conversion.h.

Referenced by Conversion(), Convert(), and operator=().

6.13.5.2 ColorSpace Vision::Conversion::ProcessedColorSpace

The destination colorspace

Definition at line 25 of file Conversion.h.

Referenced by Conversion(), Convert(), and operator=().

6.13.5.3 float Vision::Conversion::whitePoint[3] [private]

Initial value:

```
= {
      0.9504, 1.0000, 1.0889}
```

Natural whitepoint in XYZ colorspace D65 according to Matlab

Definition at line 46 of file Conversion.h.

Referenced by XYZ2Lab().

6.13.5.4 float Vision::Conversion::XYZmat[3][3] [private]

Initial value:

```
= {{0.412453, 0.357580, 0.180423},
                        {0.212671, 0.715160, 0.072169},
                        {0.019334, 0.119194, 0.950227}}
```

< Conversion matrix used in the conversion between RGB and CIE XYZ

Definition at line 42 of file Conversion.h.

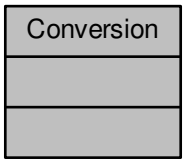
Referenced by RGB2XYZ().

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

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.cpp

6.14 Conversion Class Reference

Collaboration diagram for Conversion:



6.14.1 Detailed Description

class which converts a cv::Mat image from one colorspace to the next colorspace

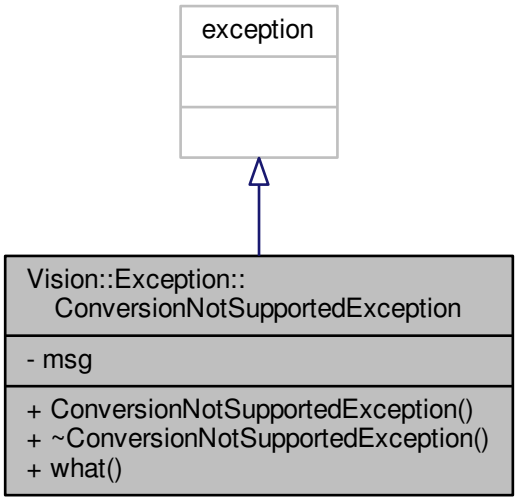
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Conversion.cpp

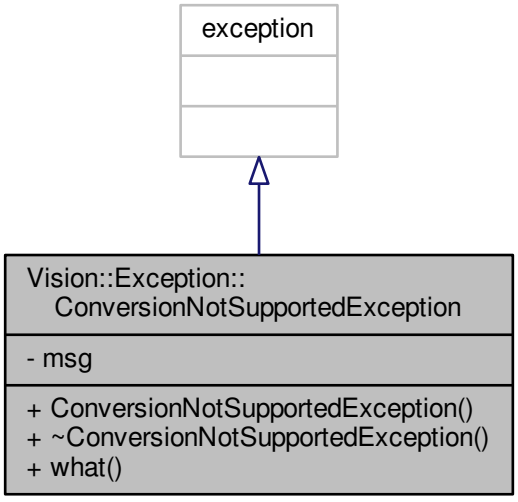
6.15 Vision::Exception::ConversionNotSupportedException Class Reference

```
#include <ConversionNotSupportedException.h>
```

Inheritance diagram for Vision::Exception::ConversionNotSupportedException:



Collaboration diagram for Vision::Exception::ConversionNotSupportedException:



Public Member Functions

- ConversionNotSupportedException (string m="Requested conversion is not supported!")
- ~ConversionNotSupportedException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.15.1 Detailed Description

Definition at line 20 of file ConversionNotSupportedException.h.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 Vision::Exception::ConversionNotSupportedException::ConversionNotSupportedException (string *m* = "Requested conversion is not supported!") [inline]

Definition at line 22 of file ConversionNotSupportedException.h.

6.15.2.2 Vision::Exception::ConversionNotSupportedException::~ConversionNotSupportedException ( ) [inline]

Definition at line 25 of file ConversionNotSupportedException.h.

6.15.3 Member Function Documentation

6.15.3.1 const char* Vision::Exception::ConversionNotSupportedException::what ( ) const [inline]

Definition at line 26 of file ConversionNotSupportedException.h.

6.15.4 Member Data Documentation

6.15.4.1 string Vision::Exception::ConversionNotSupportedException::msg [private]

Definition at line 26 of file ConversionNotSupportedException.h.

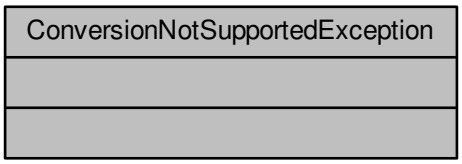
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ConversionNotSupportedException.h

6.16 ConversionNotSupportedException Class Reference

`#include <ConversionNotSupportedException.h>`

Collaboration diagram for ConversionNotSupportedException:



6.16.1 Detailed Description

Exception class which is thrown when an illegal conversion is requested.

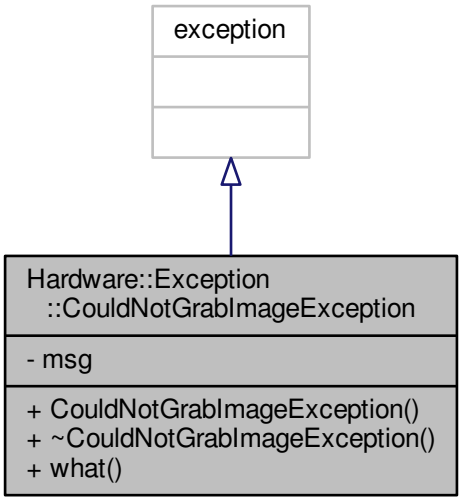
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ConversionNotSupportedException.h

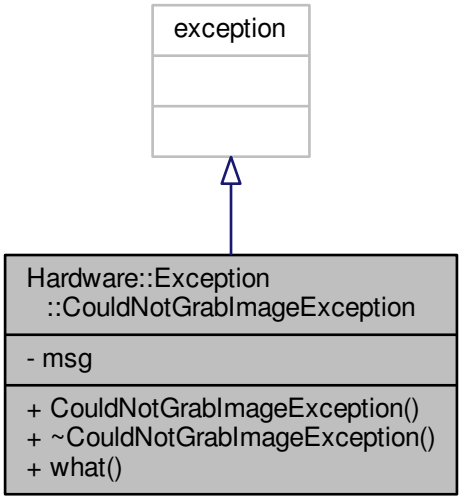
6.17 Hardware::Exception::CouldNotGrabImageException Class Reference

`#include <CouldNotGrabImageException.h>`

Inheritance diagram for Hardware::Exception::CouldNotGrabImageException:



Collaboration diagram for Hardware::Exception::CouldNotGrabImageException:



Public Member Functions

- CouldNotGrabImageException (string m="Unable to grab the next image!")
- ~CouldNotGrabImageException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.17.1 Detailed Description

Definition at line 16 of file CouldNotGrabImageException.h.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 Hardware::Exception::CouldNotGrabImageException::CouldNotGrabImageException (string *m* = "Unable to grab the next image!") [inline]

Definition at line 18 of file CouldNotGrabImageException.h.

6.17.2.2 Hardware::Exception::CouldNotGrabImageException::~CouldNotGrabImageException ( ) [inline]

Definition at line 20 of file CouldNotGrabImageException.h.

6.17.3 Member Function Documentation

6.17.3.1 const char* Hardware::Exception::CouldNotGrabImageException::what ( ) const [inline]

Definition at line 21 of file CouldNotGrabImageException.h.

6.17.4 Member Data Documentation

6.17.4.1 string Hardware::Exception::CouldNotGrabImageException::msg [private]

Definition at line 21 of file CouldNotGrabImageException.h.

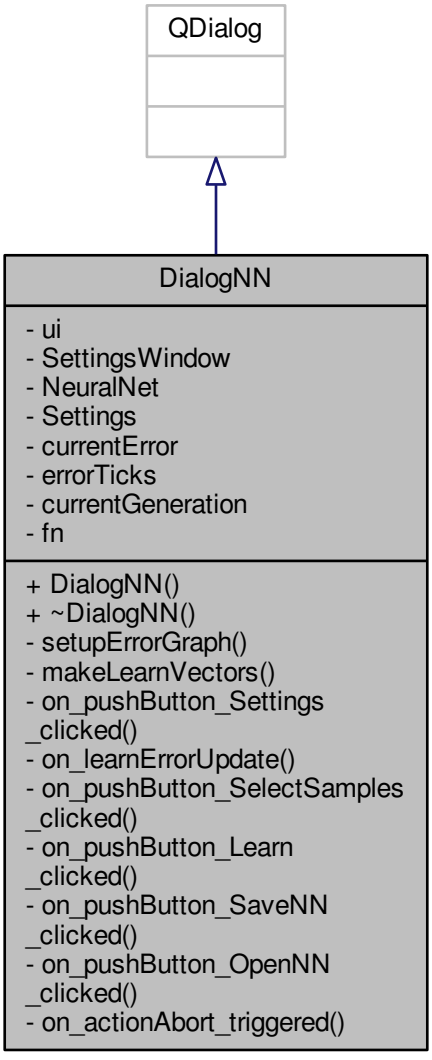
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/CouldNotGrabImageException.h

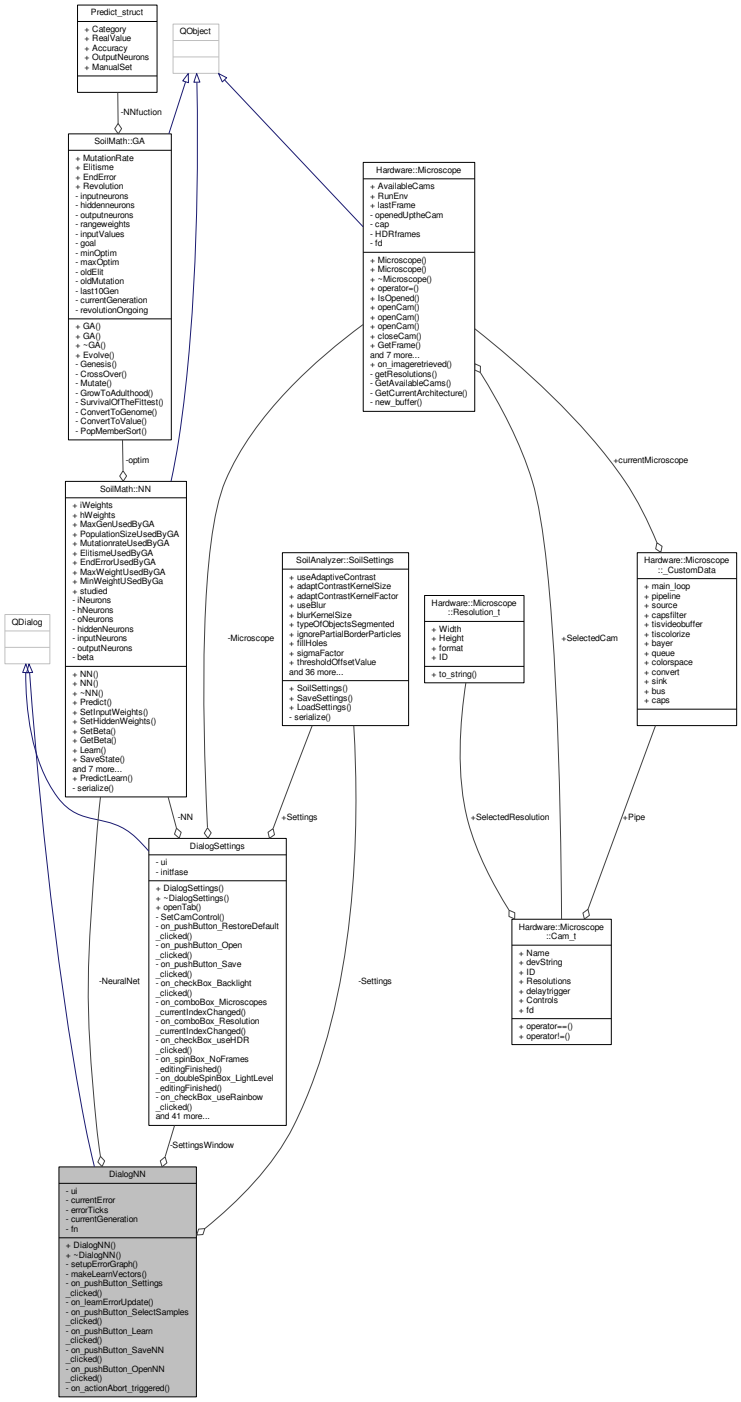
6.18 DialogNN Class Reference

```
#include <dialognn.h>
```

Inheritance diagram for DialogNN:



Collaboration diagram for DialogNN:



Public Member Functions

- DialogNN (QWidget *parent=0, SoilMath::NN *neuralnet=nullptr, SoilAnalyzer::SoilSettings *settings=nullptr, DialogSettings *setting↩ Window=nullptr)
- ~DialogNN ()

Private Slots

- void on_pushButton_Settings_clicked ()
- void on_learnErrorUpdate (double newError)
- void on_pushButton_SelectSamples_clicked ()
- void on_pushButton_Learn_clicked ()
- void on_pushButton_SaveNN_clicked ()
- void on_pushButton_OpenNN_clicked ()
- void on_actionAbort_triggered ()

Private Member Functions

- void setupErrorGraph ()
- void makeLearnVectors (InputLearnVector_t &input, OutputLearnVector_t &output)

Private Attributes

- Ui::DialogNN * ui
- DialogSettings * SettingsWindow = nullptr
- SoilMath::NN * NeuralNet = nullptr
- SoilAnalyzer::SoilSettings * Settings = nullptr
- QVector< double > currentError
- QVector< double > errorTicks
- double currentGeneration = 0
- QStringList fn

6.18.1 Detailed Description

Definition at line 15 of file dialognn.h.

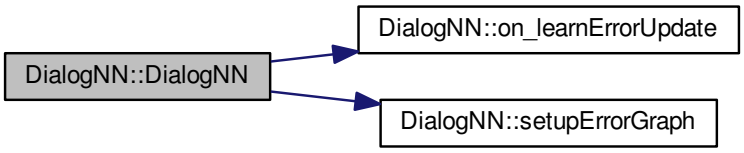
6.18.2 Constructor & Destructor Documentation

6.18.2.1 DialogNN::DialogNN (QWidget * *parent* = `0`, SoilMath::NN * *neuralnet* = `nullptr`, SoilAnalyzer::SoilSettings * *settings* = `nullptr`, DialogSettings * *settingWindow* = `nullptr`) [explicit]

Definition at line 4 of file dialognn.cpp.

References NeuralNet, on_learnErrorUpdate(), Settings, SettingsWindow, setupErrorGraph(), and ui.

Here is the call graph for this function:



6.18.2.2 DialogNN::~DialogNN ( )

Definition at line 42 of file dialognn.cpp.

References ui.

6.18.3 Member Function Documentation

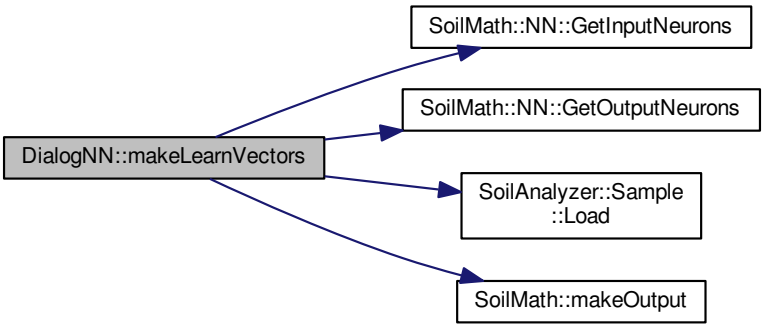
6.18.3.1 void DialogNN::makeLearnVectors (InputLearnVector_t & *input*, OutputLearnVector_t & *output*) [private]

Definition at line 97 of file dialognn.cpp.

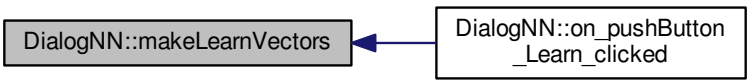
References fn, SoilMath::NN::GetInputNeurons(), SoilMath::NN::GetOutputNeurons(), SoilAnalyzer::Sample::Load(), SoilMath::makeOutput(), NeuralNet, Predict_struct::OutputNeurons, and SoilAnalyzer::Sample::ParticlePopulation.

Referenced by on_pushButton_Learn_clicked().

Here is the call graph for this function:



Here is the caller graph for this function:



6.18.3.2 void DialogNN::on_actionAbort_triggered ( ) [private], [slot]

Definition at line 147 of file dialognn.cpp.

References SoilMath::NN::EndErrorUsedByGA, NeuralNet, and ui.

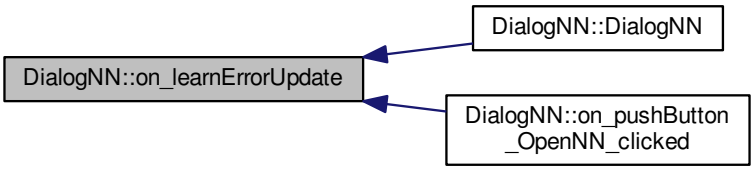
6.18.3.3 void DialogNN::on_learnErrorUpdate (double *newError*) [private], [slot]

Definition at line 50 of file dialognn.cpp.

References currentGeneration, and ui.

Referenced by DialogNN(), and on_pushButton_OpenNN_clicked().

Here is the caller graph for this function:

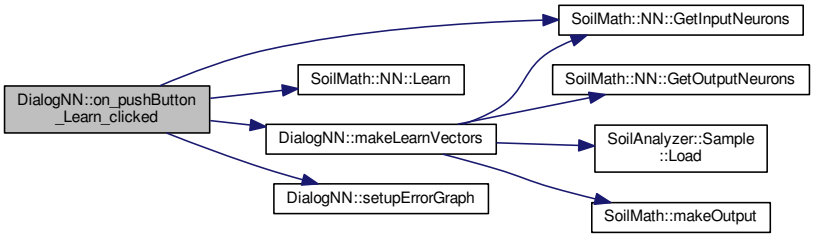


6.18.3.4 void DialogNN::on_pushButton_Learn_clicked ( ) [private], [slot]

Definition at line 86 of file dialognn.cpp.

References fn, SoilMath::NN::GetInputNeurons(), SoilMath::NN::Learn(), makeLearnVectors(), NeuralNet, and setupErrorGraph().

Here is the call graph for this function:

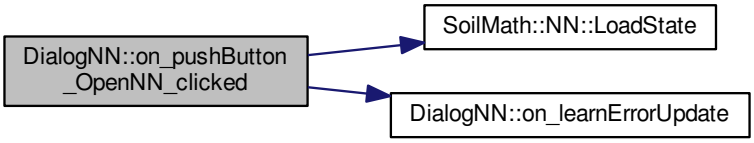


6.18.3.5 void DialogNN::on_pushButton_OpenNN_clicked ( ) [private], [slot]

Definition at line 130 of file dialognn.cpp.

References fn, SoilMath::NN::LoadState(), NeuralNet, on_learnErrorUpdate(), SoilAnalyzer::SoilSettings::SampleFolder, and Settings.

Here is the call graph for this function:

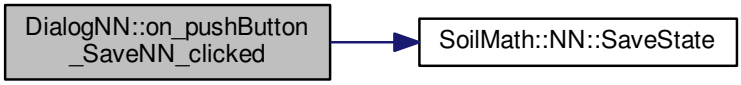


6.18.3.6 void DialogNN::on_pushButton_SaveNN_clicked ( ) [private], [slot]

Definition at line 118 of file dialognn.cpp.

References fn, NeuralNet, SoilAnalyzer::SoilSettings::NNFolder, SoilMath::NN::SaveState(), and Settings.

Here is the call graph for this function:



6.18.3.7 void DialogNN::on_pushButton_SelectSamples_clicked ( ) [private], [slot]

Definition at line 75 of file dialognn.cpp.

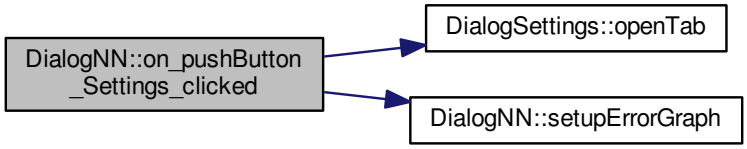
References fn, SoilAnalyzer::SoilSettings::SampleFolder, and Settings.

6.18.3.8 void DialogNN::on_pushButton_Settings_clicked ( ) [private], [slot]

Definition at line 44 of file dialognn.cpp.

References DialogSettings::openTab(), SettingsWindow, and setupErrorGraph().

Here is the call graph for this function:



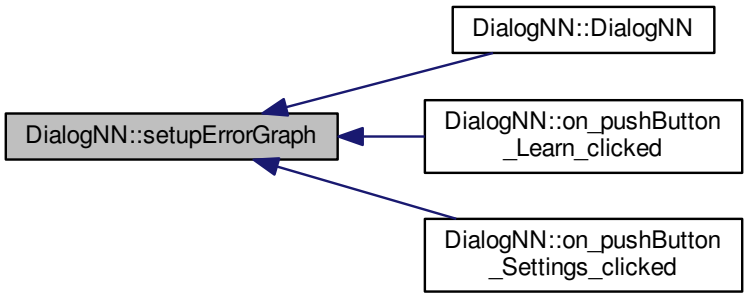
6.18.3.9 void DialogNN::setupErrorGraph ( ) [private]

Definition at line 58 of file dialognn.cpp.

References SoilMath::NN::EndErrorUsedByGA, errorTicks, SoilMath::NN::MaxGenUsedByGA, NeuralNet, and ui.

Referenced by DialogNN(), on_pushButton_Learn_clicked(), and on_pushButton_Settings_clicked().

Here is the caller graph for this function:



6.18.4 Member Data Documentation

6.18.4.1 QVector<double> DialogNN::currentError [private]

Definition at line 48 of file dialognn.h.

6.18.4.2 double DialogNN::currentGeneration = 0 [private]

Definition at line 50 of file dialognn.h.

Referenced by on_learnErrorUpdate().

6.18.4.3 QVector<double> DialogNN::errorTicks [private]

Definition at line 49 of file dialognn.h.

Referenced by setupErrorGraph().

6.18.4.4 QStringList DialogNN::fn [private]

Definition at line 51 of file dialognn.h.

Referenced by makeLearnVectors(), on_pushButton_Learn_clicked(), on_pushButton_OpenNN_clicked(), on_pushButton_SaveNN_clicked(), and on_pushButton_SelectSamples_clicked().

6.18.4.5 SoilMath::NN* DialogNN::NeuralNet = nullptr [private]

Definition at line 42 of file dialognn.h.

Referenced by DialogNN(), makeLearnVectors(), on_actionAbort_triggered(), on_pushButton_Learn_clicked(), on_pushButton_OpenNN_↩ clicked(), on_pushButton_SaveNN_clicked(), and setupErrorGraph().

6.18.4.6 SoilAnalyzer::SoilSettings* DialogNN::Settings = nullptr [private]

Definition at line 43 of file dialognn.h.

Referenced by DialogNN(), on_pushButton_OpenNN_clicked(), on_pushButton_SaveNN_clicked(), and on_pushButton_SelectSamples_↩ clicked().

6.18.4.7 DialogSettings* DialogNN::SettingsWindow = nullptr [private]

Definition at line 41 of file dialognn.h.

Referenced by DialogNN(), and on_pushButton_Settings_clicked().

6.18.4.8 Ui::DialogNN* DialogNN::ui [private]

Definition at line 40 of file dialognn.h.

Referenced by DialogNN(), on_actionAbort_triggered(), on_learnErrorUpdate(), setupErrorGraph(), and ~DialogNN().

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

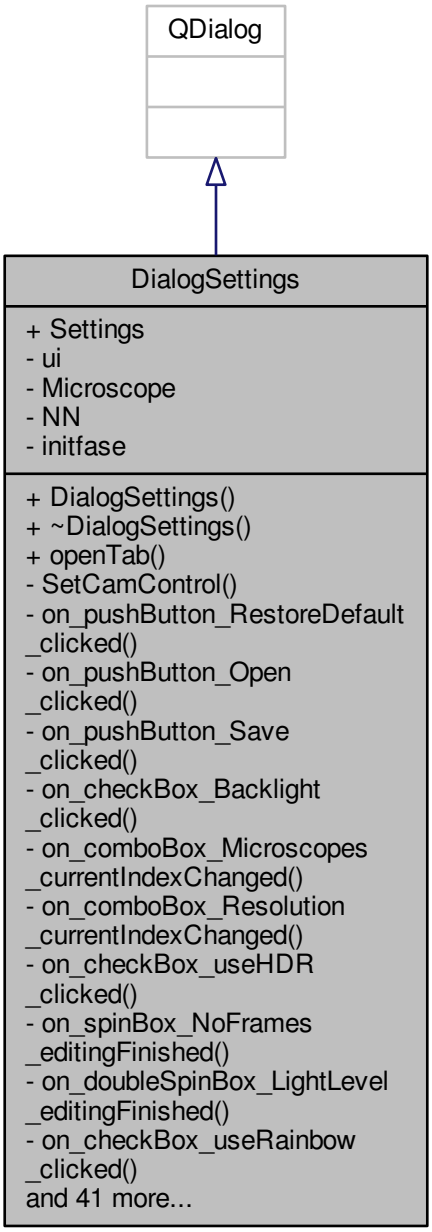
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/dialognn.h

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/dialognn.cpp

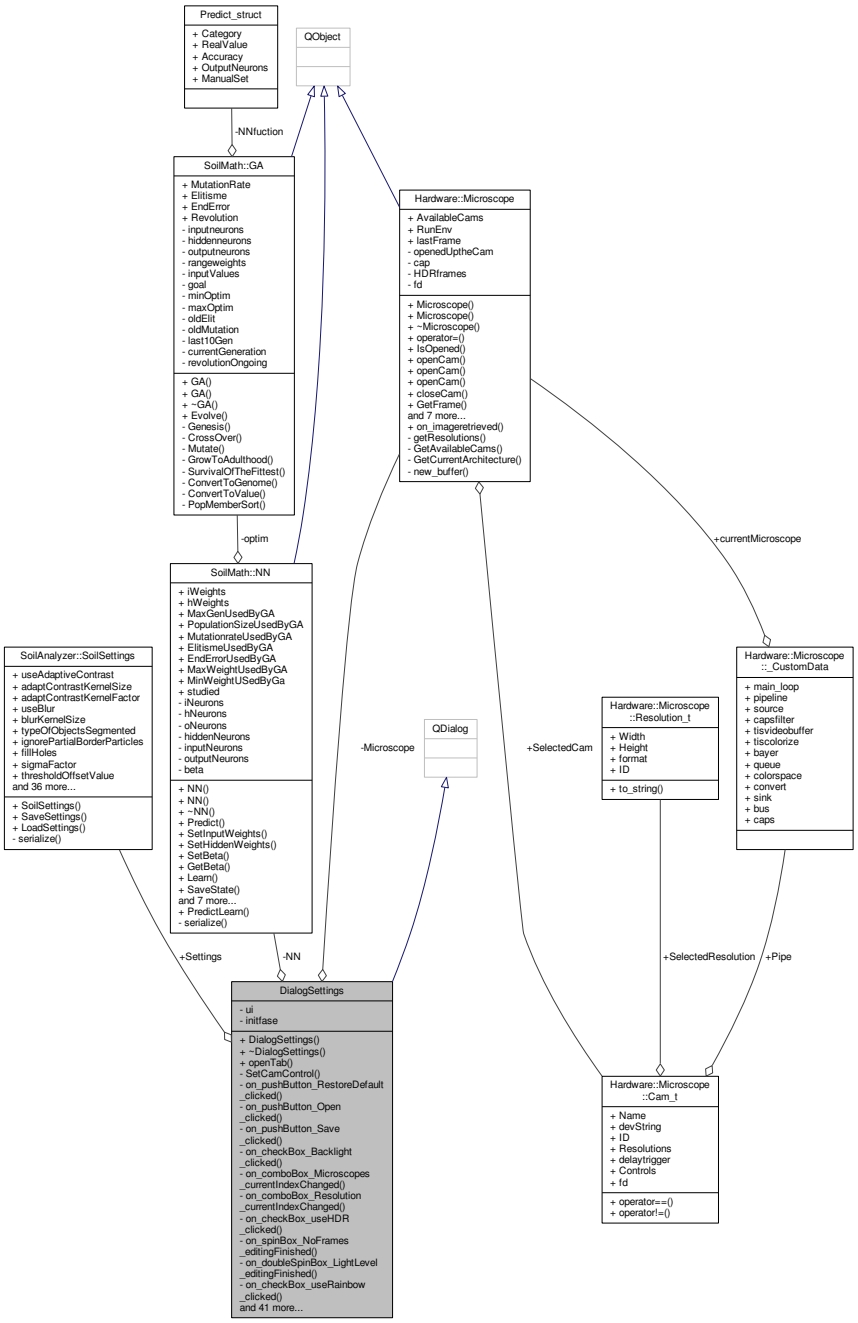
6.19 DialogSettings Class Reference

```
#include <dialogsettings.h>
```

Inheritance diagram for DialogSettings:



Collaboration diagram for DialogSettings:



Public Member Functions

- DialogSettings (QWidget *parent=0, SoilAnalyzer::SoilSettings *settings=nullptr, Hardware::Microscope *microscope=nullptr, SoilMath::NN *nn=nullptr, bool openNN=false)
- ~DialogSettings ()
- void openTab (int newValue)

Public Attributes

- SoilAnalyzer::SoilSettings * Settings = nullptr

Private Slots

- void on_pushButton_RestoreDefault_clicked ()
- void on_pushButton_Open_clicked ()
- void on_pushButton_Save_clicked ()
- void on_checkBox_Backlight_clicked (bool checked)
- void on_comboBox_Microscopes_currentIndexChanged (const QString &arg1)

- void on_comboBox_Resolution_currentIndexChanged (int index)
- void on_checkBox_useHDR_clicked (bool checked)
- void on_spinBox_NoFrames_editingFinished ()
- void on_doubleSpinBox_LightLevel_editingFinished ()
- void on_checkBox_useRainbow_clicked (bool checked)
- void on_checkBox_InvertEncoder_clicked (bool checked)
- void on_checkBox_useCUDA_clicked (bool checked)
- void on_horizontalSlider_BrightFront_valueChanged (int value)
- void on_horizontalSlider_ContrastFront_valueChanged (int value)
- void on_horizontalSlider_SaturationFront_valueChanged (int value)
- void on_horizontalSlider_HueFront_valueChanged (int value)
- void on_horizontalSlider_SharpnessFront_valueChanged (int value)
- void on_horizontalSlider_BrightProj_valueChanged (int value)
- void on_horizontalSlider_ContrastProj_valueChanged (int value)
- void on_horizontalSlider_SaturationProj_valueChanged (int value)
- void on_horizontalSlider_HueProj_valueChanged (int value)
- void on_horizontalSlider_SharpnessProj_valueChanged (int value)
- void on_cb_use_adaptContrast_3_clicked (bool checked)
- void on_cb_useBlur_3_clicked (bool checked)
- void on_rb_useDark_3_toggled (bool checked)
- void on_cb_ignoreBorder_3_clicked (bool checked)
- void on_cb_fillHoles_3_clicked (bool checked)
- void on_sb_sigmaFactor_3_editingFinished ()
- void on_rb_useOpen_3_clicked (bool checked)
- void on_rb_useClose_3_clicked (bool checked)
- void on_rb_useErode_3_clicked (bool checked)
- void on_rb_useDilate_3_clicked (bool checked)
- void on_sb_morphMask_3_editingFinished ()
- void on_spinBox_MaxGen_editingFinished ()
- void on_spinBox_PopSize_editingFinished ()
- void on_doubleSpinBox_MutationRate_editingFinished ()
- void on_spinBox_Elitisme_editingFinished ()
- void on_doubleSpinBox_endError_editingFinished ()
- void on_doubleSpinBox_maxWeight_editingFinished ()
- void on_doubleSpinBox_MinWeight_editingFinished ()
- void on_doubleSpinBox_Beta_editingFinished ()
- void on_spinBox_InputNeurons_editingFinished ()
- void on_spinBox_HiddenNeurons_editingFinished ()
- void on_spinBox_OutputNeurons_editingFinished ()
- void on_pushButton_selectSampleFolder_clicked ()
- void on_pushButton_SelectSettingFolder_clicked ()
- void on_pushButton_SelectNNFolder_clicked ()
- void on_pushButton_SelectNN_clicked ()
- void on_spinBox_NoShots_editingFinished ()
- void on_checkBox_PredictShape_clicked (bool checked)
- void on_checkBox_revolt_clicked (bool checked)

Private Member Functions

- void SetCamControl (Hardware::Microscope::Cam_t *selectedCam, QSlider *Brightness, QSlider *Contrast, QSlider *Saturation, QSlider *Hue, QSlider *Sharpness)

Private Attributes

- Ui::DialogSettings * ui
- Hardware::Microscope * Microscope
- SoilMath::NN * NN
- bool initfase = true

6.19.1 Detailed Description

Definition at line 16 of file dialogsettings.h.

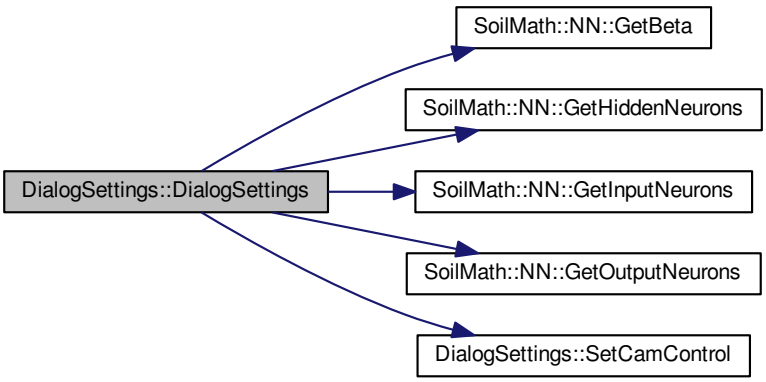
6.19.2 Constructor & Destructor Documentation

6.19.2.1 DialogSettings::DialogSettings (QWidget * *parent* = 0, SoilAnalyzer::SoilSettings * *settings* = nullptr, Hardware::Microscope * *microscope* = nullptr, SoilMath::NN * *nn* = nullptr, bool *openNN* = false) [explicit]

Definition at line 5 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::adaptContrastKernelFactor, SoilAnalyzer::SoilSettings::adaptContrastKernelSize, SoilAnalyzer::SoilSettings::blurKernelSize, Vision::Segment::Bright, SoilAnalyzer::SoilSettings::Brightness_front, SoilAnalyzer::SoilSettings::Brightness_proj, Vision::MorphologicalFilter::CLOSE, SoilAnalyzer::SoilSettings::Contrast_front, SoilAnalyzer::SoilSettings::Contrast_proj, Vision::Segment::Dark, Vision::MorphologicalFilter::DILATE, SoilMath::NN::ElitismeUsedByGA, SoilAnalyzer::SoilSettings::enableRainbow, SoilAnalyzer::SoilSettings::encInv, SoilMath::NN::EndErrorUsedByGA, Vision::MorphologicalFilter::ERODE, SoilAnalyzer::SoilSettings::fillHoles, SoilAnalyzer::SoilSettings::filterMaskSize, SoilMath::NN::GetBeta(), SoilMath::NN::GetHiddenNeurons(), SoilMath::NN::GetInputNeurons(), SoilMath::NN::GetOutputNeurons(), SoilAnalyzer::SoilSettings::HDRframes, SoilAnalyzer::SoilSettings::Hue_front, SoilAnalyzer::SoilSettings::Hue_proj, SoilAnalyzer::SoilSettings::ignorePartialBorderParticles, initfase, SoilMath::NN::MaxGenUsedByGA, SoilMath::NN::MaxWeightUsedByGA, SoilMath::NN::MinWeightUSedByGa, SoilAnalyzer::SoilSettings::morphFilterType, SoilMath::NN::MutationrateUsedByGA, SoilAnalyzer::SoilSettings::NNFolder, SoilAnalyzer::SoilSettings::NNlocation, Vision::MorphologicalFilter::OPEN, SoilMath::NN::PopulationSizeUsedByGA, SoilAnalyzer::SoilSettings::PredictTheShape, SoilAnalyzer::SoilSettings::Revolution, SoilAnalyzer::SoilSettings::SampleFolder, SoilAnalyzer::SoilSettings::Saturation_front, SoilAnalyzer::SoilSettings::Saturation_proj, SetCamControl(), Settings, SoilAnalyzer::SoilSettings::SettingsFolder, SoilAnalyzer::SoilSettings::Sharpness_front, SoilAnalyzer::SoilSettings::Sharpness_proj, SoilAnalyzer::SoilSettings::sigmaFactor, SoilAnalyzer::SoilSettings::StandardNumberOfShots, SoilAnalyzer::SoilSettings::StandardPrinter, SoilAnalyzer::SoilSettings::StandardSentTo, SoilAnalyzer::SoilSettings::typeOfObjectsSegmented, ui, SoilAnalyzer::SoilSettings::useAdaptiveContrast, SoilAnalyzer::SoilSettings::useBacklightProjection, SoilAnalyzer::SoilSettings::useBlur, SoilAnalyzer::SoilSettings::useCUDA, SoilAnalyzer::SoilSettings::useHDR, and Hardware::Microscope::X64.

Here is the call graph for this function:



6.19.2.2 DialogSettings::~DialogSettings ( )

Definition at line 188 of file dialogsettings.cpp.

References ui.

6.19.3 Member Function Documentation

6.19.3.1 void DialogSettings::on_cb_fillHoles_3_clicked (bool *checked*) [private], [slot]

Definition at line 393 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::fillHoles, and Settings.

6.19.3.2 void DialogSettings::on_cb_ignoreBorder_3_clicked (bool *checked*) [private], [slot]

Definition at line 389 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::ignorePartialBorderParticles, and Settings.

6.19.3.3 void DialogSettings::on_cb_use_adaptContrast_3_clicked (bool *checked*) [private], [slot]

Definition at line 370 of file dialogsettings.cpp.

References Settings, ui, and SoilAnalyzer::SoilSettings::useAdaptiveContrast.

6.19.3.4 void DialogSettings::on_cb_useBlur_3_clicked (bool *checked*) [private], [slot]

Definition at line 376 of file dialogsettings.cpp.

References Settings, ui, and SoilAnalyzer::SoilSettings::useBlur.

6.19.3.5 void DialogSettings::on_checkBox_Backlight_clicked (bool *checked*) [private], [slot]

Definition at line 222 of file dialogsettings.cpp.

References Settings, ui, and SoilAnalyzer::SoilSettings::useBacklightProjection.

6.19.3.6 void DialogSettings::on_checkBox_InvertEncoder_clicked (bool *checked*) [private], [slot]

Definition at line 299 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::encInv, and Settings.

6.19.3.7 void DialogSettings::on_checkBox_PredictShape_clicked (bool *checked*) [private], [slot]

Definition at line 515 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::PredictTheShape, and Settings.

6.19.3.8 void DialogSettings::on_checkBox_revolt_clicked (bool *checked*) [private], [slot]

Definition at line 519 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::Revolution, and Settings.

6.19.3.9 void DialogSettings::on_checkBox_useCUDA_clicked (bool *checked*) [private], [slot]

Definition at line 303 of file dialogsettings.cpp.

References Settings, and SoilAnalyzer::SoilSettings::useCUDA.

6.19.3.10 void DialogSettings::on_checkBox_useHDR_clicked (bool *checked*) [private], [slot]

Definition at line 256 of file dialogsettings.cpp.

References Settings, ui, and SoilAnalyzer::SoilSettings::useHDR.

6.19.3.11 void DialogSettings::on_checkBox_useRainbow_clicked (bool *checked*) [private], [slot]

Definition at line 295 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::enableRainbow, and Settings.

6.19.3.12 void DialogSettings::on_comboBox_Microscopes_currentIndexChanged (const QString & *arg1*) [private], [slot]

Definition at line 227 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::defaultWebcam, initfase, Settings, and ui.

6.19.3.13 void DialogSettings::on_comboBox_Resolution_currentIndexChanged (int *index*) [private], [slot]

Definition at line 247 of file dialogsettings.cpp.

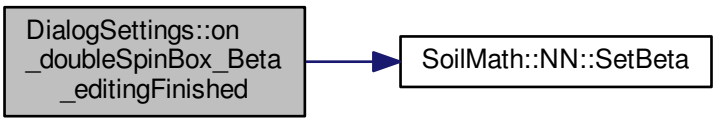
References initfase, SoilAnalyzer::SoilSettings::selectedResolution, and Settings.

6.19.3.14 void DialogSettings::on_doubleSpinBox_Beta_editingFinished ( ) [private], [slot]

Definition at line 449 of file dialogsettings.cpp.

References SoilMath::NN::SetBeta(), and ui.

Here is the call graph for this function:



6.19.3.15 void DialogSettings::on_doubleSpinBox_endError_editingFinished ( ) [private], [slot]

Definition at line 437 of file dialogsettings.cpp.

References SoilMath::NN::EndErrorUsedByGA, and ui.

6.19.3.16 void DialogSettings::on_doubleSpinBox_LightLevel_editingFinished ( ) [private], [slot]

Definition at line 290 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::lightLevel, Settings, and ui.

6.19.3.17 void DialogSettings::on_doubleSpinBox_maxWeight_editingFinished ( ) [private], [slot]

Definition at line 441 of file dialogsettings.cpp.

References SoilMath::NN::MaxWeightUsedByGA, and ui.

6.19.3.18 void DialogSettings::on_doubleSpinBox_MinWeight_editingFinished ( ) [private], [slot]

Definition at line 445 of file dialogsettings.cpp.

References SoilMath::NN::MinWeightUSedByGa, and ui.

6.19.3.19 void DialogSettings::on_doubleSpinBox_MutationRate_editingFinished ( ) [private], [slot]

Definition at line 429 of file dialogsettings.cpp.

References SoilMath::NN::MutationrateUsedByGA, and ui.

6.19.3.20 void DialogSettings::on_horizontalSlider_BrightFront_valueChanged (int *value*) [private], [slot]

Definition at line 307 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::Brightness_front, initfase, and Settings.

6.19.3.21 void DialogSettings::on_horizontalSlider_BrightProj_valueChanged (int *value*) [private], [slot]

Definition at line 339 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::Brightness_proj, initfase, and Settings.

6.19.3.22 void DialogSettings::on_horizontalSlider_ContrastFront_valueChanged (int *value*) [private], [slot]

Definition at line 313 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::Contrast_front, initfase, and Settings.

6.19.3.23 void DialogSettings::on_horizontalSlider_ContrastProj_valueChanged (int *value*) [private], [slot]

Definition at line 345 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::Contrast_proj, initfase, and Settings.

6.19.3.24 void DialogSettings::on_horizontalSlider_HueFront_valueChanged (int *value*) [private], [slot]

Definition at line 326 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::Hue_front, initfase, and Settings.

6.19.3.25 void DialogSettings::on_horizontalSlider_HueProj_valueChanged (int *value*) [private], [slot]

Definition at line 358 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::Hue_proj, initfase, and Settings.

6.19.3.26 void DialogSettings::on_horizontalSlider_SaturationFront_valueChanged (int *value*) [private], [slot]

Definition at line 319 of file dialogsettings.cpp.

References initfase, SoilAnalyzer::SoilSettings::Saturation_front, and Settings.

6.19.3.27 void DialogSettings::on_horizontalSlider_SaturationProj_valueChanged (int *value*) [private], [slot]

Definition at line 351 of file dialogsettings.cpp.

References initfase, SoilAnalyzer::SoilSettings::Saturation_proj, and Settings.

6.19.3.28 void DialogSettings::on_horizontalSlider_SharpnessFront_valueChanged (int *value*) [private], [slot]

Definition at line 332 of file dialogsettings.cpp.

References initfase, Settings, and SoilAnalyzer::SoilSettings::Sharpness_front.

6.19.3.29 void DialogSettings::on_horizontalSlider_SharpnessProj_valueChanged (int *value*) [private], [slot]

Definition at line 364 of file dialogsettings.cpp.

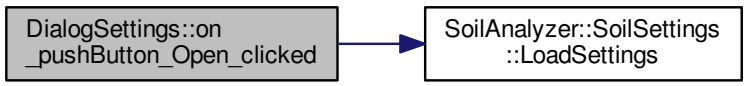
References initfase, Settings, and SoilAnalyzer::SoilSettings::Sharpness_proj.

6.19.3.30 void DialogSettings::on_pushButton_Open_clicked ( ) [private], [slot]

Definition at line 200 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::LoadSettings(), and Settings.

Here is the call graph for this function:

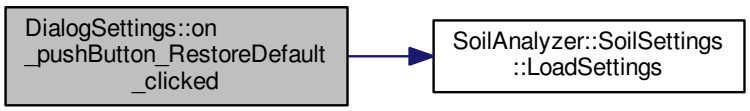


6.19.3.31 void DialogSettings::on_pushButton_RestoreDefault_clicked ( ) [private], [slot]

Definition at line 196 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::LoadSettings(), and Settings.

Here is the call graph for this function:

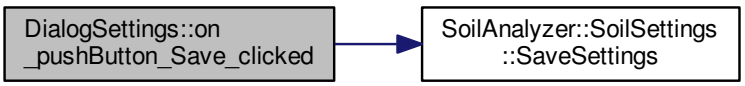


6.19.3.32 void DialogSettings::on_pushButton_Save_clicked ( ) [private], [slot]

Definition at line 211 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::SaveSettings(), and Settings.

Here is the call graph for this function:



6.19.3.33 void DialogSettings::on_pushButton_SelectNN_clicked ( ) [private], [slot]

Definition at line 498 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::NNlocation, Settings, and ui.

6.19.3.34 void DialogSettings::on_pushButton_SelectNNFolder_clicked ( ) [private], [slot]

Definition at line 487 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::NNFolder, Settings, and ui.

6.19.3.35 void DialogSettings::on_pushButton_selectSampleFolder_clicked ( ) [private], [slot]

Definition at line 465 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::SampleFolder, Settings, and ui.

6.19.3.36 void DialogSettings::on_pushButton_SelectSettingFolder_clicked ( ) [private], [slot]

Definition at line 476 of file dialogsettings.cpp.

References Settings, SoilAnalyzer::SoilSettings::SettingsFolder, and ui.

6.19.3.37 void DialogSettings::on_rb_useClose_3_clicked (bool *checked*) [private], [slot]

Definition at line 405 of file dialogsettings.cpp.

References Vision::MorphologicalFilter::CLOSE, SoilAnalyzer::SoilSettings::morphFilterType, and Settings.

6.19.3.38 void DialogSettings::on_rb_useDark_3_toggled (bool *checked*) [private], [slot]

Definition at line 381 of file dialogsettings.cpp.

References Vision::Segment::Bright, Vision::Segment::Dark, Settings, and SoilAnalyzer::SoilSettings::typeOfObjectsSegmented.

6.19.3.39 void DialogSettings::on_rb_useDilate_3_clicked (bool *checked*) [private], [slot]

Definition at line 413 of file dialogsettings.cpp.

References Vision::MorphologicalFilter::DILATE, SoilAnalyzer::SoilSettings::morphFilterType, and Settings.

6.19.3.40 void DialogSettings::on_rb_useErode_3_clicked (bool *checked*) [private], [slot]

Definition at line 409 of file dialogsettings.cpp.

References Vision::MorphologicalFilter::ERODE, SoilAnalyzer::SoilSettings::morphFilterType, and Settings.

6.19.3.41 void DialogSettings::on_rb_useOpen_3_clicked (bool *checked*) [private], [slot]

Definition at line 401 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::morphFilterType, Vision::MorphologicalFilter::OPEN, and Settings.

6.19.3.42 void DialogSettings::on_sb_morphMask_3_editingFinished ( ) [private], [slot]

Definition at line 417 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::filterMaskSize, Settings, and ui.

6.19.3.43 void DialogSettings::on_sb_sigmaFactor_3_editingFinished ( ) [private], [slot]

Definition at line 397 of file dialogsettings.cpp.

References Settings, SoilAnalyzer::SoilSettings::sigmaFactor, and ui.

6.19.3.44 void DialogSettings::on_spinBox_Elitisme_editingFinished ( ) [private], [slot]

Definition at line 433 of file dialogsettings.cpp.

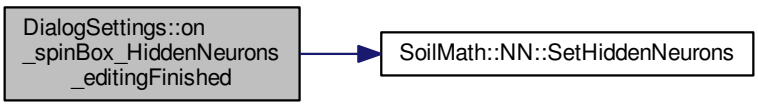
References SoilMath::NN::ElitismeUsedByGA, and ui.

6.19.3.45 void DialogSettings::on_spinBox_HiddenNeurons_editingFinished ( ) [private], [slot]

Definition at line 457 of file dialogsettings.cpp.

References SoilMath::NN::SetHiddenNeurons(), and ui.

Here is the call graph for this function:

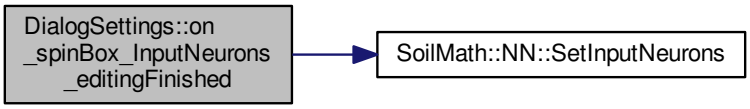


6.19.3.46 void DialogSettings::on_spinBox_InputNeurons_editingFinished ( ) [private], [slot]

Definition at line 453 of file dialogsettings.cpp.

References SoilMath::NN::SetInputNeurons(), and ui.

Here is the call graph for this function:



6.19.3.47 void DialogSettings::on_spinBox_MaxGen_editingFinished ( ) [private], [slot]

Definition at line 421 of file dialogsettings.cpp.

References SoilMath::NN::MaxGenUsedByGA, and ui.

6.19.3.48 void DialogSettings::on_spinBox_NoFrames_editingFinished ( ) [private], [slot]

Definition at line 286 of file dialogsettings.cpp.

References SoilAnalyzer::SoilSettings::HDRframes, Settings, and ui.

6.19.3.49 void DialogSettings::on_spinBox_NoShots_editingFinished ( ) [private], [slot]

Definition at line 511 of file dialogsettings.cpp.

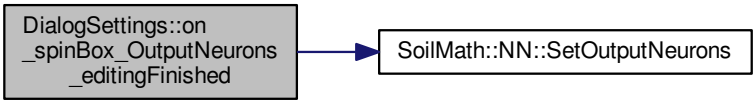
References Settings, SoilAnalyzer::SoilSettings::StandardNumberOfShots, and ui.

6.19.3.50 void DialogSettings::on_spinBox_OutputNeurons_editingFinished ( ) [private], [slot]

Definition at line 461 of file dialogsettings.cpp.

References SoilMath::NN::SetOutputNeurons(), and ui.

Here is the call graph for this function:



6.19.3.51 void DialogSettings::on_spinBox_PopSize_editingFinished ( ) [private],[slot]

Definition at line 425 of file dialogsettings.cpp.

References SoilMath::NN::PopulationSizeUsedByGA, and ui.

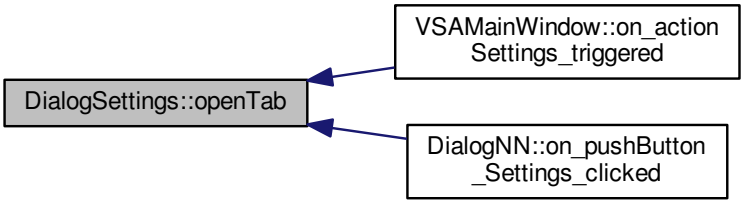
6.19.3.52 void DialogSettings::openTab (int *newValue*)

Definition at line 190 of file dialogsettings.cpp.

References ui.

Referenced by VSAMainWindow::on_actionSettings_triggered(), and DialogNN::on_pushButton_Settings_clicked().

Here is the caller graph for this function:



6.19.3.53 void DialogSettings::SetCamControl (Hardware::Microscope::Cam_t * *selectedCam,* QSlider * *Brightness,* QSlider * *Contrast,* QSlider * *Saturation,* QSlider * *Hue,* QSlider * *Sharpness*) [private]

Definition at line 262 of file dialogsettings.cpp.

References Hardware::Microscope::Cam_t::Controls.

Referenced by DialogSettings().

Here is the caller graph for this function:

6.19.4 Member Data Documentation

6.19.4.1 bool DialogSettings::initfase = true [private]

Definition at line 136 of file dialogsettings.h.

Referenced by DialogSettings(), on_comboBox_Microscopes_currentIndexChanged(), on_comboBox_Resolution_currentIndexChanged(), on_horizontalSlider_BrightFront_valueChanged(), on_horizontalSlider_BrightProj_valueChanged(), on_horizontalSlider_ContrastFront_value↩

Changed(), on_horizontalSlider_ContrastProj_valueChanged(), on_horizontalSlider_HueFront_valueChanged(), on_horizontalSlider_HueProj_valueChanged(), on_horizontalSlider_SaturationFront_valueChanged(), on_horizontalSlider_SaturationProj_valueChanged(), on_horizontalSlider_SharpnessFront_valueChanged(), and on_horizontalSlider_SharpnessProj_valueChanged().

6.19.4.2 Hardware::Microscope* DialogSettings::Microscope [private]

Definition at line 134 of file dialogsettings.h.

6.19.4.3 SoilMath::NN* DialogSettings::NN [private]

Definition at line 135 of file dialogsettings.h.

6.19.4.4 SoilAnalyzer::SoilSettings* DialogSettings::Settings = nullptr

Definition at line 20 of file dialogsettings.h.

Referenced by DialogSettings(), on_cb_fillHoles_3_clicked(), on_cb_ignoreBorder_3_clicked(), on_cb_use_adaptContrast_3_clicked(), on_cb_useBlur_3_clicked(), on_checkBox_Backlight_clicked(), on_checkBox_InvertEncoder_clicked(), on_checkBox_PredictShape_clicked(), on_checkBox_revolt_clicked(), on_checkBox_useCUDA_clicked(), on_checkBox_useHDR_clicked(), on_checkBox_useRainbow_clicked(), on_comboBox_Microscopes_currentIndexChanged(), on_comboBox_Resolution_currentIndexChanged(), on_doubleSpinBox_LightLevel_editingFinished(), on_horizontalSlider_BrightFront_valueChanged(), on_horizontalSlider_BrightProj_valueChanged(), on_horizontalSlider_ContrastFront_valueChanged(), on_horizontalSlider_ContrastProj_valueChanged(), on_horizontalSlider_HueFront_valueChanged(), on_horizontalSlider_HueProj_valueChanged(), on_horizontalSlider_SaturationFront_valueChanged(), on_horizontalSlider_SaturationProj_valueChanged(), on_horizontalSlider_SharpnessFront_valueChanged(), on_horizontalSlider_SharpnessProj_valueChanged(), on_pushButton_Open_clicked(), on_pushButton_RestoreDefault_clicked(), on_pushButton_Save_clicked(), on_pushButton_SelectNN_clicked(), on_pushButton_SelectNNFolder_clicked(), on_pushButton_selectSampleFolder_clicked(), on_pushButton_SelectSettingFolder_clicked(), on_rb_useClose_3_clicked(), on_rb_useDark_3_toggled(), on_rb_useDilate_3_clicked(), on_rb_useErode_3_clicked(), on_rb_useOpen_3_clicked(), on_sb_morphMask_3_editingFinished(), on_sb_sigmaFactor_3_editingFinished(), on_spinBox_NoFrames_editingFinished(), and on_spinBox_NoShots_editingFinished().

6.19.4.5 Ui::DialogSettings* DialogSettings::ui [private]

Definition at line 133 of file dialogsettings.h.

Referenced by DialogSettings(), on_cb_use_adaptContrast_3_clicked(), on_cb_useBlur_3_clicked(), on_checkBox_Backlight_clicked(), on_checkBox_useHDR_clicked(), on_comboBox_Microscopes_currentIndexChanged(), on_doubleSpinBox_Beta_editingFinished(), on_doubleSpinBox_endError_editingFinished(), on_doubleSpinBox_LightLevel_editingFinished(), on_doubleSpinBox_maxWeight_editingFinished(), on_doubleSpinBox_MinWeight_editingFinished(), on_doubleSpinBox_MutationRate_editingFinished(), on_pushButton_SelectNN_clicked(), on_pushButton_SelectNNFolder_clicked(), on_pushButton_selectSampleFolder_clicked(), on_pushButton_SelectSettingFolder_clicked(), on_sb_morphMask_3_editingFinished(), on_sb_sigmaFactor_3_editingFinished(), on_spinBox_Elitisme_editingFinished(), on_spinBox_HiddenNeurons_editingFinished(), on_spinBox_InputNeurons_editingFinished(), on_spinBox_MaxGen_editingFinished(), on_spinBox_NoFrames_editingFinished(), on_spinBox_NoShots_editingFinished(), on_spinBox_OutputNeurons_editingFinished(), on_spinBox_PopSize_editingFinished(), openTab(), and ~DialogSettings().

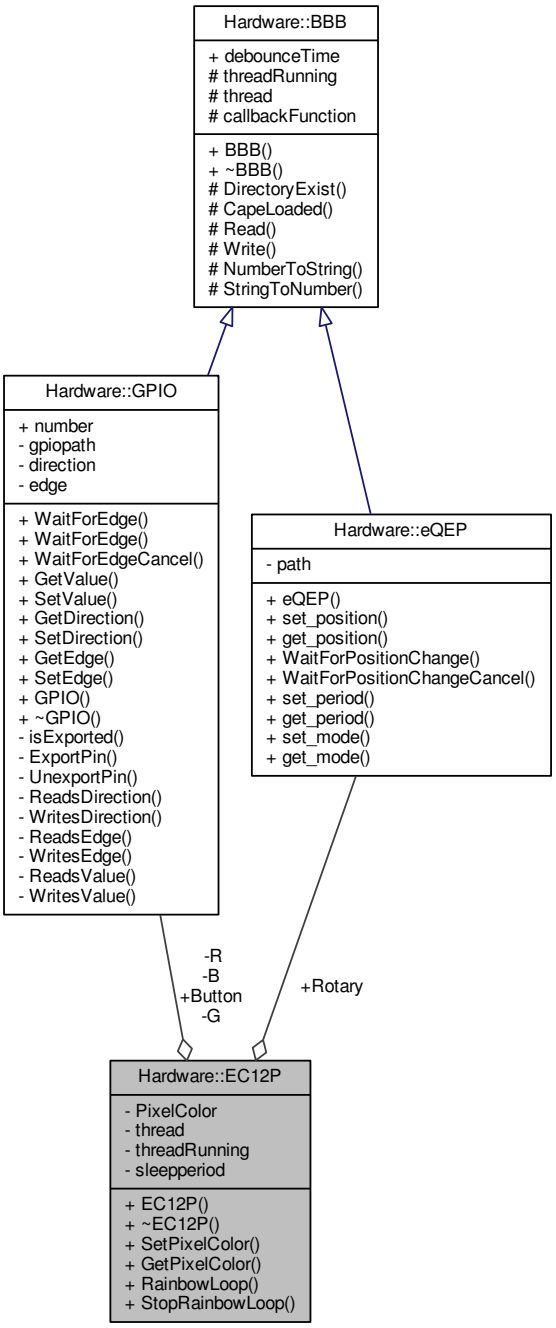
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/dialogsettings.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/dialogsettings.cpp

6.20 Hardware::EC12P Class Reference

`#include <EC12P.h>`

Collaboration diagram for Hardware::EC12P:



Public Types

- enum Color {
  Red, Pink, Blue, SkyBlue,
  Green, Yellow, White, None }

Public Member Functions

- EC12P ()
- ~EC12P ()
- void SetPixelColor (Color value)
- Color GetPixelColor ()
- void RainbowLoop (int sleepperiod)
- void StopRainbowLoop ()

Public Attributes

- eQEP Rotary {eQEP2, eQEP::eQEP_Mode_Absolute}
- GPIO Button {68}

Private Attributes

- Color PixelColor
- GPIO R {31}
- GPIO B {48}
- GPIO G {51}
- pthread_t thread
- bool threadRunning
- int sleepperiod

Friends

- void * colorLoop (void *value)

6.20.1 Detailed Description

Definition at line 23 of file EC12P.h.

6.20.2 Member Enumeration Documentation

6.20.2.1 enum Hardware::EC12P::Color

Enumerator indicating the color of the encoder shaft

Enumerator

Red Red

Pink Pink

Blue Blue

SkyBlue SkyBlue

Green Green

Yellow Yellow

White White

None Off

Definition at line 29 of file EC12P.h.

6.20.3 Constructor & Destructor Documentation

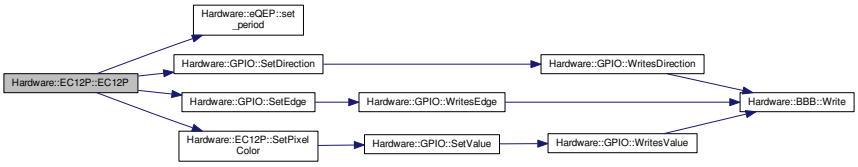
6.20.3.1 EC12P::EC12P ( )

Constructor

Definition at line 12 of file EC12P.cpp.

References B, Button, G, Hardware::GPIO::Input, None, Hardware::GPIO::Output, R, Hardware::GPIO::Rising, Rotary, Hardware::eQEP::set↩
_period(), Hardware::GPIO::SetDirection(), Hardware::GPIO::SetEdge(), SetPixelColor(), and threadRunning.

Here is the call graph for this function:



6.20.3.2 EC12P::~EC12P ( )

De-constructor

Definition at line 30 of file EC12P.cpp.

6.20.4 Member Function Documentation

6.20.4.1 Color Hardware::EC12P::GetPixelColor ( ) [inline]

Definition at line 41 of file EC12P.h.

6.20.4.2 void EC12P::RainbowLoop (int *sleepperiod*)

Loops through all the colors except of as a thread

Definition at line 82 of file EC12P.cpp.

References colorLoop, sleepperiod, thread, and threadRunning.

6.20.4.3 void EC12P::SetPixelColor (Color *value*)

Set the shaft color

Parameters

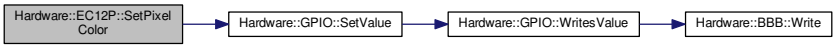
value	as Color enumerator

Definition at line 35 of file EC12P.cpp.

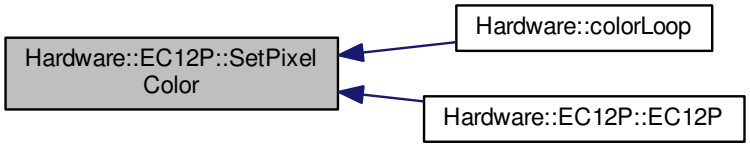
References B, Blue, G, Green, Hardware::GPIO::High, Hardware::GPIO::Low, None, Pink, PixelColor, R, Red, Hardware::GPIO::SetValue(), SkyBlue, White, and Yellow.

Referenced by Hardware::colorLoop(), and EC12P().

Here is the call graph for this function:



Here is the caller graph for this function:



6.20.4.4 void Hardware::EC12P::StopRainbowLoop ( ) [inline]

Definition at line 44 of file EC12P.h.

6.20.5 Friends And Related Function Documentation

6.20.5.1 void* colorLoop (void * *value*) [friend]

The thread function that runs trough all the colors

Definition at line 91 of file EC12P.cpp.

Referenced by RainbowLoop().

6.20.6 Member Data Documentation

6.20.6.1 GPIO Hardware::EC12P::B {48} [private]

Blue LED

Definition at line 53 of file EC12P.h.

Referenced by EC12P(), and SetPixelColor().

6.20.6.2 GPIO Hardware::EC12P::Button {68}

The pushbutton

Definition at line 47 of file EC12P.h.

Referenced by EC12P().

6.20.6.3 GPIO Hardware::EC12P::G {51} [private]

Green LED

Definition at line 54 of file EC12P.h.

Referenced by EC12P(), and SetPixelColor().

6.20.6.4 Color Hardware::EC12P::PixelColor [private]

Current shaft color

Definition at line 50 of file EC12P.h.

Referenced by SetPixelColor().

6.20.6.5 GPIO Hardware::EC12P::R {31} [private]

Red LED

Definition at line 52 of file EC12P.h.

Referenced by EC12P(), and SetPixelColor().

6.20.6.6 eQEP Hardware::EC12P::Rotary {eQEP2, eQEP::eQEP_Mode_Absolute}

The encoder

Definition at line 46 of file EC12P.h.

Referenced by EC12P().

6.20.6.7 int Hardware::EC12P::sleepperiod [private]

Sleep period

Definition at line 58 of file EC12P.h.

Referenced by Hardware::colorLoop(), and RainbowLoop().

6.20.6.8 pthread_t Hardware::EC12P::thread [private]

the thread

Definition at line 56 of file EC12P.h.

Referenced by RainbowLoop().

6.20.6.9 bool Hardware::EC12P::threadRunning [private]

Bool used to stop the thread

Definition at line 57 of file EC12P.h.

Referenced by Hardware::colorLoop(), EC12P(), and RainbowLoop().

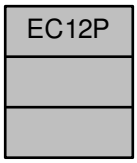
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/EC12P.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/EC12P.cpp

6.21 EC12P Class Reference

`#include <EC12P.h>`

Collaboration diagram for EC12P:



6.21.1 Detailed Description

Interaction with the sparksfun RGB encoder

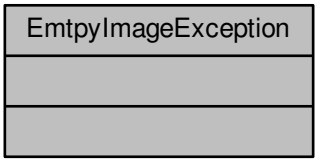
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/EC12P.h

6.22 EmtpyImageException Class Reference

`#include <EmptyImageException.h>`

Collaboration diagram for EmtpyImageException:



6.22.1 Detailed Description

Exception class which is thrown when operations are about to start on a empty image.

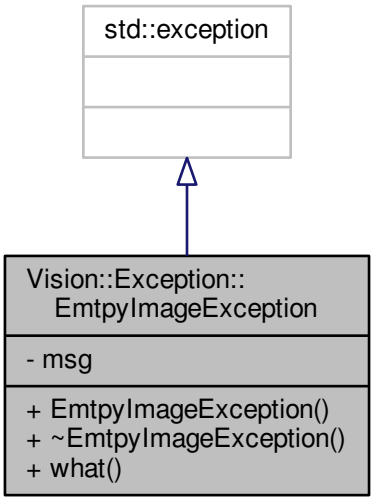
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/EmptyImageException.h

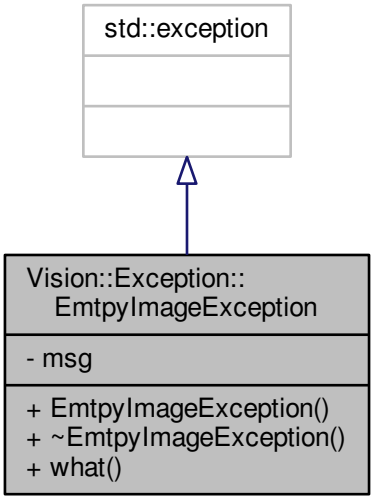
6.23 Vision::Exception::EmtpyImageException Class Reference

`#include <EmptyImageException.h>`

Inheritance diagram for Vision::Exception::EmtpyImageException:



Collaboration diagram for Vision::Exception::EmtpyImageException:



Public Member Functions

- EmtpyImageException (string m="Empty Image!")
- ~EmtpyImageException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.23.1 Detailed Description

Definition at line 22 of file EmptyImageException.h.

6.23.2 Constructor & Destructor Documentation

6.23.2.1 Vision::Exception::EmtpyImageException::EmtpyImageException (string *m* = "Empty Image!") [inline]

Definition at line 24 of file EmptyImageException.h.

6.23.2.2 Vision::Exception::EmtpyImageException::~EmtpyImageException ( ) [inline]

Definition at line 25 of file EmptyImageException.h.

6.23.3 Member Function Documentation

6.23.3.1 const char* Vision::Exception::EmtpyImageException::what ( ) const [inline]

Definition at line 26 of file EmptyImageException.h.

6.23.4 Member Data Documentation

6.23.4.1 string Vision::Exception::EmtpyImageException::msg [private]

Definition at line 26 of file EmptyImageException.h.

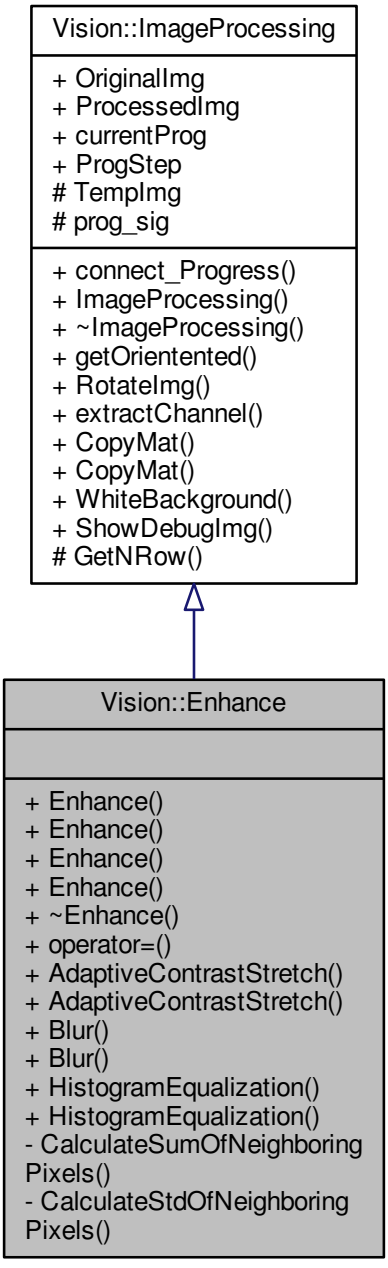
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/EmptyImageException.h

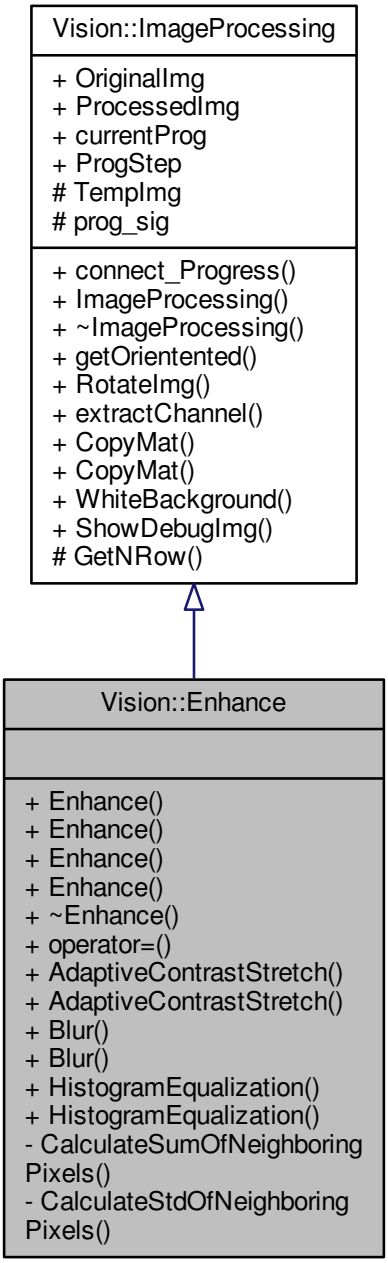
6.24 Vision::Enhance Class Reference

```
#include <Enhance.h>
```

Inheritance diagram for Vision::Enhance:



Collaboration diagram for Vision::Enhance:



Public Types

- enum EnhanceOperation { _AdaptiveContrastStretch, _Blur, _HistogramEqualization }

Public Member Functions

- Enhance ()
- Enhance (const Mat &src)
- Enhance (const Mat &src, Mat &dst, uint8_t kernelsize=9, float factor=1.0, EnhanceOperation operation=_Blur)
- Enhance (const Enhance &rhs)
- ~Enhance ()
- Enhance & operator= (Enhance rhs)
- void AdaptiveContrastStretch (uint8_t kernelsize, float factor, bool chain=false)
- void AdaptiveContrastStretch (const Mat &src, Mat &dst, uint8_t kernelsize, float factor)
- void Blur (uint8_t kernelsize, bool chain=false)
- void Blur (const Mat &src, Mat &dst, uint8_t kernelsize)

- void HistogramEqualization (bool chain=false)
- void HistogramEqualization (const Mat &src, Mat &dst)

Private Member Functions

- void CalculateSumOfNeighboringPixels (uchar *O, int i, int hKsize, int nCols, uint32_t &sum)
- float CalculateStdOfNeighboringPixels (uchar *O, int i, int hKsize, int nCols, int noNeighboursPix, float mean)

Additional Inherited Members

6.24.1 Detailed Description

Definition at line 18 of file Enhance.h.

6.24.2 Member Enumeration Documentation

6.24.2.1 enum Vision::Enhance::EnhanceOperation

Enumerator indicating the requested enhancement operation

Enumerator

_AdaptiveContrastStretch custom adaptive contrast stretch operation

_Blur Blur operation

_HistogramEqualization Histogram equalization

Definition at line 27 of file Enhance.h.

6.24.3 Constructor & Destructor Documentation

6.24.3.1 Enhance::Enhance ( )

Constructor

Definition at line 15 of file Enhance.cpp.

6.24.3.2 Enhance::Enhance (const Mat & *src*)

Constructor

Parameters

src	cv::Mat source image

Definition at line 20 of file Enhance.cpp.

References Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

6.24.3.3 Vision::Enhance::Enhance (const Mat & *src*, Mat & *dst*, uint8_t *kernelsize* = 9, float *factor* = 1.0, EnhanceOperation *operation* = _Blur)

6.24.3.4 Enhance::Enhance (const Enhance & *rhs*)

Definition at line 25 of file Enhance.cpp.

References Vision::ImageProcessing::OriginalImg, Vision::ImageProcessing::ProcessedImg, and Vision::ImageProcessing::TempImg.

6.24.3.5 Enhance::~Enhance ( )

Dec-constructor

Definition at line 60 of file Enhance.cpp.

6.24.4 Member Function Documentation

6.24.4.1 void Vision::Enhance::AdaptiveContrastStretch (uint8_t *kernelsize*, float *factor*, bool *chain* = false)

6.24.4.2 void Vision::Enhance::AdaptiveContrastStretch (const Mat & *src*, Mat & *dst*, uint8_t *kernelsize*, float *factor*)

6.24.4.3 void Vision::Enhance::Blur (uint8_t *kernelsize*, bool *chain* = `false`)

6.24.4.4 void Vision::Enhance::Blur (const Mat & *src*, Mat & *dst*, uint8_t *kernelsize*)

6.24.4.5 float Enhance::CalculateStdOfNeighboringPixels (uchar * *O*, int *i*, int *hKsize*, int *nCols*, int *noNeighboursPix*, float *mean*) [private]

Calculate the standard deviation of the neighboring pixels

Parameters

O	uchar pointer to the current pixel of the original image
i	current counter
hKsize	half the kernelsize
nCols	total number of columns
noNeighboursPix	total number of neighboring pixels
mean	mean value of the neighboring pixels

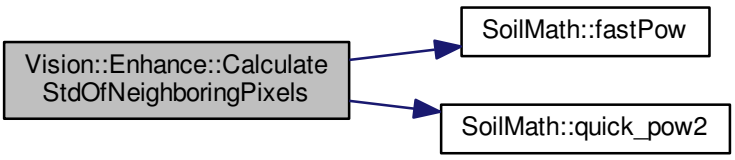
Returns

standard deviation

Definition at line 80 of file Enhance.cpp.

References SoilMath::fastPow(), and SoilMath::quick_pow2().

Here is the call graph for this function:



6.24.4.6 void Enhance::CalculateSumOfNeighboringPixels (uchar * *O*, int *i*, int *hKsize*, int *nCols*, uint32_t & *sum*) [private]

Calculate the sum of the neighboring pixels

Parameters

O	uchar pointer to the current pixel of the original image
i	current counter
hKsize	half the kernelsize
nCols	total number of columns
sum	Total sum of the neighboringpixels

Definition at line 105 of file Enhance.cpp.

6.24.4.7 void Enhance::HistogramEqualization (bool *chain* = `false`)

Stretches the image using a histogram

Parameters

chain	use the results from the previous operation default value = false;

Definition at line 277 of file Enhance.cpp.

References CHAIN_PROCESS, EMPTY_CHECK, SoilMath::Stats< T1, T2, T3 >::max, SoilMath::Stats< T1, T2, T3 >::min, Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

6.24.4.8 void Vision::Enhance::HistogramEqualization (const Mat & *src*, Mat & *dst*)

6.24.4.9 Enhance & Enhance::operator= (Enhance *rhs*)

Definition at line 62 of file Enhance.cpp.

References Vision::ImageProcessing::OriginalImg, Vision::ImageProcessing::ProcessedImg, and Vision::ImageProcessing::TempImg.

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

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.cpp

6.25 Enhance Class Reference

Collaboration diagram for Enhance:

6.25.1 Detailed Description

class which enhances a greyscale cv::Mat image

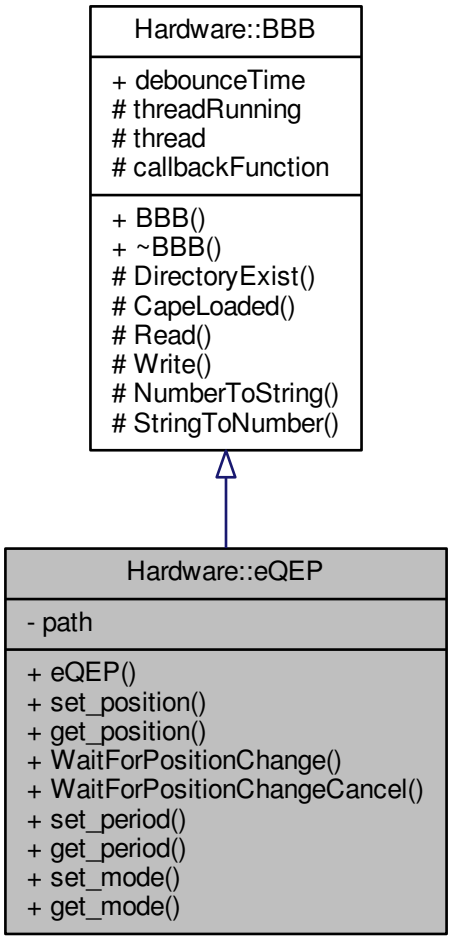
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Enhance.cpp

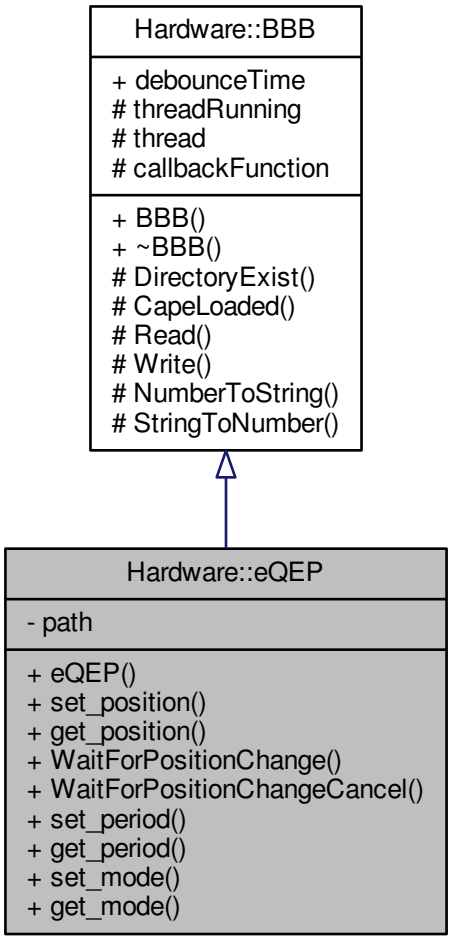
6.26 Hardware::eQEP Class Reference

```
#include <eqep.h>
```

Inheritance diagram for Hardware::eQEP:



Collaboration diagram for Hardware::eQEP:



Public Types

- enum eQEP_Mode { eQEP_Mode_Absolute = 0, eQEP_Mode_Relative = 1, eQEP_Mode_Error = 2 }

Public Member Functions

- eQEP (std::string _path, eQEP_Mode _mode)
- void set_position (int32_t position)
- int32_t get_position (bool _poll=true)
- int WaitForPositionChange (CallbackType callback)
- void WaitForPositionChangeCancel ()
- void set_period (long long unsigned int period)
- uint64_t get_period ()
- void set_mode (eQEP_Mode mode)
- eQEP_Mode get_mode ()

Private Attributes

- std::string path

Friends

- void * threadedPolleqep (void *value)

Additional Inherited Members

6.26.1 Detailed Description

Definition at line 39 of file eqep.h.

6.26.2 Member Enumeration Documentation

6.26.2.1 enum Hardware::eQEP::eQEP_Mode

Enumerator

eQEP_Mode_Absolute

eQEP_Mode_Relative

eQEP_Mode_Error

Definition at line 45 of file eqep.h.

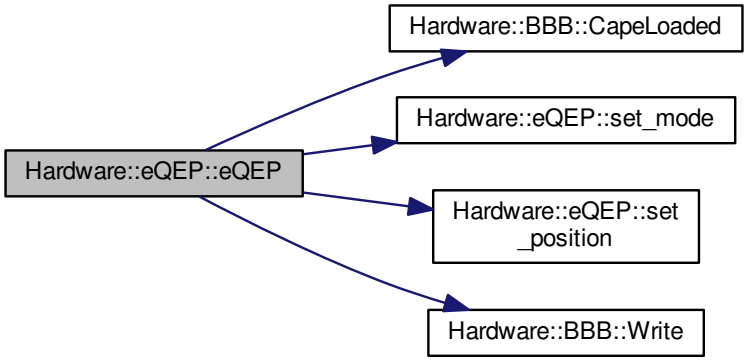
6.26.3 Constructor & Destructor Documentation

6.26.3.1 Hardware::eQEP::eQEP (std::string *_path,* eQEP::eQEP_Mode *_mode*)

Definition at line 42 of file eqep.cpp.

References Hardware::BBB::CapeLoaded(), eQEP0, eQEP1, eQEP2, set_mode(), set_position(), SLOTS, and Hardware::BBB::Write().

Here is the call graph for this function:



6.26.4 Member Function Documentation

6.26.4.1 eQEP::eQEP_Mode Hardware::eQEP::get_mode ( )

Definition at line 219 of file eqep.cpp.

References eQEP_Mode_Error, and path.

6.26.4.2 uint64_t Hardware::eQEP::get_period ( )

Definition at line 195 of file eqep.cpp.

References path.

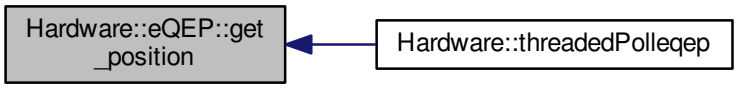
6.26.4.3 int32_t Hardware::eQEP::get_position (bool *_poll* = `true`)

Definition at line 137 of file eqep.cpp.

References path.

Referenced by Hardware::threadedPolleqep().

Here is the caller graph for this function:



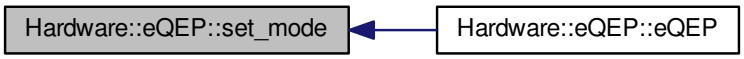
6.26.4.4 void Hardware::eQEP::set_mode (eQEP::eQEP_Mode *_mode*)

Definition at line 105 of file eqep.cpp.

References path.

Referenced by eQEP().

Here is the caller graph for this function:



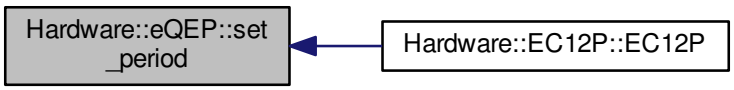
6.26.4.5 void Hardware::eQEP::set_period (long long unsigned int *period*)

Definition at line 85 of file eqep.cpp.

References path.

Referenced by Hardware::EC12P::EC12P().

Here is the caller graph for this function:



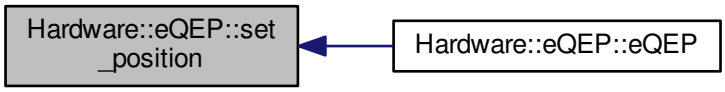
6.26.4.6 void Hardware::eQEP::set_position (int32_t *position*)

Definition at line 65 of file eqep.cpp.

References path.

Referenced by eQEP().

Here is the caller graph for this function:



6.26.4.7 int Hardware::eQEP::WaitForPositionChange (CallbackType *callback*)

Definition at line 124 of file eqep.cpp.

References Hardware::BBB::callbackFunction, Hardware::BBB::thread, threadedPolleqep, and Hardware::BBB::threadRunning.

6.26.4.8 void Hardware::eQEP::WaitForPositionChangeCancel ( ) [inline]

Definition at line 68 of file eqep.h.

References Hardware::BBB::threadRunning.

6.26.5 Friends And Related Function Documentation

6.26.5.1 void* threadedPolleqep (void * *value*) [friend]

Definition at line 242 of file eqep.cpp.

Referenced by WaitForPositionChange().

6.26.6 Member Data Documentation

6.26.6.1 std::string Hardware::eQEP::path [private]

Definition at line 41 of file eqep.h.

Referenced by get_mode(), get_period(), get_position(), set_mode(), set_period(), and set_position().

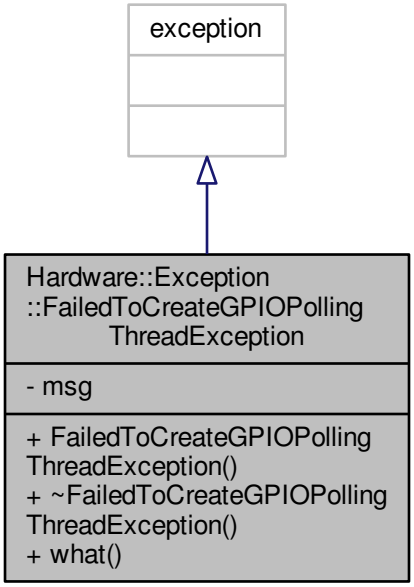
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/eqep.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/eqep.cpp

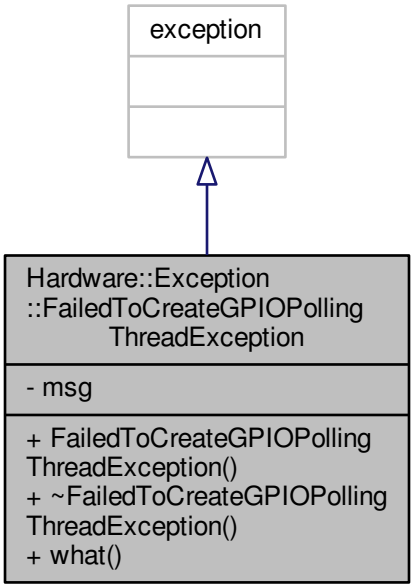
6.27 Hardware::Exception::FailedToCreateGPIOPollingThreadException Class Reference

`#include <FailedToCreateGPIOPollingThreadException.h>`

Inheritance diagram for Hardware::Exception::FailedToCreateGPIOPollingThreadException:



Collaboration diagram for Hardware::Exception::FailedToCreateGPIOPollingThreadException:



Public Member Functions

- FailedToCreateGPIOPollingThreadException (string m="Failed to create GPIO polling thread!")
- ~FailedToCreateGPIOPollingThreadException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.27.1 Detailed Description

Definition at line 17 of file FailedToCreateGPIOPollingThreadException.h.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Hardware::Exception::FailedToCreateGPIOPollingThreadException::FailedToCreateGPIOPollingThreadException (string *m* = "Failed to create GPIO polling thread!") [inline]

Definition at line 19 of file FailedToCreateGPIOPollingThreadException.h.

6.27.2.2 Hardware::Exception::FailedToCreateGPIOPollingThreadException::~FailedToCreateGPIOPollingThreadException ( ) [inline]

Definition at line 22 of file FailedToCreateGPIOPollingThreadException.h.

6.27.3 Member Function Documentation

6.27.3.1 const char* Hardware::Exception::FailedToCreateGPIOPollingThreadException::what ( ) const [inline]

Definition at line 23 of file FailedToCreateGPIOPollingThreadException.h.

6.27.4 Member Data Documentation

6.27.4.1 string Hardware::Exception::FailedToCreateGPIOPollingThreadException::msg [private]

Definition at line 23 of file FailedToCreateGPIOPollingThreadException.h.

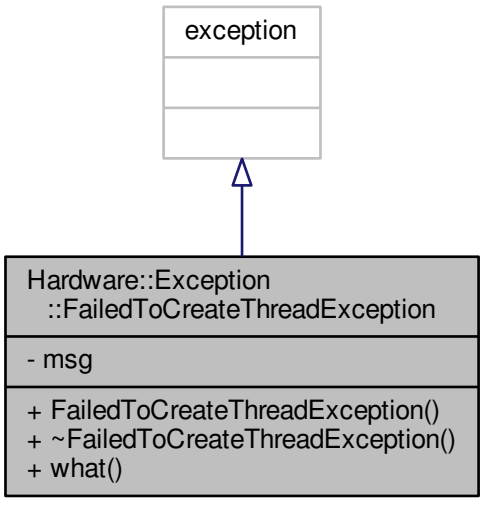
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/FailedToCreateGPIOPollingThreadException.h

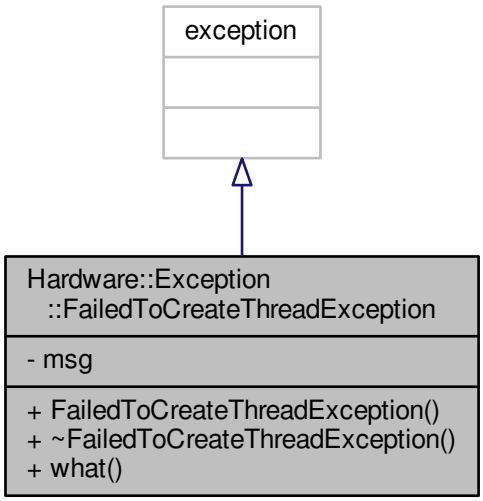
6.28 Hardware::Exception::FailedToCreateThreadException Class Reference

`#include <FailedToCreateThreadException.h>`

Inheritance diagram for Hardware::Exception::FailedToCreateThreadException:



Collaboration diagram for Hardware::Exception::FailedToCreateThreadException:



Public Member Functions

- FailedToCreateThreadException (string m="Couldn't create the thread!")
- ~FailedToCreateThreadException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.28.1 Detailed Description

Definition at line 17 of file FailedToCreateThreadException.h.

6.28.2 Constructor & Destructor Documentation

6.28.2.1 Hardware::Exception::FailedToCreateThreadException::FailedToCreateThreadException (string *m* = "Couldn't create the thread!") [inline]

Definition at line 19 of file FailedToCreateThreadException.h.

6.28.2.2 Hardware::Exception::FailedToCreateThreadException::~FailedToCreateThreadException ( ) [inline]

Definition at line 21 of file FailedToCreateThreadException.h.

6.28.3 Member Function Documentation

6.28.3.1 const char* Hardware::Exception::FailedToCreateThreadException::what ( ) const [inline]

Definition at line 22 of file FailedToCreateThreadException.h.

6.28.4 Member Data Documentation

6.28.4.1 string Hardware::Exception::FailedToCreateThreadException::msg [private]

Definition at line 22 of file FailedToCreateThreadException.h.

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

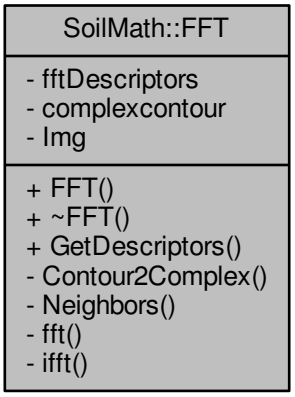
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/FailedToCreateThreadException.h

6.29 SoilMath::FFT Class Reference

Fast Fourier Transform class.

`#include <FFT.h>`

Collaboration diagram for SoilMath::FFT:



Public Member Functions

- FFT ()

  Standard constructor.
- ~FFT ()

  Standard deconstructor.
- ComplexVect_t GetDescriptors (const cv::Mat &img)

  Transforming the img to the frequency domain and returning the Fourier Descriptors.

Private Member Functions

- ComplexVect_t Contour2Complex (const cv::Mat &img, float centerCol, float centerRow)

  Contour2Complex a private function which translates a continous contour image to a vector of complex values. The contour is found using a depth first search with extension list. The alghorithm is based upon `MIT opencourseware 6-034-artificial-intelligence lecture 4`
- iContour_t Neighbors (uchar *O, int pixel, uint32_t columns, uint32_t rows)

  Neighbors a private function returning the neighboring pixels which belong to a contour.
- void fft (ComplexArray_t &CA)

  fft a private function calculating the Fast Fourier Transform let m *be an integer and let* $N = 2^m$ *also* $CA = [x_0, \ldots, x_{N-1}]$ *is an* N *dimensional complex vector let* $\omega = \exp(\frac{-2\pi i}{N})$ *then* $c_k = \frac{1}{N} \sum_{j=0}^{j=N-1} CA_j \omega^{jk}$
- void ifft (ComplexArray_t &CA)

  ifft

Private Attributes

- ComplexVect_t fftDescriptors
- ComplexVect_t complexcontour
- cv::Mat Img

6.29.1 Detailed Description

Fast Fourier Transform class.

Use this class to transform a black and white blob presented as a cv::Mat with values 0 or 1 to a vector of complex values representing the Fourier Descriptors.

Definition at line 31 of file FFT.h.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 SoilMath::FFT::FFT ( )

Standard constructor.

Definition at line 11 of file FFT.cpp.

6.29.2.2 SoilMath::FFT::~FFT ( )

Standard deconstructor.

Definition at line 13 of file FFT.cpp.

6.29.3 Member Function Documentation

6.29.3.1 ComplexVect_t SoilMath::FFT::Contour2Complex (const cv::Mat & *img*, float *centerCol*, float *centerRow*) [private]

Contour2Complex a private function which translates a continous contour image to a vector of complex values. The contour is found using a depth first search with extension list. The alghorithm is based upon MIT opencourseware 6-034-artificial-intelligence lecture 4

Parameters

img	contour in the form of a cv::Mat type CV_8UC1. Which should consist of a continous contour. $\{img \in \mathbb{Z} \| 0 \leq img \leq 1\}$
centerCol	centre of the contour X value
centerRow	centre of the contour Y value

Returns

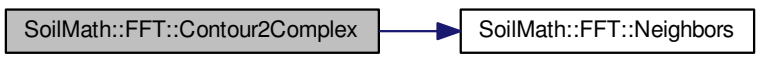
a vector with complex values, represing the contour as a function

Definition at line 64 of file FFT.cpp.

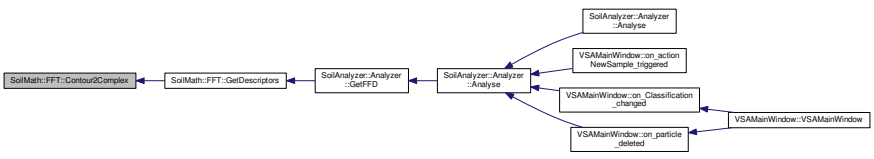
References EXCEPTION_NO_CONTOUR_FOUND, EXCEPTION_NO_CONTOUR_FOUND_NR, and Neighbors().

Referenced by GetDescriptors().

Here is the call graph for this function:



Here is the caller graph for this function:



6.29.3.2 void SoilMath::FFT::fft (ComplexArray_t & *CA*) [private]

fft a private function calculating the Fast Fourier Transform let m be an integer and let $N = 2^m$ also $CA = [x_0, \ldots, x_{N-1}]$ is an N dimensional complex vector let $\omega = \exp(\frac{-2\pi i}{N})$ then $c_k = \frac{1}{N} \sum_{j=0}^{j=N-1} CA_j \omega^{jk}$

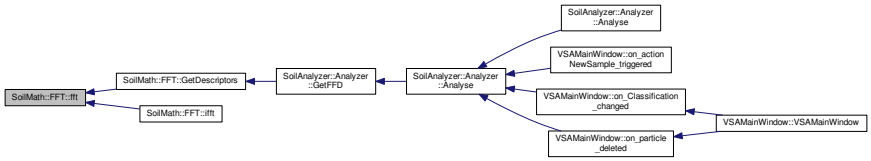
Parameters

CA	a $CA = [x_0, \ldots, x_{N-1}]$ is an N dimensional complex vector

Definition at line 149 of file FFT.cpp.

Referenced by GetDescriptors(), and ifft().

Here is the caller graph for this function:



6.29.3.3 ComplexVect_t SoilMath::FFT::GetDescriptors (const cv::Mat & *img*)

Transforming the img to the frequency domain and returning the Fourier Descriptors.

Parameters

img	contour in the form of a cv::Mat type CV_8UC1. Which should consist of a continous contour. $\{img \in \mathbb{Z} \vert 0 \leq img \leq 1\}$

Returns

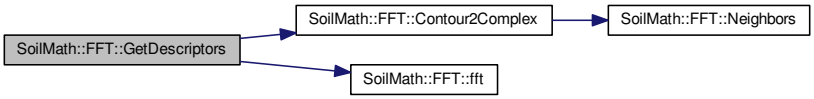
a vector with complex values, represing the contour in the frequency domain, expressed as Fourier Descriptors

Definition at line 15 of file FFT.cpp.

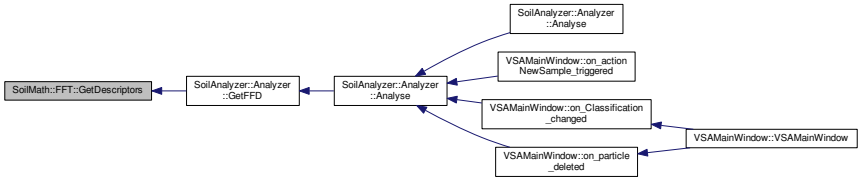
References complexcontour, Contour2Complex(), fft(), and fftDescriptors.

Referenced by SoilAnalyzer::Analyzer::GetFFD().

Here is the call graph for this function:



Here is the caller graph for this function:



6.29.3.4 void SoilMath::FFT::ifft (ComplexArray_t & *CA*) [private]

ifft

Parameters

CA	

Definition at line 169 of file FFT.cpp.

References fft().

Here is the call graph for this function:

6.29.3.5 iContour_t SoilMath::FFT::Neighbors (uchar * *O*, int *pixel*, uint32_t *columns*, uint32_t *rows*) [private]

Neighbors a private function returning the neighboring pixels which belong to a contour.

Parameters

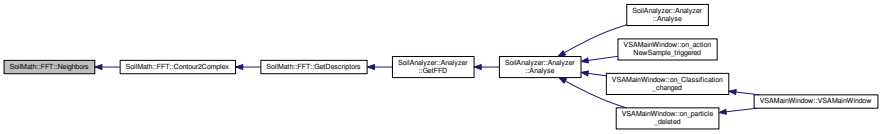
O	uchar pointer to the data
pixel	current counter
columns	total number of columns
rows	total number of rows

Returns

Definition at line 43 of file FFT.cpp.

Referenced by Contour2Complex().

Here is the caller graph for this function:



6.29.4 Member Data Documentation

6.29.4.1 ComplexVect_t SoilMath::FFT::complexcontour [private]

Vector with complex values which represent the contour

Definition at line 59 of file FFT.h.

Referenced by GetDescriptors().

6.29.4.2 ComplexVect_t SoilMath::FFT::fftDescriptors [private]

Vector with complex values which represent the descriptors

Definition at line 56 of file FFT.h.

Referenced by GetDescriptors().

6.29.4.3 cv::Mat SoilMath::FFT::Img [private]

Img which will be analysed

Definition at line 61 of file FFT.h.

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

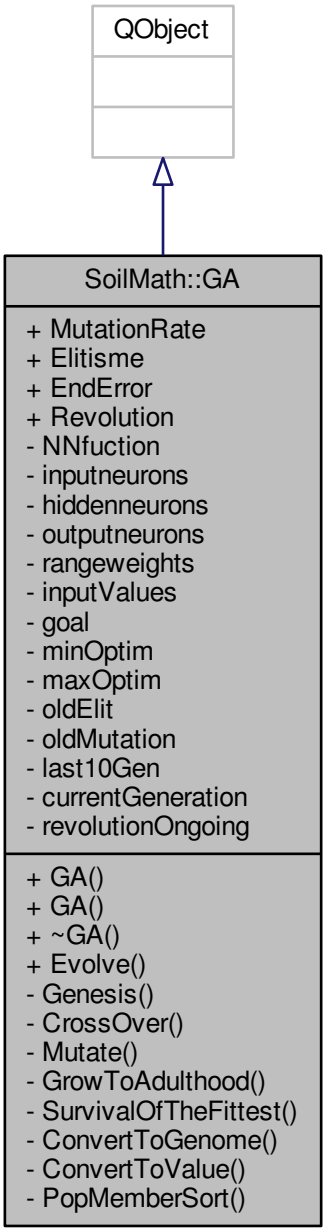
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/FFT.h

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/FFT.cpp

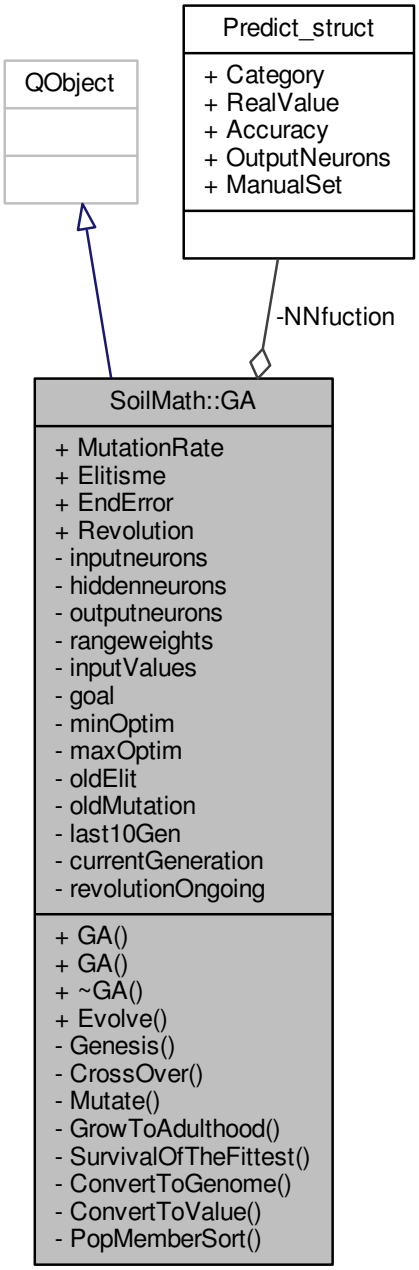
6.30 SoilMath::GA Class Reference

`#include <GA.h>`

Inheritance diagram for SoilMath::GA:



Collaboration diagram for SoilMath::GA:



Signals

- void learnErrorUpdate (double newError)

Public Member Functions

- GA ()

  GA Standard constructor.
- GA (NNfunctionType nnfunction, uint32_t inputneurons, uint32_t hiddenneurons, uint32_t outputneurons)

  GA Construction with a Neural Network initializers.
- ~GA ()

  GA standard de constructor.
- void Evolve (const InputLearnVector_t &inputValues, Weight_t &weights, MinMaxWeight_t rangeweights, OutputLearnVector_t &goal, uint32_t maxGenerations=200, uint32_t popSize=30)

  Evolve Darwin would be proud!!! This function creates a population and itterates through the generation till the maximum number off itterations has been reached of the error is acceptable.

Public Attributes

- float MutationRate = 0.075f
- uint32_t Elitisme = 4
- float EndError = 0.001f
- bool Revolution = true

Private Member Functions

- Population_t Genesis (const Weight_t &weights, uint32_t popSize)

  Genesis private function which is the spark of live, using a random seed.
- void CrossOver (Population_t &pop)

  CrossOver a private function where the partners mate with each other The values or PopMember_t are expressed as bits or ar cut at the point CROSSOVER the population members are paired with the nearest neighbor and new members are created pairing the Genome_t of each other at the CROSSOVER point. Afterwards all the top tiers partners are allowed to mate again.
- void Mutate (Population_t &pop)

  Mutate a private function where individual bits from the Genome_t are mutated at a random uniform distribution event defined by the MUTATIONR↩ ATE.
- void GrowToAdulthood (Population_t &pop, float &totalFitness)

  GrowToAdulthood a private function where the new population members serve as the the input for the Neural Network prediction function. The results are weight against the goal and this weight determine the fitness of the population member.
- bool SurvivalOfTheFittest (Population_t &pop, float &totalFitness)

  SurvivalOfTheFittest a private function where a battle to the death commences The fittest population members have the best chance of survival. Death is instigated with a random uniform distibution. The elite members don't partake in this desctruction The ELITISME rate indicate how many top tier members survive this catastrophic event.
- template<typename T >
  Genome_t ConvertToGenome (T value, std::pair< T, T > range)

  Conversion of the value of type T to Genome_t.
- template<typename T >
  T ConvertToValue (Genome_t gen, std::pair< T, T > range)

  Conversion of the Genome to a value.

Static Private Member Functions

- static bool PopMemberSort (PopMember_t i, PopMember_t j)

  PopMemberSort a private function where the members are sorted according to there fitness ranking.

Private Attributes

- NNfunctionType NNfuction
- uint32_t inputneurons
- uint32_t hiddenneurons
- uint32_t outputneurons
- MinMaxWeight_t rangeweights
- InputLearnVector_t inputValues
- OutputLearnVector_t goal
- float minOptim = 0
- float maxOptim = 0
- uint32_t oldElit = 0
- float oldMutation = 0.
- std::list< double > last10Gen
- uint32_t currentGeneration = 0
- bool revolutionOngoing = false

6.30.1 Detailed Description

Definition at line 36 of file GA.h.

6.30.2 Constructor & Destructor Documentation

6.30.2.1 SoilMath::GA::GA ( )

GA Standard constructor.

Definition at line 11 of file GA.cpp.

6.30.2.2 SoilMath::GA::GA (NNfunctionType *nnfunction*, uint32_t *inputneurons*, uint32_t *hiddenneurons*, uint32_t *outputneurons*)

GA Construction with a Neural Network initializers.

Parameters

nnfunction	the Neural Network prediction function which results will be optimized
inputneurons	the number of input neurons in the Neural Network don't count the bias
hiddenneurons	the number of hidden neurons in the Neural Network don't count the bias
outputneurons	the number of output neurons in the Neural Network

Definition at line 13 of file GA.cpp.

References hiddenneurons, inputneurons, NNfuction, and outputneurons.

6.30.2.3 SoilMath::GA::~GA ( )

GA standard de constructor.

Definition at line 21 of file GA.cpp.

6.30.3 Member Function Documentation

6.30.3.1 template<typename T > Genome_t SoilMath::GA::ConvertToGenome (T *value*, std::pair< T, T > *range*) [inline], [private]

Conversion of the value of type T to Genome_t.

Usage: Use `ConvertToGenome<Type>(type, range)`

Parameters

value	The current value wich should be converted to a Genome_t
range	the range in which the value should fall, this is to have a Genome_t which utilizes the complete range 0000...n till 1111...n

Definition at line 191 of file GA.h.

6.30.3.2 template<typename T > T SoilMath::GA::ConvertToValue (Genome_t *gen*, std::pair< T, T > *range*) [inline], [private]

Conversion of the Genome to a value.

Usage: use `ConvertToValue<Type>(genome, range)`

Parameters

gen	is the Genome which is to be converted
range	is the range in which the value should fall

Definition at line 205 of file GA.h.

6.30.3.3 void SoilMath::GA::CrossOver (Population_t & *pop*) [private]

CrossOver a private function where the partners mate with each other The values or PopMember_t are expressed as bits or ar cut at the point CROSSOVER the population members are paired with the nearest neighbor and new members are created pairing the Genome_t of each other at the CROSSOVER point. Afterwards all the top tiers partners are allowed to mate again.

Parameters

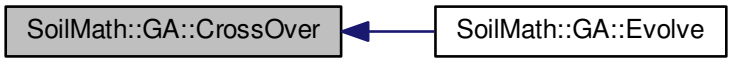
pop	reference to the population

Definition at line 72 of file GA.cpp.

References CROSSOVER, GENE_MAX, and PopMemberStruct::weightsGen.

Referenced by Evolve().

Here is the caller graph for this function:



6.30.3.4 void SoilMath::GA::Evolve (const InputLearnVector_t & *inputValues*, Weight_t & *weights*, MinMaxWeight_t *rangeweights*, OutputLearnVector_t & *goal*, uint32_t *maxGenerations* = 200, uint32_t *popSize* = 30)

Evolve Darwin would be proud!!! This function creates a population and itterates through the generation till the maximum number off itterations has been reached of the error is acceptable.

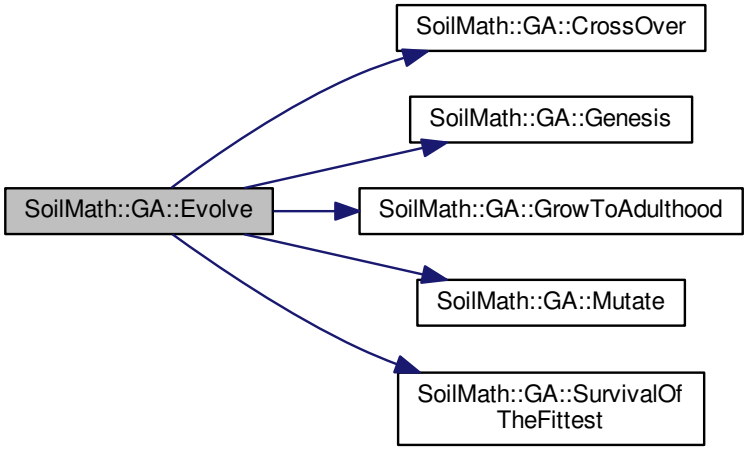
Parameters

inputValues	complex vector with a reference to the inputvalues
weights	reference to the vector of weights which will be optimized
rangeweights	reference to the range of weights, currently it doesn't support indivudal ranges this is because of the crossing
goal	target value towards the Neural Network prediction function will be optimized
maxGenerations	maximum number of itterations default value is 200
popSize	maximum number of population, this should be an even number

Definition at line 23 of file GA.cpp.

References CrossOver(), Elitisme, Genesis(), goal, GrowToAdulthood(), inputValues, maxOptim, minOptim, Mutate(), MutationRate, oldElit, oldMutation, rangeweights, and SurvivalOfTheFittest().

Here is the call graph for this function:



6.30.3.5 Population_t SoilMath::GA::Genesis (const Weight_t & *weights*, uint32_t *popSize*) [private]

Genesis private function which is the spark of live, using a random seed.

Parameters

weights	a reference to the used Weight_t vector
rangeweights	pointer to the range of weights, currently it doesn't support indivudal ranges
popSize	maximum number of population, this should be an even number

Returns

Definition at line 50 of file GA.cpp.

References rangeweights, PopMemberStruct::weights, and PopMemberStruct::weightsGen.

Referenced by Evolve().

Here is the caller graph for this function:

6.30.3.6 void SoilMath::GA::GrowToAdulthood (Population_t & *pop*, float & *totalFitness*) [private]

GrowToAdulthood a private function where the new population members serve as the the input for the Neural Network prediction function. The results are weight against the goal and this weight determine the fitness of the population member.

Parameters

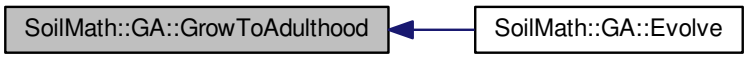
pop	reference to the population
inputValues	a InputLearnVector_t with a reference to the inputvalues
rangeweights	pointer to the range of weights, currently it doesn't support indivudal ranges
goal	a Predict_t type with the expected value
totalFitness	a reference to the total population fitness

Definition at line 152 of file GA.cpp.

References goal, hiddenneurons, inputneurons, inputValues, NNfuction, Predict_struct::OutputNeurons, outputneurons, and rangeweights.

Referenced by Evolve().

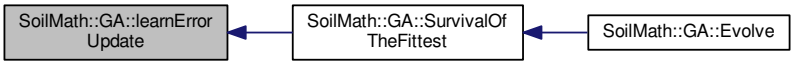
Here is the caller graph for this function:



6.30.3.7 void SoilMath::GA::learnErrorUpdate (double *newError*) [signal]

Referenced by SurvivalOfTheFittest().

Here is the caller graph for this function:



6.30.3.8 void SoilMath::GA::Mutate (Population_t & *pop*) [private]

Mutate a private function where individual bits from the Genome_t are mutated at a random uniform distribution event defined by the MUTATI↩ ONRATE.

Parameters

pop	reference to the population

Definition at line 135 of file GA.cpp.

References GENE_MAX, and MutationRate.

Referenced by Evolve().

Here is the caller graph for this function:

6.30.3.9 static bool SoilMath::GA::PopMemberSort (PopMember_t *i*, PopMember_t *j*) [inline], [static], [private]

PopMemberSort a private function where the members are sorted according to there fitness ranking.

Parameters

i	left hand population member
j	right hand population member

Returns

true if the left member is closser to the goal as the right member.

Definition at line 178 of file GA.h.

References PopMemberStruct::Fitness.

6.30.3.10 bool SoilMath::GA::SurvivalOfTheFittest (Population_t & *pop*, float & *totalFitness*) [private]

SurvivalOfTheFittest a private function where a battle to the death commences The fittest population members have the best chance of survival. Death is instigated with a random uniform distibution. The elite members don't partake in this desctruction The ELITISME rate indicate how many top tier members survive this catastrophic event.

Parameters

inputValues	a InputLearnVector_t with a reference to the inputvalues
totalFitness	a reference to the total population fitness

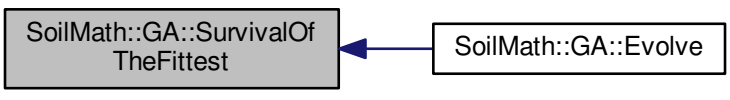
Returns

Definition at line 189 of file GA.cpp.

References currentGeneration, Elitisme, EndError, PopMemberStruct::Fitness, last10Gen, learnErrorUpdate(), maxOptim, minOptim, MutationRate, oldElit, oldMutation, and revolutionOngoing.

Referenced by Evolve().

Here is the caller graph for this function:



6.30.4 Member Data Documentation

6.30.4.1 uint32_t SoilMath::GA::currentGeneration = 0 [private]

Definition at line 105 of file GA.h.

Referenced by SurvivalOfTheFittest().

6.30.4.2 uint32_t SoilMath::GA::Elitisme = 4

total number of the elite bastard

Definition at line 41 of file GA.h.

Referenced by Evolve(), and SurvivalOfTheFittest().

6.30.4.3 float SoilMath::GA::EndError = 0.001f

acceptable error between last itteration

Definition at line 42 of file GA.h.

Referenced by SurvivalOfTheFittest().

6.30.4.4 OutputLearnVector_t SoilMath::GA::goal [private]

Definition at line 98 of file GA.h.

Referenced by Evolve(), and GrowToAdulthood().

6.30.4.5 uint32_t SoilMath::GA::hiddenneurons [private]

the total number of hidden neurons

Definition at line 93 of file GA.h.

Referenced by GA(), and GrowToAdulthood().

6.30.4.6 uint32_t SoilMath::GA::inputneurons [private]

the total number of input neurons

Definition at line 92 of file GA.h.

Referenced by GA(), and GrowToAdulthood().

6.30.4.7 InputLearnVector_t SoilMath::GA::inputValues [private]

Definition at line 97 of file GA.h.

Referenced by Evolve(), and GrowToAdulthood().

6.30.4.8 std::list<double> SoilMath::GA::last10Gen [private]

Definition at line 104 of file GA.h.

Referenced by SurvivalOfTheFittest().

6.30.4.9 float SoilMath::GA::maxOptim = 0 [private]

Definition at line 101 of file GA.h.

Referenced by Evolve(), and SurvivalOfTheFittest().

6.30.4.10 float SoilMath::GA::minOptim = 0 [private]

Definition at line 100 of file GA.h.

Referenced by Evolve(), and SurvivalOfTheFittest().

6.30.4.11 float SoilMath::GA::MutationRate = 0.075f

mutation rate

Definition at line 40 of file GA.h.

Referenced by Evolve(), Mutate(), and SurvivalOfTheFittest().

6.30.4.12 NNfunctionType SoilMath::GA::NNfuction [private]

The Neural Net work function

Definition at line 91 of file GA.h.

Referenced by GA(), and GrowToAdulthood().

6.30.4.13 uint32_t SoilMath::GA::oldElit = 0 [private]

Definition at line 102 of file GA.h.

Referenced by Evolve(), and SurvivalOfTheFittest().

6.30.4.14 float SoilMath::GA::oldMutation = 0. [private]

Definition at line 103 of file GA.h.

Referenced by Evolve(), and SurvivalOfTheFittest().

6.30.4.15 uint32_t SoilMath::GA::outputneurons [private]

the total number of output neurons

Definition at line 94 of file GA.h.

Referenced by GA(), and GrowToAdulthood().

6.30.4.16 MinMaxWeight_t SoilMath::GA::rangeweights [private]

Definition at line 96 of file GA.h.

Referenced by Evolve(), Genesis(), and GrowToAdulthood().

6.30.4.17 bool SoilMath::GA::Revolution = true

Definition at line 43 of file GA.h.

6.30.4.18 bool SoilMath::GA::revolutionOngoing = false [private]

Definition at line 106 of file GA.h.

Referenced by SurvivalOfTheFittest().

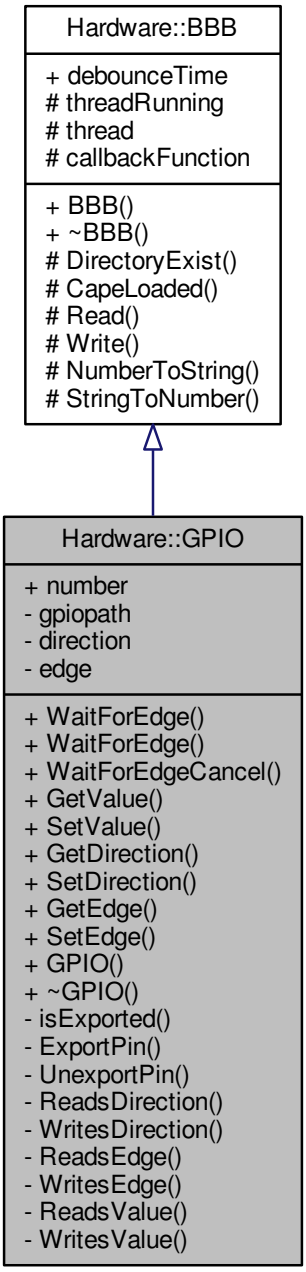
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/GA.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/GA.cpp

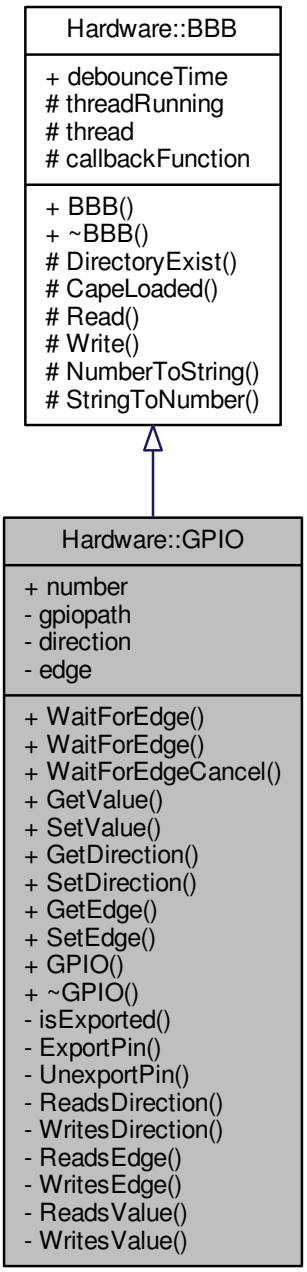
6.31 Hardware::GPIO Class Reference

```
#include <GPIO.h>
```

Inheritance diagram for Hardware::GPIO:



Collaboration diagram for Hardware::GPIO:



Public Types

- enum Direction { Input, Output }
- enum Value { Low = 0, High = 1 }
- enum Edge { None, Rising, Falling, Both }

Public Member Functions

- int WaitForEdge ()
- int WaitForEdge (CallbackType callback)
- void WaitForEdgeCancel ()
- Value GetValue ()
- void SetValue (Value value)
- Direction GetDirection ()
- void SetDirection (Direction direction)
- Edge GetEdge ()

- void SetEdge (Edge edge)
- GPIO (int number)
- ~GPIO ()

Public Attributes

- int number

Private Member Functions

- bool isExported (int number, Direction &dir, Edge &edge)
- bool ExportPin (int number)
- bool UnexportPin (int number)
- Direction ReadsDirection (const string &gpiopath)
- void WritesDirection (const string &gpiopath, Direction direction)
- Edge ReadsEdge (const string &gpiopath)
- void WritesEdge (const string &gpiopath, Edge edge)
- Value ReadsValue (const string &gpiopath)
- void WritesValue (const string &gpiopath, Value value)

Private Attributes

- string gpiopath
- Direction direction
- Edge edge

Friends

- void * threadedPollGPIO (void *value)

Additional Inherited Members

6.31.1 Detailed Description

Definition at line 25 of file GPIO.h.

6.31.2 Member Enumeration Documentation

6.31.2.1 enum Hardware::GPIO::Direction

Enumerator

Input

Output

Definition at line 27 of file GPIO.h.

6.31.2.2 enum Hardware::GPIO::Edge

Enumerator

None

Rising

Falling

Both

Definition at line 29 of file GPIO.h.

6.31.2.3 enum Hardware::GPIO::Value

Enumerator

Low

High

Definition at line 28 of file GPIO.h.

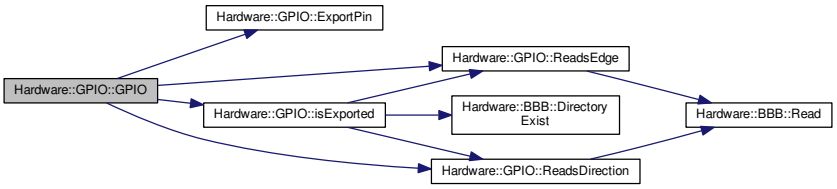
6.31.3 Constructor & Destructor Documentation

6.31.3.1 Hardware::GPIO::GPIO (int *number*)

Definition at line 11 of file GPIO.cpp.

References direction, edge, ExportPin(), gpiopath, GPIOS, isExported(), number, ReadsDirection(), and ReadsEdge().

Here is the call graph for this function:

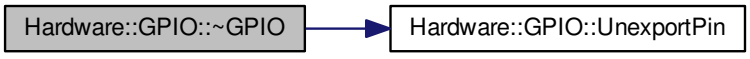


6.31.3.2 Hardware::GPIO::~GPIO ( )

Definition at line 24 of file GPIO.cpp.

References number, and UnexportPin().

Here is the call graph for this function:



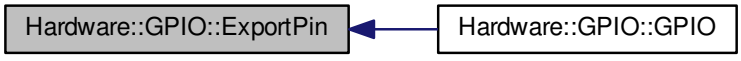
6.31.4 Member Function Documentation

6.31.4.1 bool Hardware::GPIO::ExportPin (int *number*) [private]

Definition at line 102 of file GPIO.cpp.

Referenced by GPIO().

Here is the caller graph for this function:



6.31.4.2 GPIO::Direction Hardware::GPIO::GetDirection ( )

Definition at line 77 of file GPIO.cpp.

References direction.

6.31.4.3 GPIO::Edge Hardware::GPIO::GetEdge ( )

Definition at line 83 of file GPIO.cpp.

References edge.

6.31.4.4 GPIO::Value Hardware::GPIO::GetValue ( )

Definition at line 74 of file GPIO.cpp.

References gpiopath, and ReadsValue().

Here is the call graph for this function:

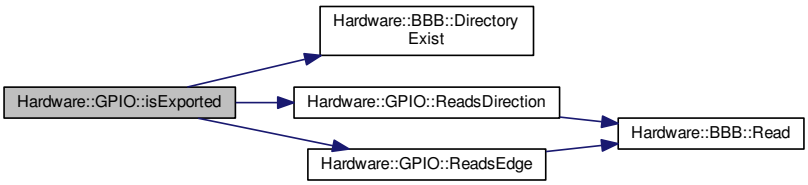
6.31.4.5 bool Hardware::GPIO::isExported (int *number*, Direction & *dir*, Edge & *edge*) [private]

Definition at line 89 of file GPIO.cpp.

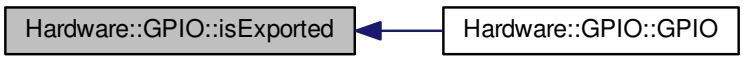
References Hardware::BBB::DirectoryExist(), gpiopath, ReadsDirection(), and ReadsEdge().

Referenced by GPIO().

Here is the call graph for this function:



Here is the caller graph for this function:



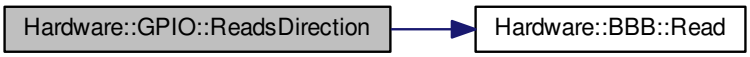
6.31.4.6 GPIO::Direction Hardware::GPIO::ReadsDirection (const string & *gpiopath*) [private]

Definition at line 205 of file GPIO.cpp.

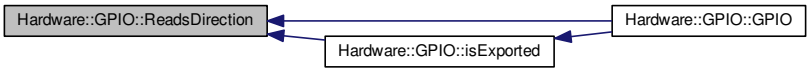
References DIRECTION, Input, Output, and Hardware::BBB::Read().

Referenced by GPIO(), and isExported().

Here is the call graph for this function:



Here is the caller graph for this function:



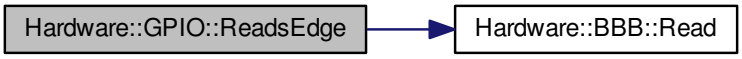
6.31.4.7 GPIO::Edge Hardware::GPIO::ReadsEdge (const string & *gpiopath*) [private]

Definition at line 224 of file GPIO.cpp.

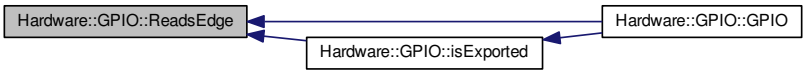
References Both, EDGE, Falling, None, Hardware::BBB::Read(), and Rising.

Referenced by GPIO(), and isExported().

Here is the call graph for this function:



Here is the caller graph for this function:



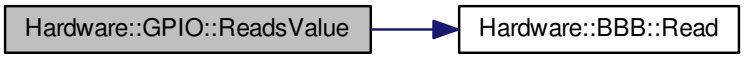
6.31.4.8 GPIO::Value Hardware::GPIO::ReadsValue (const string & *gpiopath*) [private]

Definition at line 256 of file GPIO.cpp.

References Hardware::BBB::Read(), and VALUE.

Referenced by GetValue().

Here is the call graph for this function:



Here is the caller graph for this function:

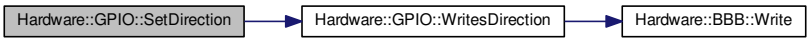
6.31.4.9 void Hardware::GPIO::SetDirection (Direction *direction*)

Definition at line 78 of file GPIO.cpp.

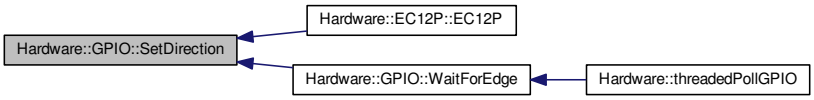
References direction, gpiopath, and WritesDirection().

Referenced by Hardware::EC12P::EC12P(), and WaitForEdge().

Here is the call graph for this function:



Here is the caller graph for this function:



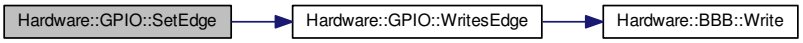
6.31.4.10 void Hardware::GPIO::SetEdge (Edge *edge*)

Definition at line 84 of file GPIO.cpp.

References edge, gpiopath, and WritesEdge().

Referenced by Hardware::EC12P::EC12P().

Here is the call graph for this function:



Here is the caller graph for this function:

6.31.4.11 void Hardware::GPIO::SetValue (GPIO::Value *value*)

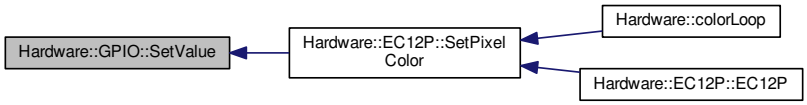
Definition at line 75 of file GPIO.cpp.

References gpiopath, and WritesValue().

Referenced by Hardware::EC12P::SetPixelColor().

Here is the call graph for this function:

Here is the caller graph for this function:

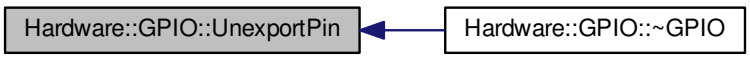


6.31.4.12 bool Hardware::GPIO::UnexportPin (int *number*) [private]

Definition at line 201 of file GPIO.cpp.

Referenced by ~GPIO().

Here is the caller graph for this function:



6.31.4.13 int Hardware::GPIO::WaitForEdge ( )

Definition at line 37 of file GPIO.cpp.

References direction, gpiopath, Input, Output, SetDirection(), and VALUE.

Referenced by Hardware::threadedPollGPIO().

Here is the call graph for this function:

Here is the caller graph for this function:

6.31.4.14 int Hardware::GPIO::WaitForEdge (CallbackType *callback*)

Definition at line 26 of file GPIO.cpp.

References Hardware::BBB::callbackFunction, Hardware::BBB::thread, threadedPollGPIO, and Hardware::BBB::threadRunning.

6.31.4.15 void Hardware::GPIO::WaitForEdgeCancel ( ) [inline]

Definition at line 35 of file GPIO.h.

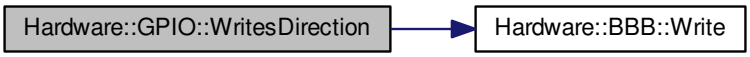
6.31.4.16 void Hardware::GPIO::WritesDirection (const string & *gpiopath*, Direction *direction*) [private]

Definition at line 213 of file GPIO.cpp.

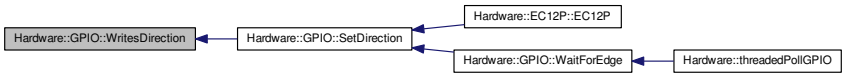
References DIRECTION, Input, Output, and Hardware::BBB::Write().

Referenced by SetDirection().

Here is the call graph for this function:



Here is the caller graph for this function:



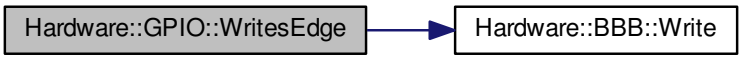
6.31.4.17 void Hardware::GPIO::WritesEdge (const string & *gpiopath*, Edge *edge*) [private]

Definition at line 237 of file GPIO.cpp.

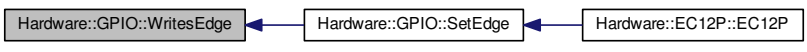
References Both, EDGE, Falling, None, Rising, and Hardware::BBB::Write().

Referenced by SetEdge().

Here is the call graph for this function:



Here is the caller graph for this function:



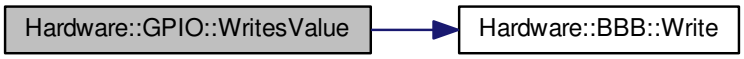
6.31.4.18 void Hardware::GPIO::WritesValue (const string & *gpiopath*, Value *value*) [private]

Definition at line 262 of file GPIO.cpp.

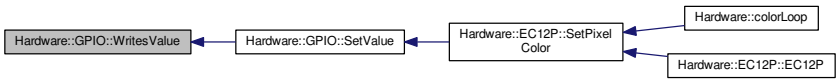
References VALUE, and Hardware::BBB::Write().

Referenced by SetValue().

Here is the call graph for this function:



Here is the caller graph for this function:



6.31.5 Friends And Related Function Documentation

6.31.5.1 void* threadedPollGPIO (void * *value*) [friend]

Definition at line 266 of file GPIO.cpp.

Referenced by WaitForEdge().

6.31.6 Member Data Documentation

6.31.6.1 Direction Hardware::GPIO::direction [private]

Definition at line 51 of file GPIO.h.

Referenced by GetDirection(), GPIO(), SetDirection(), and WaitForEdge().

6.31.6.2 Edge Hardware::GPIO::edge [private]

Definition at line 52 of file GPIO.h.

Referenced by GetEdge(), GPIO(), and SetEdge().

6.31.6.3 string Hardware::GPIO::gpiopath [private]

Definition at line 50 of file GPIO.h.

Referenced by GetValue(), GPIO(), isExported(), SetDirection(), SetEdge(), SetValue(), and WaitForEdge().

6.31.6.4 int Hardware::GPIO::number

Definition at line 31 of file GPIO.h.

Referenced by GPIO(), and ~GPIO().

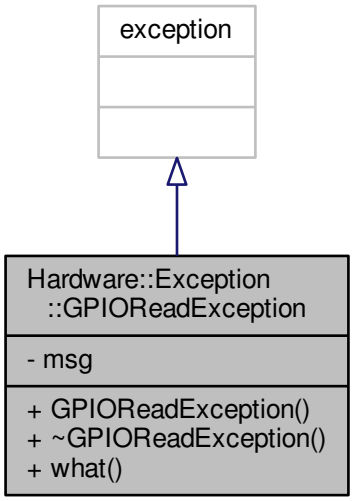
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/GPIO.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/GPIO.cpp

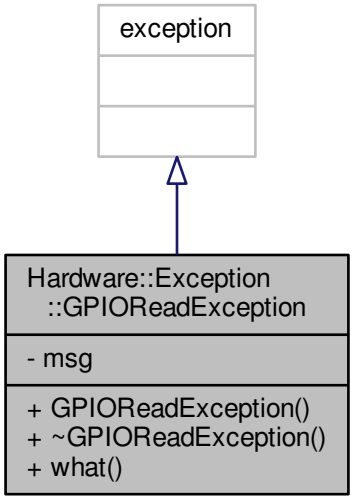
6.32 Hardware::Exception::GPIOReadException Class Reference

`#include <GPIOReadException.h>`

Inheritance diagram for Hardware::Exception::GPIOReadException:



Collaboration diagram for Hardware::Exception::GPIOReadException:



Public Member Functions

- GPIOReadException (string m="Can't read GPIO data!")
- ~GPIOReadException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.32.1 Detailed Description

Definition at line 17 of file GPIOReadException.h.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 Hardware::Exception::GPIOReadException::GPIOReadException (string *m* = "Can't read GPIO data!") [inline]

Definition at line 19 of file GPIOReadException.h.

6.32.2.2 Hardware::Exception::GPIOReadException::~GPIOReadException ( ) [inline]

Definition at line 20 of file GPIOReadException.h.

6.32.3 Member Function Documentation

6.32.3.1 const char* Hardware::Exception::GPIOReadException::what ( ) const [inline]

Definition at line 21 of file GPIOReadException.h.

6.32.4 Member Data Documentation

6.32.4.1 string Hardware::Exception::GPIOReadException::msg [private]

Definition at line 21 of file GPIOReadException.h.

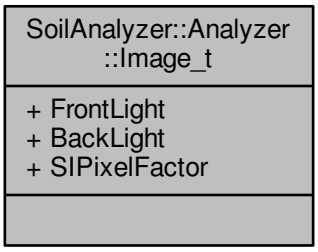
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/GPIOReadException.h

6.33 SoilAnalyzer::Analyzer::Image_t Struct Reference

`#include <analyzer.h>`

Collaboration diagram for SoilAnalyzer::Analyzer::Image_t:



Public Attributes

- cv::Mat FrontLight
- cv::Mat BackLight
- float SIPixelFactor = 0.0111915

6.33.1 Detailed Description

Definition at line 39 of file analyzer.h.

6.33.2 Member Data Documentation

6.33.2.1 cv::Mat SoilAnalyzer::Analyzer::Image_t::BackLight

Definition at line 41 of file analyzer.h.

Referenced by VSAMainWindow::TakeSnapShots().

6.33.2.2 cv::Mat SoilAnalyzer::Analyzer::Image_t::FrontLight

Definition at line 40 of file analyzer.h.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), and VSAMainWindow::TakeSnapShots().

6.33.2.3 float SoilAnalyzer::Analyzer::Image_t::SIPixelFactor = 0.0111915

Definition at line 42 of file analyzer.h.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), and VSAMainWindow::TakeSnapShots().

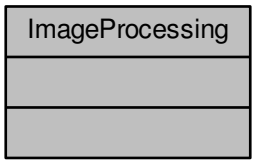
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/analyzer.h

6.34 ImageProcessing Class Reference

Core class of all the image classes Core class of all the image classes with a few commonly shared functions and variables.

Collaboration diagram for ImageProcessing:



6.34.1 Detailed Description

Core class of all the image classes Core class of all the image classes with a few commonly shared functions and variables.

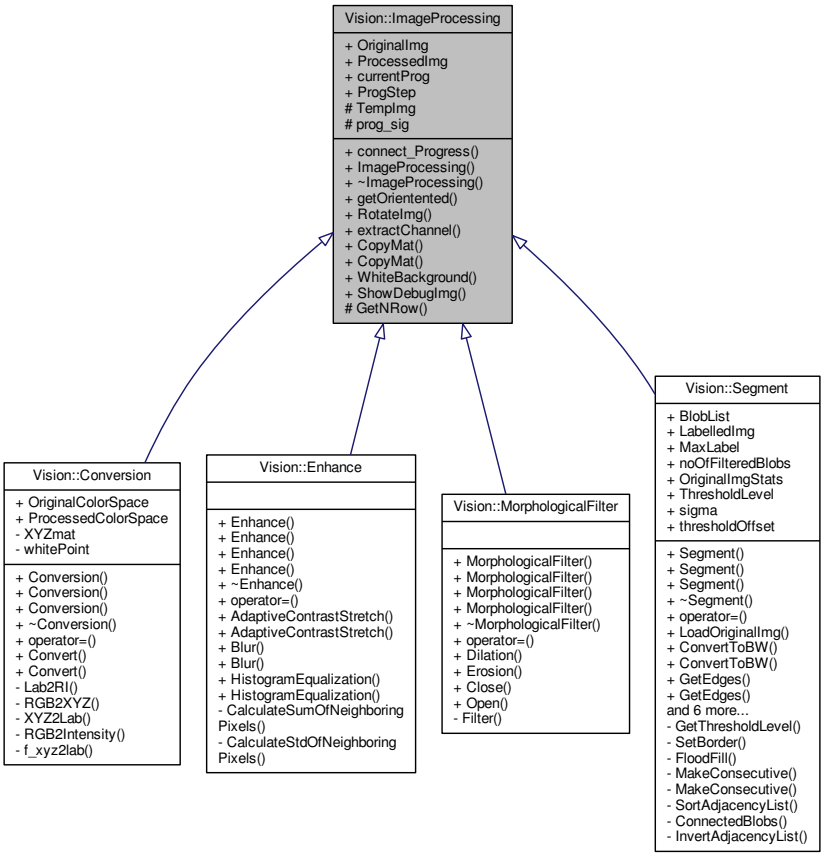
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.cpp

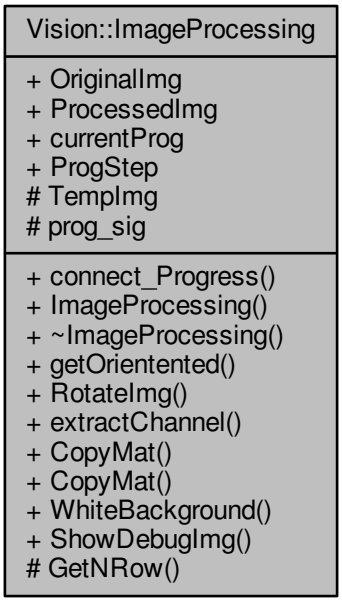
6.35 Vision::ImageProcessing Class Reference

```
#include <ImageProcessing.h>
```

Inheritance diagram for Vision::ImageProcessing:



Collaboration diagram for Vision::ImageProcessing:



Public Types

- typedef boost::signals2::signal< void(float, std::string)> Progress_t

Public Member Functions

- boost::signals2::connection connect_Progress (const Progress_t::slot_type &subscriber)
- ImageProcessing ()
- ~ImageProcessing ()

Static Public Member Functions

- static void getOrientented (Mat &BW, cv::Point_< double > &centroid, double &theta, double &eccentricty)
- static void RotateImg (Mat &src, Mat &dst, double &theta, cv::Point_< double > &Centroid, Rect &ROI)
- static std::vector< Mat > extractChannel (const Mat &src)
- template<typename T1 , typename T2 >
  static Mat CopyMat (const Mat &src, T1 *LUT, int cvType)
- template<typename T1 >
  static Mat CopyMat (const Mat &src, const Mat &mask, int cvType)
- static cv::Mat WhiteBackground (const cv::Mat &src)
- template<typename T1 >
  static void ShowDebugImg (cv::Mat img, T1 maxVal, std::string windowName, bool scale=true)

Public Attributes

- Mat OriginalImg
- Mat ProcessedImg
- double currentProg = 0.
- double ProgStep = 0.

Protected Member Functions

- uchar * GetNRow (int nData, int hKsize, int nCols, uint32_t totalRows)

Protected Attributes

- Mat TempImg
- Progress_t prog_sig

6.35.1 Detailed Description

Definition at line 48 of file ImageProcessing.h.

6.35.2 Member Typedef Documentation

6.35.2.1 typedef boost::signals2::signal<void(float, std::string)> Vision::ImageProcessing::Progress_t

Definition at line 50 of file ImageProcessing.h.

6.35.3 Constructor & Destructor Documentation

6.35.3.1 ImageProcessing::ImageProcessing ( )

Constructor of the core class

Definition at line 17 of file ImageProcessing.cpp.

6.35.3.2 ImageProcessing::~ImageProcessing ( )

De-constructor of the core class

Definition at line 20 of file ImageProcessing.cpp.

6.35.4 Member Function Documentation

6.35.4.1 boost::signals2::connection ImageProcessing::connect_Progress (const Progress_t::slot_type & *subscriber*)

Definition at line 130 of file ImageProcessing.cpp.

References prog_sig.

6.35.4.2 template<typename T1 , typename T2 > static Mat Vision::ImageProcessing::CopyMat (const Mat & *src*, T1 * *LUT*, int *cvType*) [inline], [static]

Copy a matrix to a new matrix with a LUT mask

Parameters

src	the source image
*LUT	type T with a LUT to filter out unwanted pixel values
cvType	an in where you can pas CV_UC8C1 etc.

Returns

The new matrix

Definition at line 82 of file ImageProcessing.h.

6.35.4.3 template<typename T1 > static Mat Vision::ImageProcessing::CopyMat (const Mat & *src*, const Mat & *mask*, int *cvType*) [inline], [static]

Copy a matrix to a new matrix with a mask

Parameters

src	the source image
*LUT	type T with a LUT to filter out unwanted pixel values
cvType	an in where you can pas CV_UC8C1 etc.

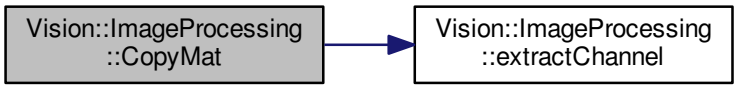
Returns

The new matrix

Definition at line 121 of file ImageProcessing.h.

References extractChannel().

Here is the call graph for this function:

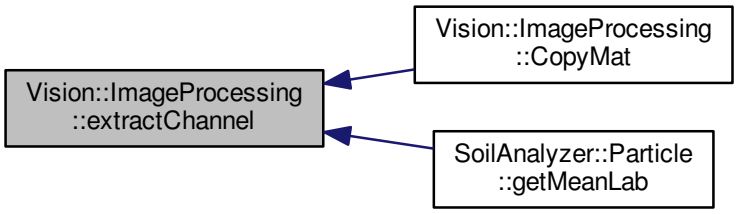


6.35.4.4 std::vector< Mat > ImageProcessing::extractChannel (const Mat & *src*) [static]

Definition at line 42 of file ImageProcessing.cpp.

Referenced by CopyMat(), and SoilAnalyzer::Particle::getMeanLab().

Here is the caller graph for this function:



6.35.4.5 uchar * ImageProcessing::GetNRow (int *nData*, int *hKsize*, int *nCols*, uint32_t *totalRows*) [protected]

Create a LUT indicating which iteration variable i is the end of an row

Parameters

nData	an int indicating total pixels
hKsize	int half the size of the kernel, if any. which acts as an offset from the border pixels
nCols	int number of columns in a row

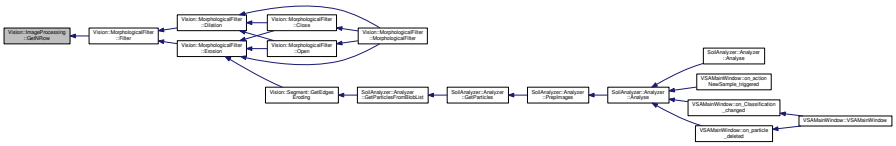
Returns

array of uchars where a zero is a middle column and a 1 indicates an end of an row minus the offset from half the kernel size

Definition at line 30 of file ImageProcessing.cpp.

Referenced by Vision::MorphologicalFilter::Filter().

Here is the caller graph for this function:

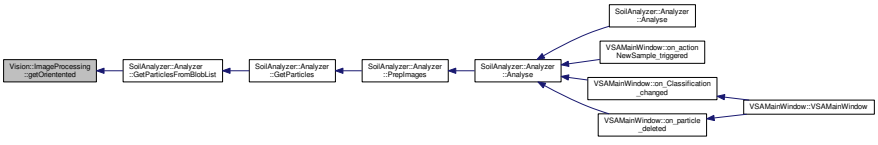


6.35.4.6 void ImageProcessing::getOrientented (Mat & *BW*, cv::Point_< double > & *centroid*, double & *theta*, double & *eccentricty*) [static]

Definition at line 48 of file ImageProcessing.cpp.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList().

Here is the caller graph for this function:

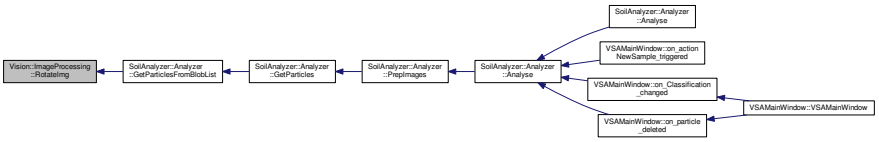


6.35.4.7 void ImageProcessing::RotateImg (Mat & *src*, Mat & *dst*, double & *theta*, cv::Point_< double > & *Centroid*, Rect & *ROI*) [static]

Definition at line 70 of file ImageProcessing.cpp.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList().

Here is the caller graph for this function:



6.35.4.8 template<typename T1 > static void Vision::ImageProcessing::ShowDebugImg (cv::Mat *img,* T1 *maxVal,* std::string *windowName,* bool *scale =* `true`) [inline], [static]

Definition at line 156 of file ImageProcessing.h.

6.35.4.9 static cv::Mat Vision::ImageProcessing::WhiteBackground (const cv::Mat & *src*) [inline], [static]

Definition at line 149 of file ImageProcessing.h.

6.35.5 Member Data Documentation

6.35.5.1 double Vision::ImageProcessing::currentProg = 0.

Definition at line 70 of file ImageProcessing.h.

Referenced by Vision::Conversion::Convert().

6.35.5.2 Mat Vision::ImageProcessing::OriginalImg

Definition at line 63 of file ImageProcessing.h.

Referenced by Vision::Segment::ConnectedBlobs(), Vision::Conversion::Conversion(), Vision::Conversion::Convert(), Vision::Segment::ConvertToBW(), Vision::Enhance::Enhance(), Vision::Segment::FillHoles(), Vision::MorphologicalFilter::Filter(), Vision::Segment::GetBlobList(), Vision::Segment::GetEdges(), Vision::Segment::GetEdgesEroding(), Vision::Segment::GetThresholdLevel(), Vision::Enhance::HistogramEqualization(), Vision::Segment::LabelBlobs(), Vision::Segment::LoadOriginalImg(), Vision::MorphologicalFilter::MorphologicalFilter(), Vision::MorphologicalFilter::operator=(), Vision::Conversion::operator=(), Vision::Enhance::operator=(), Vision::Segment::operator=(), Vision::Segment::RemoveBorderBlobs(), Vision::Conversion::RGB2XYZ(), Vision::Segment::Segment(), Vision::Segment::SetBorder(), and Vision::Segment::Threshold().

6.35.5.3 Mat Vision::ImageProcessing::ProcessedImg

Definition at line 64 of file ImageProcessing.h.

Referenced by SoilAnalyzer::Analyzer::CalibrateSI(), Vision::Conversion::Conversion(), Vision::Conversion::Convert(), Vision::Segment::ConvertToBW(), Vision::Enhance::Enhance(), Vision::Segment::FillHoles(), Vision::MorphologicalFilter::Filter(), SoilAnalyzer::Analyzer::GetBW(), Vision::Segment::GetEdges(), Vision::Segment::GetEdgesEroding(), SoilAnalyzer::Particle::getLabImg(), SoilAnalyzer::Particle::GetMeanRI(), SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), Vision::Enhance::HistogramEqualization(), Vision::Segment::LabelBlobs(), Vision::Segment::LoadOriginalImg(), Vision::MorphologicalFilter::MorphologicalFilter(), Vision::MorphologicalFilter::operator=(), Vision::Conversion::operator=(), Vision::Enhance::operator=(), Vision::Segment::operator=(), Vision::Segment::RemoveBorderBlobs(), Vision::Segment::Segment(), and Vision::Segment::Threshold().

6.35.5.4 Progress_t Vision::ImageProcessing::prog_sig [protected]

Definition at line 58 of file ImageProcessing.h.

Referenced by connect_Progress(), and Vision::Conversion::Convert().

6.35.5.5 double Vision::ImageProcessing::ProgStep = 0.

Definition at line 71 of file ImageProcessing.h.

Referenced by Vision::Conversion::Convert().

6.35.5.6 Mat Vision::ImageProcessing::TempImg [protected]

Definition at line 56 of file ImageProcessing.h.

Referenced by Vision::Conversion::Conversion(), Vision::Enhance::Enhance(), Vision::Segment::FillHoles(), Vision::Segment::GetEdgesEroding(), Vision::Segment::LabelBlobs(), Vision::MorphologicalFilter::MorphologicalFilter(), Vision::MorphologicalFilter::operator=(), Vision::Conversion::operator=(), Vision::Enhance::operator=(), Vision::Segment::operator=(), Vision::Segment::RemoveBorderBlobs(), and Vision::Segment::Segment().

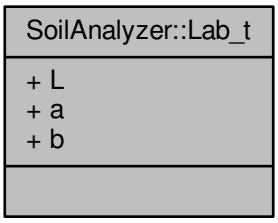
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/ImageProcessing.cpp

6.36 SoilAnalyzer::Lab_t Struct Reference

`#include <soilanalyzertypes.h>`

Collaboration diagram for SoilAnalyzer::Lab_t:



Public Attributes

- float L
- float a
- float b

6.36.1 Detailed Description

Definition at line 10 of file soilanalyzertypes.h.

6.36.2 Member Data Documentation

6.36.2.1 float SoilAnalyzer::Lab_t::a

Definition at line 12 of file soilanalyzertypes.h.

Referenced by SoilAnalyzer::Particle::getMeanLab(), boost::serialization::serialize(), and SoilAnalyzer::Particle::serialize().

6.36.2.2 float SoilAnalyzer::Lab_t::b

Definition at line 13 of file soilanalyzertypes.h.

Referenced by SoilAnalyzer::Particle::getMeanLab(), boost::serialization::serialize(), and SoilAnalyzer::Particle::serialize().

6.36.2.3 float SoilAnalyzer::Lab_t::L

Definition at line 11 of file soilanalyzertypes.h.

Referenced by SoilAnalyzer::Particle::getMeanLab(), boost::serialization::serialize(), and SoilAnalyzer::Particle::serialize().

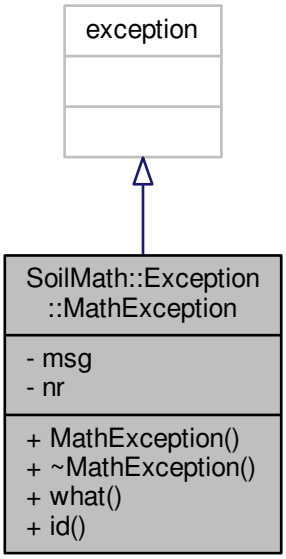
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilanalyzertypes.h

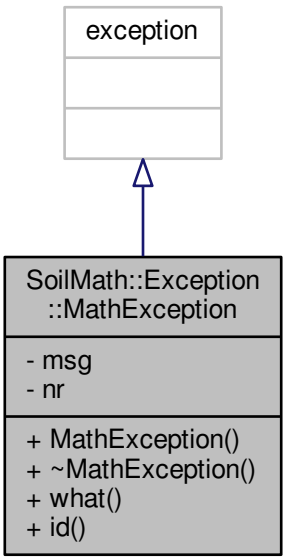
6.37 SoilMath::Exception::MathException Class Reference

`#include <MathException.h>`

Inheritance diagram for SoilMath::Exception::MathException:



Collaboration diagram for SoilMath::Exception::MathException:



Public Member Functions

- MathException (std::string m=EXCEPTION_MATH, int n=EXCEPTION_MATH_NR)
- ~MathException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT
- const int * id () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- std::string msg
- int nr

6.37.1 Detailed Description

Definition at line 28 of file MathException.h.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 SoilMath::Exception::MathException::MathException (std::string *m* = EXCEPTION_MATH, int *n* = EXCEPTION_MATH_NR) [inline]

Definition at line 30 of file MathException.h.

6.37.2.2 SoilMath::Exception::MathException::~MathException ( ) [inline]

Definition at line 32 of file MathException.h.

6.37.3 Member Function Documentation

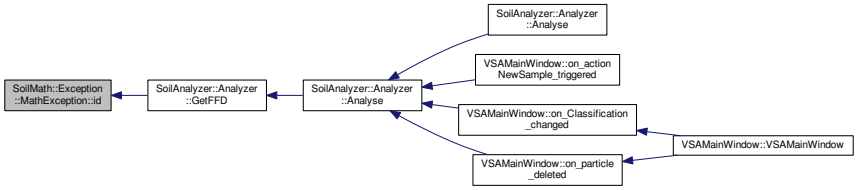
6.37.3.1 const int* SoilMath::Exception::MathException::id ( ) const [inline]

Definition at line 34 of file MathException.h.

References nr.

Referenced by SoilAnalyzer::Analyzer::GetFFD().

Here is the caller graph for this function:



6.37.3.2 const char* SoilMath::Exception::MathException::what ( ) const [inline]

Definition at line 33 of file MathException.h.

References msg.

6.37.4 Member Data Documentation

6.37.4.1 std::string SoilMath::Exception::MathException::msg [private]

Definition at line 37 of file MathException.h.

Referenced by what().

6.37.4.2 int SoilMath::Exception::MathException::nr [private]

Definition at line 38 of file MathException.h.

Referenced by id().

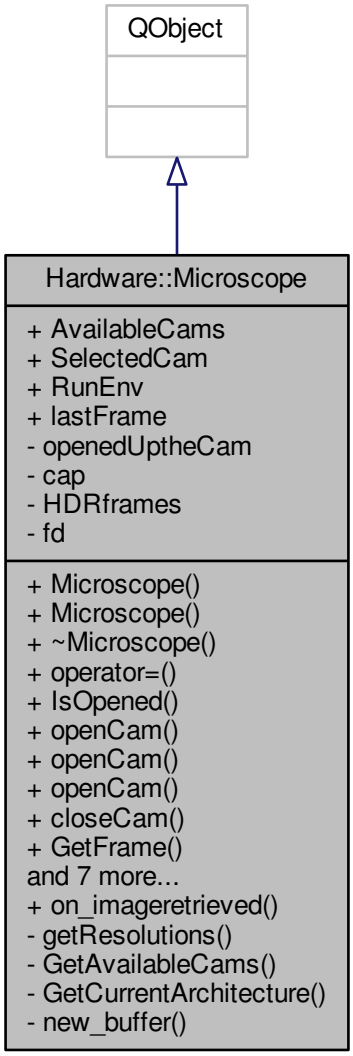
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/MathException.h

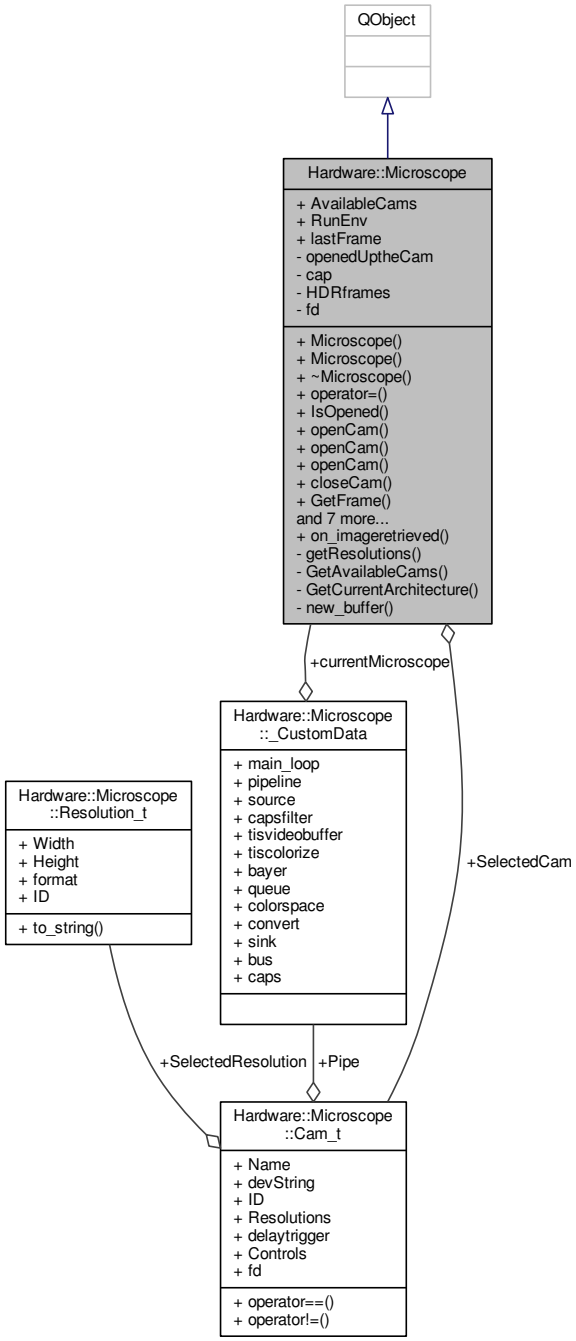
6.38 Hardware::Microscope Class Reference

`#include <Microscope.h>`

Inheritance diagram for Hardware::Microscope:



Collaboration diagram for Hardware::Microscope:



Classes

- struct _CustomData
- struct Cam_t
- struct Control_t
- struct Resolution_t

Public Types

- enum Arch { ARM, X64 }
- enum PixelFormat { YUYV, MJPG, GREY }
- typedef std::vector< Control_t > Controls_t
- typedef struct Hardware::Microscope::_CustomData CustomData

Public Slots

- void on_imageretrieved ()

Signals

- void imageretrieved ()

Public Member Functions

- Microscope ()
- Microscope (const Microscope &rhs)
- ~Microscope ()
- Microscope operator= (Microscope const &rhs)
- bool IsOpened ()
- bool openCam (Cam_t *cam)
- bool openCam (int &cam)
- bool openCam (std::string &cam)
- bool closeCam (Cam_t *cam)
- void GetFrame (cv::Mat &dst)
- void GetGstreamFrame (cv::Mat &dst)
- void GetHDRFrame (cv::Mat &dst, uint32_t noframes=3)
- Control_t * GetControl (const std::string name)
- void SetControl (Control_t *control)
- Cam_t * FindCam (std::string cam)
- Cam_t * FindCam (int cam)
- void SendImageRetrieved ()

Public Attributes

- std::vector< Cam_t > AvailableCams
- Cam_t * SelectedCam = nullptr
- Arch RunEnv
- cv::Mat lastFrame

Private Member Functions

- void getResolutions (Cam_t &currentCam, int FormatType)
- std::vector< Cam_t > GetAvailableCams ()
- Arch GetCurrentArchitecture ()

Static Private Member Functions

- static void new_buffer (GstElement *sink, CustomData *data)

Private Attributes

- bool openedUptheCam = false
- cv::VideoCapture * cap = nullptr
- std::vector< cv::Mat > HDRframes
- int fd

6.38.1 Detailed Description

Definition at line 49 of file Microscope.h.

6.38.2 Member Typedef Documentation

6.38.2.1 typedef std::vector<Control_t> Hardware::Microscope::Controls_t

Definition at line 103 of file Microscope.h.

6.38.2.2 typedef struct Hardware::Microscope::_CustomData Hardware::Microscope::CustomData

6.38.3 Member Enumeration Documentation

6.38.3.1 enum Hardware::Microscope::Arch

Enumerator

ARM

X64

Definition at line 53 of file Microscope.h.

6.38.3.2 enum Hardware::Microscope::PixelFormat

Enumerator

YUYV

MJPG

GREY

Definition at line 55 of file Microscope.h.

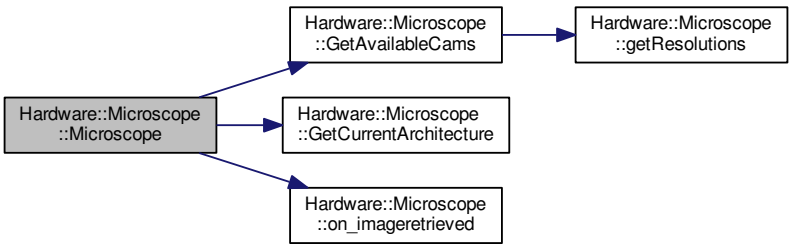
6.38.4 Constructor & Destructor Documentation

6.38.4.1 Microscope::Microscope ( )

Definition at line 12 of file Microscope.cpp.

References AvailableCams, GetAvailableCams(), GetCurrentArchitecture(), imageretrieved(), on_imageretrieved(), and RunEnv.

Here is the call graph for this function:

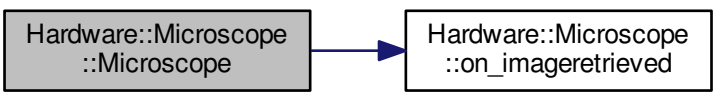


6.38.4.2 Microscope::Microscope (const Microscope & *rhs*)

Definition at line 21 of file Microscope.cpp.

References AvailableCams, cap, fd, HDRframes, imageretrieved(), on_imageretrieved(), RunEnv, and SelectedCam.

Here is the call graph for this function:



6.38.4.3 Microscope::~Microscope ( )

Definition at line 32 of file Microscope.cpp.

References cap.

6.38.5 Member Function Documentation

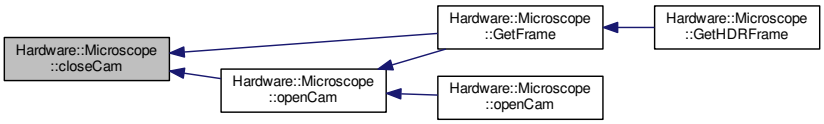
6.38.5.1 bool Microscope::closeCam (Cam_t * *cam*)

Definition at line 311 of file Microscope.cpp.

References openedUptheCam, Hardware::Microscope::Cam_t::Pipe, and Hardware::Microscope::_CustomData::pipeline.

Referenced by GetFrame(), and openCam().

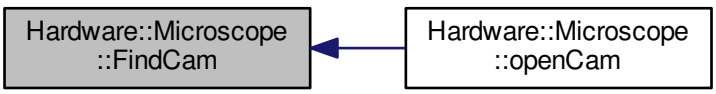
Here is the caller graph for this function:



6.38.5.2 Cam_t* Hardware::Microscope::FindCam (std::string *cam*)

Referenced by openCam().

Here is the caller graph for this function:



6.38.5.3 Microscope::Cam_t * Microscope::FindCam (int *cam*)

Definition at line 293 of file Microscope.cpp.

References AvailableCams.

6.38.5.4 std::vector< Microscope::Cam_t > Microscope::GetAvailableCams ( ) [private]

Definition at line 47 of file Microscope.cpp.

References Hardware::Microscope::Cam_t::Controls, Hardware::Microscope::Control_t::current_value, Hardware::Microscope::Control_t::default_value, Hardware::Microscope::Cam_t::devString, EXCEPTION_NOCAMS, EXCEPTION_NOCAMS_NR, EXCEPTION_QUERY, EXCEPTION_QUERY_NR, Hardware::Microscope::Cam_t::fd, getResolutions(), Hardware::Microscope::Control_t::ID, Hardware::Microscope::Cam_t::ID, Hardware::Microscope::Control_t::maximum, Hardware::Microscope::Control_t::minimum, Hardware::Microscope::Control_t::name, Hardware::Microscope::Cam_t::Name, and Hardware::Microscope::Control_t::step.

Referenced by Microscope().

Here is the call graph for this function:

Here is the caller graph for this function:

6.38.5.5 Microscope::Control_t * Microscope::GetControl (const std::string *name*)

Definition at line 364 of file Microscope.cpp.

References Hardware::Microscope::Cam_t::Controls, and SelectedCam.

Referenced by GetHDRFrame().

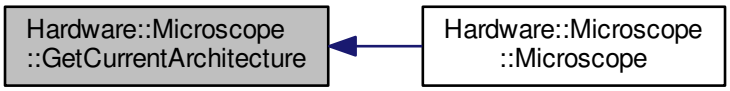
Here is the caller graph for this function:

6.38.5.6 Microscope::Arch Microscope::GetCurrentArchitecture ( ) [private]

Definition at line 34 of file Microscope.cpp.

Referenced by Microscope().

Here is the caller graph for this function:



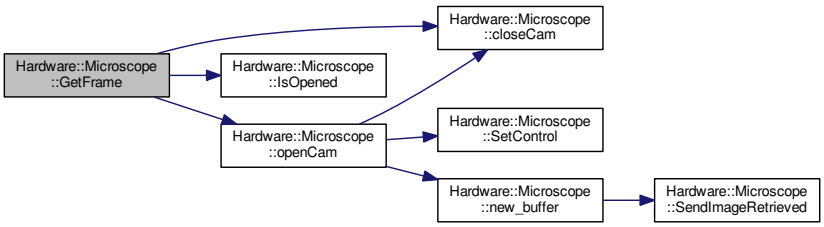
6.38.5.7 void Microscope::GetFrame (cv::Mat & *dst*)

Definition at line 319 of file Microscope.cpp.

References closeCam(), imageretrieved(), IsOpened(), lastFrame, openCam(), Hardware::Microscope::Cam_t::Pipe, Hardware::Microscope::↩
_CustomData::pipeline, and SelectedCam.

Referenced by GetHDRFrame().

Here is the call graph for this function:



Here is the caller graph for this function:

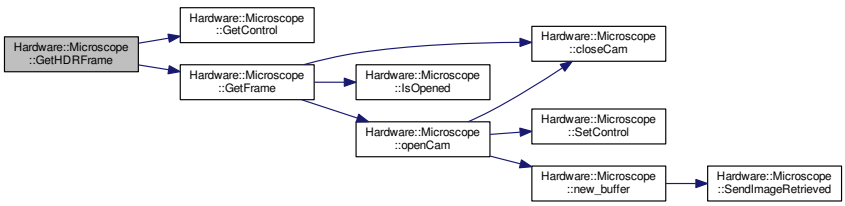
6.38.5.8 void Hardware::Microscope::GetGstreamFrame (cv::Mat & *dst*)

6.38.5.9 void Microscope::GetHDRFrame (cv::Mat & *dst,* uint32_t *noframes* = 3)

Definition at line 333 of file Microscope.cpp.

References Hardware::Microscope::Control_t::current_value, GetControl(), GetFrame(), HDRframes, Hardware::Microscope::Control_t↩
::maximum, and Hardware::Microscope::Control_t::minimum.

Here is the call graph for this function:



6.38.5.10 void Microscope::getResolutions (Cam_t & *currentCam,* int *FormatType*) [private]

Definition at line 123 of file Microscope.cpp.

References Hardware::Microscope::Cam_t::fd, Hardware::Microscope::Resolution_t::format, Hardware::Microscope::Resolution_t::Height, Hardware::Microscope::Resolution_t::ID, Hardware::Microscope::Cam_t::Resolutions, and Hardware::Microscope::Resolution_t::Width.

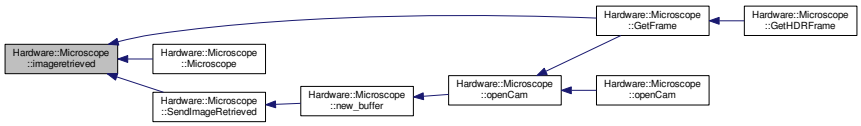
Referenced by GetAvailableCams().

Here is the caller graph for this function:

6.38.5.11 void Hardware::Microscope::imageretrieved ( ) [signal]

Referenced by GetFrame(), Microscope(), and SendImageRetrieved().

Here is the caller graph for this function:



6.38.5.12 bool Microscope::IsOpened ( )

Definition at line 165 of file Microscope.cpp.

References openedUptheCam.

Referenced by GetFrame().

Here is the caller graph for this function:

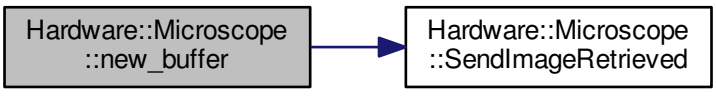
6.38.5.13 void Microscope::new_buffer (GstElement * *sink*, CustomData * *data*) [static], [private]

Definition at line 413 of file Microscope.cpp.

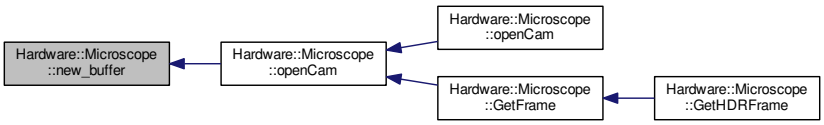
References Hardware::Microscope::_CustomData::currentMicroscope, Hardware::Microscope::Resolution_t::Height, lastFrame, SelectedCam, Hardware::Microscope::Cam_t::SelectedResolution, SendImageRetrieved(), and Hardware::Microscope::Resolution_t::Width.

Referenced by openCam().

Here is the call graph for this function:



Here is the caller graph for this function:



6.38.5.14 void Microscope::on_imageretrieved ( ) [slot]

Definition at line 331 of file Microscope.cpp.

Referenced by Microscope().

Here is the caller graph for this function:

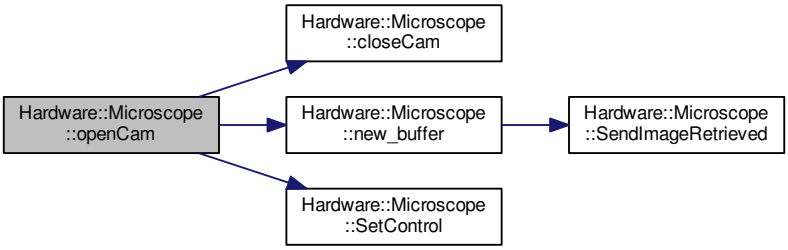
6.38.5.15 bool Microscope::openCam (Cam_t * *cam*)

Definition at line 167 of file Microscope.cpp.

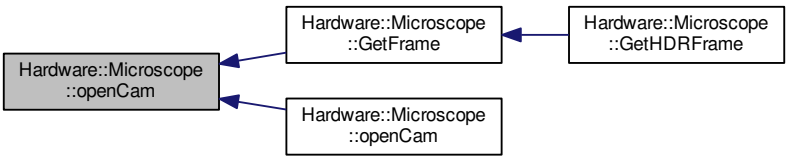
References AvailableCams, Hardware::Microscope::_CustomData::bayer, Hardware::Microscope::_CustomData::bus, Hardware::Microscope::_CustomData::caps, Hardware::Microscope::_CustomData::capsfilter, closeCam(), Hardware::Microscope::_CustomData::colorspace, Hardware::Microscope::Cam_t::Controls, Hardware::Microscope::_CustomData::convert, Hardware::Microscope::_CustomData::currentMicroscope, Hardware::Microscope::Cam_t::devString, EXCEPTION_GSTREAM_ELEM_EXCEPTION, EXCEPTION_GSTREAM_ELEM_EXCEPTION_NR, EXCEPTION_GSTREAM_INIT_EXCEPTION, EXCEPTION_GSTREAM_INIT_EXCEPTION_NR, Hardware::Microscope::Resolution_t::format, Hardware::Microscope::Resolution_t::Height, Hardware::Microscope::Cam_t::Name, new_buffer(), openedUptheCam, Hardware::Microscope::Cam_t::Pipe, Hardware::Microscope::_CustomData::pipeline, Hardware::Microscope::_CustomData::queue, SelectedCam, Hardware::Microscope::Cam_t::SelectedResolution, SetControl(), Hardware::Microscope::_CustomData::sink, Hardware::Microscope::_CustomData::source, Hardware::Microscope::_CustomData::tiscolorize, Hardware::Microscope::_CustomData::tisvideobuffer, and Hardware::Microscope::Resolution_t::Width.

Referenced by GetFrame(), and openCam().

Here is the call graph for this function:



Here is the caller graph for this function:

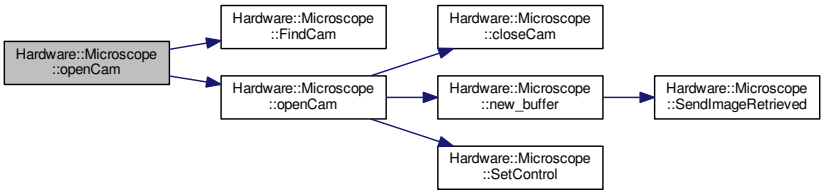


6.38.5.16 bool Microscope::openCam (int & *cam*)

Definition at line 291 of file Microscope.cpp.

References FindCam(), and openCam().

Here is the call graph for this function:

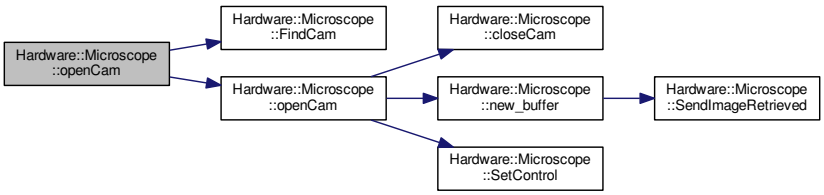


6.38.5.17 bool Microscope::openCam (std::string & *cam*)

Definition at line 289 of file Microscope.cpp.

References FindCam(), and openCam().

Here is the call graph for this function:



6.38.5.18 Microscope Hardware::Microscope::operator= (Microscope const & *rhs*)

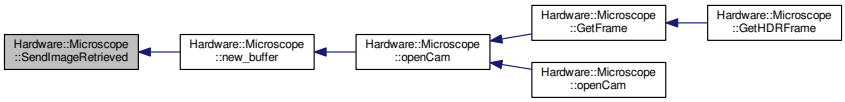
6.38.5.19 void Microscope::SendImageRetrieved ( )

Definition at line 411 of file Microscope.cpp.

References imageretrieved().

Referenced by new_buffer().

Here is the caller graph for this function:



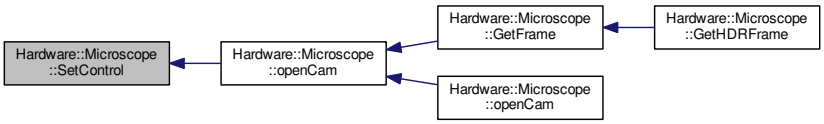
6.38.5.20 void Microscope::SetControl (Control_t * *control*)

Definition at line 374 of file Microscope.cpp.

References Hardware::Microscope::Control_t::current_value, Hardware::Microscope::Cam_t::devString, EXCEPTION_CTRL_NOT_FOUND, EXCEPTION_CTRL_NOT_FOUND_NR, EXCEPTION_NOCAMS, EXCEPTION_NOCAMS_NR, EXCEPTION_QUERY, EXCEPTION_QUERY_NR, Hardware::Microscope::Cam_t::fd, Hardware::Microscope::Control_t::ID, and SelectedCam.

Referenced by openCam().

Here is the caller graph for this function:



6.38.6 Member Data Documentation

6.38.6.1 std::vector<Cam_t> Hardware::Microscope::AvailableCams

Definition at line 148 of file Microscope.h.

Referenced by FindCam(), Microscope(), and openCam().

6.38.6.2 cv::VideoCapture* Hardware::Microscope::cap = nullptr [private]

Definition at line 189 of file Microscope.h.

Referenced by Microscope(), and ~Microscope().

6.38.6.3 int Hardware::Microscope::fd [private]

Definition at line 195 of file Microscope.h.

Referenced by Microscope().

6.38.6.4 std::vector<cv::Mat> Hardware::Microscope::HDRframes [private]

Definition at line 191 of file Microscope.h.

Referenced by GetHDRFrame(), and Microscope().

6.38.6.5 cv::Mat Hardware::Microscope::lastFrame

Definition at line 175 of file Microscope.h.

Referenced by GetFrame(), and new_buffer().

6.38.6.6 bool Hardware::Microscope::openedUptheCam = false [private]

Definition at line 188 of file Microscope.h.

Referenced by closeCam(), IsOpened(), and openCam().

6.38.6.7 Arch Hardware::Microscope::RunEnv

Definition at line 150 of file Microscope.h.

Referenced by Microscope().

6.38.6.8 Cam_t* Hardware::Microscope::SelectedCam = nullptr

Definition at line 149 of file Microscope.h.

Referenced by GetControl(), GetFrame(), Microscope(), new_buffer(), openCam(), and SetControl().

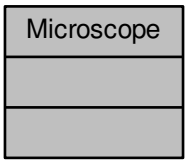
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.cpp

6.39 Microscope Class Reference

`#include <Microscope.h>`

Collaboration diagram for Microscope:



6.39.1 Detailed Description

Interaction with the microscope

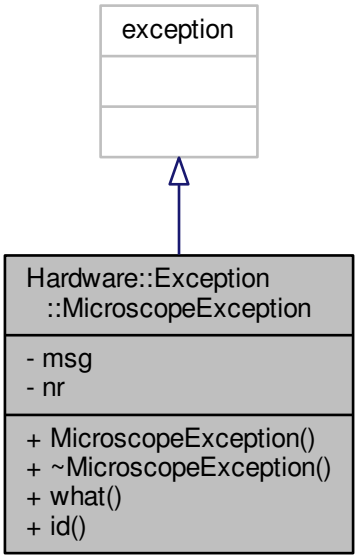
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h

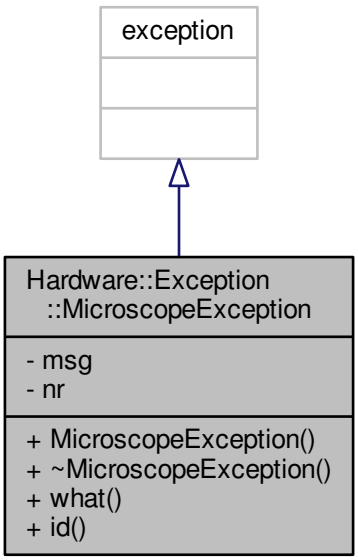
6.40 Hardware::Exception::MicroscopeException Class Reference

`#include <MicroscopeNotFoundException.h>`

Inheritance diagram for Hardware::Exception::MicroscopeException:



Collaboration diagram for Hardware::Exception::MicroscopeException:



Public Member Functions

- MicroscopeException (string m=EXCEPTION_OPENCAM, int n=EXCEPTION_OPENCAM_NR)
- ~MicroscopeException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT
- const int * id () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg
- int nr

6.40.1 Detailed Description

Definition at line 35 of file MicroscopeNotFoundException.h.

6.40.2 Constructor & Destructor Documentation

6.40.2.1 Hardware::Exception::MicroscopeException::MicroscopeException (string *m* = EXCEPTION_OPENCAM, int *n* = EXCEPTION_OPENCAM_NR) [inline]

Definition at line 37 of file MicroscopeNotFoundException.h.

6.40.2.2 Hardware::Exception::MicroscopeException::~MicroscopeException ( ) [inline]

Definition at line 39 of file MicroscopeNotFoundException.h.

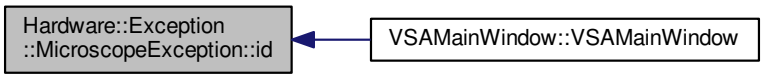
6.40.3 Member Function Documentation

6.40.3.1 const int* Hardware::Exception::MicroscopeException::id ( ) const [inline]

Definition at line 41 of file MicroscopeNotFoundException.h.

Referenced by VSAMainWindow::VSAMainWindow().

Here is the caller graph for this function:



6.40.3.2 const char* Hardware::Exception::MicroscopeException::what ( ) const [inline]

Definition at line 40 of file MicroscopeNotFoundException.h.

6.40.4 Member Data Documentation

6.40.4.1 string Hardware::Exception::MicroscopeException::msg [private]

Definition at line 44 of file MicroscopeNotFoundException.h.

6.40.4.2 int Hardware::Exception::MicroscopeException::nr [private]

Definition at line 45 of file MicroscopeNotFoundException.h.

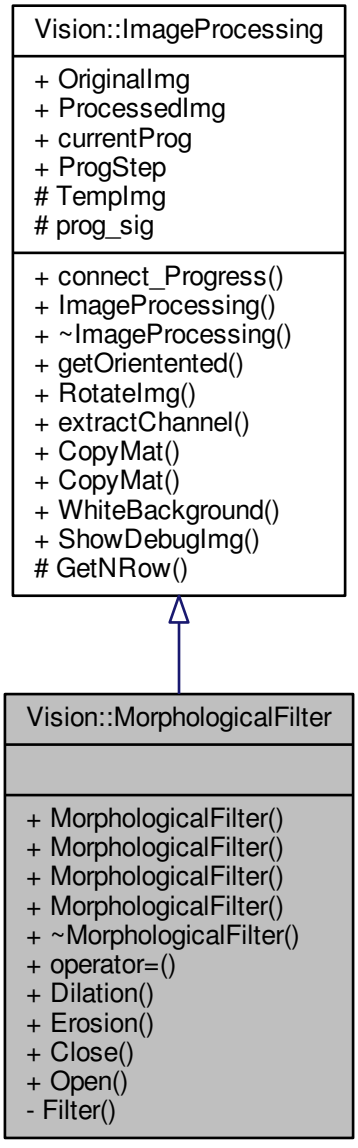
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/MicroscopeNotFoundException.h

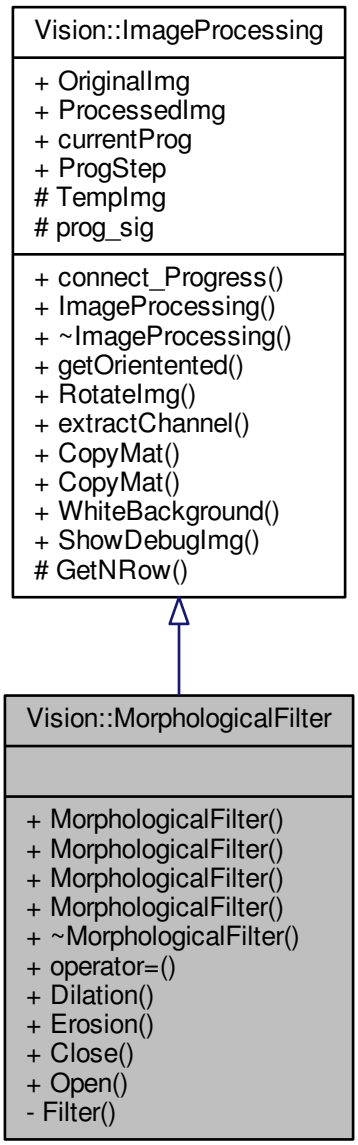
6.41 Vision::MorphologicalFilter Class Reference

`#include <MorphologicalFilter.h>`

Inheritance diagram for Vision::MorphologicalFilter:



Collaboration diagram for Vision::MorphologicalFilter:



Public Types

- enum FilterType {
  OPEN, CLOSE, ERODE, DILATE,
  NONE }

Public Member Functions

- MorphologicalFilter ()
- MorphologicalFilter (FilterType filtertype)
- MorphologicalFilter (const Mat &src, FilterType filtertype=FilterType::NONE)
- MorphologicalFilter (const MorphologicalFilter &rhs)
- ~MorphologicalFilter ()
- MorphologicalFilter & operator= (MorphologicalFilter &rhs)
- void Dilation (const Mat &mask, bool chain=false)
- void Erosion (const Mat &mask, bool chain=false)
- void Close (const Mat &mask, bool chain=false)
- void Open (const Mat &mask, bool chain=false)

Private Member Functions

- void Filter (const Mat &mask, bool chain, uchar startVal, uchar newVal, uchar switchVal)

Additional Inherited Members

6.41.1 Detailed Description

Definition at line 14 of file MorphologicalFilter.h.

6.41.2 Member Enumeration Documentation

6.41.2.1 enum Vision::MorphologicalFilter::FilterType

Enumerator

OPEN
CLOSE
ERODE
DILATE
NONE

Definition at line 16 of file MorphologicalFilter.h.

6.41.3 Constructor & Destructor Documentation

6.41.3.1 Vision::MorphologicalFilter::MorphologicalFilter ( )

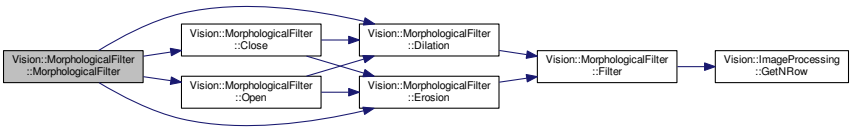
Definition at line 11 of file MorphologicalFilter.cpp.

6.41.3.2 Vision::MorphologicalFilter::MorphologicalFilter (FilterType *filtertype*)

Definition at line 13 of file MorphologicalFilter.cpp.

References Close(), Dilation(), Erosion(), Open(), and Vision::ImageProcessing::OriginalImg.

Here is the call graph for this function:

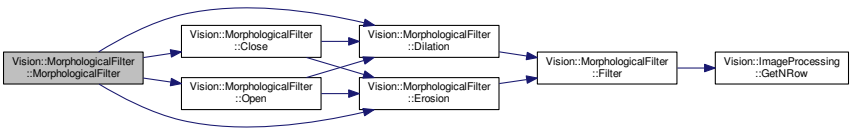


6.41.3.3 Vision::MorphologicalFilter::MorphologicalFilter (const Mat & *src,* FilterType *filtertype* = `FilterType::NONE`)

Definition at line 32 of file MorphologicalFilter.cpp.

References Close(), Dilation(), Erosion(), Open(), Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

Here is the call graph for this function:



6.41.3.4 Vision::MorphologicalFilter::MorphologicalFilter (const MorphologicalFilter & *rhs*)

Definition at line 54 of file MorphologicalFilter.cpp.

References Vision::ImageProcessing::OriginalImg, Vision::ImageProcessing::ProcessedImg, and Vision::ImageProcessing::TempImg.

6.41.3.5 Vision::MorphologicalFilter::~MorphologicalFilter ( )

Definition at line 60 of file MorphologicalFilter.cpp.

6.41.4 Member Function Documentation

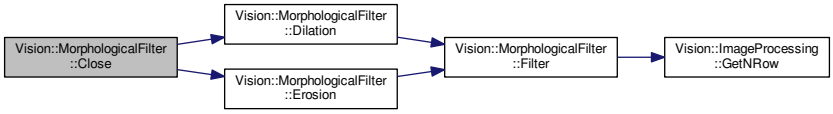
6.41.4.1 void Vision::MorphologicalFilter::Close (const Mat & *mask*, bool *chain* = `false`)

Definition at line 76 of file MorphologicalFilter.cpp.

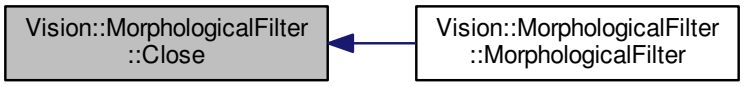
References Dilation(), and Erosion().

Referenced by MorphologicalFilter().

Here is the call graph for this function:



Here is the caller graph for this function:



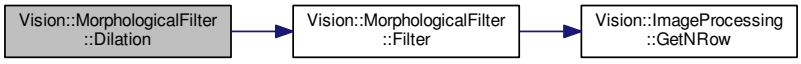
6.41.4.2 void Vision::MorphologicalFilter::Dilation (const Mat & *mask*, bool *chain* = `false`)

Definition at line 81 of file MorphologicalFilter.cpp.

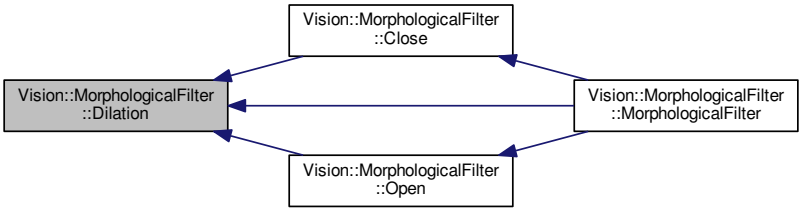
References Filter().

Referenced by Close(), MorphologicalFilter(), and Open().

Here is the call graph for this function:



Here is the caller graph for this function:



6.41.4.3 void Vision::MorphologicalFilter::Erosion (const Mat & *mask*, bool *chain* = `false`)

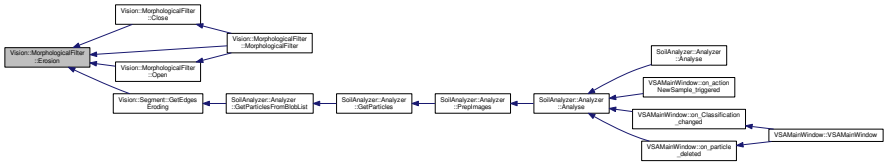
Definition at line 85 of file MorphologicalFilter.cpp.

References Filter().

Referenced by Close(), Vision::Segment::GetEdgesEroding(), MorphologicalFilter(), and Open().

Here is the call graph for this function:

Here is the caller graph for this function:



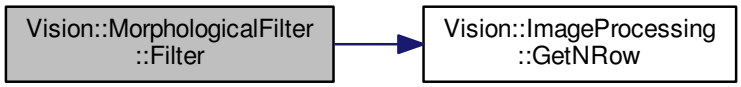
6.41.4.4 void Vision::MorphologicalFilter::Filter (const Mat & *mask*, bool *chain*, uchar *startVal*, uchar *newVal*, uchar *switchVal*) [private]

Definition at line 89 of file MorphologicalFilter.cpp.

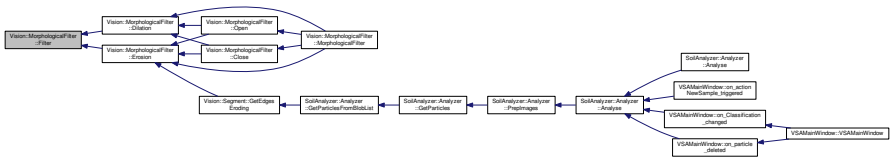
References EMPTY_CHECK, Vision::ImageProcessing::GetNRow(), Vision::ImageProcessing::OriginalImg, Vision::ImageProcessing::↩ ProcessedImg, and SHOW_DEBUG_IMG.

Referenced by Dilation(), and Erosion().

Here is the call graph for this function:



Here is the caller graph for this function:



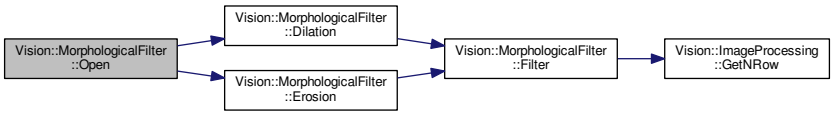
6.41.4.5 void Vision::MorphologicalFilter::Open (const Mat & *mask*, bool *chain* = `false`)

Definition at line 71 of file MorphologicalFilter.cpp.

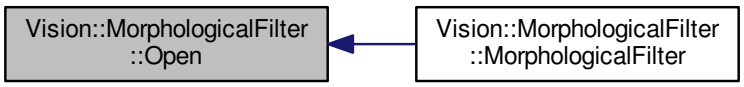
References Dilation(), and Erosion().

Referenced by MorphologicalFilter().

Here is the call graph for this function:



Here is the caller graph for this function:



6.41.4.6 MorphologicalFilter & Vision::MorphologicalFilter::operator= (MorphologicalFilter & *rhs*)

Definition at line 62 of file MorphologicalFilter.cpp.

References Vision::ImageProcessing::OriginalImg, Vision::ImageProcessing::ProcessedImg, and Vision::ImageProcessing::TempImg.

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

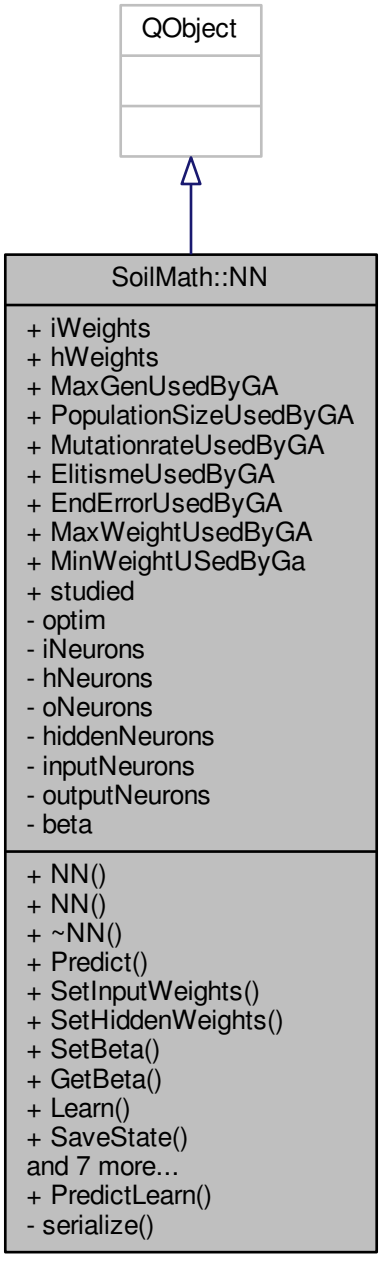
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/MorphologicalFilter.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/MorphologicalFilter.cpp

6.42 SoilMath::NN Class Reference

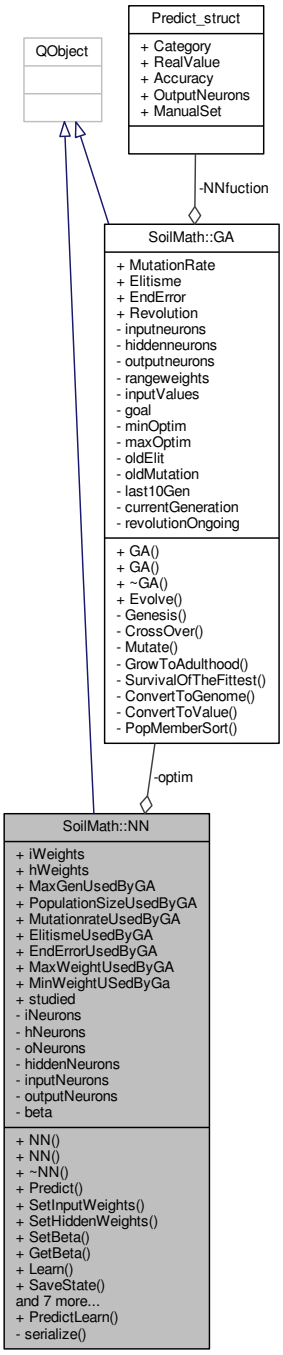
The Neural Network class.

`#include <NN.h>`

Inheritance diagram for SoilMath::NN:



Collaboration diagram for SoilMath::NN:



Signals

- void learnErrorUpdate (double newError)

Public Member Functions

- NN (uint32_t inputneurons, uint32_t hiddenneurons, uint32_t outputneurons)

  NN constructor for the Neural Net.
- NN ()

  NN constructor for the Neural Net.
- virtual ~NN ()

  ~NN virtual deconstructor for the Neural Net
- Predict_t Predict (ComplexVect_t input)

  Predict The prediction function.
- void SetInputWeights (Weight_t value)

SetInputWeights a function to set the input weights.

- void SetHiddenWeights (Weight_t value)

  SetHiddenWeights a function to set the hidden weights.
- void SetBeta (float value)

  SetBeta a function to set the beta value.
- float GetBeta ()
- void Learn (InputLearnVector_t input, OutputLearnVector_t cat, uint32_t noOfDescriptorsUsed)

  Learn the learning function.
- void SaveState (std::string filename)

  SaveState Serialize and save the values of the Neural Net to disk.
- void LoadState (std::string filename)

  LoadState Loads the previouse saved Neural Net from disk.
- uint32_t GetInputNeurons ()
- void SetInputNeurons (uint32_t value)
- uint32_t GetHiddenNeurons ()
- void SetHiddenNeurons (uint32_t value)
- uint32_t GetOutputNeurons ()
- void SetOutputNeurons (uint32_t value)

Static Public Member Functions

- static Predict_t PredictLearn (ComplexVect_t input, Weight_t inputweights, Weight_t hiddenweights, uint32_t inputneurons, uint32_t hiddenneurons, uint32_t outputneurons)

  PredictLearn a static function used in learning of the weights.

Public Attributes

- Weight_t iWeights
- Weight_t hWeights
- uint32_t MaxGenUsedByGA = 200
- uint32_t PopulationSizeUsedByGA = 30
- float MutationrateUsedByGA = 0.075f
- uint32_t ElitismeUsedByGA = 4
- float EndErrorUsedByGA = 0.001
- float MaxWeightUsedByGA = 50
- float MinWeightUSedByGa = -50
- bool studied

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  serialization function

Private Attributes

- GA * optim = nullptr
- std::vector< float > iNeurons
- std::vector< float > hNeurons
- std::vector< float > oNeurons
- uint32_t hiddenNeurons = 50
- uint32_t inputNeurons = 20
- uint32_t outputNeurons = 18
- float beta

Friends

- class boost::serialization::access

6.42.1 Detailed Description

The Neural Network class.

This class is used to make prediction on large data set. Using self learning algoritmes

Definition at line 33 of file NN.h.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 SoilMath::NN::NN (uint32_t *inputneurons*, uint32_t *hiddenneurons*, uint32_t *outputneurons*)

NN constructor for the Neural Net.

Parameters

inputneurons	number of input neurons
hiddenneurons	number of hidden neurons
outputneurons	number of output neurons

Definition at line 14 of file NN.cpp.

6.42.2.2 SoilMath::NN::NN ( )

NN constructor for the Neural Net.

Definition at line 12 of file NN.cpp.

6.42.2.3 SoilMath::NN::~NN ( ) [virtual]

~NN virtual deconstructor for the Neural Net

Definition at line 27 of file NN.cpp.

6.42.3 Member Function Documentation

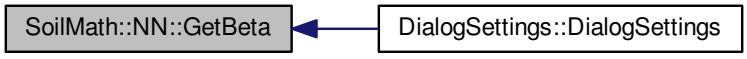
6.42.3.1 float SoilMath::NN::GetBeta ( ) [inline]

Definition at line 102 of file NN.h.

References beta.

Referenced by DialogSettings::DialogSettings().

Here is the caller graph for this function:



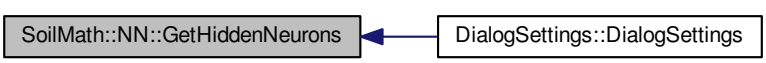
6.42.3.2 uint32_t SoilMath::NN::GetHiddenNeurons ( ) [inline]

Definition at line 143 of file NN.h.

References hiddenNeurons.

Referenced by DialogSettings::DialogSettings().

Here is the caller graph for this function:



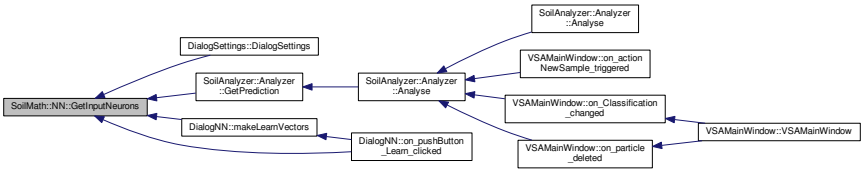
6.42.3.3 uint32_t SoilMath::NN::GetInputNeurons ( ) [inline]

Definition at line 140 of file NN.h.

References inputNeurons.

Referenced by DialogSettings::DialogSettings(), SoilAnalyzer::Analyzer::GetPrediction(), DialogNN::makeLearnVectors(), and DialogNN::on_↩ pushButton_Learn_clicked().

Here is the caller graph for this function:



6.42.3.4 uint32_t SoilMath::NN::GetOutputNeurons ( ) [inline]

Definition at line 146 of file NN.h.

References outputNeurons.

Referenced by DialogSettings::DialogSettings(), and DialogNN::makeLearnVectors().

Here is the caller graph for this function:

6.42.3.5 void SoilMath::NN::Learn (InputLearnVector_t *input*, OutputLearnVector_t *cat*, uint32_t *noOfDescriptorsUsed*)

Learn the learning function.

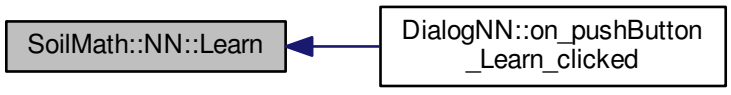
Parameters

input	a vector of vectors with complex input values
cat	a vector of vectors with the know output values
noOfDescriptors↩ Used	the total number of descriptos which should be used

Definition at line 113 of file NN.cpp.

Referenced by DialogNN::on_pushButton_Learn_clicked().

Here is the caller graph for this function:



6.42.3.6 void SoilMath::NN::learnErrorUpdate (double *newError*) [signal]

6.42.3.7 void SoilMath::NN::LoadState (std::string *filename*)

LoadState Loads the previouse saved Neural Net from disk.

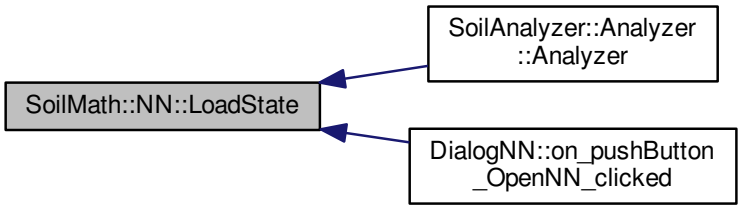
Parameters

filename	a string indicating the file location and name

Definition at line 34 of file NN.cpp.

Referenced by SoilAnalyzer::Analyzer::Analyzer(), and DialogNN::on_pushButton_OpenNN_clicked().

Here is the caller graph for this function:



6.42.3.8 Predict_t SoilMath::NN::Predict (ComplexVect_t *input*)

Predict The prediction function.

In this function the neural net is setup and the input which are the complex values descriping the contour in the frequency domein serve as input. The absolute value of these im. number because I'm not interrested in the orrientation of the particle but more in the degree of variations.

Parameters

input	vector of complex input values, these're the Fourier descriptors

Returns

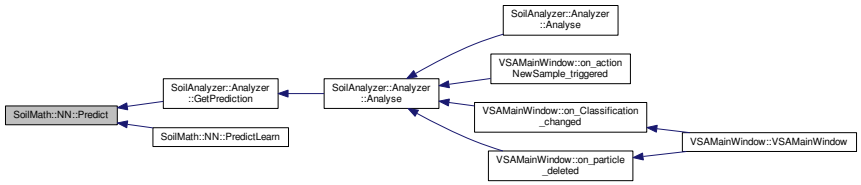
a real valued vector of the output neurons

Definition at line 56 of file NN.cpp.

References EXCEPTION_NEURAL_NET_NOT_STUDIED, EXCEPTION_NEURAL_NET_NOT_STUDIED_NR, EXCEPTION_SIZE_OF_INP↩UT_NEURONS, EXCEPTION_SIZE_OF_INPUT_NEURONS_NR, Predict_struct::ManualSet, and Predict_struct::OutputNeurons.

Referenced by SoilAnalyzer::Analyzer::GetPrediction(), and PredictLearn().

Here is the caller graph for this function:



6.42.3.9 Predict_t SoilMath::NN::PredictLearn (ComplexVect_t *input*, Weight_t *inputweights*, Weight_t *hiddenweights*, uint32_t *inputneurons*, uint32_t *hiddenneurons*, uint32_t *outputneurons*) [static]

PredictLearn a static function used in learning of the weights.

It starts a new Neural Network object and passes all the paramaters in to this newly created object. After this the predict function is called and the value is returned. This work around was needed to pass the neural network to the Genetic Algorithm class.

Parameters

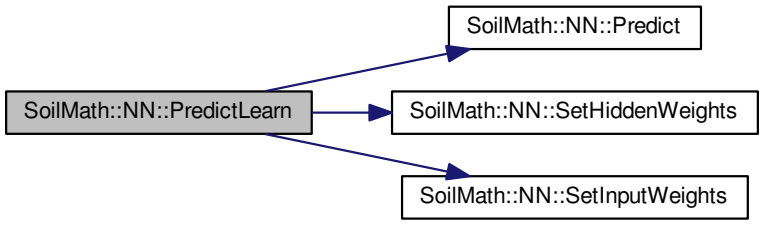
input	a complex vector of input values
inputweights	the input weights
hiddenweights	the hidden weights
inputneurons	the input neurons
hiddenneurons	the hidden neurons
outputneurons	the output neurons

Returns

Definition at line 46 of file NN.cpp.

References Predict(), SetHiddenWeights(), SetInputWeights(), and studied.

Here is the call graph for this function:



6.42.3.10 void SoilMath::NN::SaveState (std::string *filename*)

SaveState Serialize and save the values of the Neural Net to disk.

Save the Neural Net in XML valued text file to disk so that a object can be reconstructed on a latter stadia.

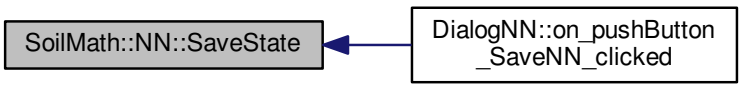
Parameters

filename	a string indicating the file location and name

Definition at line 40 of file NN.cpp.

Referenced by DialogNN::on_pushButton_SaveNN_clicked().

Here is the caller graph for this function:



6.42.3.11 template<class Archive > void SoilMath::NN::serialize (Archive & *ar*, const unsigned int *version*) [inline], [private]

serialization function

Parameters

ar	the object
version	the version of the class

Definition at line 181 of file NN.h.

6.42.3.12 void SoilMath::NN::SetBeta (float *value*) [inline]

SetBeta a function to set the beta value.

Parameters

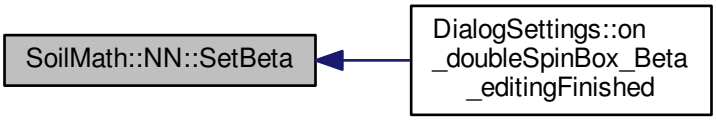
value	a floating value ussualy between 0.5 and 1.5

Definition at line 101 of file NN.h.

References beta.

Referenced by DialogSettings::on_doubleSpinBox_Beta_editingFinished().

Here is the caller graph for this function:

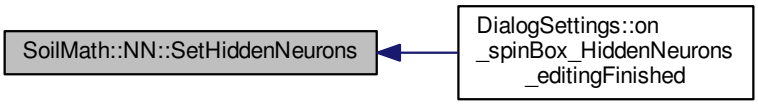


6.42.3.13 void SoilMath::NN::SetHiddenNeurons (uint32_t *value*)

Definition at line 150 of file NN.cpp.

Referenced by DialogSettings::on_spinBox_HiddenNeurons_editingFinished().

Here is the caller graph for this function:



6.42.3.14 void SoilMath::NN::SetHiddenWeights (Weight_t *value*) [inline]

SetHiddenWeights a function to set the hidden weights.

Parameters

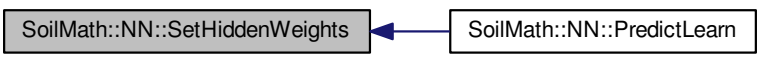
value	the real valued vector with the values

Definition at line 95 of file NN.h.

References hWeights.

Referenced by PredictLearn().

Here is the caller graph for this function:

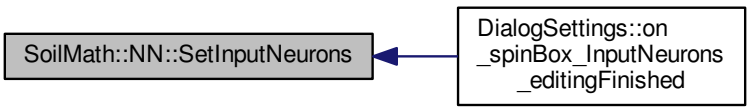


6.42.3.15 void SoilMath::NN::SetInputNeurons (uint32_t *value*)

Definition at line 141 of file NN.cpp.

Referenced by DialogSettings::on_spinBox_InputNeurons_editingFinished().

Here is the caller graph for this function:



6.42.3.16 void SoilMath::NN::SetInputWeights (Weight_t *value*) [inline]

SetInputWeights a function to set the input weights.

Parameters

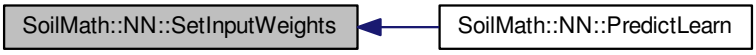
value	the real valued vector with the values

Definition at line 89 of file NN.h.

References iWeights.

Referenced by PredictLearn().

Here is the caller graph for this function:

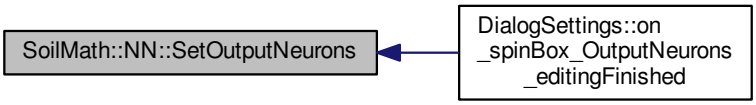


6.42.3.17 void SoilMath::NN::SetOutputNeurons (uint32_t *value*)

Definition at line 159 of file NN.cpp.

Referenced by DialogSettings::on_spinBox_OutputNeurons_editingFinished().

Here is the caller graph for this function:



6.42.4 Friends And Related Function Documentation

6.42.4.1 friend class boost::serialization::access [friend]

a private friend class so the serialization can access all the needed functions

Definition at line 172 of file NN.h.

6.42.5 Member Data Documentation

6.42.5.1 float SoilMath::NN::beta [private]

the beta value, this indicates the steepness of the sigmoid function

Definition at line 169 of file NN.h.

Referenced by GetBeta(), and SetBeta().

6.42.5.2 uint32_t SoilMath::NN::ElitismeUsedByGA = 4

Definition at line 135 of file NN.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_spinBox_Elitisme_editingFinished().

6.42.5.3 float SoilMath::NN::EndErrorUsedByGA = 0.001

Definition at line 136 of file NN.h.

Referenced by DialogSettings::DialogSettings(), DialogNN::on_actionAbort_triggered(), DialogSettings::on_doubleSpinBox_endError_editing↩Finished(), and DialogNN::setupErrorGraph().

6.42.5.4 uint32_t SoilMath::NN::hiddenNeurons = 50 [private]

number of hidden neurons minus bias

Definition at line 166 of file NN.h.

Referenced by GetHiddenNeurons().

6.42.5.5 std::vector<float> SoilMath::NN::hNeurons [private]

a vector of hidden values, the bias is included and is the first value

Definition at line 162 of file NN.h.

6.42.5.6 Weight_t SoilMath::NN::hWeights

a vector of real valued floating point hidden weight

Definition at line 130 of file NN.h.

Referenced by SetHiddenWeights().

6.42.5.7 std::vector<float> SoilMath::NN::iNeurons [private]

a vector of input values, the bias is included, the bias is included and is the first value

Definition at line 158 of file NN.h.

6.42.5.8 uint32_t SoilMath::NN::inputNeurons = 20 [private]

number of input neurons minus bias

Definition at line 167 of file NN.h.

Referenced by GetInputNeurons().

6.42.5.9 Weight_t SoilMath::NN::iWeights

a vector of real valued floating point input weights

Definition at line 129 of file NN.h.

Referenced by SetInputWeights().

6.42.5.10 uint32_t SoilMath::NN::MaxGenUsedByGA = 200

Definition at line 132 of file NN.h.

Referenced by DialogSettings::DialogSettings(), DialogSettings::on_spinBox_MaxGen_editingFinished(), and DialogNN::setupErrorGraph().

6.42.5.11 float SoilMath::NN::MaxWeightUsedByGA = 50

Definition at line 137 of file NN.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_doubleSpinBox_maxWeight_editingFinished().

6.42.5.12 float SoilMath::NN::MinWeightUSedByGa = -50

Definition at line 138 of file NN.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_doubleSpinBox_MinWeight_editingFinished().

6.42.5.13 float SoilMath::NN::MutationrateUsedByGA = 0.075f

Definition at line 134 of file NN.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_doubleSpinBox_MutationRate_editingFinished().

6.42.5.14 std::vector<float> SoilMath::NN::oNeurons [private]

a vector of output values

Definition at line 164 of file NN.h.

6.42.5.15 GA* SoilMath::NN::optim = nullptr [private]

Definition at line 157 of file NN.h.

6.42.5.16 uint32_t SoilMath::NN::outputNeurons = 18 [private]

number of output neurons

Definition at line 168 of file NN.h.

Referenced by GetOutputNeurons().

6.42.5.17 uint32_t SoilMath::NN::PopulationSizeUsedByGA = 30

Definition at line 133 of file NN.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_spinBox_PopSize_editingFinished().

6.42.5.18 bool SoilMath::NN::studied

Initial value:

```
=
    false
```

a value indicating if the weights are a results of a learning curve

Definition at line 149 of file NN.h.

Referenced by PredictLearn().

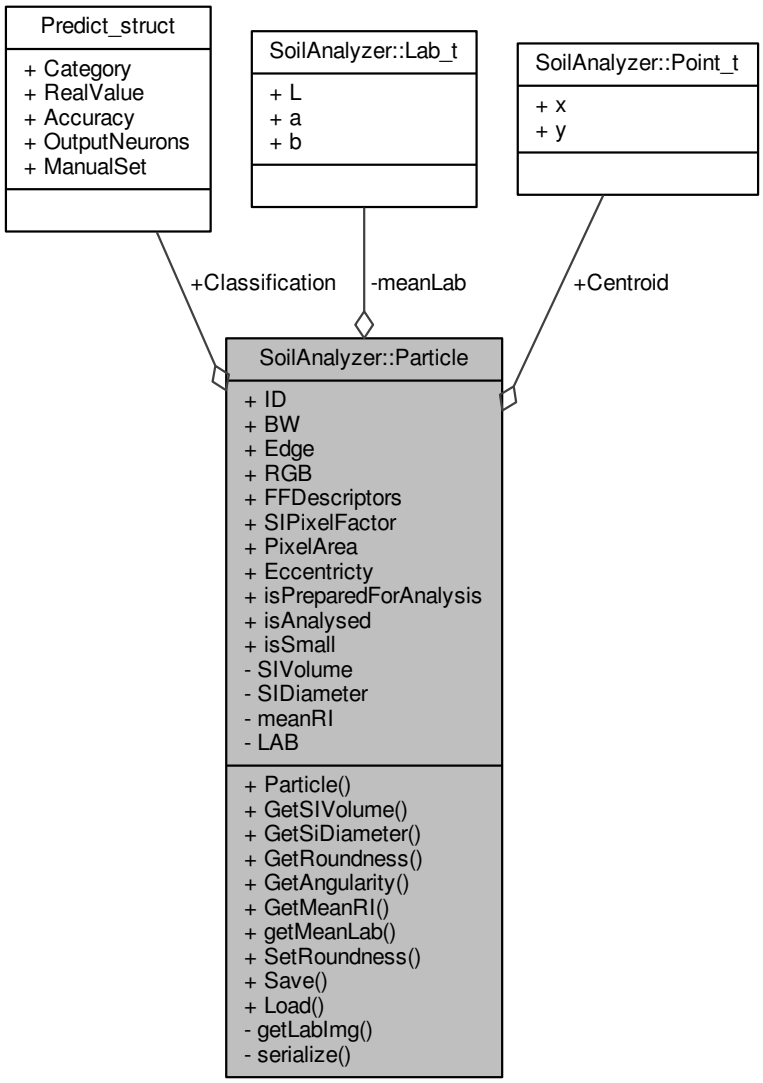
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/NN.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/NN.cpp

6.43 SoilAnalyzer::Particle Class Reference

```
#include <particle.h>
```

Collaboration diagram for SoilAnalyzer::Particle:



Public Types

- typedef std::vector< Particle > ParticleVector_t
- typedef std::vector< double > PSDVector_t
- typedef std::vector< uint8_t > ClassVector_t
- typedef std::vector< float > floatVector_t
- typedef std::vector< double > doubleVector_t

Public Member Functions

- Particle ()
- float GetSIVolume ()

  Particle::GetSIVolume.
- float GetSiDiameter ()
- uint8_t GetRoundness ()
- uint8_t GetAngularity ()
- float GetMeanRI ()
- Lab_t getMeanLab ()
- void SetRoundness ()
- void Save (const std::string &filename)

  Particle::Save.
- void Load (const std::string &filename)

  Particle::Load.

Public Attributes

- uint32_t ID
- cv::Mat BW
- cv::Mat Edge
- cv::Mat RGB
- Point_t Centroid = {0, 0}
- std::vector< Complex_t > FFDescriptors
- Predict_t Classification
- double SIPixelFactor = 0.0111915
- uint32_t PixelArea = 0
- double Eccentricty = 1
- bool isPreparedForAnalysis = false
- bool isAnalysed = false
- bool isSmall = false

Private Member Functions

- void getLabImg ()
- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

Private Attributes

- float SIVolume = 0.
- float SIDiameter = 0.
- float meanRI = 0
- Lab_t meanLab {0,0,0}
- cv::Mat LAB

Friends

- class boost::serialization::access

6.43.1 Detailed Description

Definition at line 28 of file particle.h.

6.43.2 Member Typedef Documentation

6.43.2.1 typedef std::vector<uint8_t> SoilAnalyzer::Particle::ClassVector_t

a vector used in the classification histogram

Definition at line 34 of file particle.h.

6.43.2.2 typedef std::vector<double> SoilAnalyzer::Particle::doubleVector_t

Definition at line 36 of file particle.h.

6.43.2.3 typedef std::vector<float> SoilAnalyzer::Particle::floatVector_t

Definition at line 35 of file particle.h.

6.43.2.4 typedef std::vector<Particle> SoilAnalyzer::Particle::ParticleVector_t

a vector consisting of individual particles

Definition at line 31 of file particle.h.

6.43.2.5 typedef std::vector<double> SoilAnalyzer::Particle::PSDVector_t

a vector used in the PSD

Definition at line 32 of file particle.h.

6.43.3 Constructor & Destructor Documentation

6.43.3.1 SoilAnalyzer::Particle::Particle ( )

Definition at line 13 of file particle.cpp.

6.43.4 Member Function Documentation

6.43.4.1 uint8_t SoilAnalyzer::Particle::GetAngularity ( )

Definition at line 79 of file particle.cpp.

References Predict_struct::Category, and Classification.

Referenced by SetRoundness().

Here is the caller graph for this function:

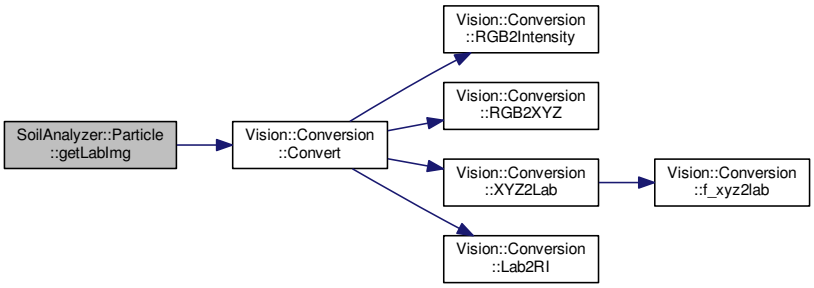
6.43.4.2 void SoilAnalyzer::Particle::getLabImg ( ) [private]

Definition at line 138 of file particle.cpp.

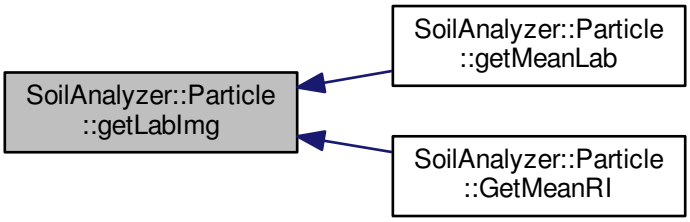
References Vision::Conversion::CIE_lab, Vision::Conversion::Convert(), LAB, Vision::ImageProcessing::ProcessedImg, Vision::Conversion::↩ RGB, and RGB.

Referenced by getMeanLab(), and GetMeanRI().

Here is the call graph for this function:



Here is the caller graph for this function:

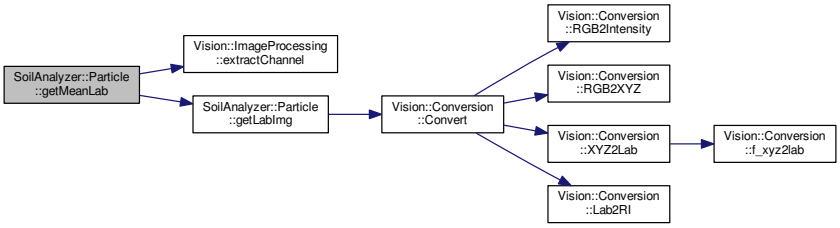


6.43.4.3 Lab_t SoilAnalyzer::Particle::getMeanLab ( )

Definition at line 96 of file particle.cpp.

References SoilAnalyzer::Lab_t::a, SoilAnalyzer::Lab_t::b, BW, EXCEPTION_NO_IMAGES_PRESENT, EXCEPTION_NO_IMAGES_PRESENT_NR, Vision::ImageProcessing::extractChannel(), getLabImg(), SoilAnalyzer::Lab_t::L, LAB, SoilMath::Stats< T1, T2, T3 >::Mean, meanLab, and RGB.

Here is the call graph for this function:

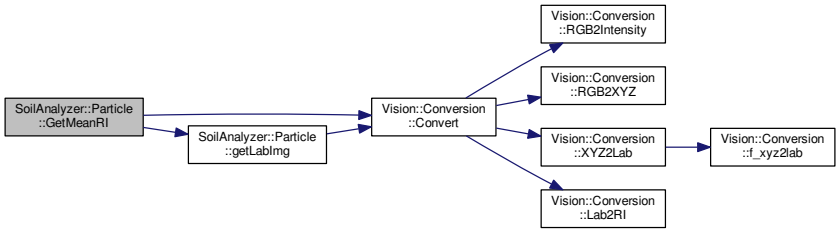


6.43.4.4 float SoilAnalyzer::Particle::GetMeanRI ( )

Definition at line 120 of file particle.cpp.

References BW, Vision::Conversion::CIE_lab, Vision::Conversion::Convert(), EXCEPTION_NO_IMAGES_PRESENT, EXCEPTION_NO_IMAGES_PRESENT_NR, getLabImg(), LAB, SoilMath::Stats< T1, T2, T3 >::Mean, meanRI, Vision::ImageProcessing::ProcessedImg, RGB, and Vision::Conversion::RI.

Here is the call graph for this function:



6.43.4.5 uint8_t SoilAnalyzer::Particle::GetRoundness ( )

Definition at line 84 of file particle.cpp.

References Predict_struct::Category, and Classification.

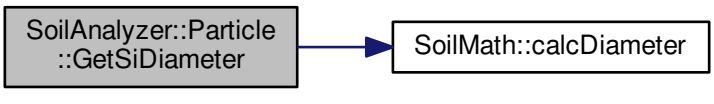
6.43.4.6 float SoilAnalyzer::Particle::GetSiDiameter ( )

Definition at line 68 of file particle.cpp.

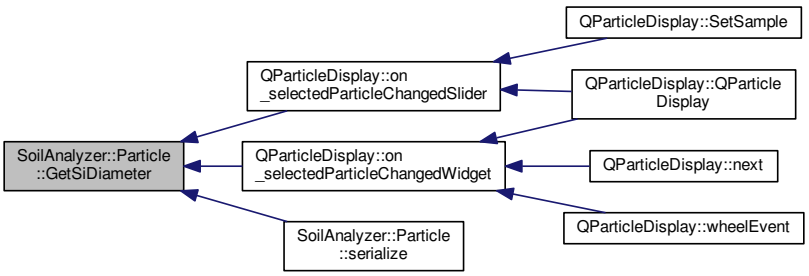
References SoilMath::calcDiameter(), Eccentricty, EXCEPTION_PARTICLE_NOT_ANALYZED, EXCEPTION_PARTICLE_NOT_ANALYZED_NR, PixelArea, SIDiameter, and SIPixelFactor.

Referenced by QParticleDisplay::on_selectedParticleChangedSlider(), QParticleDisplay::on_selectedParticleChangedWidget(), and serialize().

Here is the call graph for this function:



Here is the caller graph for this function:



6.43.4.7 float SoilAnalyzer::Particle::GetSIVolume ( )

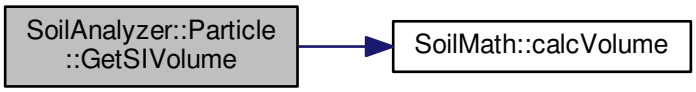
Particle::GetSIVolume.

Returns

Definition at line 57 of file particle.cpp.

References SoilMath::calcVolume(), Eccentricty, EXCEPTION_PARTICLE_NOT_ANALYZED, EXCEPTION_PARTICLE_NOT_ANALYZED_↩NR, PixelArea, SIPixelFactor, and SIVolume.

Here is the call graph for this function:



6.43.4.8 void SoilAnalyzer::Particle::Load (const std::string & *filename*)

Particle::Load.

Parameters

filename	

Definition at line 38 of file particle.cpp.

6.43.4.9 void SoilAnalyzer::Particle::Save (const std::string & *filename*)

Particle::Save.

Parameters

filename	

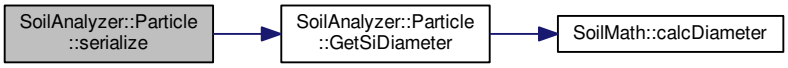
Definition at line 19 of file particle.cpp.

6.43.4.10 template<class Archive > void SoilAnalyzer::Particle::serialize (Archive & *ar,* const unsigned int *version*) [inline], [private]

Definition at line 83 of file particle.h.

References SoilAnalyzer::Lab_t::a, SoilAnalyzer::Lab_t::b, BW, Classification, Eccentricty, Edge, FFDescriptors, GetSiDiameter(), ID, is↩Analysed, isPreparedForAnalysis, isSmall, SoilAnalyzer::Lab_t::L, meanLab, meanRI, PixelArea, RGB, SIDiameter, SIPixelFactor, SIVolume, SoilAnalyzer::Point_t::x, and SoilAnalyzer::Point_t::y.

Here is the call graph for this function:



6.43.4.11 void SoilAnalyzer::Particle::SetRoundness ( )

Definition at line 89 of file particle.cpp.

References Predict_struct::Category, Classification, Eccentricty, GetAngularity(), and Predict_struct::ManualSet.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList().

Here is the call graph for this function:

Here is the caller graph for this function:

6.43.5 Friends And Related Function Documentation

6.43.5.1 friend class boost::serialization::access [friend]

Definition at line 81 of file particle.h.

6.43.6 Member Data Documentation

6.43.6.1 cv::Mat SoilAnalyzer::Particle::BW

The binary image of the particle

Definition at line 42 of file particle.h.

Referenced by QParticleDisplay::ConvertParticleToQImage(), getMeanLab(), GetMeanRI(), SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), and serialize().

6.43.6.2 Point_t SoilAnalyzer::Particle::Centroid = {0, 0}

Definition at line 46 of file particle.h.

6.43.6.3 Predict_t SoilAnalyzer::Particle::Classification

The classification prediction

Definition at line 50 of file particle.h.

Referenced by GetAngularity(), GetRoundness(), QParticleDisplay::on_selectedParticleChangedSlider(), QParticleDisplay::on_selectedParticleChangedWidget(), serialize(), and SetRoundness().

6.43.6.4 double SoilAnalyzer::Particle::Eccentricty = 1

Definition at line 53 of file particle.h.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), GetSiDiameter(), GetSIVolume(), serialize(), and SetRoundness().

6.43.6.5 cv::Mat SoilAnalyzer::Particle::Edge

The binary edge image of the particle

Definition at line 43 of file particle.h.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), and serialize().

6.43.6.6 std::vector<Complex_t> SoilAnalyzer::Particle::FFDescriptors

The Fast Fourier Descriptors describing the contour in the Frequency domain

Definition at line 47 of file particle.h.

Referenced by serialize().

6.43.6.7 uint32_t SoilAnalyzer::Particle::ID

The particle ID

Definition at line 40 of file particle.h.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), and serialize().

6.43.6.8 bool SoilAnalyzer::Particle::isAnalysed = false

is the particle analyzed

Definition at line 68 of file particle.h.

Referenced by serialize().

6.43.6.9 bool SoilAnalyzer::Particle::isPreparedForAnalysis = false

is the particle ready for analysis

Definition at line 67 of file particle.h.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), and serialize().

6.43.6.10 bool SoilAnalyzer::Particle::isSmall = false

Definition at line 69 of file particle.h.

Referenced by serialize().

6.43.6.11 cv::Mat SoilAnalyzer::Particle::LAB [private]

Definition at line 77 of file particle.h.

Referenced by getLabImg(), getMeanLab(), and GetMeanRI().

6.43.6.12 Lab_t SoilAnalyzer::Particle::meanLab {0,0,0} [private]

Definition at line 76 of file particle.h.

Referenced by getMeanLab(), and serialize().

6.43.6.13 float SoilAnalyzer::Particle::meanRI = 0 [private]

Definition at line 75 of file particle.h.

Referenced by GetMeanRI(), and serialize().

6.43.6.14 uint32_t SoilAnalyzer::Particle::PixelArea = 0

The total area of the binary image

Definition at line 52 of file particle.h.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), GetSiDiameter(), GetSIVolume(), and serialize().

6.43.6.15 cv::Mat SoilAnalyzer::Particle::RGB

The RGB image of the particle

Definition at line 44 of file particle.h.

Referenced by QParticleDisplay::ConvertParticleToQImage(), getLabImg(), getMeanLab(), GetMeanRI(), SoilAnalyzer::Analyzer::GetParticles↩FromBlobList(), and serialize().

6.43.6.16 float SoilAnalyzer::Particle::SIDiameter = 0. [private]

Definition at line 73 of file particle.h.

Referenced by GetSiDiameter(), and serialize().

6.43.6.17 double SoilAnalyzer::Particle::SIPixelFactor = 0.0111915

The conversion factor from pixel to SI

Definition at line 51 of file particle.h.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList(), GetSiDiameter(), GetSIVolume(), and serialize().

6.43.6.18 float SoilAnalyzer::Particle::SIVolume = 0. [private]

The correspondening SI volume

Definition at line 72 of file particle.h.

Referenced by GetSIVolume(), and serialize().

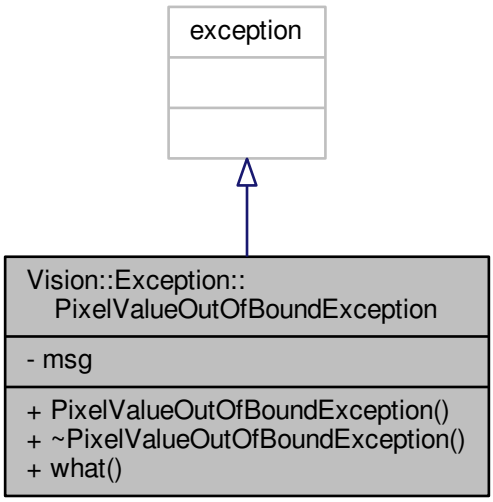
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/particle.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/particle.cpp

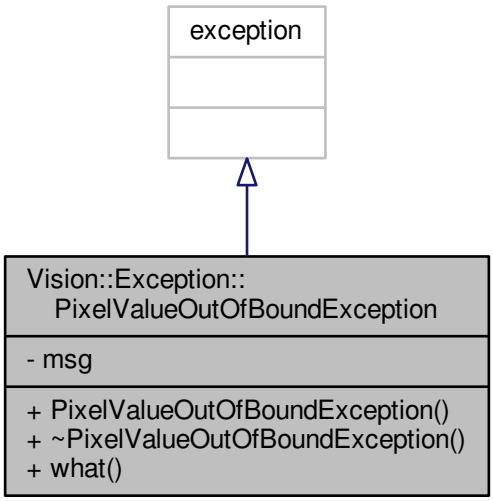
6.44 Vision::Exception::PixelValueOutOfBoundException Class Reference

`#include <PixelValueOutOfBoundException.h>`

Inheritance diagram for Vision::Exception::PixelValueOutOfBoundException:



Collaboration diagram for Vision::Exception::PixelValueOutOfBoundException:



Public Member Functions

- PixelValueOutOfBoundException (string m="Current pixel value out of bounds!")
- ~PixelValueOutOfBoundException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.44.1 Detailed Description

Definition at line 21 of file PixelValueOutOfBoundException.h.

6.44.2 Constructor & Destructor Documentation

6.44.2.1 Vision::Exception::PixelValueOutOfBoundException::PixelValueOutOfBoundException (string *m* = "Current pixel value out of bounds!") [inline]

Definition at line 23 of file PixelValueOutOfBoundException.h.

6.44.2.2 Vision::Exception::PixelValueOutOfBoundException::~PixelValueOutOfBoundException ( ) [inline]

Definition at line 25 of file PixelValueOutOfBoundException.h.

6.44.3 Member Function Documentation

6.44.3.1 const char* Vision::Exception::PixelValueOutOfBoundException::what ( ) const [inline]

Definition at line 26 of file PixelValueOutOfBoundException.h.

6.44.4 Member Data Documentation

6.44.4.1 string Vision::Exception::PixelValueOutOfBoundException::msg [private]

Definition at line 26 of file PixelValueOutOfBoundException.h.

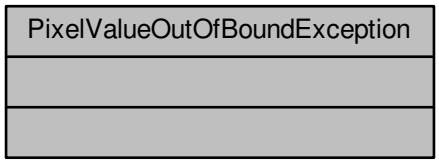
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/PixelValueOutOfBoundException.h

6.45 PixelValueOutOfBoundException Class Reference

`#include <PixelValueOutOfBoundException.h>`

Collaboration diagram for PixelValueOutOfBoundException:



6.45.1 Detailed Description

Exception class which is thrown when an unexpected pixel value has to be computed

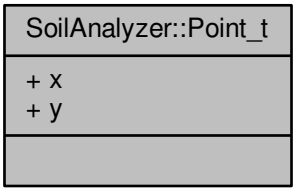
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/PixelValueOutOfBoundException.h

6.46 SoilAnalyzer::Point_t Struct Reference

`#include <soilanalyzertypes.h>`

Collaboration diagram for SoilAnalyzer::Point_t:



Public Attributes

- double x
- double y

6.46.1 Detailed Description

Definition at line 5 of file soilanalyzertypes.h.

6.46.2 Member Data Documentation

6.46.2.1 double SoilAnalyzer::Point_t::x

Definition at line 6 of file soilanalyzertypes.h.

Referenced by SoilAnalyzer::Particle::serialize().

6.46.2.2 double SoilAnalyzer::Point_t::y

Definition at line 7 of file soilanalyzertypes.h.

Referenced by SoilAnalyzer::Particle::serialize().

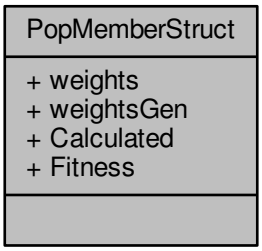
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilanalyzertypes.h

6.47 PopMemberStruct Struct Reference

`#include <SoilMathTypes.h>`

Collaboration diagram for PopMemberStruct:



Public Attributes

- Weight_t weights
- GenVect_t weightsGen
- float Calculated = 0.0
- float Fitness = 0.0

6.47.1 Detailed Description

Definition at line 33 of file SoilMathTypes.h.

6.47.2 Member Data Documentation

6.47.2.1 float PopMemberStruct::Calculated = 0.0

the calculated value

Definition at line 36 of file SoilMathTypes.h.

6.47.2.2 float PopMemberStruct::Fitness = 0.0

the fitness of the population member

Definition at line 37 of file SoilMathTypes.h.

Referenced by SoilMath::GA::PopMemberSort(), and SoilMath::GA::SurvivalOfTheFittest().

6.47.2.3 Weight_t PopMemberStruct::weights

the weights the core of a population member

Definition at line 34 of file SoilMathTypes.h.

Referenced by SoilMath::GA::Genesis().

6.47.2.4 GenVect_t PopMemberStruct::weightsGen

the weights as genomes

Definition at line 35 of file SoilMathTypes.h.

Referenced by SoilMath::GA::CrossOver(), and SoilMath::GA::Genesis().

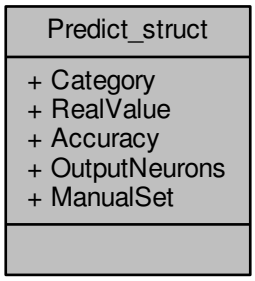
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/SoilMathTypes.h

6.48 Predict_struct Struct Reference

`#include <SoilMathTypes.h>`

Collaboration diagram for Predict_struct:



Public Attributes

- uint8_t Category = 1
- float RealValue = 1.
- float Accuracy = 1.
- std::vector< float > OutputNeurons
- bool ManualSet = true

6.48.1 Detailed Description

Definition at line 43 of file SoilMathTypes.h.

6.48.2 Member Data Documentation

6.48.2.1 float Predict_struct::Accuracy = 1.

the accuracy of the category

Definition at line 47 of file SoilMathTypes.h.

Referenced by boost::serialization::serialize().

6.48.2.2 uint8_t Predict_struct::Category = 1

the category number

Definition at line 44 of file SoilMathTypes.h.

Referenced by SoilAnalyzer::Particle::GetAngularity(), SoilAnalyzer::Particle::GetRoundness(), QParticleDisplay::on_selectedParticle↩ ChangedSlider(), QParticleDisplay::on_selectedParticleChangedWidget(), boost::serialization::serialize(), and SoilAnalyzer::Particle::Set↩ Roundness().

6.48.2.3 bool Predict_struct::ManualSet = true

Definition at line 49 of file SoilMathTypes.h.

Referenced by SoilMath::NN::Predict(), and SoilAnalyzer::Particle::SetRoundness().

6.48.2.4 std::vector<float> Predict_struct::OutputNeurons

the output Neurons

Definition at line 48 of file SoilMathTypes.h.

Referenced by SoilMath::GA::GrowToAdulthood(), DialogNN::makeLearnVectors(), SoilMath::NN::Predict(), and boost::serialization::serialize().

6.48.2.5 float Predict_struct::RealValue = 1.

category number as float in order to estimate how precise to outcome is

Definition at line 45 of file SoilMathTypes.h.

Referenced by boost::serialization::serialize().

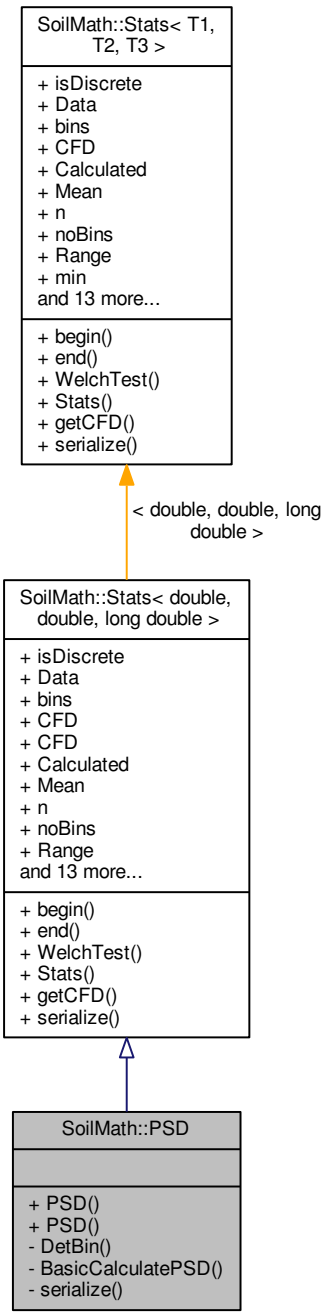
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/SoilMathTypes.h

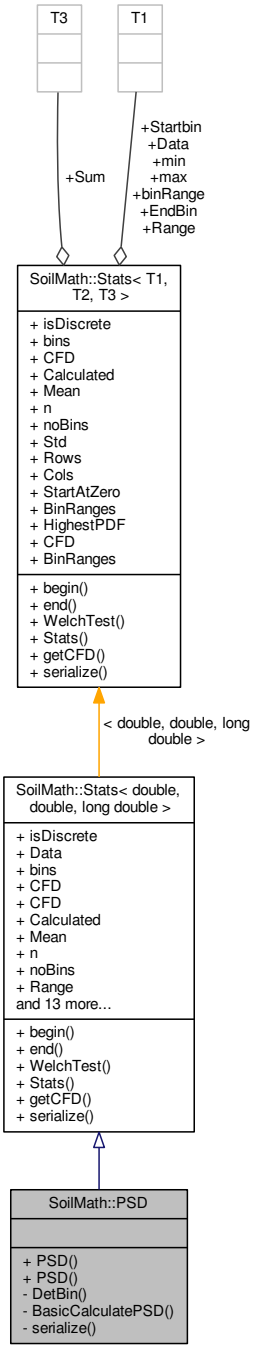
6.49 SoilMath::PSD Class Reference

`#include <psd.h>`

Inheritance diagram for SoilMath::PSD:



Collaboration diagram for SoilMath::PSD:



Public Member Functions

- PSD ()
- PSD (double *data, uint32_t nodata, double *binranges, uint32_t nobins, uint32_t endbin)

Private Member Functions

- uint32_t DetBin (float value)
- void BasicCalculatePSD ()
- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

Friends

- class boost::serialization::access

Additional Inherited Members

6.49.1 Detailed Description

Definition at line 14 of file psd.h.

6.49.2 Constructor & Destructor Documentation

6.49.2.1 SoilMath::PSD::PSD ( ) [inline]

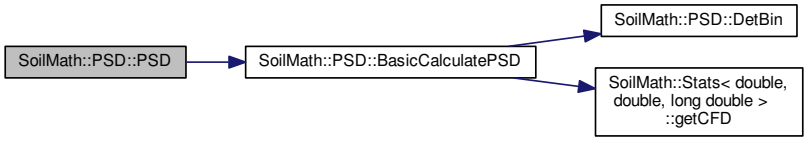
Definition at line 62 of file psd.h.

6.49.2.2 SoilMath::PSD::PSD (double * *data*, uint32_t *nodata*, double * *binranges*, uint32_t *nobins*, uint32_t *endbin*) [inline]

Definition at line 64 of file psd.h.

References BasicCalculatePSD(), SoilMath::Stats< double, double, long double >::BinRanges, SoilMath::Stats< double, double, long double >::Cols, SoilMath::Stats< double, double, long double >::Data, and SoilMath::Stats< double, double, long double >::Rows.

Here is the call graph for this function:



6.49.3 Member Function Documentation

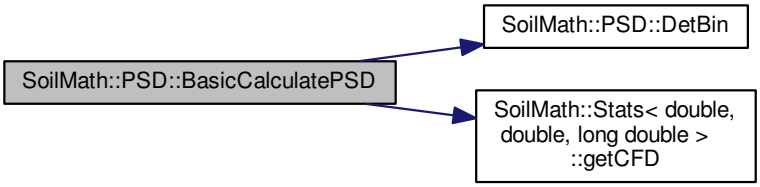
6.49.3.1 void SoilMath::PSD::BasicCalculatePSD ( ) [inline], [private]

Definition at line 27 of file psd.h.

References SoilMath::Stats< double, double, long double >::bins, SoilMath::Stats< double, double, long double >::Calculated, SoilMath::Stats< double, double, long double >::Cols, SoilMath::Stats< double, double, long double >::Data, DetBin(), SoilMath::Stats< double, double, long double >::getCFD(), SoilMath::Stats< double, double, long double >::max, SoilMath::Stats< double, double, long double >::Mean, SoilMath::Stats< double, double, long double >::min, SoilMath::Stats< double, double, long double >::n, SoilMath::Stats< double, double, long double >::Range, SoilMath::Stats< double, double, long double >::Rows, SoilMath::Stats< double, double, long double >::Std, and SoilMath::Stats< double, double, long double >::Sum.

Referenced by PSD().

Here is the call graph for this function:



Here is the caller graph for this function:

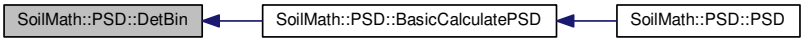
6.49.3.2 uint32_t SoilMath::PSD::DetBin (float *value*) [inline], [private]

Definition at line 16 of file psd.h.

References SoilMath::Stats< double, double, long double >::BinRanges, and SoilMath::Stats< double, double, long double >::noBins.

Referenced by BasicCalculatePSD().

Here is the caller graph for this function:



6.49.3.3 template<class Archive > void SoilMath::PSD::serialize (Archive & *ar,* const unsigned int *version*) [inline], [private]

Definition at line 54 of file psd.h.

6.49.4 Friends And Related Function Documentation

6.49.4.1 friend class boost::serialization::access [friend]

Definition at line 51 of file psd.h.

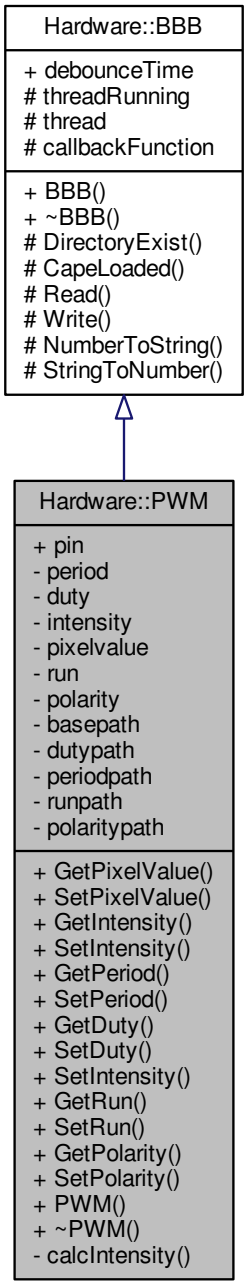
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/psd.h

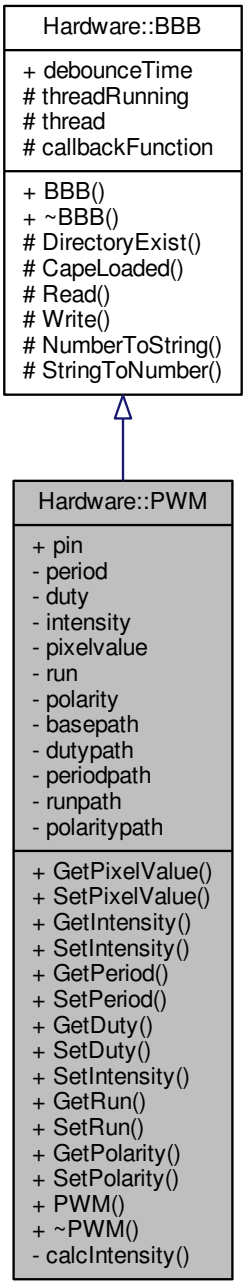
6.50 Hardware::PWM Class Reference

`#include <PWM.h>`

Inheritance diagram for Hardware::PWM:



Collaboration diagram for Hardware::PWM:



Public Types

- enum Pin { P8_13, P8_19, P9_14, P9_16 }
- enum Run { On = 1, Off = 0 }
- enum Polarity { Normal = 1, Inverted = 0 }

Public Member Functions

- uint8_t GetPixelValue ()
- void SetPixelValue (uint8_t value)

  Set the output as a corresponding uint8_t value
- float GetIntensity ()
- void SetIntensity (float value)

  Set the intensity level as percentage
- int GetPeriod ()
- void SetPeriod (int value)

Set the period of the signal

- int GetDuty ()
- void SetDuty (int value)

  Set the duty of the signal
- void SetIntensity ()
- Run GetRun ()
- void SetRun (Run value)

  Run the signal
- Polarity GetPolarity ()
- void SetPolarity (Polarity value)

  Set the polarity
- PWM (Pin pin)

  Constructeur
- ~PWM ()

Public Attributes

- Pin pin

Private Member Functions

- void calcIntensity ()

  Calculate the current intensity

Private Attributes

- int period
- int duty
- float intensity
- uint8_t pixelvalue
- Run run
- Polarity polarity
- string basepath
- string dutypath
- string periodpath
- string runpath
- string polaritypath

Additional Inherited Members

6.50.1 Detailed Description

Definition at line 16 of file PWM.h.

6.50.2 Member Enumeration Documentation

6.50.2.1 enum Hardware::PWM::Pin

Enumerator

P8_13

P8_19

P9_14

P9_16

Definition at line 18 of file PWM.h.

6.50.2.2 enum Hardware::PWM::Polarity

Enumerator

Normal

Inverted

Definition at line 26 of file PWM.h.

6.50.2.3 enum Hardware::PWM::Run

Enumerator

On

Off

Definition at line 23 of file PWM.h.

6.50.3 Constructor & Destructor Documentation

6.50.3.1 Hardware::PWM::PWM (Pin *pin*)

Constructeur

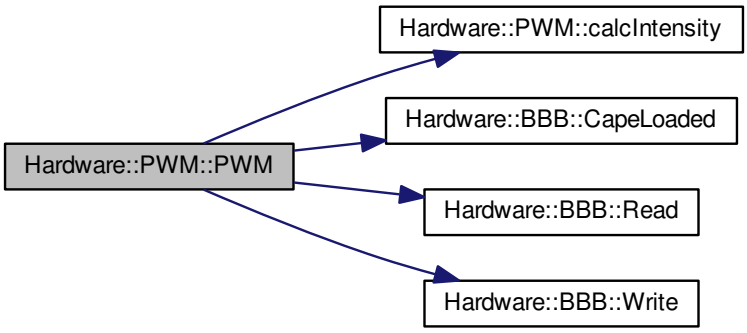
Parameters

pin	Pin

Definition at line 15 of file PWM.cpp.

References basepath, calcIntensity(), Hardware::BBB::CapeLoaded(), duty, dutypath, OCP_PATH, P8_13, P8_19, P9_14, P9_16, period, periodpath, pin, polarity, polaritypath, PWM_CAPE, Hardware::BBB::Read(), run, runpath, SLOTS, and Hardware::BBB::Write().

Here is the call graph for this function:



6.50.3.2 Hardware::PWM::~PWM ( )

Definition at line 65 of file PWM.cpp.

6.50.4 Member Function Documentation

6.50.4.1 void Hardware::PWM::calcIntensity ( ) [private]

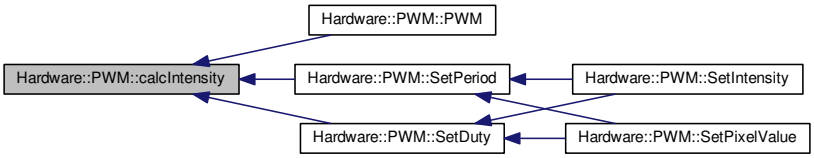
Calculate the current intensity

Definition at line 70 of file PWM.cpp.

References duty, intensity, Normal, period, and polarity.

Referenced by PWM(), SetDuty(), and SetPeriod().

Here is the caller graph for this function:



6.50.4.2 int Hardware::PWM::GetDuty ( ) [inline]

Definition at line 41 of file PWM.h.

References duty.

6.50.4.3 float Hardware::PWM::GetIntensity ( ) [inline]

Definition at line 35 of file PWM.h.

References intensity.

6.50.4.4 int Hardware::PWM::GetPeriod ( ) [inline]

Definition at line 38 of file PWM.h.

References period.

6.50.4.5 uint8_t Hardware::PWM::GetPixelValue ( ) [inline]

Definition at line 32 of file PWM.h.

References pixelvalue.

6.50.4.6 Polarity Hardware::PWM::GetPolarity ( ) [inline]

Definition at line 48 of file PWM.h.

References polarity.

6.50.4.7 Run Hardware::PWM::GetRun ( ) [inline]

Definition at line 45 of file PWM.h.

References run.

6.50.4.8 void Hardware::PWM::SetDuty (int *value*)

Set the duty of the signal

Parameters

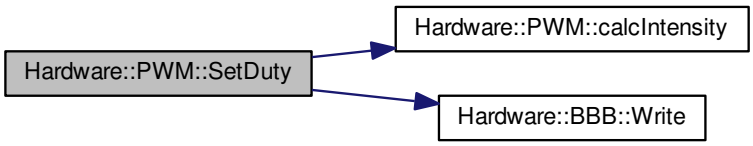
value	duty : int

Definition at line 126 of file PWM.cpp.

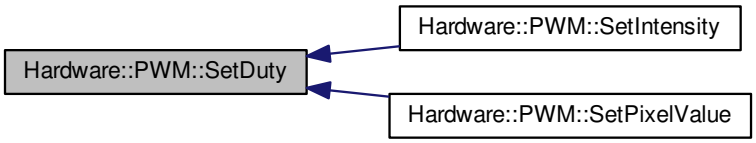
References calcIntensity(), duty, dutypath, and Hardware::BBB::Write().

Referenced by SetIntensity(), and SetPixelValue().

Here is the call graph for this function:



Here is the caller graph for this function:



6.50.4.9 void Hardware::PWM::SetIntensity (float *value*)

Set the intensity level as percentage

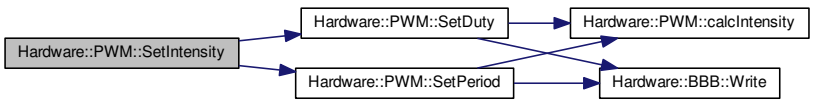
Parameters

value	floating value multipication factor

Definition at line 90 of file PWM.cpp.

References duty, Normal, period, polarity, SetDuty(), and SetPeriod().

Here is the call graph for this function:



6.50.4.10 void Hardware::PWM::SetIntensity ( )

6.50.4.11 void Hardware::PWM::SetPeriod (int *value*)

Set the period of the signal

Parameters

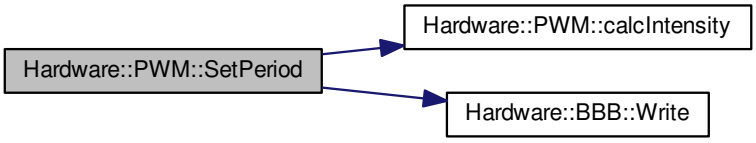
value	period : int

Definition at line 114 of file PWM.cpp.

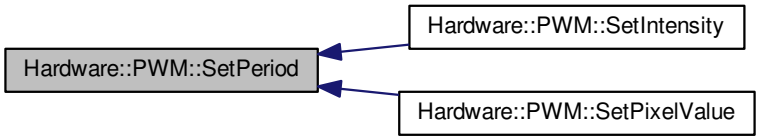
References calcIntensity(), period, periodpath, and Hardware::BBB::Write().

Referenced by SetIntensity(), and SetPixelValue().

Here is the call graph for this function:



Here is the caller graph for this function:



6.50.4.12 void Hardware::PWM::SetPixelValue (uint8_t *value*)

Set the output as a corresponding uint8_t value

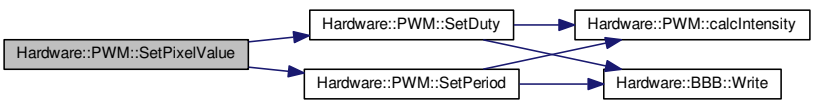
Parameters

value	pixel value 0-255

Definition at line 102 of file PWM.cpp.

References period, pixelvalue, SetDuty(), and SetPeriod().

Here is the call graph for this function:



6.50.4.13 void Hardware::PWM::SetPolarity (Polarity *value*)

Set the polarity

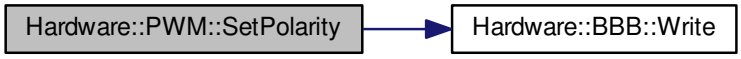
Parameters

value	Normal or Inverted signal

Definition at line 149 of file PWM.cpp.

References polarity, runpath, and Hardware::BBB::Write().

Here is the call graph for this function:



6.50.4.14 void Hardware::PWM::SetRun (Run *value*)

Run the signal

Parameters

value	On or Off

Definition at line 138 of file PWM.cpp.

References run, runpath, and Hardware::BBB::Write().

Here is the call graph for this function:

6.50.5 Member Data Documentation

6.50.5.1 string Hardware::PWM::basepath [private]

Definition at line 62 of file PWM.h.

Referenced by PWM().

6.50.5.2 int Hardware::PWM::duty [private]

Definition at line 56 of file PWM.h.

Referenced by calcIntensity(), GetDuty(), PWM(), SetDuty(), and SetIntensity().

6.50.5.3 string Hardware::PWM::dutypath [private]

Definition at line 63 of file PWM.h.

Referenced by PWM(), and SetDuty().

6.50.5.4 float Hardware::PWM::intensity [private]

Definition at line 57 of file PWM.h.

Referenced by calcIntensity(), and GetIntensity().

6.50.5.5 int Hardware::PWM::period [private]

Definition at line 55 of file PWM.h.

Referenced by calcIntensity(), GetPeriod(), PWM(), SetIntensity(), SetPeriod(), and SetPixelValue().

6.50.5.6 string Hardware::PWM::periodpath [private]

Definition at line 64 of file PWM.h.

Referenced by PWM(), and SetPeriod().

6.50.5.7 Pin Hardware::PWM::pin

Definition at line 30 of file PWM.h.

Referenced by PWM().

6.50.5.8 uint8_t Hardware::PWM::pixelvalue [private]

Definition at line 58 of file PWM.h.

Referenced by GetPixelValue(), and SetPixelValue().

6.50.5.9 Polarity Hardware::PWM::polarity [private]

Definition at line 60 of file PWM.h.

Referenced by calcIntensity(), GetPolarity(), PWM(), SetIntensity(), and SetPolarity().

6.50.5.10 string Hardware::PWM::polaritypath [private]

Definition at line 66 of file PWM.h.

Referenced by PWM().

6.50.5.11 Run Hardware::PWM::run [private]

Definition at line 59 of file PWM.h.

Referenced by GetRun(), PWM(), and SetRun().

6.50.5.12 string Hardware::PWM::runpath [private]

Definition at line 65 of file PWM.h.

Referenced by PWM(), SetPolarity(), and SetRun().

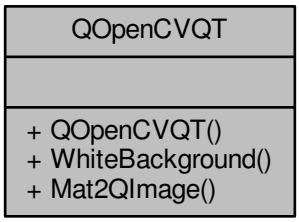
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/PWM.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/PWM.cpp

6.51 QOpenCVQT Class Reference

`#include <qopencvqt.h>`

Collaboration diagram for QOpenCVQT:



Public Member Functions

- QOpenCVQT ()

Static Public Member Functions

- static cv::Mat WhiteBackground (const cv::Mat &src)
- static QImage Mat2QImage (const cv::Mat &src)

6.51.1 Detailed Description

Definition at line 16 of file qopencvqt.h.

6.51.2 Constructor & Destructor Documentation

6.51.2.1 QOpenCVQT::QOpenCVQT ( )

Definition at line 11 of file qopencvqt.cpp.

6.51.3 Member Function Documentation

6.51.3.1 static QImage QOpenCVQT::Mat2QImage (const cv::Mat & *src*) [inline], [static]

Definition at line 26 of file qopencvqt.h.

6.51.3.2 static cv::Mat QOpenCVQT::WhiteBackground (const cv::Mat & *src*) [inline], [static]

Definition at line 20 of file qopencvqt.h.

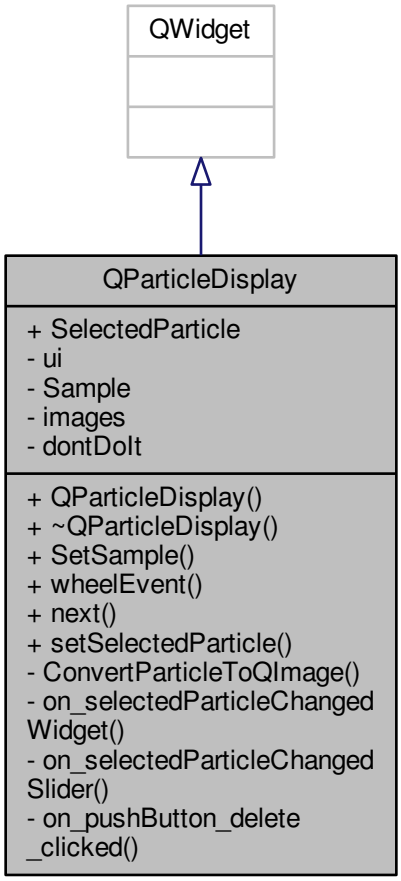
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QOpenCVQT/qopencvqt.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QOpenCVQT/qopencvqt.cpp

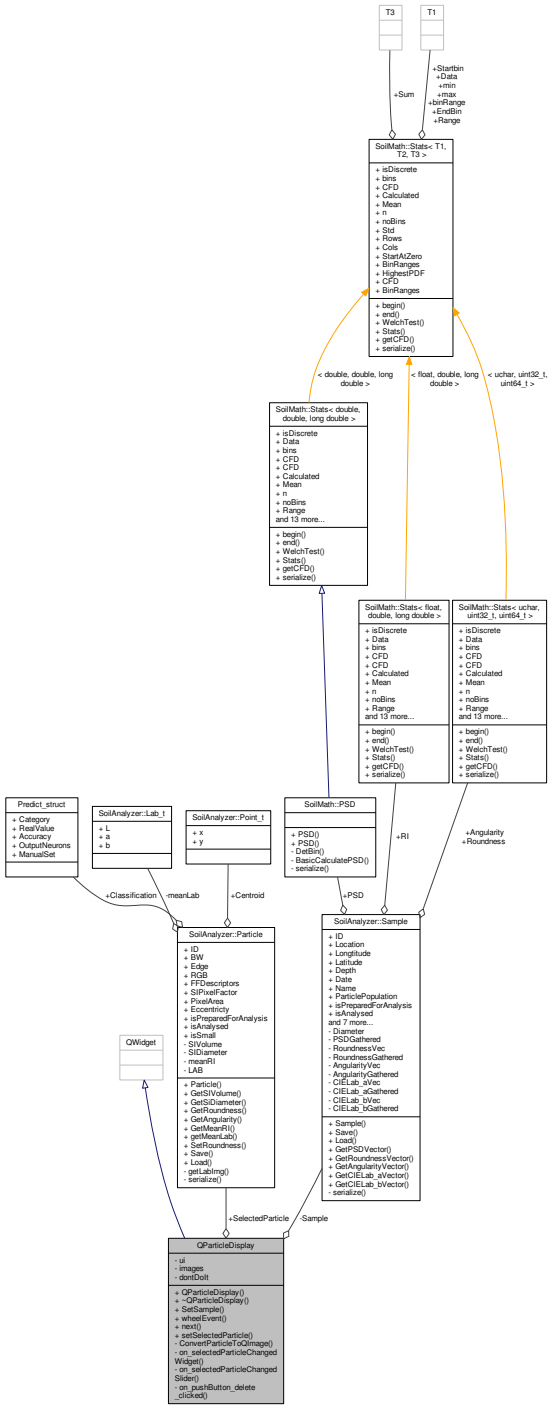
6.52 QParticleDisplay Class Reference

`#include <qparticledisplay.h>`

Inheritance diagram for QParticleDisplay:



Collaboration diagram for QParticleDisplay:



Public Slots

- void setSelectedParticle (int newValue)

Signals

- void particleChanged (int newValue)
- void shapeClassificationChanged (int newValue)
- void particleDeleted ()

Public Member Functions

- QParticleDisplay (QWidget *parent=0)
- ~QParticleDisplay ()
- void SetSample (SoilAnalyzer::Sample *sample)
- void wheelEvent (QWheelEvent *event)

- void next ()

Public Attributes

- SoilAnalyzer::Particle * SelectedParticle

Private Slots

- void on_selectedParticleChangedWidget (int value)
- void on_selectedParticleChangedSlider (int value)
- void on_pushButton_delete_clicked ()

Private Member Functions

- QImage ConvertParticleToQImage (SoilAnalyzer::Particle *particle)

Private Attributes

- Ui::QParticleDisplay * ui
- SoilAnalyzer::Sample * Sample
- QVector< QImage > images
- bool dontDoIt = false

6.52.1 Detailed Description

Definition at line 21 of file qparticledisplay.h.

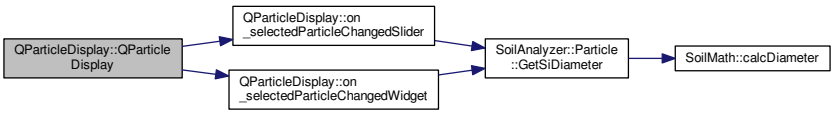
6.52.2 Constructor & Destructor Documentation

6.52.2.1 QParticleDisplay::QParticleDisplay (QWidget * *parent* = 0) [explicit]

Definition at line 11 of file qparticledisplay.cpp.

References on_selectedParticleChangedSlider(), on_selectedParticleChangedWidget(), and ui.

Here is the call graph for this function:



6.52.2.2 QParticleDisplay::~QParticleDisplay ( )

Definition at line 22 of file qparticledisplay.cpp.

References ui.

6.52.3 Member Function Documentation

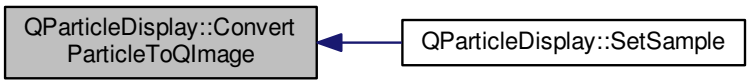
6.52.3.1 QImage QParticleDisplay::ConvertParticleToQImage (SoilAnalyzer::Particle * *particle*) [private]

Definition at line 50 of file qparticledisplay.cpp.

References SoilAnalyzer::Particle::BW, and SoilAnalyzer::Particle::RGB.

Referenced by SetSample().

Here is the caller graph for this function:



6.52.3.2 void QParticleDisplay::next ( )

Definition at line 150 of file qparticledisplay.cpp.

References on_selectedParticleChangedWidget(), and ui.

Here is the call graph for this function:

6.52.3.3 void QParticleDisplay::on_pushButton_delete_clicked ( ) [private], [slot]

Definition at line 96 of file qparticledisplay.cpp.

References SoilAnalyzer::Sample::ChangesSinceLastSave, SoilAnalyzer::Sample::ColorChange, SoilAnalyzer::Sample::ParticleChanged↩
StateAngularity, SoilAnalyzer::Sample::ParticleChangedStatePSD, SoilAnalyzer::Sample::ParticleChangedStateRoundness, particleDeleted(), SoilAnalyzer::Sample::ParticlePopulation, Sample, SelectedParticle, and ui.

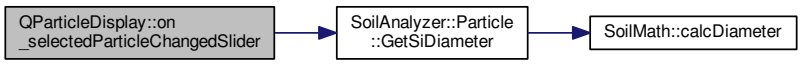
6.52.3.4 void QParticleDisplay::on_selectedParticleChangedSlider (int *value*) [private], [slot]

Definition at line 124 of file qparticledisplay.cpp.

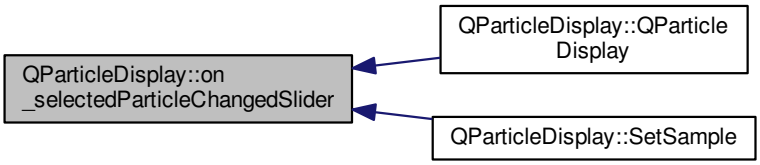
References Predict_struct::Category, SoilAnalyzer::Particle::Classification, dontDoIt, SoilAnalyzer::Particle::GetSiDiameter(), particleChanged(), SoilAnalyzer::Sample::ParticlePopulation, Sample, SelectedParticle, shapeClassificationChanged(), and ui.

Referenced by QParticleDisplay(), and SetSample().

Here is the call graph for this function:



Here is the caller graph for this function:



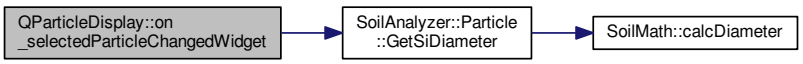
6.52.3.5 void QParticleDisplay::on_selectedParticleChangedWidget (int *value*) [private], [slot]

Definition at line 110 of file qparticledisplay.cpp.

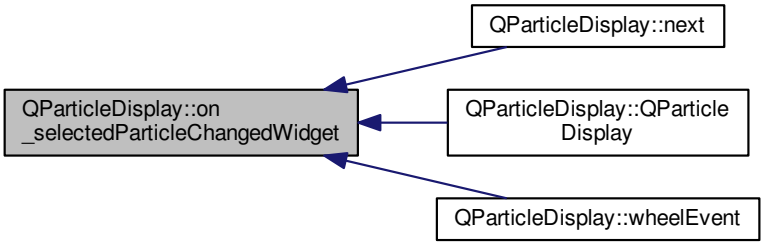
References Predict_struct::Category, SoilAnalyzer::Particle::Classification, dontDoIt, SoilAnalyzer::Particle::GetSiDiameter(), particleChanged(), SoilAnalyzer::Sample::ParticlePopulation, Sample, SelectedParticle, shapeClassificationChanged(), and ui.

Referenced by next(), QParticleDisplay(), and wheelEvent().

Here is the call graph for this function:



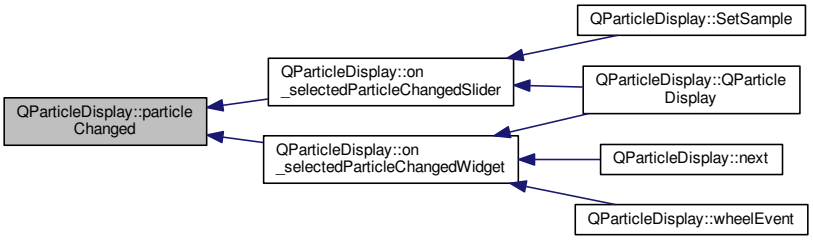
Here is the caller graph for this function:



6.52.3.6 void QParticleDisplay::particleChanged (int *newValue*) [signal]

Referenced by on_selectedParticleChangedSlider(), and on_selectedParticleChangedWidget().

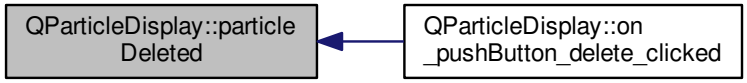
Here is the caller graph for this function:



6.52.3.7 void QParticleDisplay::particleDeleted ( ) [signal]

Referenced by on_pushButton_delete_clicked().

Here is the caller graph for this function:

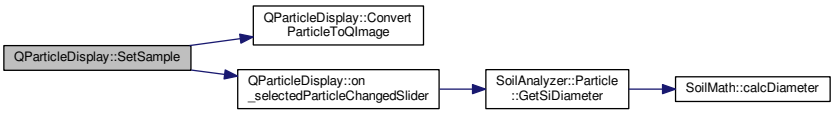


6.52.3.8 void QParticleDisplay::SetSample (SoilAnalyzer::Sample * *sample*)

Definition at line 35 of file qparticledisplay.cpp.

References ConvertParticleToQImage(), images, on_selectedParticleChangedSlider(), SoilAnalyzer::Sample::ParticlePopulation, Sample, SelectedParticle, and ui.

Here is the call graph for this function:



6.52.3.9 void QParticleDisplay::setSelectedParticle (int *newValue*) [slot]

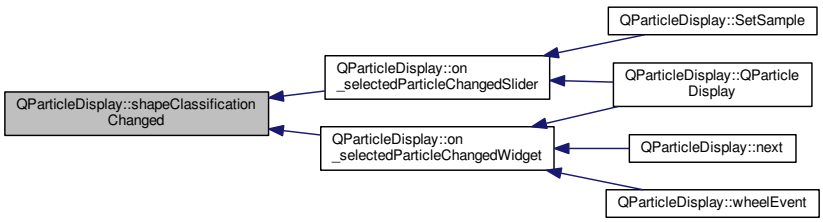
Definition at line 30 of file qparticledisplay.cpp.

References ui.

6.52.3.10 void QParticleDisplay::shapeClassificationChanged (int *newValue*) [signal]

Referenced by on_selectedParticleChangedSlider(), and on_selectedParticleChangedWidget().

Here is the caller graph for this function:

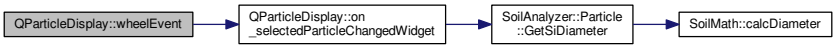


6.52.3.11 void QParticleDisplay::wheelEvent (QWheelEvent * *event*)

Definition at line 138 of file qparticledisplay.cpp.

References on_selectedParticleChangedWidget(), and ui.

Here is the call graph for this function:



6.52.4 Member Data Documentation

6.52.4.1 bool QParticleDisplay::dontDoIt = false [private]

Definition at line 51 of file qparticledisplay.h.

Referenced by on_selectedParticleChangedSlider(), and on_selectedParticleChangedWidget().

6.52.4.2 QVector<QImage> QParticleDisplay::images [private]

Definition at line 49 of file qparticledisplay.h.

Referenced by SetSample().

6.52.4.3 SoilAnalyzer::Sample* QParticleDisplay::Sample [private]

Definition at line 48 of file qparticledisplay.h.

Referenced by on_pushButton_delete_clicked(), on_selectedParticleChangedSlider(), on_selectedParticleChangedWidget(), and SetSample().

6.52.4.4 SoilAnalyzer::Particle* QParticleDisplay::SelectedParticle

Definition at line 29 of file qparticledisplay.h.

Referenced by on_pushButton_delete_clicked(), on_selectedParticleChangedSlider(), on_selectedParticleChangedWidget(), and SetSample().

6.52.4.5 Ui::QParticleDisplay* QParticleDisplay::ui [private]

Definition at line 47 of file qparticledisplay.h.

Referenced by next(), on_pushButton_delete_clicked(), on_selectedParticleChangedSlider(), on_selectedParticleChangedWidget(), QParticle↩Display(), SetSample(), setSelectedParticle(), wheelEvent(), and ~QParticleDisplay().

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

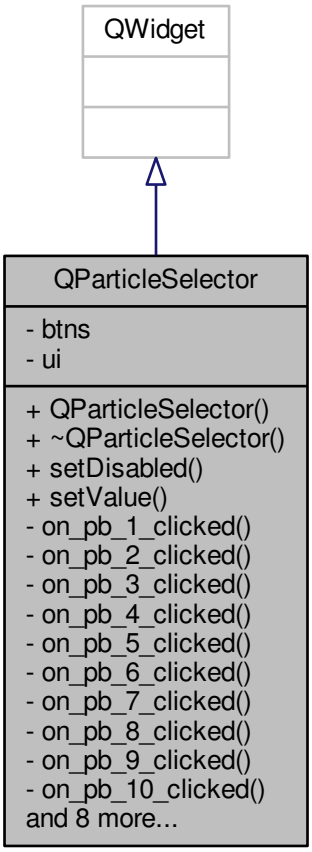
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleDisplay/qparticledisplay.h

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleDisplay/qparticledisplay.cpp

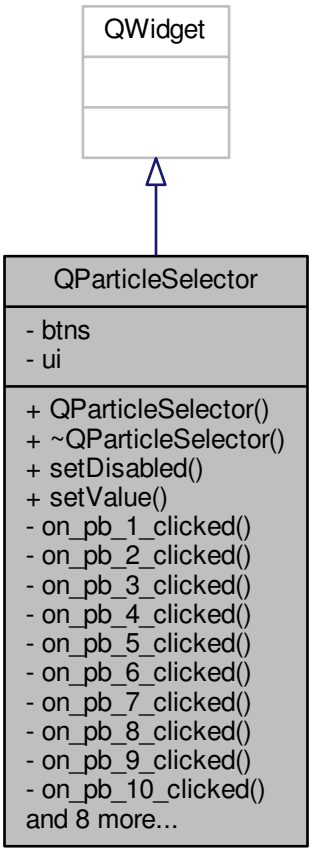
6.53 QParticleSelector Class Reference

```
#include <qparticleselector.h>
```

Inheritance diagram for QParticleSelector:



Collaboration diagram for QParticleSelector:



Public Slots

- void setValue (int newValue)

Signals

- void valueChanged (int newValue)

Public Member Functions

- QParticleSelector (QWidget *parent=0)
- ~QParticleSelector ()
- void setDisabled (bool value, int currentClass=1)

Private Slots

- void on_pb_1_clicked (bool checked)
- void on_pb_2_clicked (bool checked)
- void on_pb_3_clicked (bool checked)
- void on_pb_4_clicked (bool checked)
- void on_pb_5_clicked (bool checked)
- void on_pb_6_clicked (bool checked)
- void on_pb_7_clicked (bool checked)
- void on_pb_8_clicked (bool checked)
- void on_pb_9_clicked (bool checked)
- void on_pb_10_clicked (bool checked)
- void on_pb_11_clicked (bool checked)

- void on_pb_12_clicked (bool checked)
- void on_pb_13_clicked (bool checked)
- void on_pb_14_clicked (bool checked)
- void on_pb_15_clicked (bool checked)
- void on_pb_16_clicked (bool checked)
- void on_pb_17_clicked (bool checked)
- void on_pb_18_clicked (bool checked)

Private Attributes

- QVector< QPushButton * > btns
- Ui::QParticleSelector * ui

6.53.1 Detailed Description

Definition at line 11 of file qparticleselector.h.

6.53.2 Constructor & Destructor Documentation

6.53.2.1 QParticleSelector::QParticleSelector (QWidget * *parent* = 0) [explicit]

Definition at line 4 of file qparticleselector.cpp.

References btns, and ui.

6.53.2.2 QParticleSelector::~QParticleSelector ( )

Definition at line 27 of file qparticleselector.cpp.

References btns, and ui.

6.53.3 Member Function Documentation

6.53.3.1 void QParticleSelector::on_pb_10_clicked (bool *checked*) [private], [slot]

Definition at line 104 of file qparticleselector.cpp.

References valueChanged().

6.53.3.2 void QParticleSelector::on_pb_11_clicked (bool *checked*) [private], [slot]

Definition at line 110 of file qparticleselector.cpp.

References valueChanged().

6.53.3.3 void QParticleSelector::on_pb_12_clicked (bool *checked*) [private], [slot]

Definition at line 116 of file qparticleselector.cpp.

References valueChanged().

6.53.3.4 void QParticleSelector::on_pb_13_clicked (bool *checked*) [private], [slot]

Definition at line 122 of file qparticleselector.cpp.

References valueChanged().

6.53.3.5 void QParticleSelector::on_pb_14_clicked (bool *checked*) [private], [slot]

Definition at line 128 of file qparticleselector.cpp.

References valueChanged().

6.53.3.6 void QParticleSelector::on_pb_15_clicked (bool *checked*) [private], [slot]

Definition at line 134 of file qparticleselector.cpp.

References valueChanged().

6.53.3.7 void QParticleSelector::on_pb_16_clicked (bool *checked*) [private], [slot]

Definition at line 140 of file qparticleselector.cpp.

References valueChanged().

6.53.3.8 void QParticleSelector::on_pb_17_clicked (bool *checked*) [private], [slot]

Definition at line 146 of file qparticleselector.cpp.

References valueChanged().

6.53.3.9 void QParticleSelector::on_pb_18_clicked (bool *checked*) [private], [slot]

Definition at line 152 of file qparticleselector.cpp.

References valueChanged().

6.53.3.10 void QParticleSelector::on_pb_1_clicked (bool *checked*) [private], [slot]

Definition at line 50 of file qparticleselector.cpp.

References valueChanged().

6.53.3.11 void QParticleSelector::on_pb_2_clicked (bool *checked*) [private], [slot]

Definition at line 56 of file qparticleselector.cpp.

References valueChanged().

6.53.3.12 void QParticleSelector::on_pb_3_clicked (bool *checked*) [private], [slot]

Definition at line 62 of file qparticleselector.cpp.

References valueChanged().

6.53.3.13 void QParticleSelector::on_pb_4_clicked (bool *checked*) [private], [slot]

Definition at line 68 of file qparticleselector.cpp.

References valueChanged().

6.53.3.14 void QParticleSelector::on_pb_5_clicked (bool *checked*) [private], [slot]

Definition at line 74 of file qparticleselector.cpp.

References valueChanged().

6.53.3.15 void QParticleSelector::on_pb_6_clicked (bool *checked*) [private], [slot]

Definition at line 80 of file qparticleselector.cpp.

References valueChanged().

6.53.3.16 void QParticleSelector::on_pb_7_clicked (bool *checked*) [private], [slot]

Definition at line 86 of file qparticleselector.cpp.

References valueChanged().

6.53.3.17 void QParticleSelector::on_pb_8_clicked (bool *checked*) [private], [slot]

Definition at line 92 of file qparticleselector.cpp.

References valueChanged().

6.53.3.18 void QParticleSelector::on_pb_9_clicked (bool *checked*) [private], [slot]

Definition at line 98 of file qparticleselector.cpp.

References valueChanged().

6.53.3.19 void QParticleSelector::setDisabled (bool *value,* int *currentClass* = 1)

Definition at line 39 of file qparticleselector.cpp.

References btns.

6.53.3.20 void QParticleSelector::setValue (int *newValue*) [slot]

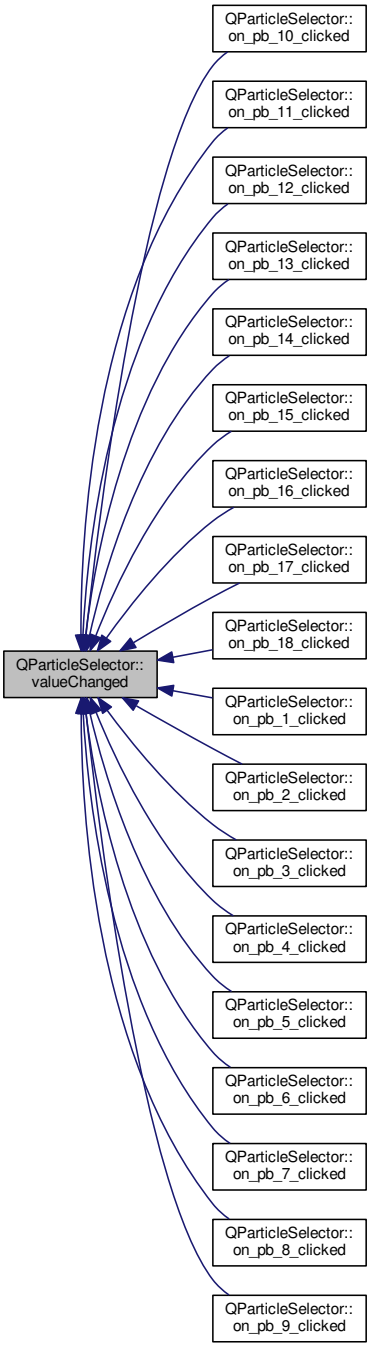
Definition at line 35 of file qparticleselector.cpp.

References btns.

6.53.3.21 void QParticleSelector::valueChanged (int *newValue*) [signal]

Referenced by on_pb_10_clicked(), on_pb_11_clicked(), on_pb_12_clicked(), on_pb_13_clicked(), on_pb_14_clicked(), on_pb_15_clicked(), on_pb_16_clicked(), on_pb_17_clicked(), on_pb_18_clicked(), on_pb_1_clicked(), on_pb_2_clicked(), on_pb_3_clicked(), on_pb_4_clicked(), on_pb_5_clicked(), on_pb_6_clicked(), on_pb_7_clicked(), on_pb_8_clicked(), and on_pb_9_clicked().

Here is the caller graph for this function:



6.53.4 Member Data Documentation

6.53.4.1 QVector<QPushButton *> QParticleSelector::btns [private]

Definition at line 65 of file qparticleselector.h.

Referenced by QParticleSelector(), setDisabled(), setValue(), and ~QParticleSelector().

6.53.4.2 Ui::QParticleSelector* QParticleSelector::ui [private]

Definition at line 66 of file qparticleselector.h.

Referenced by QParticleSelector(), and ~QParticleSelector().

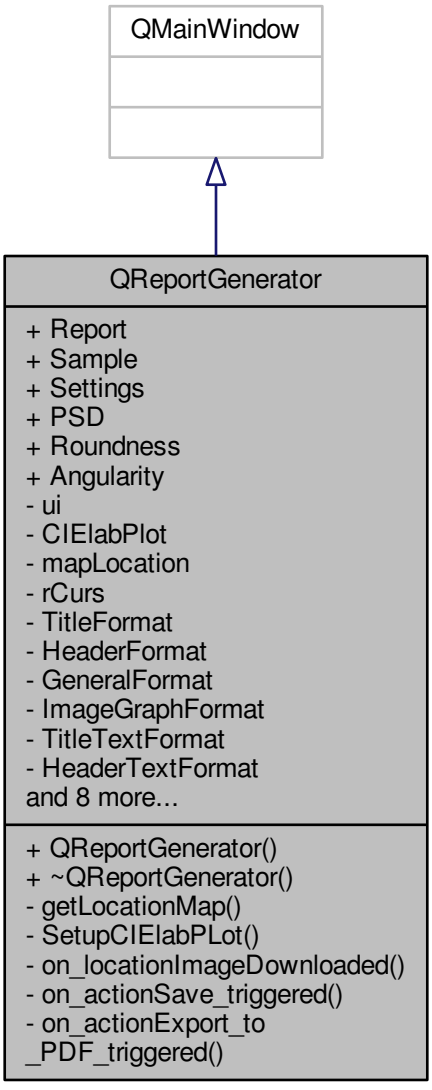
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleSelector/qparticleselector.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QParticleSelector/qparticleselector.cpp

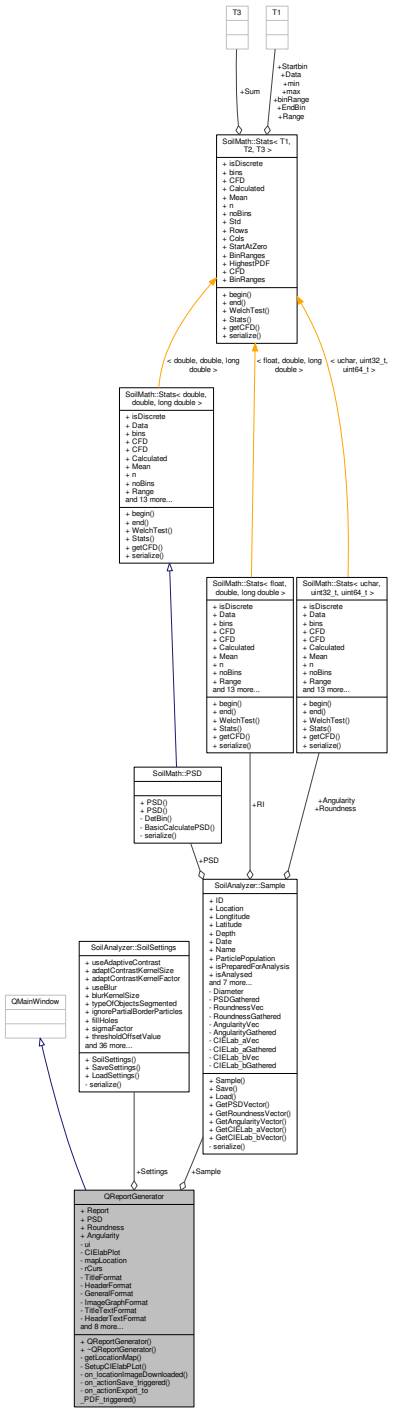
6.54 QReportGenerator Class Reference

`#include <qreportgenerator.h>`

Inheritance diagram for QReportGenerator:



Collaboration diagram for QReportGenerator:



Public Member Functions

- QReportGenerator (QWidget *parent=0, SoilAnalyzer::Sample *sample=nullptr, SoilAnalyzer::SoilSettings *settings=nullptr, QCustom↩ Plot *psd=nullptr, QCustomPlot *roundness=nullptr, QCustomPlot *angularity=nullptr)
- ~QReportGenerator ()

Public Attributes

- QTextDocument * Report = nullptr
- SoilAnalyzer::Sample * Sample = nullptr
- SoilAnalyzer::SoilSettings * Settings = nullptr
- QCustomPlot * PSD = nullptr
- QCustomPlot * Roundness = nullptr
- QCustomPlot * Angularity = nullptr

Private Slots

- void on_locationImageDownloaded (QNetworkReply *reply)
- void on_actionSave_triggered ()
- void on_actionExport_to_PDF_triggered ()

Private Member Functions

- void getLocationMap (double &latitude, double &longtitude)
- void SetupCIElabPLot ()

Private Attributes

- Ui::QReportGenerator * ui
- QCustomPlot * CIElabPlot = nullptr
- QImage * mapLocation = nullptr
- QTextCursor rCurs
- QTextBlockFormat TitleFormat
- QTextBlockFormat HeaderFormat
- QTextBlockFormat GeneralFormat
- QTextBlockFormat ImageGraphFormat
- QTextCharFormat TitleTextFormat
- QTextCharFormat HeaderTextFormat
- QTextCharFormat GtxtFormat
- QTextCharFormat GFieldtxtFormat
- QTextListFormat GeneralSampleList
- QTextTableFormat GeneralTextTableFormat
- QFont TitleFont
- QFont HeaderFont
- QFont GeneralFont
- QFont FieldFont

6.54.1 Detailed Description

Definition at line 25 of file qreportgenerator.h.

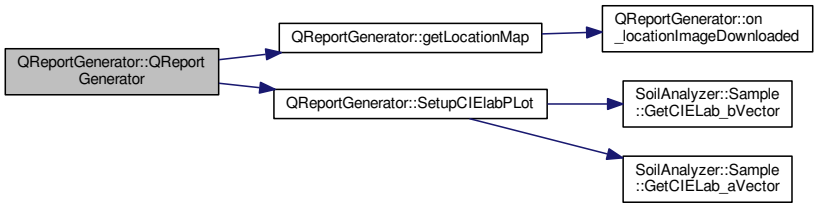
6.54.2 Constructor & Destructor Documentation

6.54.2.1 QReportGenerator::QReportGenerator (QWidget * *parent* = 0, SoilAnalyzer::Sample * *sample* = nullptr, SoilAnalyzer::SoilSettings * *settings* = nullptr, QCustomPlot * *psd* = nullptr, QCustomPlot * *roundness* = nullptr, QCustomPlot * *angularity* = nullptr) [explicit]

Definition at line 4 of file qreportgenerator.cpp.

References Angularity, SoilAnalyzer::Sample::Angularity, SoilMath::Stats< T1, T2, T3 >::bins, SoilMath::Stats< T1, T2, T3 >::CFD, CIElabPlot, SoilAnalyzer::Sample::Date, SoilAnalyzer::Sample::Depth, FieldFont, GeneralFont, GeneralFormat, GeneralSampleList, GeneralTextTableFormat, getLocationMap(), GFieldtxtFormat, GtxtFormat, HeaderFont, HeaderFormat, HeaderTextFormat, SoilAnalyzer::Sample::ID, ImageGraphFormat, SoilAnalyzer::Sample::Latitude, SoilAnalyzer::Sample::Longtitude, SoilMath::Stats< T1, T2, T3 >::max, SoilMath::Stats< T1, T2, T3 >::Mean, SoilMath::Stats< T1, T2, T3 >::min, SoilMath::Stats< T1, T2, T3 >::n, SoilAnalyzer::Sample::Name, SoilAnalyzer::Sample::PSD, SoilMath::Stats< T1, T2, T3 >::Range, rCurs, Report, Roundness, SoilAnalyzer::Sample::Roundness, Sample, Settings, SetupCIElabPLot(), SoilMath::Stats< T1, T2, T3 >::Std, TitleFont, TitleFormat, TitleTextFormat, and ui.

Here is the call graph for this function:



6.54.2.2 QReportGenerator::~QReportGenerator ( )

Definition at line 394 of file qreportgenerator.cpp.

References CIElabPlot, mapLocation, and ui.

6.54.3 Member Function Documentation

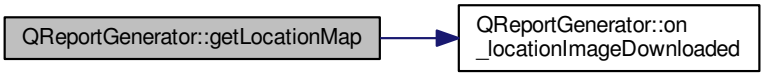
6.54.3.1 void QReportGenerator::getLocationMap (double & *latitude*, double & *longtitude*) [private]

Definition at line 357 of file qreportgenerator.cpp.

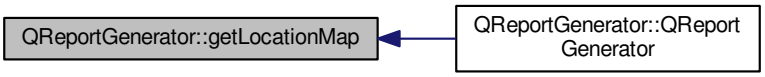
References on_locationImageDownloaded().

Referenced by QReportGenerator().

Here is the call graph for this function:



Here is the caller graph for this function:



6.54.3.2 void QReportGenerator::on_actionExport_to_PDF_triggered ( ) [private], [slot]

Definition at line 416 of file qreportgenerator.cpp.

References Report, SoilAnalyzer::SoilSettings::SampleFolder, and Settings.

6.54.3.3 void QReportGenerator::on_actionSave_triggered ( ) [private], [slot]

Definition at line 401 of file qreportgenerator.cpp.

References Report, SoilAnalyzer::SoilSettings::SampleFolder, and Settings.

6.54.3.4 void QReportGenerator::on_locationImageDownloaded (QNetworkReply * *reply*) [private], [slot]

Definition at line 376 of file qreportgenerator.cpp.

References ImageGraphFormat, mapLocation, and Report.

Referenced by getLocationMap().

Here is the caller graph for this function:

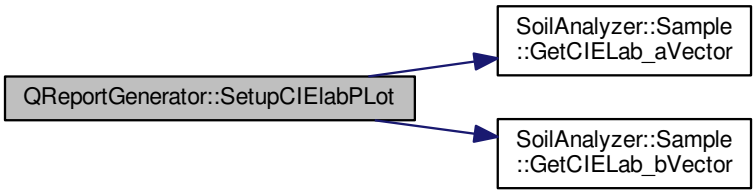
6.54.3.5 void QReportGenerator::SetupCIElabPLot ( ) [private]

Definition at line 431 of file qreportgenerator.cpp.

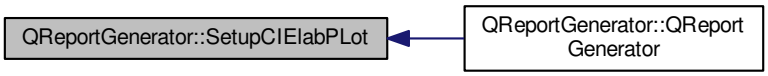
References CIElabPlot, SoilAnalyzer::Sample::GetCIELab_aVector(), SoilAnalyzer::Sample::GetCIELab_bVector(), and Sample.

Referenced by QReportGenerator().

Here is the call graph for this function:



Here is the caller graph for this function:



6.54.4 Member Data Documentation

6.54.4.1 QCustomPlot* QReportGenerator::Angularity = nullptr

Definition at line 35 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.2 QCustomPlot* QReportGenerator::CIElabPlot = nullptr [private]

Definition at line 49 of file qreportgenerator.h.

Referenced by QReportGenerator(), SetupCIElabPLot(), and ~QReportGenerator().

6.54.4.3 QFont QReportGenerator::FieldFont [private]

Definition at line 76 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.4 QFont QReportGenerator::GeneralFont [private]

Definition at line 75 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.5 QTextBlockFormat QReportGenerator::GeneralFormat [private]

Definition at line 61 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.6 QTextListFormat QReportGenerator::GeneralSampleList [private]

Definition at line 69 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.7 QTextTableFormat QReportGenerator::GeneralTextTableFormat [private]

Definition at line 70 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.8 QTextCharFormat QReportGenerator::GFieldtxtFormat [private]

Definition at line 67 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.9 QTextCharFormat QReportGenerator::GtxtFormat [private]

Definition at line 66 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.10 QFont QReportGenerator::HeaderFont [private]

Definition at line 74 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.11 QTextBlockFormat QReportGenerator::HeaderFormat [private]

Definition at line 60 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.12 QTextCharFormat QReportGenerator::HeaderTextFormat [private]

Definition at line 65 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.13 QTextBlockFormat QReportGenerator::ImageGraphFormat [private]

Definition at line 62 of file qreportgenerator.h.

Referenced by on_locationImageDownloaded(), and QReportGenerator().

6.54.4.14 QImage* QReportGenerator::mapLocation = nullptr [private]

Definition at line 54 of file qreportgenerator.h.

Referenced by on_locationImageDownloaded(), and ~QReportGenerator().

6.54.4.15 QCustomPlot* QReportGenerator::PSD = nullptr

Definition at line 33 of file qreportgenerator.h.

6.54.4.16 QTextCursor QReportGenerator::rCurs [private]

Definition at line 56 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.17 QTextDocument* QReportGenerator::Report = nullptr

Definition at line 30 of file qreportgenerator.h.

Referenced by on_actionExport_to_PDF_triggered(), on_actionSave_triggered(), on_locationImageDownloaded(), and QReportGenerator().

6.54.4.18 QCustomPlot* QReportGenerator::Roundness = nullptr

Definition at line 34 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.19 SoilAnalyzer::Sample* QReportGenerator::Sample = nullptr

Definition at line 31 of file qreportgenerator.h.

Referenced by QReportGenerator(), and SetupCIElabPLot().

6.54.4.20 SoilAnalyzer::SoilSettings* QReportGenerator::Settings = nullptr

Definition at line 32 of file qreportgenerator.h.

Referenced by on_actionExport_to_PDF_triggered(), on_actionSave_triggered(), and QReportGenerator().

6.54.4.21 QFont QReportGenerator::TitleFont [private]

Definition at line 73 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.22 QTextBlockFormat QReportGenerator::TitleFormat [private]

Definition at line 59 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.23 QTextCharFormat QReportGenerator::TitleTextFormat [private]

Definition at line 64 of file qreportgenerator.h.

Referenced by QReportGenerator().

6.54.4.24 Ui::QReportGenerator* QReportGenerator::ui [private]

Definition at line 48 of file qreportgenerator.h.

Referenced by QReportGenerator(), and ~QReportGenerator().

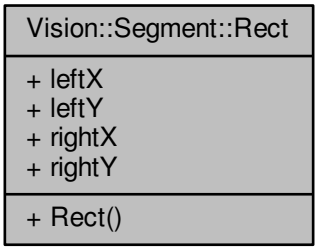
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QReportGenerator/qreportgenerator.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/QReportGenerator/qreportgenerator.cpp

6.55 Vision::Segment::Rect Struct Reference

`#include <Segment.h>`

Collaboration diagram for Vision::Segment::Rect:



Public Member Functions

- Rect (uint16_t lx, uint16_t ly, uint16_t rx, uint16_t ry)

Public Attributes

- uint16_t leftX
- uint16_t leftY
- uint16_t rightX
- uint16_t rightY

6.55.1 Detailed Description

Coordinates for the region of interest

Definition at line 30 of file Segment.h.

6.55.2 Constructor & Destructor Documentation

6.55.2.1 Vision::Segment::Rect::Rect (uint16_t *lx*, uint16_t *ly*, uint16_t *rx*, uint16_t *ry*) [inline]

Definition at line 35 of file Segment.h.

6.55.3 Member Data Documentation

6.55.3.1 uint16_t Vision::Segment::Rect::leftX

Left X coordinate

Definition at line 31 of file Segment.h.

6.55.3.2 uint16_t Vision::Segment::Rect::leftY

Left Y coordinate

Definition at line 32 of file Segment.h.

6.55.3.3 uint16_t Vision::Segment::Rect::rightX

Right X coordinate

Definition at line 33 of file Segment.h.

6.55.3.4 uint16_t Vision::Segment::Rect::rightY

Right Y coordinate

Definition at line 34 of file Segment.h.

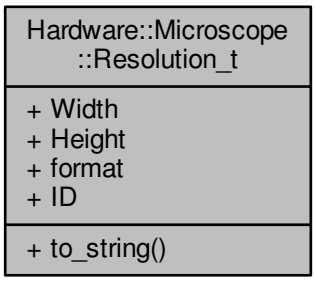
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.h

6.56 Hardware::Microscope::Resolution_t Struct Reference

`#include <Microscope.h>`

Collaboration diagram for Hardware::Microscope::Resolution_t:



Public Member Functions

- std::string to_string ()

Public Attributes

- uint16_t Width = 2048
- uint16_t Height = 1536
- PixelFormat format = PixelFormat::MJPG
- uint32_t ID

6.56.1 Detailed Description

Definition at line 57 of file Microscope.h.

6.56.2 Member Function Documentation

6.56.2.1 std::string Hardware::Microscope::Resolution_t::to_string ( ) [inline]

Definition at line 61 of file Microscope.h.

6.56.3 Member Data Documentation

6.56.3.1 PixelFormat Hardware::Microscope::Resolution_t::format = PixelFormat::MJPG

Definition at line 60 of file Microscope.h.

Referenced by Hardware::Microscope::getResolutions(), and Hardware::Microscope::openCam().

6.56.3.2 uint16_t Hardware::Microscope::Resolution_t::Height = 1536

Definition at line 59 of file Microscope.h.

Referenced by Hardware::Microscope::getResolutions(), Hardware::Microscope::new_buffer(), and Hardware::Microscope::openCam().

6.56.3.3 uint32_t Hardware::Microscope::Resolution_t::ID

Definition at line 76 of file Microscope.h.

Referenced by Hardware::Microscope::getResolutions().

6.56.3.4 uint16_t Hardware::Microscope::Resolution_t::Width = 2048

Definition at line 58 of file Microscope.h.

Referenced by Hardware::Microscope::getResolutions(), Hardware::Microscope::new_buffer(), and Hardware::Microscope::openCam().

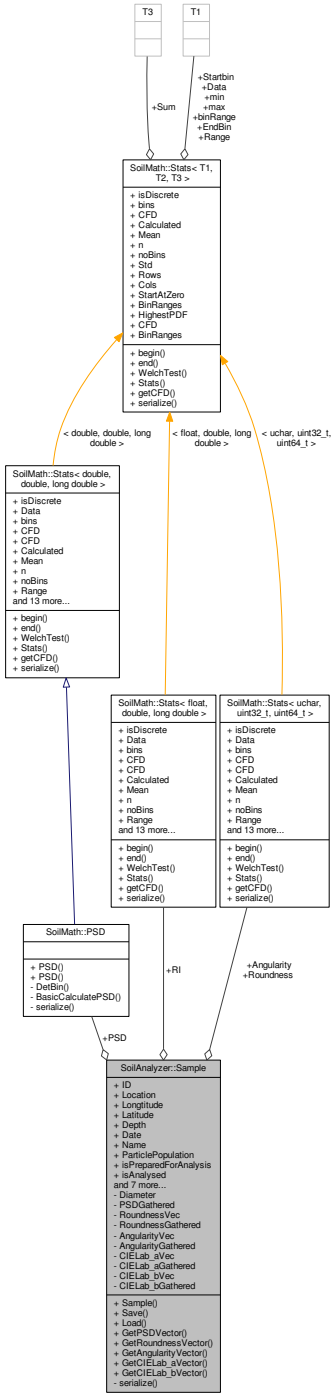
The documentation for this struct was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/Microscope.h

6.57 SoilAnalyzer::Sample Class Reference

```
#include <sample.h>
```

Collaboration diagram for SoilAnalyzer::Sample:



Public Member Functions

- Sample ()

  Sample::Sample.
- void Save (const std::string &filename)

  Sample::Save.
- void Load (const std::string &filename)

  Sample::Load.
- Particle::PSDVector_t * GetPSDVector ()

  Sample::GetPSDVector.
- Particle::ClassVector_t * GetRoundnessVector ()
- Particle::ClassVector_t * GetAngularityVector ()
- Particle::doubleVector_t * GetCIELab_aVector ()
- Particle::doubleVector_t * GetCIELab_bVector ()

Public Attributes

- uint32_t ID
- std::string Location
- double Longtitude = 4.629618299999947
- double Latitude = 51.8849149
- double Depth = 0
- std::string Date = "01-09-2015"
- std::string Name
- Particle::ParticleVector_t ParticlePopulation
- SoilMath::PSD PSD
- ucharStat_t Roundness
- ucharStat_t Angularity
- floatStat_t RI
- bool isPreparedForAnalysis
- bool isAnalysed = false
- bool ChangesSinceLastSave = false
- bool ParticleChangedStatePSD = false
- bool ParticleChangedStateClass = false
- bool ParticleChangedStateRoundness = false
- bool ParticleChangedStateAngularity = false
- bool ColorChange = false
- bool IsLoadedFromDisk = false

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

Private Attributes

- Particle::PSDVector_t Diameter
- bool PSDGathered = false
- Particle::ClassVector_t RoundnessVec
- bool RoundnessGathered = false
- Particle::ClassVector_t AngularityVec
- bool AngularityGathered = false
- Particle::doubleVector_t CIELab_aVec
- bool CIELab_aGathered = false
- Particle::doubleVector_t CIELab_bVec
- bool CIELab_bGathered = false

Friends

- class boost::serialization::access

6.57.1 Detailed Description

Definition at line 28 of file sample.h.

6.57.2 Constructor & Destructor Documentation

6.57.2.1 SoilAnalyzer::Sample::Sample ( )

Sample::Sample.

Definition at line 17 of file sample.cpp.

6.57.3 Member Function Documentation

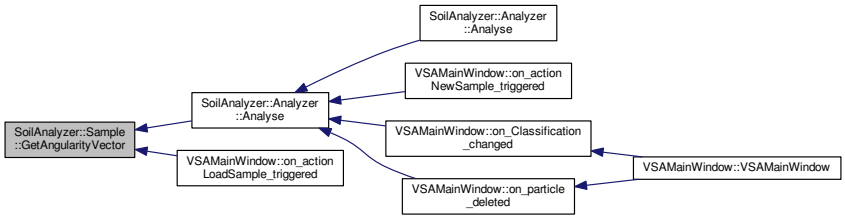
6.57.3.1 Particle::ClassVector_t * SoilAnalyzer::Sample::GetAngularityVector ( )

Definition at line 72 of file sample.cpp.

References AngularityGathered, AngularityVec, ParticleChangedStateAngularity, and ParticlePopulation.

Referenced by SoilAnalyzer::Analyzer::Analyse(), and VSAMainWindow::on_actionLoadSample_triggered().

Here is the caller graph for this function:



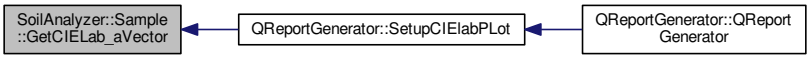
6.57.3.2 Particle::doubleVector_t * SoilAnalyzer::Sample::GetCIELab_aVector ( )

Definition at line 94 of file sample.cpp.

References CIELab_aGathered, CIELab_aVec, ColorChange, and ParticlePopulation.

Referenced by QReportGenerator::SetupCIElabPLot().

Here is the caller graph for this function:



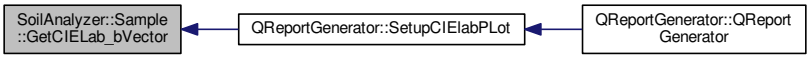
6.57.3.3 Particle::doubleVector_t * SoilAnalyzer::Sample::GetCIELab_bVector ( )

Definition at line 104 of file sample.cpp.

References CIELab_bGathered, CIELab_bVec, ColorChange, and ParticlePopulation.

Referenced by QReportGenerator::SetupCIElabPLot().

Here is the caller graph for this function:



6.57.3.4 Particle::PSDVector_t * SoilAnalyzer::Sample::GetPSDVector ( )

Sample::GetPSDVector.

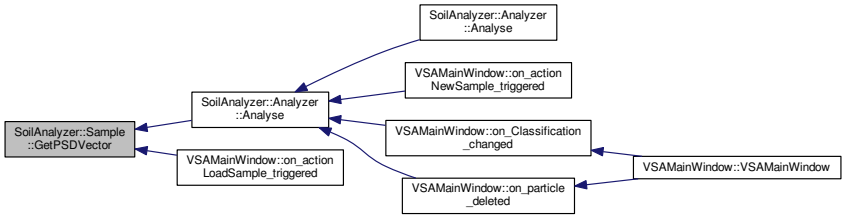
Returns

Definition at line 61 of file sample.cpp.

References Diameter, ParticleChangedStatePSD, ParticlePopulation, and PSDGathered.

Referenced by SoilAnalyzer::Analyzer::Analyse(), and VSAMainWindow::on_actionLoadSample_triggered().

Here is the caller graph for this function:



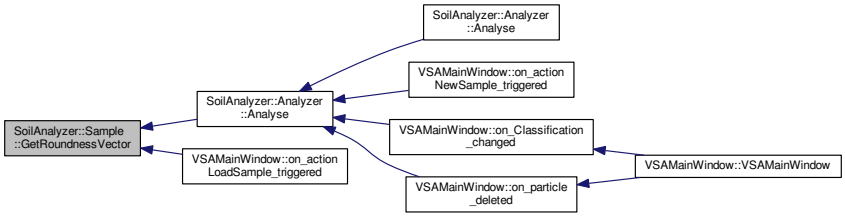
6.57.3.5 Particle::ClassVector_t * SoilAnalyzer::Sample::GetRoundnessVector ( )

Definition at line 83 of file sample.cpp.

References ParticleChangedStateRoundness, ParticlePopulation, RoundnessGathered, and RoundnessVec.

Referenced by SoilAnalyzer::Analyzer::Analyse(), and VSAMainWindow::on_actionLoadSample_triggered().

Here is the caller graph for this function:



6.57.3.6 void SoilAnalyzer::Sample::Load (const std::string & *filename*)

Sample::Load.

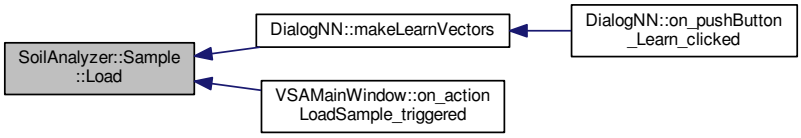
Parameters

filename	

Definition at line 42 of file sample.cpp.

Referenced by DialogNN::makeLearnVectors(), and VSAMainWindow::on_actionLoadSample_triggered().

Here is the caller graph for this function:



6.57.3.7 void SoilAnalyzer::Sample::Save (const std::string & *filename*)

Sample::Save.

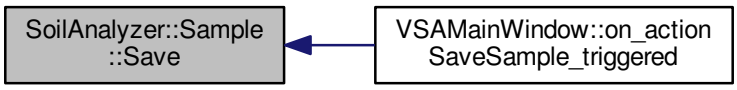
Parameters

filename	

Definition at line 23 of file sample.cpp.

Referenced by VSAMainWindow::on_actionSaveSample_triggered().

Here is the caller graph for this function:



6.57.3.8 template<class Archive > void SoilAnalyzer::Sample::serialize (Archive & *ar,* const unsigned int *version*) [inline],[private]

Definition at line 85 of file sample.h.

References Angularity, AngularityGathered, AngularityVec, ChangesSinceLastSave, CIELab_aGathered, CIELab_aVec, CIELab_bGathered, CIELab_bVec, ColorChange, Date, Depth, Diameter, ID, isAnalysed, IsLoadedFromDisk, isPreparedForAnalysis, Latitude, Location, Longtitude, Name, ParticleChangedStateAngularity, ParticleChangedStateClass, ParticleChangedStatePSD, ParticleChangedStateRoundness, ParticlePopulation, PSD, PSDGathered, RI, Roundness, RoundnessGathered, and RoundnessVec.

6.57.4 Friends And Related Function Documentation

6.57.4.1 friend class boost::serialization::access [friend]

Definition at line 83 of file sample.h.

6.57.5 Member Data Documentation

6.57.5.1 ucharStat_t SoilAnalyzer::Sample::Angularity

Definition at line 45 of file sample.h.

Referenced by SoilAnalyzer::Analyzer::Analyse(), VSAMainWindow::on_actionLoadSample_triggered(), QReportGenerator::QReportGenerator(), serialize(), and VSAMainWindow::setAngularityHistogram().

6.57.5.2 bool SoilAnalyzer::Sample::AngularityGathered = false [private]

Definition at line 77 of file sample.h.

Referenced by GetAngularityVector(), and serialize().

6.57.5.3 Particle::ClassVector_t SoilAnalyzer::Sample::AngularityVec [private]

Definition at line 76 of file sample.h.

Referenced by GetAngularityVector(), and serialize().

6.57.5.4 bool SoilAnalyzer::Sample::ChangesSinceLastSave = false

Definition at line 62 of file sample.h.

Referenced by VSAMainWindow::on_actionLoadSample_triggered(), VSAMainWindow::on_actionNewSample_triggered(), VSAMainWindow::on_actionSaveSample_triggered(), VSAMainWindow::on_Classification_changed(), QParticleDisplay::on_pushButton_delete_clicked(), and serialize().

6.57.5.5 bool SoilAnalyzer::Sample::CIELab_aGathered = false [private]

Definition at line 79 of file sample.h.

Referenced by GetCIELab_aVector(), and serialize().

6.57.5.6 Particle::doubleVector_t SoilAnalyzer::Sample::CIELab_aVec [private]

Definition at line 78 of file sample.h.

Referenced by GetCIELab_aVector(), and serialize().

6.57.5.7 bool SoilAnalyzer::Sample::CIELab_bGathered = false [private]

Definition at line 81 of file sample.h.

Referenced by GetCIELab_bVector(), and serialize().

6.57.5.8 Particle::doubleVector_t SoilAnalyzer::Sample::CIELab_bVec [private]

Definition at line 80 of file sample.h.

Referenced by GetCIELab_bVector(), and serialize().

6.57.5.9 bool SoilAnalyzer::Sample::ColorChange = false

Definition at line 67 of file sample.h.

Referenced by GetCIELab_aVector(), GetCIELab_bVector(), QParticleDisplay::on_pushButton_delete_clicked(), and serialize().

6.57.5.10 std::string SoilAnalyzer::Sample::Date = "01-09-2015"

Definition at line 37 of file sample.h.

Referenced by QReportGenerator::QReportGenerator(), and serialize().

6.57.5.11 double SoilAnalyzer::Sample::Depth = 0

Definition at line 36 of file sample.h.

Referenced by QReportGenerator::QReportGenerator(), and serialize().

6.57.5.12 Particle::PSDVector_t SoilAnalyzer::Sample::Diameter [private]

The PSD raw data

Definition at line 72 of file sample.h.

Referenced by GetPSDVector(), and serialize().

6.57.5.13 uint32_t SoilAnalyzer::Sample::ID

The sample ID

Definition at line 32 of file sample.h.

Referenced by QReportGenerator::QReportGenerator(), and serialize().

6.57.5.14 bool SoilAnalyzer::Sample::isAnalysed = false

is the sample analyzed

Definition at line 60 of file sample.h.

Referenced by serialize().

6.57.5.15 bool SoilAnalyzer::Sample::IsLoadedFromDisk = false

Definition at line 69 of file sample.h.

Referenced by SoilAnalyzer::Analyzer::Analyse(), VSAMainWindow::on_actionSaveSample_triggered(), and serialize().

6.57.5.16 bool SoilAnalyzer::Sample::isPreparedForAnalysis

Initial value:

```
=
    false
```

is the sample ready for analysis, are all the particles extracted

Definition at line 57 of file sample.h.

Referenced by SoilAnalyzer::Analyzer::Analyse(), SoilAnalyzer::Analyzer::PrepImages(), and serialize().

6.57.5.17 double SoilAnalyzer::Sample::Latitude = 51.8849149

Definition at line 35 of file sample.h.

Referenced by QReportGenerator::QReportGenerator(), and serialize().

6.57.5.18 std::string SoilAnalyzer::Sample::Location

The Location where the sample was taken

Definition at line 33 of file sample.h.

Referenced by serialize().

6.57.5.19 double SoilAnalyzer::Sample::Longtitude = 4.629618299999947

Definition at line 34 of file sample.h.

Referenced by QReportGenerator::QReportGenerator(), and serialize().

6.57.5.20 std::string SoilAnalyzer::Sample::Name

The sample name identifier

Definition at line 38 of file sample.h.

Referenced by QReportGenerator::QReportGenerator(), and serialize().

6.57.5.21 bool SoilAnalyzer::Sample::ParticleChangedStateAngularity = false

Definition at line 66 of file sample.h.

Referenced by GetAngularityVector(), VSAMainWindow::on_Classification_changed(), QParticleDisplay::on_pushButton_delete_clicked(), and serialize().

6.57.5.22 bool SoilAnalyzer::Sample::ParticleChangedStateClass = false

Definition at line 64 of file sample.h.

Referenced by serialize().

6.57.5.23 bool SoilAnalyzer::Sample::ParticleChangedStatePSD = false

Definition at line 63 of file sample.h.

Referenced by GetPSDVector(), QParticleDisplay::on_pushButton_delete_clicked(), and serialize().

6.57.5.24 bool SoilAnalyzer::Sample::ParticleChangedStateRoundness = false

Definition at line 65 of file sample.h.

Referenced by GetRoundnessVector(), VSAMainWindow::on_Classification_changed(), QParticleDisplay::on_pushButton_delete_clicked(), and serialize().

6.57.5.25 Particle::ParticleVector_t SoilAnalyzer::Sample::ParticlePopulation

the individual particles of the sample

Definition at line 41 of file sample.h.

Referenced by SoilAnalyzer::Analyzer::Analyse(), SoilAnalyzer::Analyzer::CalcMaxProgressAnalyze(), GetAngularityVector(), GetCIELab_aVector(), GetCIELab_bVector(), GetPSDVector(), GetRoundnessVector(), DialogNN::makeLearnVectors(), VSAMainWindow::on_actionNewSample_triggered(), VSAMainWindow::on_analyzer_finished(), QParticleDisplay::on_pushButton_delete_clicked(), QParticleDisplay::on_selectedParticleChangedSlider(), QParticleDisplay::on_selectedParticleChangedWidget(), SoilAnalyzer::Analyzer::PrepImages(), serialize(), and QParticleDisplay::SetSample().

6.57.5.26 SoilMath::PSD SoilAnalyzer::Sample::PSD

The Particle Size Distribution

Definition at line 43 of file sample.h.

Referenced by SoilAnalyzer::Analyzer::Analyse(), VSAMainWindow::on_actionLoadSample_triggered(), QReportGenerator::QReportGenerator(), serialize(), and VSAMainWindow::SetPSDgraph().

6.57.5.27 bool SoilAnalyzer::Sample::PSDGathered = false [private]

is the raw data gathered

Definition at line 73 of file sample.h.

Referenced by GetPSDVector(), and serialize().

6.57.5.28 floatStat_t SoilAnalyzer::Sample::RI

The statistical Redness Index data

Definition at line 46 of file sample.h.

Referenced by serialize().

6.57.5.29 ucharStat_t SoilAnalyzer::Sample::Roundness

Definition at line 44 of file sample.h.

Referenced by SoilAnalyzer::Analyzer::Analyse(), VSAMainWindow::on_actionLoadSample_triggered(), QReportGenerator::QReportGenerator(), serialize(), and VSAMainWindow::setRoundnessHistogram().

6.57.5.30 bool SoilAnalyzer::Sample::RoundnessGathered = false [private]

Definition at line 75 of file sample.h.

Referenced by GetRoundnessVector(), and serialize().

6.57.5.31 Particle::ClassVector_t SoilAnalyzer::Sample::RoundnessVec [private]

Definition at line 74 of file sample.h.

Referenced by GetRoundnessVector(), and serialize().

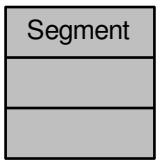
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/sample.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/sample.cpp

6.58 Segment Class Reference

Segmentation algorithms With this class, various segmentation routines can be applied to a greyscale or black and white source image.

Collaboration diagram for Segment:



6.58.1 Detailed Description

Segmentation algorithms With this class, various segmentation routines can be applied to a greyscale or black and white source image.

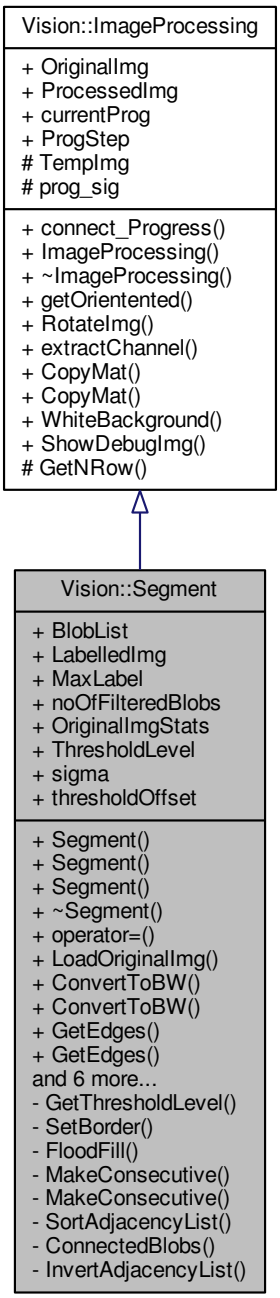
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.cpp

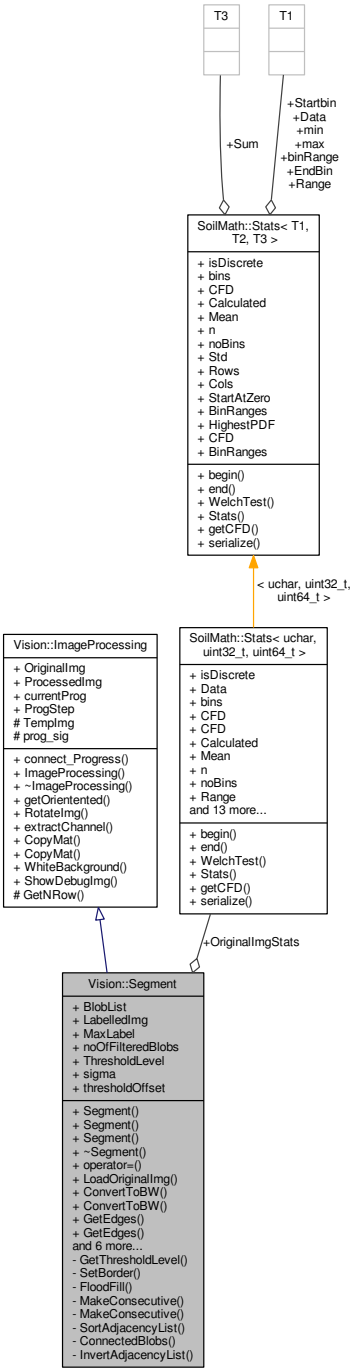
6.59 Vision::Segment Class Reference

```
#include <Segment.h>
```

Inheritance diagram for Vision::Segment:



Collaboration diagram for Vision::Segment:



Classes

- struct Blob
- struct Rect

Public Types

- enum TypeOfObjects { Bright, Dark }
- enum Connected { Four, Eight }
- enum SegmentationType { Normal, LabNeuralNet, GraphMinCut }
- typedef struct Vision::Segment::Rect Rect_t
- typedef std::vector< Vision::Segment::Rect_t > RectList_t
- typedef struct Vision::Segment::Blob Blob_t
- typedef std::vector< Blob_t > BlobList_t

Public Member Functions

- Segment ()

  Constructor of the Segmentation class.
- Segment (const Mat &src)

  Constructor of the Segmentation class.
- Segment (const Segment &rhs)
- ~Segment ()

  De-constructor.
- Segment & operator= (Segment &rhs)
- void LoadOriginalImg (const Mat &src)
- void ConvertToBW (TypeOfObjects Typeobjects)
- void ConvertToBW (const Mat &src, Mat &dst, TypeOfObjects Typeobjects)
- void GetEdges (bool chain=false, Connected conn=Eight)
- void GetEdges (const Mat &src, Mat &dst, bool chain=false, Connected conn=Eight)
- void GetEdgesEroding (bool chain=false)
- void GetBlobList (bool chain=false, Connected conn=Eight)
- void Threshold (uchar t, TypeOfObjects Typeobjects)
- void LabelBlobs (bool chain=false, uint16_t minBlobArea=25, Connected conn=Eight)
- void RemoveBorderBlobs (uint32_t border=1, bool chain=false)
- void FillHoles (bool chain=false)

Public Attributes

- BlobList_t BlobList
- cv::Mat LabelledImg
- uint16_t MaxLabel = 0
- uint16_t noOfFilteredBlobs
- ucharStat_t OriginalImgStats
- uint8_t ThresholdLevel = 0
- float sigma = 2
- uint32_t thresholdOffset = 4

Private Member Functions

- uint8_t GetThresholdLevel (TypeOfObjects TypeObject)
- void SetBorder (uchar *P, uchar setValue)
- void FloodFill (uchar *O, uchar *P, uint16_t x, uint16_t y, uchar fillValue, uchar OldValue)
- void MakeConsecutive (uint16_t *valueArr, uint32_t noElem, uint16_t &maxlabel)

  Segment::MakeConsecutive make the valueArr consequative numbers.
- void MakeConsecutive (uint16_t *valueArr, uint16_t *keyArr, uint16_t noElem, uint16_t &maxlabel)

  Segment::MakeConsecutive probably a fault in this function. Don't use.
- void SortAdjacencyList (std::vector< std::vector< uint16_t >> &adj)

  Segment::SortAdjacencyList Sort the the sub vectors.
- void ConnectedBlobs (uchar *O, uint16_t *P, std::vector< std::vector< uint16_t >> &adj, uint32_t nCols, uint32_t nRows, Connected conn)

  Segment::ConnectedBlobs Connect all the blobs and created the adjacency list.
- void InvertAdjacencyList (std::vector< std::vector< uint16_t >> &adj, std::vector< std::vector< uint16_t >> &adjInv)

  Segment::InvertAdjacencyList invert the adjecencylist for upstream (unused)

Additional Inherited Members

6.59.1 Detailed Description

Definition at line 27 of file Segment.h.

6.59.2 Member Typedef Documentation

6.59.2.1 typedef struct Vision::Segment::Blob Vision::Segment::Blob_t

Individual blob

6.59.2.2 typedef std::vector<Blob_t> Vision::Segment::BlobList_t

Definition at line 54 of file Segment.h.

6.59.2.3 typedef struct Vision::Segment::Rect Vision::Segment::Rect_t

Coordinates for the region of interest

6.59.2.4 typedef std::vector<Vision::Segment::Rect_t> Vision::Segment::RectList_t

Definition at line 39 of file Segment.h.

6.59.3 Member Enumeration Documentation

6.59.3.1 enum Vision::Segment::Connected

Enumerator to indicate how the pixel correlate between each other in a blob

Enumerator

Four Enum Four connected, relation between Center, North, East, South and West

Eight Enum Eight connected, relation between Center, North, NorthEast, East, SouthEast, South, SouthWest, West and NorthWest

Definition at line 65 of file Segment.h.

6.59.3.2 enum Vision::Segment::SegmentationType

Enumerator

Normal Segmentation looking at the intensity of an individual pixel

LabNeuralNet Segmentation looking at the chromatic a∗ and b∗ of the processed pixel and it's surrounding pixels, feeding it in an Neural Net

GraphMinCut Segmentation using a graph function and the minimum cut

Definition at line 75 of file Segment.h.

6.59.3.3 enum Vision::Segment::TypeOfObjects

Enumerator to indicate what kind of object to extract

Enumerator

Bright Enum value Bright object

Dark Enum value Dark object.

Definition at line 58 of file Segment.h.

6.59.4 Constructor & Destructor Documentation

6.59.4.1 Segment::Segment ( )

Constructor of the Segmentation class.

Definition at line 17 of file Segment.cpp.

6.59.4.2 Segment::Segment (const Mat & *src*)

Constructor of the Segmentation class.

Definition at line 20 of file Segment.cpp.

References LabelledImg, Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

6.59.4.3 Segment::Segment (const Segment & *rhs*)

Definition at line 26 of file Segment.cpp.

References BlobList, LabelledImg, MaxLabel, noOfFilteredBlobs, Vision::ImageProcessing::OriginalImg, OriginalImgStats, Vision::ImageProcessing::ProcessedImg, Vision::ImageProcessing::TempImg, and ThresholdLevel.

6.59.4.4 Segment::~Segment ( )

De-constructor.

Definition at line 39 of file Segment.cpp.

6.59.5 Member Function Documentation

6.59.5.1 void Segment::ConnectedBlobs (uchar * *O*, uint16_t * *P*, std::vector< std::vector< uint16_t >> & *adj*, uint32_t *nCols*, uint32_t *nRows*, Connected *conn*) [private]

Segment::ConnectedBlobs Connect all the blobs and created the adjacency list.

Parameters

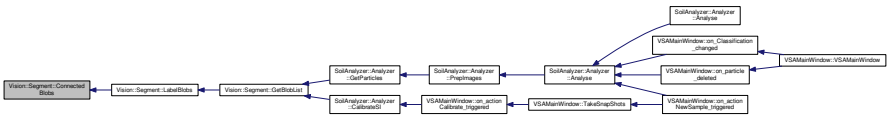
O	
P	
adj	
nCols	
nRows	
conn	

Definition at line 688 of file Segment.cpp.

References Four, and Vision::ImageProcessing::OriginalImg.

Referenced by LabelBlobs().

Here is the caller graph for this function:



6.59.5.2 void Segment::ConvertToBW (TypeOfObjects *Typeobjects*)

Convert a greyscale image to a BW using an automatic Threshold

Parameters

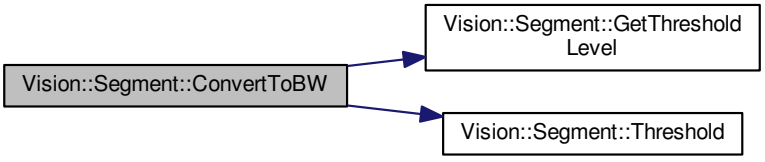
TypeObject	is an enumerator indicating if the bright or the dark pixels are the object and should be set to one

Definition at line 164 of file Segment.cpp.

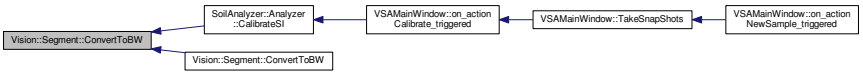
References GetThresholdLevel(), and Threshold().

Referenced by SoilAnalyzer::Analyzer::CalibrateSI(), and ConvertToBW().

Here is the call graph for this function:



Here is the caller graph for this function:



6.59.5.3 void Segment::ConvertToBW (const Mat & *src*, Mat & *dst*, TypeOfObjects *Typeobjects*)

Convert a greyscale image to a BW using an automatic Threshold

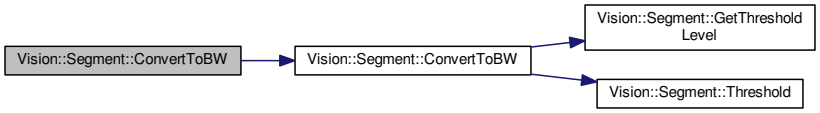
Parameters

src	is the source image as a cv::Mat
dst	destination image as a cv::Mat
TypeObject	is an enumerator indicating if the bright or the dark pixels are the object and should be set to one

Definition at line 153 of file Segment.cpp.

References ConvertToBW(), LabelledImg, Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

Here is the call graph for this function:



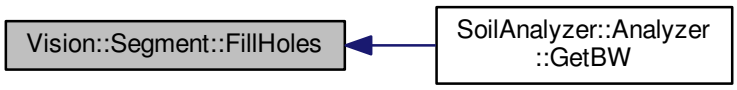
6.59.5.4 void Segment::FillHoles (bool *chain* = `false`)

Definition at line 615 of file Segment.cpp.

References CHAIN_PROCESS, EMPTY_CHECK, Vision::ImageProcessing::OriginalImg, Vision::ImageProcessing::ProcessedImg, and Vision::ImageProcessing::TempImg.

Referenced by SoilAnalyzer::Analyzer::GetBW().

Here is the caller graph for this function:



6.59.5.5 void Vision::Segment::FloodFill (uchar * *O*, uchar * *P*, uint16_t *x*, uint16_t *y*, uchar *fillValue*, uchar *OldValue*) `[private]`

6.59.5.6 void Segment::GetBlobList (bool *chain* = `false`, Connected *conn* = Eight)

Create a BlobList subtracting each individual blob out of a Labelled image. If the labelled image is empty build a new one with a BW image.

Parameters

conn	set the pixel connection eight or four
chain	use the results from the previous operation default value = false;

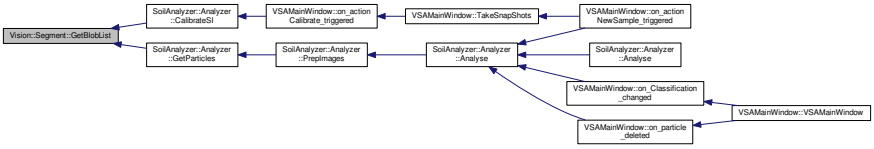
Definition at line 534 of file Segment.cpp.

References SoilMath::Stats< T1, T2, T3 >::bins, BlobList, EMPTY_CHECK, SoilMath::Stats< T1, T2, T3 >::EndBin, LabelBlobs(), LabelledImg, MaxLabel, and Vision::ImageProcessing::OriginalImg.

Referenced by SoilAnalyzer::Analyzer::CalibrateSI(), and SoilAnalyzer::Analyzer::GetParticles().

Here is the call graph for this function:

Here is the caller graph for this function:



6.59.5.7 void Segment::GetEdges (bool *chain* = `false`, Connected *conn* = Eight)

Create a BW image with only edges from a BW image

Parameters

conn	set the pixel connection eight or four
chain	use the results from the previous operation default value = false;

Definition at line 399 of file Segment.cpp.

References CHAIN_PROCESS, EMPTY_CHECK, Four, Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

Referenced by GetEdges().

Here is the caller graph for this function:

6.59.5.8 void Segment::GetEdges (const Mat & *src*, Mat & *dst*, bool *chain* = `false`, Connected *conn* = Eight)

Create a BW image with only edges from a BW image

Parameters

src	source image as a const cv::Mat
dst	destination image as a cv::Mat
conn	set the pixel connection eight or four
chain	use the results from the previous operation default value = false;

Definition at line 389 of file Segment.cpp.

References GetEdges(), Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

Here is the call graph for this function:

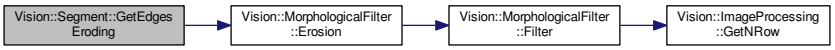
6.59.5.9 void Segment::GetEdgesEroding (bool *chain* = `false`)

Definition at line 483 of file Segment.cpp.

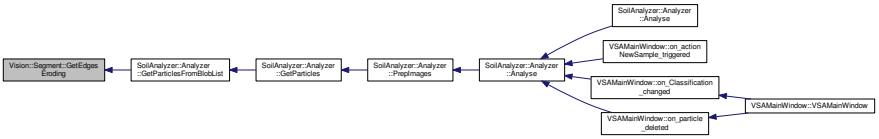
References CHAIN_PROCESS, EMPTY_CHECK, Vision::MorphologicalFilter::Erosion(), Vision::ImageProcessing::OriginalImg, Vision::↩ ImageProcessing::ProcessedImg, SHOW_DEBUG_IMG, and Vision::ImageProcessing::TempImg.

Referenced by SoilAnalyzer::Analyzer::GetParticlesFromBlobList().

Here is the call graph for this function:



Here is the caller graph for this function:



6.59.5.10 uint8_t Segment::GetThresholdLevel (TypeOfObjects *TypeObject*) [private]

Determine the threshold level by iteration, between two distribution, presumably back- and foreground. It works towards the average of the two averages and finally sets the threshold with two time the standard deviation from the mean of the set object

Parameters

TypeObject	is an enumerator indicating if the bright or the dark pixels are the object and should be set to one

Returns

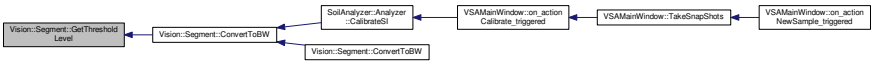
The threshold level as an uint8_t

Definition at line 69 of file Segment.cpp.

References SoilMath::Stats< T1, T2, T3 >::bins, Bright, Dark, EMPTY_CHECK, SoilMath::Stats< T1, T2, T3 >::Mean, Vision::Image↩ Processing::OriginalImg, OriginalImgStats, sigma, SoilMath::Stats< T1, T2, T3 >::Std, and thresholdOffset.

Referenced by ConvertToBW().

Here is the caller graph for this function:



6.59.5.11 void Segment::InvertAdjacencyList (std::vector< std::vector< uint16_t >> & *adj*, std::vector< std::vector< uint16_t >> & *adjInv*) [private]

Segment::InvertAdjacencyList invert the adjecencylist for upstream (unused)

Parameters

adj	
adjInv	

Definition at line 801 of file Segment.cpp.

6.59.5.12 void Segment::LabelBlobs (bool *chain* = `false`, uint16_t *minBlobArea* = `25`, Connected *conn* = Eight)

Label all the individual blobs in a BW source image. The result are written to the labelledImg as an ushort

Parameters

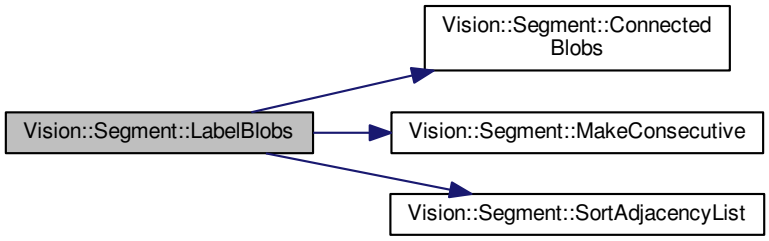
conn	set the pixel connection eight or four
chain	use the results from the previous operation default value = false;
minBlobArea	minimum area when an artifact is considered a blob

Definition at line 316 of file Segment.cpp.

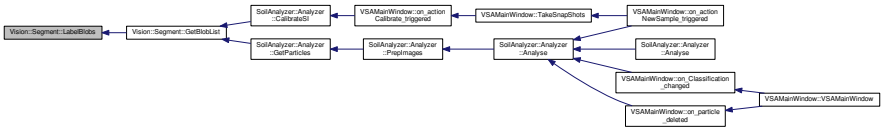
References ConnectedBlobs(), EMPTY_CHECK, LabelledImg, MakeConsecutive(), MaxLabel, Vision::ImageProcessing::OriginalImg, Vision::ImageProcessing::ProcessedImg, SortAdjacencyList(), and Vision::ImageProcessing::TempImg.

Referenced by GetBlobList().

Here is the call graph for this function:



Here is the caller graph for this function:



6.59.5.13 void Segment::LoadOriginalImg (const Mat & *src*)

Definition at line 56 of file Segment.cpp.

References LabelledImg, Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

6.59.5.14 void Segment::MakeConsecutive (uint16_t * *valueArr*, uint32_t *noElem*, uint16_t & *maxLabel*) [private]

Segment::MakeConsecutive make the valueArr consequative numbers.

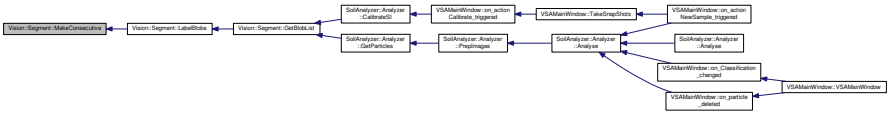
Parameters

valueArr	
noElem	
maxLabel	

Definition at line 819 of file Segment.cpp.

Referenced by LabelBlobs().

Here is the caller graph for this function:



6.59.5.15 void Segment::MakeConsecutive (uint16_t * *valueArr*, uint16_t * *keyArr*, uint16_t *noElem*, uint16_t & *maxlabel*) [private]

Segment::MakeConsecutive probably a fault in this function. Don't use.

Parameters

valueArr	
keyArr	
noElem	
maxlabel	

Definition at line 845 of file Segment.cpp.

6.59.5.16 Segment & Segment::operator= (Segment & *rhs*)

Definition at line 41 of file Segment.cpp.

References BlobList, LabelledImg, MaxLabel, noOfFilteredBlobs, Vision::ImageProcessing::OriginalImg, OriginalImgStats, Vision::ImageProcessing::ProcessedImg, Vision::ImageProcessing::TempImg, and ThresholdLevel.

6.59.5.17 void Segment::RemoveBorderBlobs (uint32_t *border* = `1`, bool *chain* = `false`)

Remove the blobs that are connected to the border

Parameters

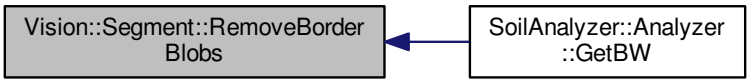
conn	set the pixel connection eight or four
chain	use the results from the previous operation default value = false;

Definition at line 245 of file Segment.cpp.

References CHAIN_PROCESS, EMPTY_CHECK, Vision::ImageProcessing::OriginalImg, Vision::ImageProcessing::ProcessedImg, SHOW_DEBUG_IMG, and Vision::ImageProcessing::TempImg.

Referenced by SoilAnalyzer::Analyzer::GetBW().

Here is the caller graph for this function:



6.59.5.18 void Segment::SetBorder (uchar * *P*, uchar *setValue*) [private]

Set all the border pixels to a set value

Parameters

**P*	uchar pointer to the Mat.data
setValue	uchar the value which is written to the border pixels

Definition at line 208 of file Segment.cpp.

References EMPTY_CHECK, and Vision::ImageProcessing::OriginalImg.

6.59.5.19 void Segment::SortAdjacencyList (std::vector< std::vector< uint16_t >> & *adj*) [private]

Segment::SortAdjacencyList Sort the the sub vectors.

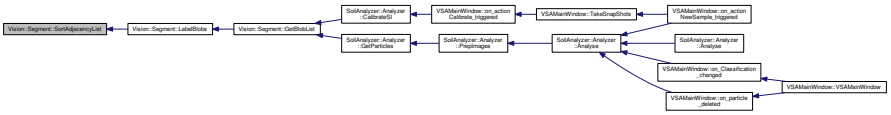
Parameters

adj	std::vector<std::vector<uint16_t>> &adj

Definition at line 658 of file Segment.cpp.

Referenced by LabelBlobs().

Here is the caller graph for this function:



6.59.5.20 void Segment::Threshold (uchar *t*, TypeOfObjects *Typeobjects*)

Convert a greyscale image to a BW

Parameters

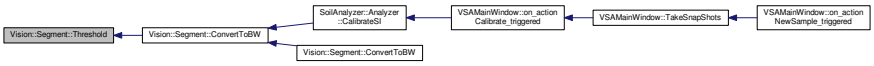
t	uchar set the value which is the tiping point
TypeObject	is an enumerator indicating if the bright or the dark pixels are the object and should be set to one

Definition at line 176 of file Segment.cpp.

References Bright, EMPTY_CHECK, Vision::ImageProcessing::OriginalImg, and Vision::ImageProcessing::ProcessedImg.

Referenced by ConvertToBW().

Here is the caller graph for this function:



6.59.6 Member Data Documentation

6.59.6.1 BlobList_t Vision::Segment::BlobList

vector with all the individual blobs

Definition at line 55 of file Segment.h.

Referenced by SoilAnalyzer::Analyzer::CalibrateSI(), GetBlobList(), SoilAnalyzer::Analyzer::GetParticles(), operator=(), and Segment().

6.59.6.2 cv::Mat Vision::Segment::LabelledImg

Image with each individual blob labeled with a individual number

Definition at line 83 of file Segment.h.

Referenced by ConvertToBW(), GetBlobList(), LabelBlobs(), LoadOriginalImg(), operator=(), and Segment().

6.59.6.3 uint16_t Vision::Segment::MaxLabel = 0

Maximum labels found in the labelled image

Definition at line 85 of file Segment.h.

Referenced by GetBlobList(), LabelBlobs(), operator=(), and Segment().

6.59.6.4 uint16_t Vision::Segment::noOfFilteredBlobs

Initial value:

=

0

Total numbers of blobs that where filtered beacuse the where smaller than the minBlobArea

Definition at line 86 of file Segment.h.

Referenced by operator=(), and Segment().

6.59.6.5 ucharStat_t Vision::Segment::OriginalImgStats

Statistical data from the original image

Definition at line 90 of file Segment.h.

Referenced by GetThresholdLevel(), operator=(), and Segment().

6.59.6.6 float Vision::Segment::sigma = 2

Definition at line 93 of file Segment.h.

Referenced by SoilAnalyzer::Analyzer::GetBW(), and GetThresholdLevel().

6.59.6.7 uint8_t Vision::Segment::ThresholdLevel = 0

Current calculated threshold level

Definition at line 91 of file Segment.h.

Referenced by operator=(), and Segment().

6.59.6.8 uint32_t Vision::Segment::thresholdOffset = 4

Definition at line 94 of file Segment.h.

Referenced by GetThresholdLevel().

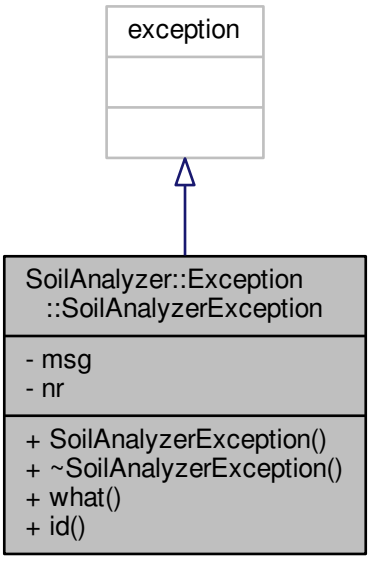
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/Segment.cpp

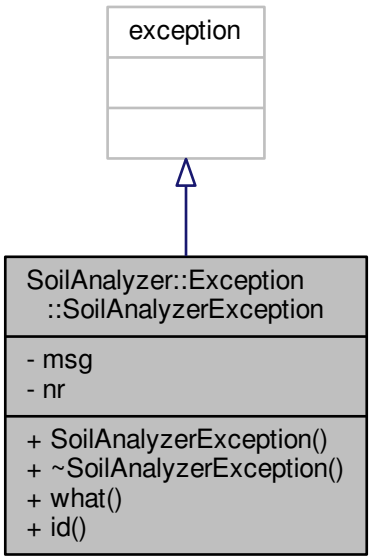
6.60 SoilAnalyzer::Exception::SoilAnalyzerException Class Reference

`#include <soilanalyzerexception.h>`

Inheritance diagram for SoilAnalyzer::Exception::SoilAnalyzerException:



Collaboration diagram for SoilAnalyzer::Exception::SoilAnalyzerException:



Public Member Functions

- SoilAnalyzerException (std::string m=EXCEPTION_PARTICLE_NOT_ANALYZED, int n=EXCEPTION_PARTICLE_NOT_ANALYZED↩ _NR)
- ~SoilAnalyzerException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT
- const int * id () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- std::string msg
- int nr

6.60.1 Detailed Description

Definition at line 20 of file soilanalyzerexception.h.

6.60.2 Constructor & Destructor Documentation

6.60.2.1 SoilAnalyzer::Exception::SoilAnalyzerException::SoilAnalyzerException (std::string *m* = EXCEPTION_PARTICLE_NOT_ANALYZED, int *n* = EXCEPTION_PARTICLE_NOT_ANALYZED_NR) [inline]

Definition at line 22 of file soilanalyzerexception.h.

6.60.2.2 SoilAnalyzer::Exception::SoilAnalyzerException::~SoilAnalyzerException ( ) [inline]

Definition at line 24 of file soilanalyzerexception.h.

6.60.3 Member Function Documentation

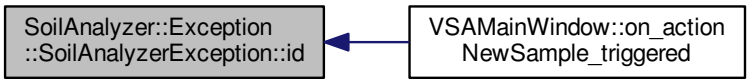
6.60.3.1 const int* SoilAnalyzer::Exception::SoilAnalyzerException::id ( ) const [inline]

Definition at line 26 of file soilanalyzerexception.h.

References nr.

Referenced by VSAMainWindow::on_actionNewSample_triggered().

Here is the caller graph for this function:



6.60.3.2 const char* SoilAnalyzer::Exception::SoilAnalyzerException::what ( ) const [inline]

Definition at line 25 of file soilanalyzerexception.h.

References msg.

6.60.4 Member Data Documentation

6.60.4.1 std::string SoilAnalyzer::Exception::SoilAnalyzerException::msg [private]

Definition at line 29 of file soilanalyzerexception.h.

Referenced by what().

6.60.4.2 int SoilAnalyzer::Exception::SoilAnalyzerException::nr [private]

Definition at line 30 of file soilanalyzerexception.h.

Referenced by id().

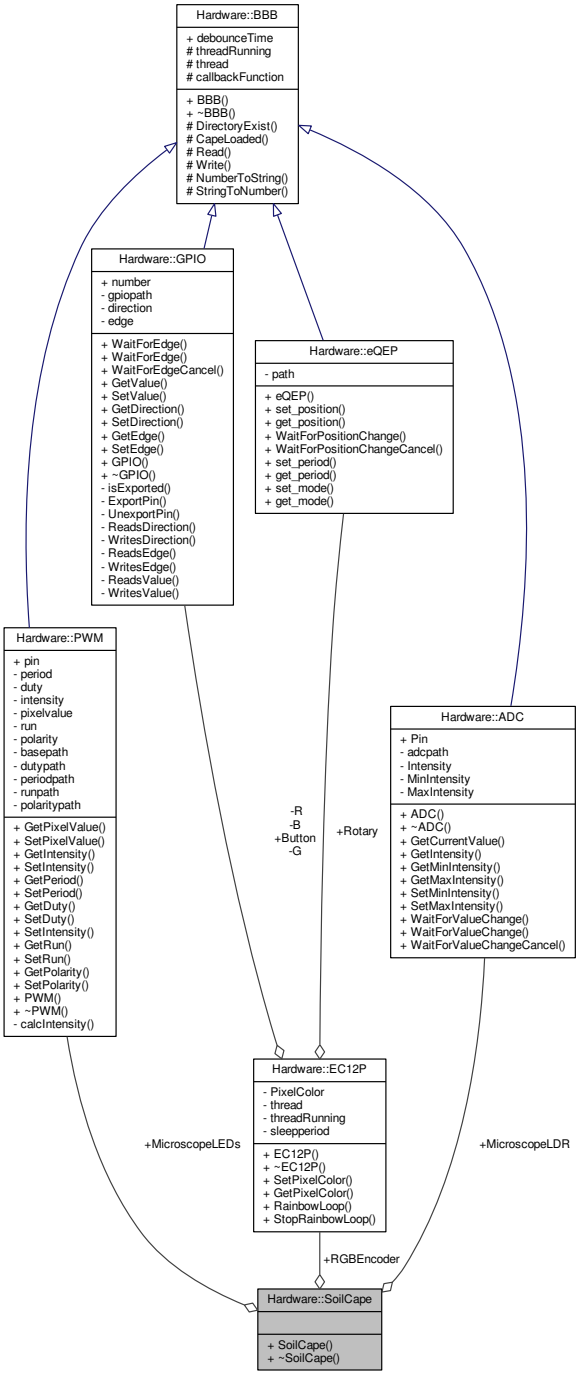
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilanalyzerexception.h

6.61 Hardware::SoilCape Class Reference

```
#include <SoilCape.h>
```

Collaboration diagram for Hardware::SoilCape:



Public Member Functions

- SoilCape ()
- ~SoilCape ()

Public Attributes

- EC12P RGBEncoder
- PWM MicroscopeLEDs {PWM::P9_14}
- ADC MicroscopeLDR {ADC::ADC0}

6.61.1 Detailed Description

Definition at line 16 of file SoilCape.h.

6.61.2 Constructor & Destructor Documentation

6.61.2.1 Hardware::SoilCape::SoilCape ( )

Definition at line 11 of file SoilCape.cpp.

6.61.2.2 Hardware::SoilCape::~SoilCape ( )

Definition at line 13 of file SoilCape.cpp.

6.61.3 Member Data Documentation

6.61.3.1 ADC Hardware::SoilCape::MicroscopeLDR {ADC::ADC0}

Definition at line 20 of file SoilCape.h.

6.61.3.2 PWM Hardware::SoilCape::MicroscopeLEDs {PWM::P9_14}

Definition at line 19 of file SoilCape.h.

6.61.3.3 EC12P Hardware::SoilCape::RGBEncoder

Definition at line 18 of file SoilCape.h.

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

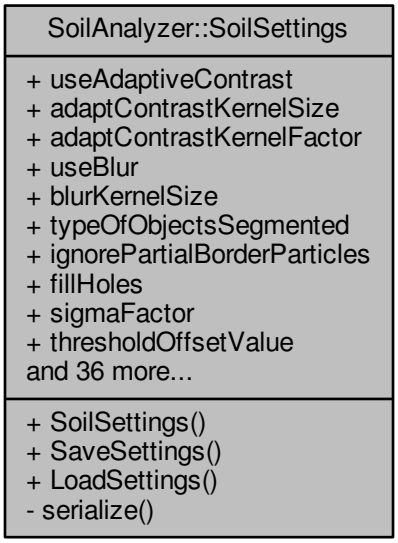
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/SoilCape.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/SoilCape.cpp

6.62 SoilAnalyzer::SoilSettings Class Reference

The SoilSettings class.

`#include <soilsettings.h>`

Collaboration diagram for SoilAnalyzer::SoilSettings:



Public Member Functions

- SoilSettings ()
- void SaveSettings (std::string filename)

  SaveSettings a function to save the settings to disk.

- void LoadSettings (std::string filename)

  LoadSettings a function to load the settings from disk.

Public Attributes

- bool useAdaptiveContrast
- uint32_t adaptContrastKernelSize
- float adaptContrastKernelFactor = 1.
- bool useBlur = false
- uint32_t blurKernelSize = 5
- Vision::Segment::TypeOfObjects typeOfObjectsSegmented
- bool ignorePartialBorderParticles
- bool fillHoles = true
- float sigmaFactor = 2
- int thresholdOffsetValue = 0
- Vision::MorphologicalFilter::FilterType morphFilterType
- uint32_t filterMaskSize = 5
- uint32_t HDRframes
- float lightLevel = 0.5
- bool encInv = false
- bool enableRainbow
- bool useBacklightProjection = true
- bool useHDR = false
- std::string defaultWebcam = "USB Microscope"
- int Brightness_front = 0
- int Brightness_proj = -10
- int Contrast_front = 36
- int Contrast_proj = 36
- int Saturation_front = 64
- int Saturation_proj = 0
- int Hue_front = 0
- int Hue_proj = -40
- int Gamma_front = 100
- int Gamma_proj = 200
- int PowerLineFrequency_front
- int PowerLineFrequency_proj
- int Sharpness_front = 12
- int Sharpness_proj = 25
- int BackLightCompensation_front
- int BackLightCompensation_proj
- std::string NNlocation = "NeuralNet/Default.NN"
- bool useCUDA = false
- int selectedResolution = 0
- std::string SampleFolder = "~/Samples"
- std::string SettingsFolder = "Settings"
- std::string NNFolder = "NeuralNet"
- std::string StandardSentTo = "j.spijker@ihcmerwede.com"
- std::string StandardPrinter = "PDF printer"
- uint32_t StandardNumberOfShots = 10
- bool PredictTheShape = true
- bool Revolution = true

Private Member Functions

- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

Friends

- class boost::serialization::access

6.62.1 Detailed Description

The SoilSettings class.

A class with which the used settings can easily transfered to setup the Sample class in one go. This class is also used in the GUI. and has a possibility to saved to disk as a serialized object

Definition at line 24 of file soilsettings.h.

6.62.2 Constructor & Destructor Documentation

6.62.2.1 SoilAnalyzer::SoilSettings::SoilSettings ( )

Definition at line 11 of file soilsettings.cpp.

6.62.3 Member Function Documentation

6.62.3.1 void SoilAnalyzer::SoilSettings::LoadSettings (std::string *filename*)

LoadSettings a function to load the settings from disk.

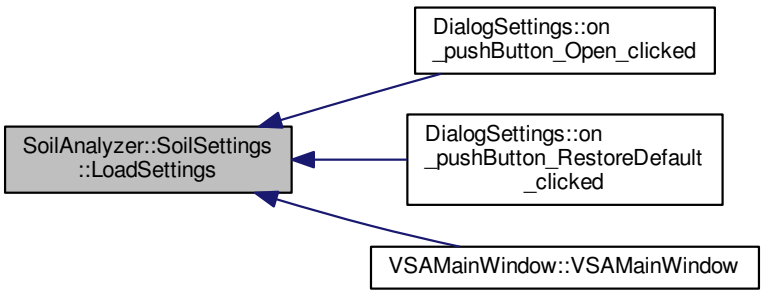
Parameters

filename	a string with the filename

Definition at line 13 of file soilsettings.cpp.

Referenced by DialogSettings::on_pushButton_Open_clicked(), DialogSettings::on_pushButton_RestoreDefault_clicked(), and VSAMainWindow::VSAMainWindow().

Here is the caller graph for this function:



6.62.3.2 void SoilAnalyzer::SoilSettings::SaveSettings (std::string *filename*)

SaveSettings a function to save the settings to disk.

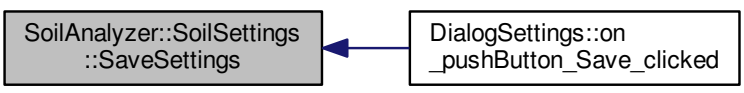
Parameters

filename	a string with the filename

Definition at line 19 of file soilsettings.cpp.

Referenced by DialogSettings::on_pushButton_Save_clicked().

Here is the caller graph for this function:



6.62.3.3 template<class Archive > void SoilAnalyzer::SoilSettings::serialize (Archive & *ar,* const unsigned int *version*) [inline], [private]

Definition at line 109 of file soilsettings.h.

6.62.4 Friends And Related Function Documentation

6.62.4.1 friend class boost::serialization::access [friend]

Definition at line 107 of file soilsettings.h.

6.62.5 Member Data Documentation

6.62.5.1 float SoilAnalyzer::SoilSettings::adaptContrastKernelFactor = 1.

the factor with which to multiply the effect of the adaptive contrast stretch

Definition at line 44 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings().

6.62.5.2 uint32_t SoilAnalyzer::SoilSettings::adaptContrastKernelSize

Initial value:

```
=
    9
```

The size of the adaptive contrast kernelsize

Definition at line 42 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings().

6.62.5.3 int SoilAnalyzer::SoilSettings::BackLightCompensation_front

Initial value:

```
=
    1
```

cam backlight compensation setting front light

Definition at line 91 of file soilsettings.h.

6.62.5.4 int SoilAnalyzer::SoilSettings::BackLightCompensation_proj

Initial value:

```
=
    1
```

cam backlight compensation setting projected light

Definition at line 93 of file soilsettings.h.

6.62.5.5 uint32_t SoilAnalyzer::SoilSettings::blurKernelSize = 5

the median blurkernel

Definition at line 49 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings().

6.62.5.6 int SoilAnalyzer::SoilSettings::Brightness_front = 0

cam brightness setting front light

Definition at line 75 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_BrightFront_valueChanged().

6.62.5.7 int SoilAnalyzer::SoilSettings::Brightness_proj = -10

cam brightness setting projected light

Definition at line 76 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_BrightProj_valueChanged().

6.62.5.8 int SoilAnalyzer::SoilSettings::Contrast_front = 36

cam contrast setting front light

Definition at line 77 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_ContrastFront_valueChanged().

6.62.5.9 int SoilAnalyzer::SoilSettings::Contrast_proj = 36

cam contrast setting projected light

Definition at line 78 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_ContrastProj_valueChanged().

6.62.5.10 std::string SoilAnalyzer::SoilSettings::defaultWebcam = "USB Microscope"

The defaultWebcam string

Definition at line 74 of file soilsettings.h.

Referenced by DialogSettings::on_comboBox_Microscopes_currentIndexChanged(), and VSAMainWindow::VSAMainWindow().

6.62.5.11 bool SoilAnalyzer::SoilSettings::enableRainbow

Initial value:

```
=
    true
```

run a rainbow loop on the RGB encoder during analysis

Definition at line 70 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_checkBox_useRainbow_clicked().

6.62.5.12 bool SoilAnalyzer::SoilSettings::enclnv = false

invert the values gained form the encoder

Definition at line 69 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_checkBox_InvertEncoder_clicked().

6.62.5.13 bool SoilAnalyzer::SoilSettings::fillHoles = true

should the holes be filled

Definition at line 55 of file soilsettings.h.

Referenced by SoilAnalyzer::Analyzer::CalcMaxProgress(), DialogSettings::DialogSettings(), SoilAnalyzer::Analyzer::GetBW(), and DialogSettings::on_cb_fillHoles_3_clicked().

6.62.5.14 uint32_t SoilAnalyzer::SoilSettings::filterMaskSize = 5

the filter mask

Definition at line 64 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), SoilAnalyzer::Analyzer::GetBW(), and DialogSettings::on_sb_morphMask_3_editingFinished().

6.62.5.15 int SoilAnalyzer::SoilSettings::Gamma_front = 100

cam gamma setting front light

Definition at line 83 of file soilsettings.h.

6.62.5.16 int SoilAnalyzer::SoilSettings::Gamma_proj = 200

cam gamma setting projected light

Definition at line 84 of file soilsettings.h.

6.62.5.17 uint32_t SoilAnalyzer::SoilSettings::HDRframes

Initial value:

```
=
    5
```

The number of frames which should be used for the HDR image

Definition at line 66 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), DialogSettings::on_spinBox_NoFrames_editingFinished(), and VSAMainWindow::TakeSnapShots().

6.62.5.18 int SoilAnalyzer::SoilSettings::Hue_front = 0

cam hue setting front light

Definition at line 81 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_HueFront_valueChanged().

6.62.5.19 int SoilAnalyzer::SoilSettings::Hue_proj = -40

cam hue setting projected light

Definition at line 82 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_HueProj_valueChanged().

6.62.5.20 bool SoilAnalyzer::SoilSettings::ignorePartialBorderParticles

Initial value:

```
=
    true
```

Indication of partial border particles should be used

Definition at line 53 of file soilsettings.h.

Referenced by SoilAnalyzer::Analyzer::CalcMaxProgress(), DialogSettings::DialogSettings(), SoilAnalyzer::Analyzer::GetBW(), and DialogSettings::on_cb_ignoreBorder_3_clicked().

6.62.5.21 float SoilAnalyzer::SoilSettings::lightLevel = 0.5

The light level of the environmental case

Definition at line 68 of file soilsettings.h.

Referenced by DialogSettings::on_doubleSpinBox_LightLevel_editingFinished().

6.62.5.22 Vision::MorphologicalFilter::FilterType SoilAnalyzer::SoilSettings::morphFilterType

Initial value:

```
=
    Vision::MorphologicalFilter::OPEN
```

Indicating which type of morhpological filter should be used

Definition at line 60 of file soilsettings.h.

Referenced by SoilAnalyzer::Analyzer::CalcMaxProgress(), DialogSettings::DialogSettings(), SoilAnalyzer::Analyzer::GetBW(), DialogSettings::on_rb_useClose_3_clicked(), DialogSettings::on_rb_useDilate_3_clicked(), DialogSettings::on_rb_useErode_3_clicked(), and DialogSettings::on_rb_useOpen_3_clicked().

6.62.5.23 std::string SoilAnalyzer::SoilSettings::NNFolder = "NeuralNet"

Definition at line 100 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), DialogNN::on_pushButton_SaveNN_clicked(), and DialogSettings::on_pushButton_SelectNNFolder_clicked().

6.62.5.24 std::string SoilAnalyzer::SoilSettings::NNlocation = "NeuralNet/Default.NN"

Definition at line 95 of file soilsettings.h.

Referenced by SoilAnalyzer::Analyzer::Analyzer(), DialogSettings::DialogSettings(), and DialogSettings::on_pushButton_SelectNN_clicked().

6.62.5.25 int SoilAnalyzer::SoilSettings::PowerLineFrequency_front

Initial value:

```
=
    1
```

cam powerline freq setting front light

Definition at line 85 of file soilsettings.h.

6.62.5.26 int SoilAnalyzer::SoilSettings::PowerLineFrequency_proj

Initial value:

```
=
    1
```

cam powerline freq setting projected light

Definition at line 87 of file soilsettings.h.

6.62.5.27 bool SoilAnalyzer::SoilSettings::PredictTheShape = true

Definition at line 104 of file soilsettings.h.

Referenced by SoilAnalyzer::Analyzer::Analyse(), DialogSettings::DialogSettings(), VSAMainWindow::on_actionAutomatic_Shape_Pediction_↩triggered(), DialogSettings::on_checkBox_PredictShape_clicked(), and VSAMainWindow::VSAMainWindow().

6.62.5.28 bool SoilAnalyzer::SoilSettings::Revolution = true

Definition at line 105 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_checkBox_revolt_clicked().

6.62.5.29 std::string SoilAnalyzer::SoilSettings::SampleFolder = "~/Samples"

Definition at line 98 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), QReportGenerator::on_actionExport_to_PDF_triggered(), VSAMainWindow::on_actionLoad↩Sample_triggered(), QReportGenerator::on_actionSave_triggered(), VSAMainWindow::on_actionSaveSample_triggered(), VSAMainWindow↩::on_compare_against(), DialogNN::on_pushButton_OpenNN_clicked(), DialogSettings::on_pushButton_selectSampleFolder_clicked(), and DialogNN::on_pushButton_SelectSamples_clicked().

6.62.5.30 int SoilAnalyzer::SoilSettings::Saturation_front = 64

cam saturation setting front light

Definition at line 79 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_SaturationFront_valueChanged().

6.62.5.31 int SoilAnalyzer::SoilSettings::Saturation_proj = 0

cam saturation setting projected light

Definition at line 80 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_SaturationProj_valueChanged().

6.62.5.32 int SoilAnalyzer::SoilSettings::selectedResolution = 0

Definition at line 97 of file soilsettings.h.

Referenced by DialogSettings::on_comboBox_Resolution_currentIndexChanged(), and VSAMainWindow::VSAMainWindow().

6.62.5.33 std::string SoilAnalyzer::SoilSettings::SettingsFolder = "Settings"

Definition at line 99 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_pushButton_SelectSettingFolder_clicked().

6.62.5.34 int SoilAnalyzer::SoilSettings::Sharpness_front = 12

cam sharpness setting front light

Definition at line 89 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_SharpnessFront_valueChanged().

6.62.5.35 int SoilAnalyzer::SoilSettings::Sharpness_proj = 25

cam sharpness setting projected light

Definition at line 90 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_horizontalSlider_SharpnessProj_valueChanged().

6.62.5.36 float SoilAnalyzer::SoilSettings::sigmaFactor = 2

The sigma factor or the bandwidth indicating which pixel intensity values count belong to an object

Definition at line 56 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), SoilAnalyzer::Analyzer::GetBW(), and DialogSettings::on_sb_sigmaFactor_3_editingFinished().

6.62.5.37 uint32_t SoilAnalyzer::SoilSettings::StandardNumberOfShots = 10

Definition at line 103 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), DialogSettings::on_spinBox_NoShots_editingFinished(), and VSAMainWindow::TakeSnapShots().

6.62.5.38 std::string SoilAnalyzer::SoilSettings::StandardPrinter = "PDF printer"

Definition at line 102 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings().

6.62.5.39 std::string SoilAnalyzer::SoilSettings::StandardSentTo = "j.spijker@ihcmerwede.com"

Definition at line 101 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings().

6.62.5.40 int SoilAnalyzer::SoilSettings::thresholdOffsetValue = 0

an tweaking offset value

Definition at line 58 of file soilsettings.h.

Referenced by SoilAnalyzer::Analyzer::GetBW().

6.62.5.41 Vision::Segment::TypeOfObjects SoilAnalyzer::SoilSettings::typeOfObjectsSegmented

Initial value:

```
=
    Vision::Segment::Dark
```

Which type of object should be segmented

Definition at line 51 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), SoilAnalyzer::Analyzer::GetBW(), and DialogSettings::on_rb_useDark_3_toggled().

6.62.5.42 bool SoilAnalyzer::SoilSettings::useAdaptiveContrast

Initial value:

```
=
    false
```

Should adaptive contrast stretch be used default is true

Definition at line 40 of file soilsettings.h.

Referenced by SoilAnalyzer::Analyzer::CalcMaxProgress(), DialogSettings::DialogSettings(), and DialogSettings::on_cb_use_adaptContrast_3_clicked().

6.62.5.43 bool SoilAnalyzer::SoilSettings::useBacklightProjection = true

use Projection

Definition at line 72 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), DialogSettings::on_checkBox_Backlight_clicked(), and VSAMainWindow::TakeSnapShots().

6.62.5.44 bool SoilAnalyzer::SoilSettings::useBlur = false

Should the mediaan blur be used during analsyis

Definition at line 48 of file soilsettings.h.

Referenced by SoilAnalyzer::Analyzer::CalcMaxProgress(), DialogSettings::DialogSettings(), and DialogSettings::on_cb_useBlur_3_clicked().

6.62.5.45 bool SoilAnalyzer::SoilSettings::useCUDA = false

CUDA enabled

Definition at line 96 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), and DialogSettings::on_checkBox_useCUDA_clicked().

6.62.5.46 bool SoilAnalyzer::SoilSettings::useHDR = false

use HDR

Definition at line 73 of file soilsettings.h.

Referenced by DialogSettings::DialogSettings(), DialogSettings::on_checkBox_useHDR_clicked(), and VSAMainWindow::TakeSnapShots().

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

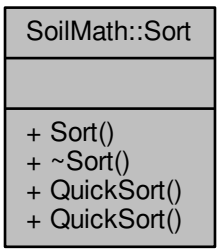
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilsettings.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilAnalyzer/soilsettings.cpp

6.63 SoilMath::Sort Class Reference

The Sort template class.

`#include <Sort.h>`

Collaboration diagram for SoilMath::Sort:



Public Member Functions

- Sort ()
- ~Sort ()

Static Public Member Functions

- template<typename T >
  static void QuickSort (T *arr, int i)

  QuickSort a static sort a Type T array with i values.
- template<typename T >
  static void QuickSort (T *arr, T *key, int i)

  QuickSort a static sort a Type T array with i values where the key are also changed accordingly.

6.63.1 Detailed Description

The Sort template class.

Definition at line 15 of file Sort.h.

6.63.2 Constructor & Destructor Documentation

6.63.2.1 SoilMath::Sort::Sort ( ) [inline]

Definition at line 17 of file Sort.h.

6.63.2.2 SoilMath::Sort::~Sort ( ) [inline]

Definition at line 18 of file Sort.h.

6.63.3 Member Function Documentation

6.63.3.1 template<typename T > static void SoilMath::Sort::QuickSort (T * *arr*, int *i*) [inline], [static]

QuickSort a static sort a Type T array with i values.

Usage: QuickSort<type>(*type , i)

Parameters

arr	an array of Type T
i	the number of elements

Definition at line 26 of file Sort.h.

6.63.3.2 template<typename T > static void SoilMath::Sort::QuickSort (T * *arr*, T * *key*, int *i*) [inline], [static]

QuickSort a static sort a Type T array with i values where the key are also changed accordingly.

Usage: QuickSort<type>(*type *type , i)

Parameters

arr	an array of Type T
key	an array of 0..i-1 representing the index
i	the number of elements

Definition at line 58 of file Sort.h.

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

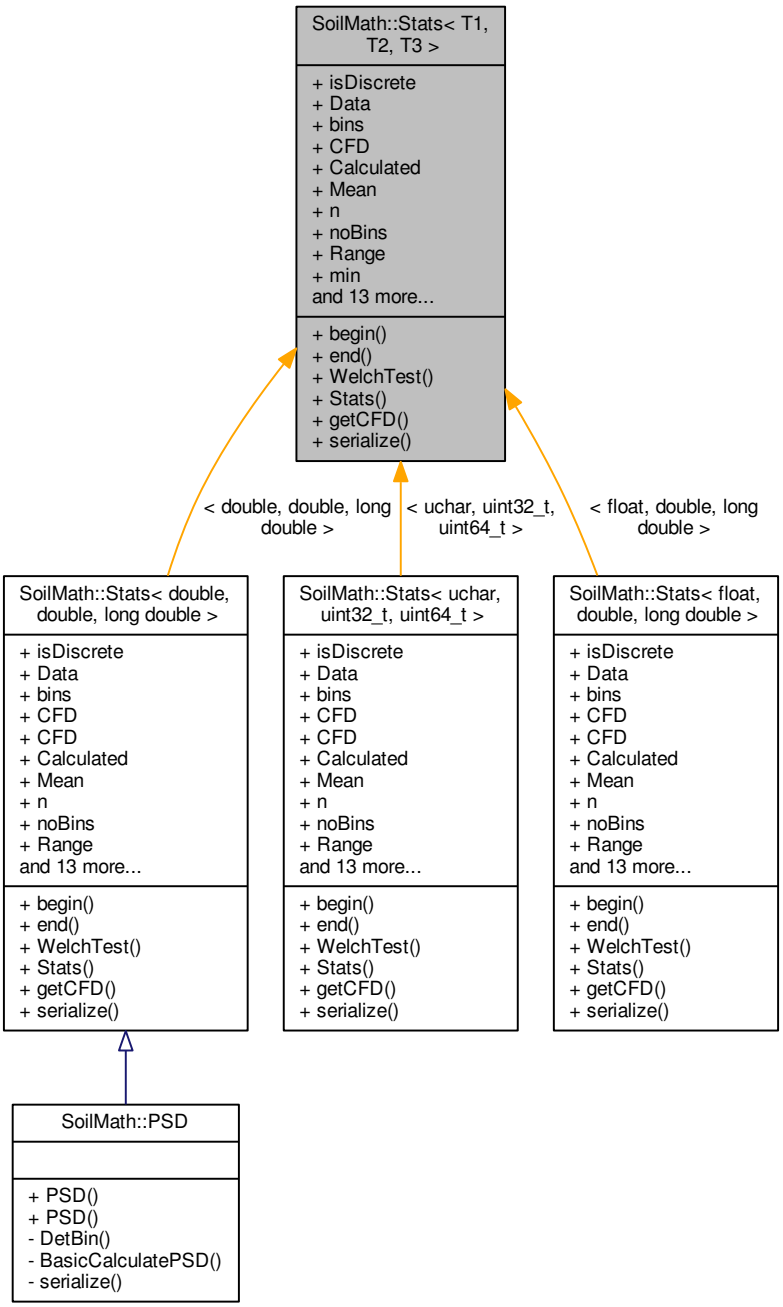
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Sort.h

6.64 SoilMath::Stats< T1, T2, T3 > Class Template Reference

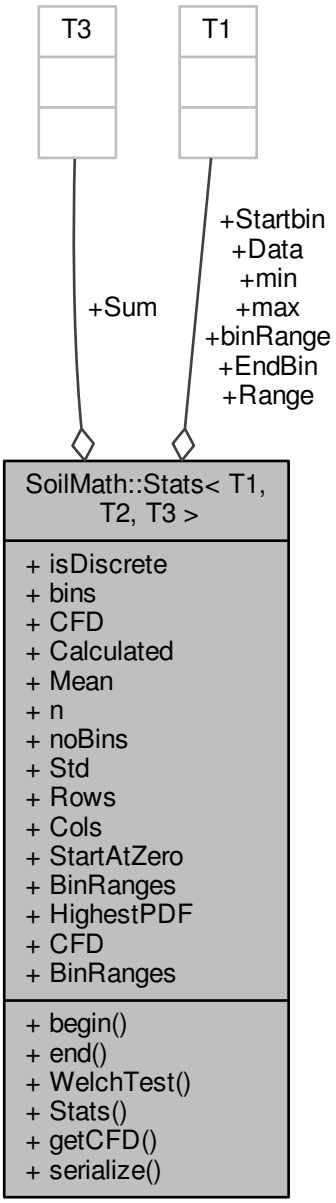
Stats class.

`#include <Stats.h>`

Inheritance diagram for SoilMath::Stats< T1, T2, T3 >:



Collaboration diagram for SoilMath::Stats< T1, T2, T3 >:



Public Member Functions

- uint32_t * begin ()
- uint32_t * end ()
- bool WelchTest (SoilMath::Stats< T1, T2, T3 > &statComp)

  WelchTest Compare the sample using the Welch's Test.
- Stats (const Stats &rhs)

  Stats Constructor.
- void getCFD ()

  getCFD get the CFD matrix;
- template<class Archive >
  void serialize (Archive &ar, const unsigned int version)

  serialize the object

Public Attributes

- bool isDiscrete = true

- T1 * Data = nullptr
- uint32_t * bins = nullptr
- double * CFD = nullptr
- bool Calculated = false
- float Mean = 0.0
- uint32_t n = 0
- uint32_t noBins = 0
- T1 Range = 0
- T1 min = 0
- T1 max = 0
- T1 Startbin = 0
- T1 EndBin = 0
- T1 binRange = 0
- float Std = 0.0
- T3 Sum = 0
- uint16_t Rows = 0
- uint16_t Cols = 0
- bool StartAtZero = true
- double * BinRanges = nullptr
- double HighestPDF = 0.
- CFD {new double[rhs.noBins]{}}
- BinRanges

Friends

- class boost::serialization::access

6.64.1 Detailed Description

template<typename T1, typename T2, typename T3>class SoilMath::Stats< T1, T2, T3 >

Stats class.

Usage Stats<type1, type2, type3>Stats() type 1, 2 and 3 shoudl be of the same value and concecuative in size

Definition at line 39 of file Stats.h.

6.64.2 Constructor & Destructor Documentation

6.64.2.1 template<typename T1, typename T2, typename T3> SoilMath::Stats< T1, T2, T3 >::Stats (const Stats< T1, T2, T3 > & *rhs*) [inline]

Stats Constructor.

Parameters

rhs	Right hand side

Definition at line 112 of file Stats.h.

6.64.3 Member Function Documentation

6.64.3.1 template<typename T1, typename T2, typename T3> uint32_t* SoilMath::Stats< T1, T2, T3 >::begin ( ) [inline]

pointer to the first bin

Definition at line 65 of file Stats.h.

6.64.3.2 template<typename T1, typename T2, typename T3> uint32_t* SoilMath::Stats< T1, T2, T3 >::end ( ) [inline]

pointer to the last + 1 bin

Definition at line 66 of file Stats.h.

6.64.3.3 template<typename T1, typename T2, typename T3> void SoilMath::Stats< T1, T2, T3 >::getCFD ( ) [inline]

getCFD get the CFD matrix;

Definition at line 629 of file Stats.h.

6.64.3.4 template<typename T1, typename T2, typename T3> template<class Archive > void SoilMath::Stats< T1, T2, T3 >::serialize (Archive & *ar,* const unsigned int *version*) [inline]

serialize the object

Parameters

ar	argument
version	

Definition at line 651 of file Stats.h.

6.64.3.5 template<typename T1, typename T2, typename T3> bool SoilMath::Stats< T1, T2, T3 >::WelchTest (SoilMath::Stats< T1, T2, T3 > & *statComp*) [inline]

WelchTest Compare the sample using the Welch's Test.

(source: http://www.boost.org/doc/libs/1_57_0/libs/math/doc/html/math_toolkit/stat_tut/weg/st_eg/two_sample_students_t.html)

Parameters

statComp	Statiscs Results of which it should be tested against

Returns

Definition at line 75 of file Stats.h.

6.64.4 Friends And Related Function Documentation

6.64.4.1 template<typename T1, typename T2, typename T3> friend class boost::serialization::access [friend]

Serialization class

Definition at line 643 of file Stats.h.

6.64.5 Member Data Documentation

6.64.5.1 template<typename T1, typename T2, typename T3> T1 SoilMath::Stats< T1, T2, T3 >::binRange = 0

the range of a single bin

Definition at line 55 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::serialize().

6.64.5.2 template<typename T1, typename T2, typename T3> double* SoilMath::Stats< T1, T2, T3 >::BinRanges = nullptr

Definition at line 62 of file Stats.h.

6.64.5.3 template<typename T1, typename T2, typename T3> SoilMath::Stats< T1, T2, T3 >::BinRanges

data end counter used with mask

Definition at line 114 of file Stats.h.

6.64.5.4 template<typename T1, typename T2, typename T3> uint32_t* SoilMath::Stats< T1, T2, T3 >::bins = nullptr

the histogram

Definition at line 44 of file Stats.h.

Referenced by Vision::Segment::GetBlobList(), Vision::Segment::GetThresholdLevel(), QReportGenerator::QReportGenerator(), VSAMainWindow::setAngularityHistogram(), and VSAMainWindow::setRoundnessHistogram().

6.64.5.5 template<typename T1, typename T2, typename T3> bool SoilMath::Stats< T1, T2, T3 >::Calculated = false

indication if the data has been calculated

Definition at line 46 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::serialize().

6.64.5.6 template<typename T1, typename T2, typename T3> double* SoilMath::Stats< T1, T2, T3 >::CFD = nullptr

the CFD

Definition at line 45 of file Stats.h.

Referenced by QReportGenerator::QReportGenerator(), and VSAMainWindow::SetPSDgraph().

6.64.5.7 template<typename T1, typename T2, typename T3> SoilMath::Stats< T1, T2, T3 >::CFD {new double[rhs.noBins]{}}

Definition at line 113 of file Stats.h.

6.64.5.8 template<typename T1, typename T2, typename T3> uint16_t SoilMath::Stats< T1, T2, T3 >::Cols = 0

number of cols from the data matrix

Definition at line 59 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::serialize().

6.64.5.9 template<typename T1, typename T2, typename T3> T1* SoilMath::Stats< T1, T2, T3 >::Data = nullptr

Pointer the data

Definition at line 43 of file Stats.h.

Referenced by VSAMainWindow::on_actionLoadSample_triggered().

6.64.5.10 template<typename T1, typename T2, typename T3> T1 SoilMath::Stats< T1, T2, T3 >::EndBin = 0

End bin value

Definition at line 54 of file Stats.h.

Referenced by Vision::Segment::GetBlobList(), and SoilMath::Stats< float, double, long double >::serialize().

6.64.5.11 template<typename T1, typename T2, typename T3> double SoilMath::Stats< T1, T2, T3 >::HighestPDF = 0.

Definition at line 63 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::serialize(), VSAMainWindow::setAngularityHistogram(), and VSAMainWindow::setRoundnessHistogram().

6.64.5.12 template<typename T1, typename T2, typename T3> bool SoilMath::Stats< T1, T2, T3 >::isDiscrete = true

indicates if the data is discrete or real

Definition at line 41 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::serialize().

6.64.5.13 template<typename T1, typename T2, typename T3> T1 SoilMath::Stats< T1, T2, T3 >::max = 0

maximum value

Definition at line 52 of file Stats.h.

Referenced by Vision::Enhance::HistogramEqualization(), QReportGenerator::QReportGenerator(), and SoilMath::Stats< float, double, long double >::serialize().

6.64.5.14 template<typename T1, typename T2, typename T3> float SoilMath::Stats< T1, T2, T3 >::Mean = 0.0

the mean value of the data

Definition at line 47 of file Stats.h.

Referenced by SoilAnalyzer::Particle::getMeanLab(), SoilAnalyzer::Particle::GetMeanRI(), Vision::Segment::GetThresholdLevel(), QReportGenerator::QReportGenerator(), SoilMath::Stats< float, double, long double >::serialize(), VSAMainWindow::setAngularityHistogram(), VSAMainWindow::setRoundnessHistogram(), and SoilMath::Stats< float, double, long double >::WelchTest().

6.64.5.15 template<typename T1, typename T2, typename T3> T1 SoilMath::Stats< T1, T2, T3 >::min = 0

minimum value

Definition at line 51 of file Stats.h.

Referenced by Vision::Enhance::HistogramEqualization(), QReportGenerator::QReportGenerator(), and SoilMath::Stats< float, double, long double >::serialize().

6.64.5.16 template<typename T1, typename T2, typename T3> uint32_t SoilMath::Stats< T1, T2, T3 >::n = 0

number of data points

Definition at line 48 of file Stats.h.

Referenced by QReportGenerator::QReportGenerator(), SoilMath::Stats< float, double, long double >::serialize(), and SoilMath::Stats< float, double, long double >::WelchTest().

6.64.5.17 template<typename T1, typename T2, typename T3> uint32_t SoilMath::Stats< T1, T2, T3 >::noBins = 0

number of bins

Definition at line 49 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::end(), SoilMath::Stats< float, double, long double >::getCFD(), and SoilMath::Stats< float, double, long double >::serialize().

6.64.5.18 template<typename T1, typename T2, typename T3> T1 SoilMath::Stats< T1, T2, T3 >::Range = 0

range of the data

Definition at line 50 of file Stats.h.

Referenced by QReportGenerator::QReportGenerator(), and SoilMath::Stats< float, double, long double >::serialize().

6.64.5.19 template<typename T1, typename T2, typename T3> uint16_t SoilMath::Stats< T1, T2, T3 >::Rows = 0

number of rows from the data matrix

Definition at line 58 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::serialize().

6.64.5.20 template<typename T1, typename T2, typename T3> bool SoilMath::Stats< T1, T2, T3 >::StartAtZero = true

indication of the minimum value starts at zero or could be less

Definition at line 60 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::serialize().

6.64.5.21 template<typename T1, typename T2, typename T3> T1 SoilMath::Stats< T1, T2, T3 >::Startbin = 0

First bin value

Definition at line 53 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::serialize().

6.64.5.22 template<typename T1, typename T2, typename T3> float SoilMath::Stats< T1, T2, T3 >::Std = 0.0

standard deviation

Definition at line 56 of file Stats.h.

Referenced by Vision::Segment::GetThresholdLevel(), QReportGenerator::QReportGenerator(), SoilMath::Stats< float, double, long double >::serialize(), and SoilMath::Stats< float, double, long double >::WelchTest().

6.64.5.23 template<typename T1, typename T2, typename T3> T3 SoilMath::Stats< T1, T2, T3 >::Sum = 0

total sum of all the data values

Definition at line 57 of file Stats.h.

Referenced by SoilMath::Stats< float, double, long double >::serialize().

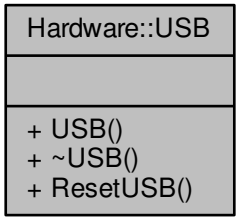
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilMath/Stats.h

6.65 Hardware::USB Class Reference

```
#include <USB.h>
```

Collaboration diagram for Hardware::USB:



Public Member Functions

- USB ()
- ∼USB ()
- void ResetUSB ()

6.65.1 Detailed Description

Definition at line 19 of file USB.h.

6.65.2 Constructor & Destructor Documentation

6.65.2.1 Hardware::USB::USB ( )

Definition at line 11 of file USB.cpp.

6.65.2.2 Hardware::USB::∼USB ( )

Definition at line 13 of file USB.cpp.

6.65.3 Member Function Documentation

6.65.3.1 void Hardware::USB::ResetUSB ( )

Definition at line 15 of file USB.cpp.

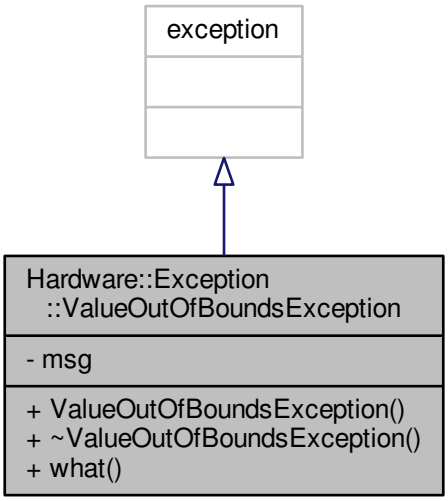
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/USB.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/USB.cpp

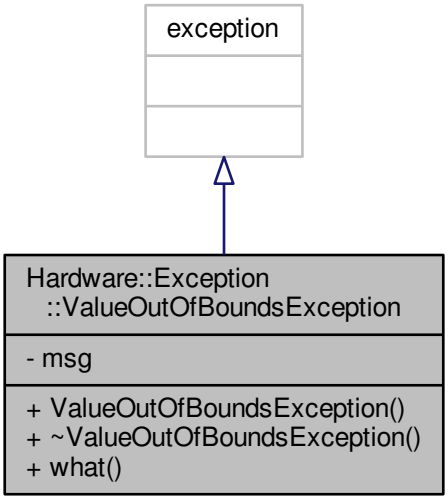
6.66 Hardware::Exception::ValueOutOfBoundsException Class Reference

`#include <ValueOutOfBoundsException.h>`

Inheritance diagram for Hardware::Exception::ValueOutOfBoundsException:



Collaboration diagram for Hardware::Exception::ValueOutOfBoundsException:



Public Member Functions

- ValueOutOfBoundsException (string m="Value out of bounds!")
- ~ValueOutOfBoundsException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.66.1 Detailed Description

Definition at line 17 of file ValueOutOfBoundsException.h.

6.66.2 Constructor & Destructor Documentation

6.66.2.1 Hardware::Exception::ValueOutOfBoundsException::ValueOutOfBoundsException (string *m* = "Value out of bounds!") [inline]

Definition at line 19 of file ValueOutOfBoundsException.h.

6.66.2.2 Hardware::Exception::ValueOutOfBoundsException::~ValueOutOfBoundsException ( ) [inline]

Definition at line 20 of file ValueOutOfBoundsException.h.

6.66.3 Member Function Documentation

6.66.3.1 const char* Hardware::Exception::ValueOutOfBoundsException::what ( ) const [inline]

Definition at line 21 of file ValueOutOfBoundsException.h.

6.66.4 Member Data Documentation

6.66.4.1 string Hardware::Exception::ValueOutOfBoundsException::msg [private]

Definition at line 21 of file ValueOutOfBoundsException.h.

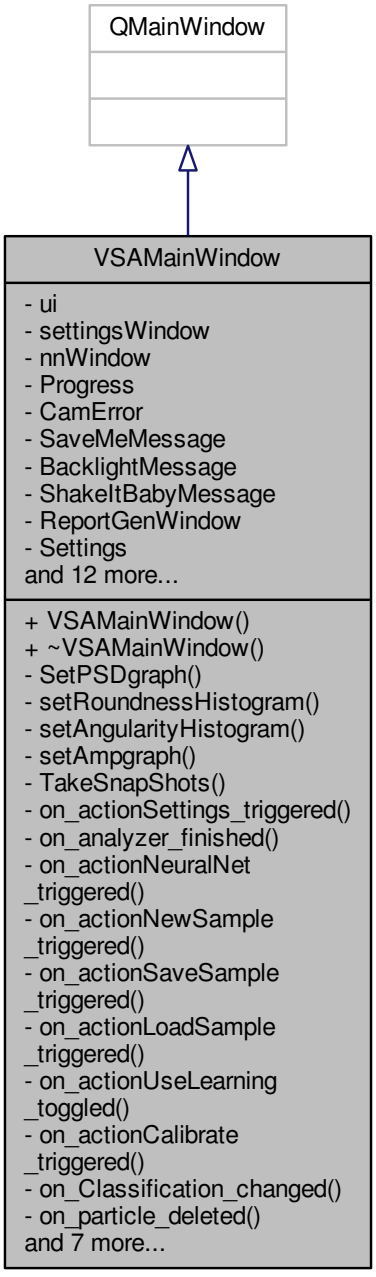
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilHardware/ValueOutOfBoundsException.h

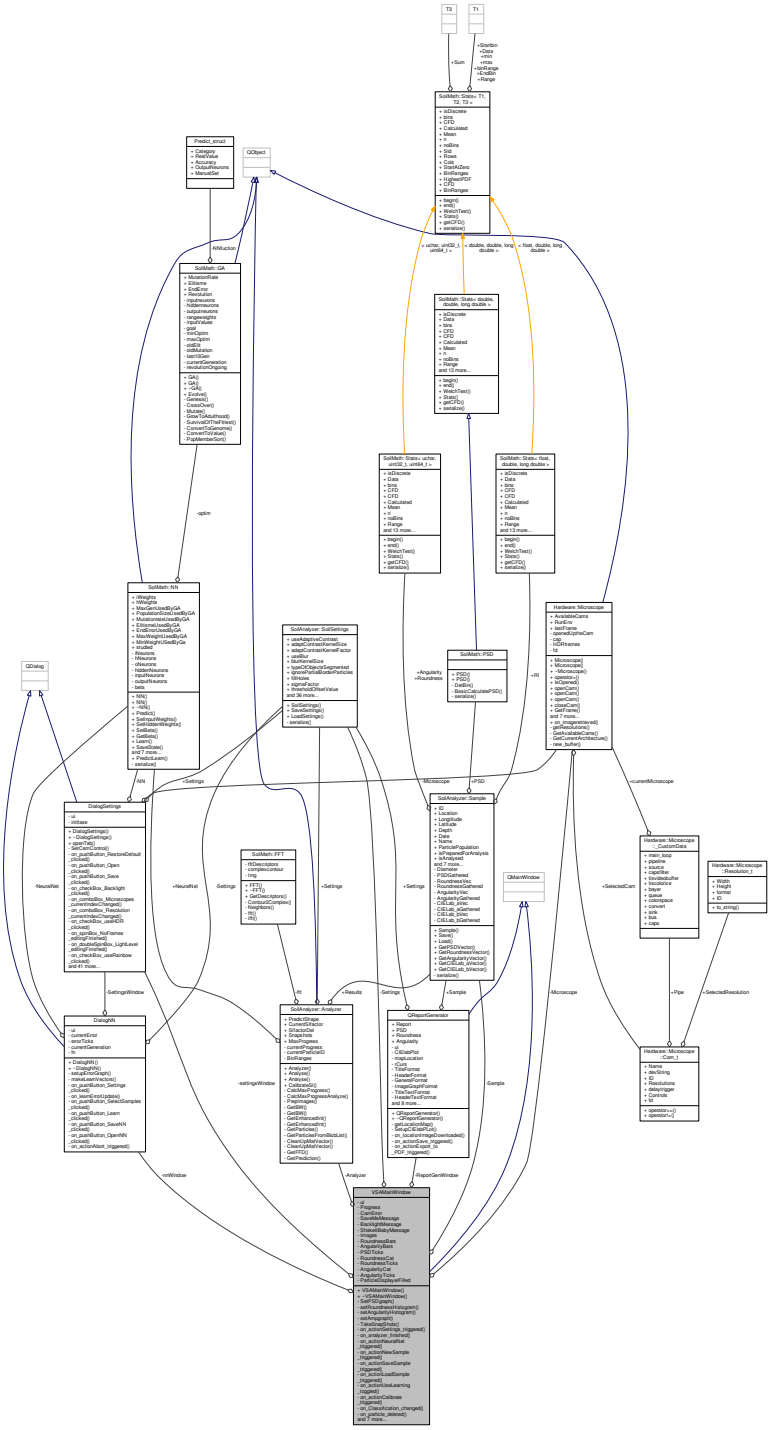
6.67 VSAMainWindow Class Reference

```
#include <vsamainwindow.h>
```

Inheritance diagram for VSAMainWindow:



Collaboration diagram for VSAMainWindow:



Public Member Functions

- VSAMainWindow (QWidget *parent=0)
- ~VSAMainWindow ()

Private Slots

- void on_actionSettings_triggered ()
- void on_analyzer_finished ()
- void on_actionNeuralNet_triggered ()
- void on_actionNewSample_triggered ()
- void on_actionSaveSample_triggered ()
- void on_actionLoadSample_triggered ()
- void on_actionUseLearning_toggled (bool arg1)
- void on_actionCalibrate_triggered ()
- void on_Classification_changed (int newValue)

- void on_particle_deleted ()
- void on_actionAutomatic_Shape_Pediction_triggered (bool checked)
- void on_reset_graph (QMouseEvent *e)
- void on_actionReport_Generator_triggered ()
- void on_particleChanged (int newPart)
- void on_PSD_contextMenuRequest (QPoint point)
- void on_compare_against ()
- void on_restore_PSD ()

Private Member Functions

- void SetPSDgraph ()
- void setRoundnessHistogram ()
- void setAngularityHistogram ()
- void setAmpgraph ()
- void TakeSnapShots ()

Private Attributes

- Ui::VSAMainWindow * ui
- DialogSettings * settingsWindow = nullptr
- DialogNN * nnWindow = nullptr
- QProgressBar * Progress
- QErrorMessage * CamError = nullptr
- QMessageBox * SaveMeMessage = nullptr
- QMessageBox * BacklightMessage = nullptr
- QMessageBox * ShakeItBabyMessage = nullptr
- QReportGenerator * ReportGenWindow = nullptr
- SoilAnalyzer::SoilSettings * Settings = nullptr
- Hardware::Microscope * Microscope = nullptr
- SoilAnalyzer::Sample * Sample = nullptr
- SoilAnalyzer::Analyzer * Analyzer = nullptr
- SoilAnalyzer::Analyzer::Images_t * Images = nullptr
- QCPBars * RoundnessBars = nullptr
- QCPBars * AngularityBars = nullptr
- std::vector< double > PSDTicks
- QVector< QString > RoundnessCat = {"High", "Medium", "Low"}
- std::vector< double > RoundnessTicks = {1, 2, 3}
- QVector< QString > AngularityCat
- std::vector< double > AngularityTicks = {1, 2, 3, 4, 5, 6}
- bool ParticleDisplayerFilled = false

6.67.1 Detailed Description

Definition at line 27 of file vsamainwindow.h.

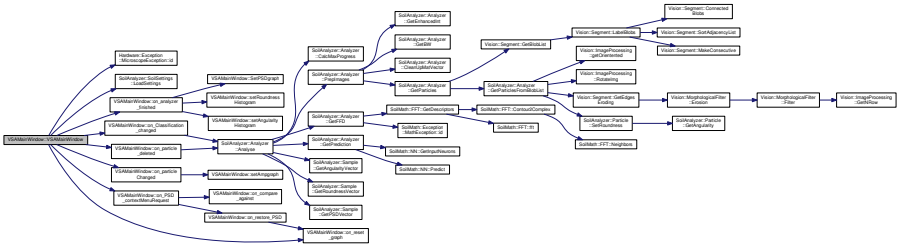
6.67.2 Constructor & Destructor Documentation

6.67.2.1 VSAMainWindow::VSAMainWindow (QWidget * *parent* = 0) [explicit]

Definition at line 4 of file vsamainwindow.cpp.

References Analyzer, AngularityBars, AngularityCat, AngularityTicks, BacklightMessage, CamError, SoilAnalyzer::SoilSettings::defaultWebcam, EXCEPTION_NOCAMS_NR, EXCEPTION_OPENCAM_NR, Hardware::Exception::MicroscopeException::id(), Images, SoilAnalyzer::SoilSettings::LoadSettings(), SoilAnalyzer::Analyzer::MaxProgress, SoilAnalyzer::Analyzer::NeuralNet, nnWindow, on_analyzer_finished(), on_Classification_changed(), on_particle_deleted(), on_particleChanged(), on_PSD_contextMenuRequest(), on_reset_graph(), SoilAnalyzer::SoilSettings::PredictTheShape, Progress, PSDTicks, RoundnessBars, RoundnessCat, RoundnessTicks, Sample, SaveMeMessage, SoilAnalyzer::SoilSettings::selectedResolution, Settings, settingsWindow, ShakeItBabyMessage, and ui.

Here is the call graph for this function:



6.67.2.2 VSAMainWindow::~VSAMainWindow ( )

Definition at line 254 of file vsamainwindow.cpp.

References Analyzer, BacklightMessage, CamError, Images, Microscope, nnWindow, Sample, SaveMeMessage, Settings, settingsWindow, ShakeItBabyMessage, and ui.

6.67.3 Member Function Documentation

6.67.3.1 void VSAMainWindow::on_actionAutomatic_Shape_Pediction_triggered (bool *checked*) [private], [slot]

Definition at line 541 of file vsamainwindow.cpp.

References SoilAnalyzer::SoilSettings::PredictTheShape, and Settings.

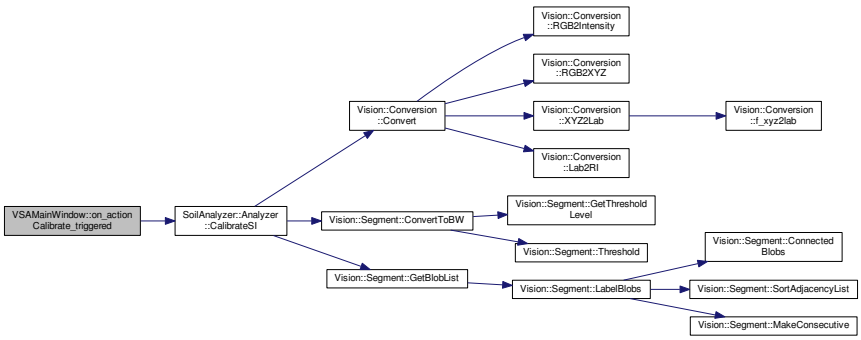
6.67.3.2 void VSAMainWindow::on_actionCalibrate_triggered ( ) [private], [slot]

Definition at line 516 of file vsamainwindow.cpp.

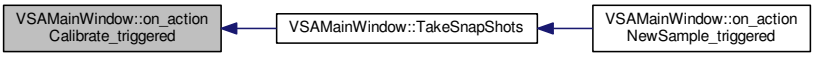
References Analyzer, and SoilAnalyzer::Analyzer::CalibrateSI().

Referenced by TakeSnapShots().

Here is the call graph for this function:



Here is the caller graph for this function:

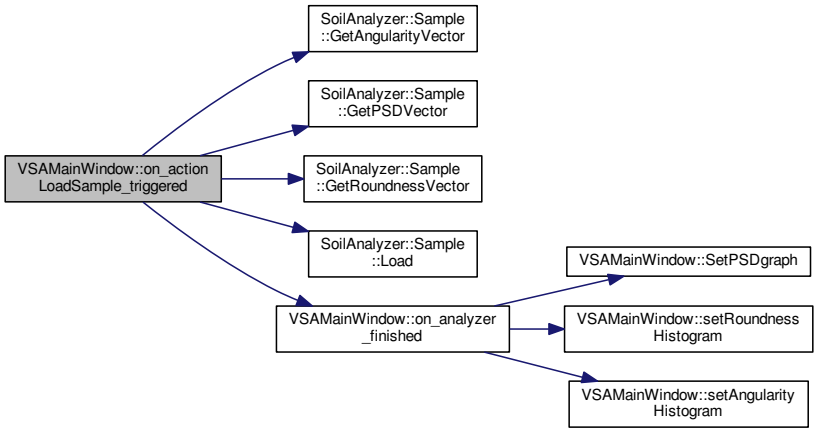


6.67.3.3 void VSAMainWindow::on_actionLoadSample_triggered ( ) [private], [slot]

Definition at line 475 of file vsamainwindow.cpp.

References Analyzer, SoilAnalyzer::Sample::Angularity, SoilAnalyzer::Sample::ChangesSinceLastSave, SoilMath::Stats< T1, T2, T3 >::↩ Data, SoilAnalyzer::Sample::GetAngularityVector(), SoilAnalyzer::Sample::GetPSDVector(), SoilAnalyzer::Sample::GetRoundnessVector(), Images, SoilAnalyzer::Sample::Load(), on_analyzer_finished(), ParticleDisplayerFilled, SoilAnalyzer::Sample::PSD, SoilAnalyzer::Analyzer::↩ Results, SoilAnalyzer::Sample::Roundness, Sample, SoilAnalyzer::SoilSettings::SampleFolder, SaveMeMessage, Settings, and ui.

Here is the call graph for this function:



6.67.3.4 void VSAMainWindow::on_actionNeuralNet_triggered ( ) [private], [slot]

Definition at line 353 of file vsamainwindow.cpp.

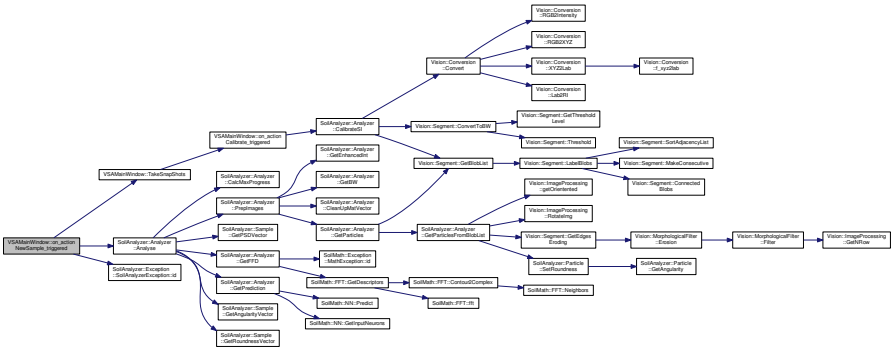
References Analyzer, SoilAnalyzer::Analyzer::NeuralNet, nnWindow, Settings, and settingsWindow.

6.67.3.5 void VSAMainWindow::on_actionNewSample_triggered ( ) [private], [slot]

Definition at line 361 of file vsamainwindow.cpp.

References SoilAnalyzer::Analyzer::Analyse(), Analyzer, CamError, SoilAnalyzer::Sample::ChangesSinceLastSave, EXCEPTION_NO_SN↩APSHOTS_NR, SoilAnalyzer::Exception::SoilAnalyzerException::id(), Images, SoilAnalyzer::Sample::ParticlePopulation, Sample, SaveMe↩Message, Settings, TakeSnapShots(), and ui.

Here is the call graph for this function:



6.67.3.6 void VSAMainWindow::on_actionReport_Generator_triggered ( ) [private], [slot]

Definition at line 552 of file vsamainwindow.cpp.

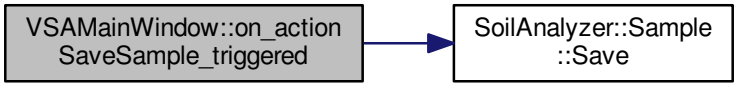
References ReportGenWindow, Sample, Settings, and ui.

6.67.3.7 void VSAMainWindow::on_actionSaveSample_triggered ( ) [private], [slot]

Definition at line 460 of file vsamainwindow.cpp.

References SoilAnalyzer::Sample::ChangesSinceLastSave, SoilAnalyzer::Sample::IsLoadedFromDisk, Sample, SoilAnalyzer::SoilSettings::↩SampleFolder, SoilAnalyzer::Sample::Save(), and Settings.

Here is the call graph for this function:

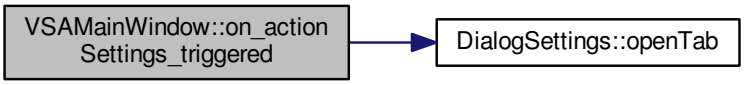


6.67.3.8 void VSAMainWindow::on_actionSettings_triggered ( ) [private], [slot]

Definition at line 270 of file vsamainwindow.cpp.

References DialogSettings::openTab(), and settingsWindow.

Here is the call graph for this function:



6.67.3.9 void VSAMainWindow::on_actionUseLearning_toggled (bool *arg1*) [private], [slot]

Definition at line 512 of file vsamainwindow.cpp.

References Analyzer, and SoilAnalyzer::Analyzer::PredictShape.

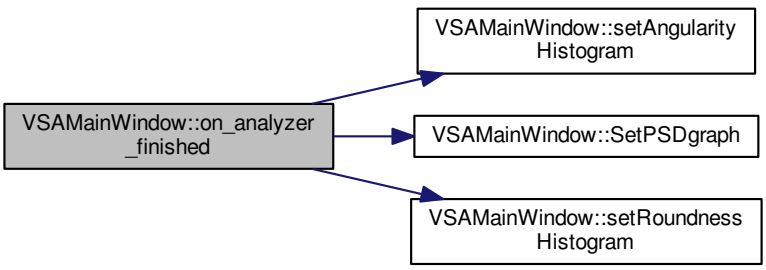
6.67.3.10 void VSAMainWindow::on_analyzer_finished ( ) [private], [slot]

Definition at line 275 of file vsamainwindow.cpp.

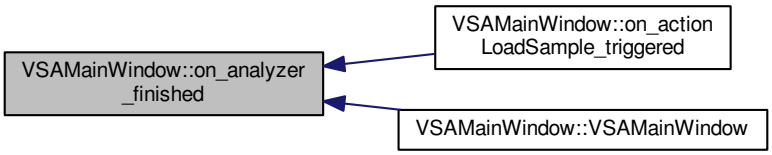
References ParticleDisplayerFilled, SoilAnalyzer::Sample::ParticlePopulation, Sample, setAngularityHistogram(), SetPSDgraph(), setRoundnessHistogram(), and ui.

Referenced by on_actionLoadSample_triggered(), and VSAMainWindow().

Here is the call graph for this function:



Here is the caller graph for this function:



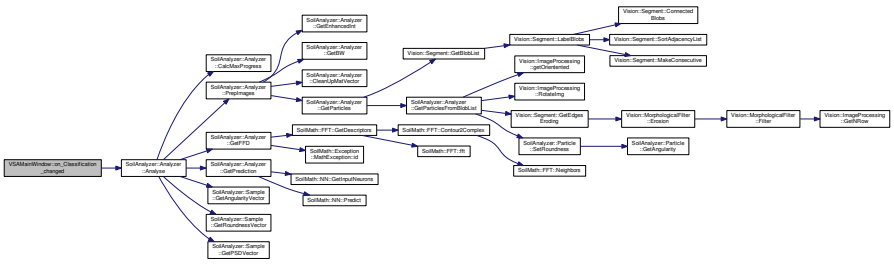
6.67.3.11 void VSAMainWindow::on_Classification_changed (int *newValue*) [private], [slot]

Definition at line 522 of file vsamainwindow.cpp.

References SoilAnalyzer::Analyzer::Analyse(), Analyzer, SoilAnalyzer::Sample::ChangesSinceLastSave, SoilAnalyzer::Sample::Particle↩ ChangedStateAngularity, SoilAnalyzer::Sample::ParticleChangedStateRoundness, Sample, and ui.

Referenced by VSAMainWindow().

Here is the call graph for this function:



Here is the caller graph for this function:

6.67.3.12 void VSAMainWindow::on_compare_against ( ) [private], [slot]

Definition at line 570 of file vsamainwindow.cpp.

References PSDTicks, SoilAnalyzer::SoilSettings::SampleFolder, Settings, and ui.

Referenced by on_PSD_contextMenuRequest().

Here is the caller graph for this function:

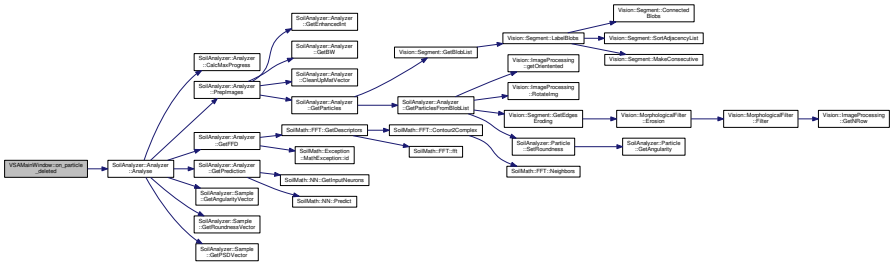
6.67.3.13 void VSAMainWindow::on_particle_deleted ( ) [private], [slot]

Definition at line 539 of file vsamainwindow.cpp.

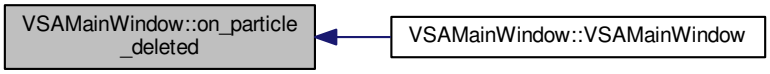
References SoilAnalyzer::Analyzer::Analyse(), and Analyzer.

Referenced by VSAMainWindow().

Here is the call graph for this function:



Here is the caller graph for this function:



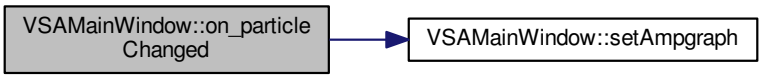
6.67.3.14 void VSAMainWindow::on_particleChanged (int *newPart*) [private], [slot]

Definition at line 351 of file vsamainwindow.cpp.

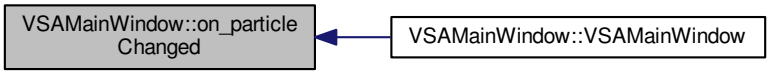
References setAmpgraph().

Referenced by VSAMainWindow().

Here is the call graph for this function:



Here is the caller graph for this function:



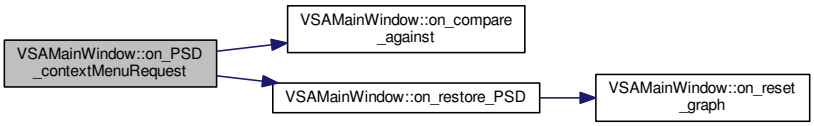
6.67.3.15 void VSAMainWindow::on_PSD_contextMenuRequest (QPoint *point*) [private], [slot]

Definition at line 561 of file vsamainwindow.cpp.

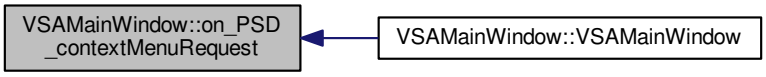
References on_compare_against(), on_restore_PSD(), and ui.

Referenced by VSAMainWindow().

Here is the call graph for this function:



Here is the caller graph for this function:



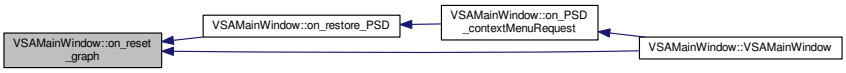
6.67.3.16 void VSAMainWindow::on_reset_graph (QMouseEvent * *e*) [private], [slot]

Definition at line 545 of file vsamainwindow.cpp.

References ui.

Referenced by on_restore_PSD(), and VSAMainWindow().

Here is the caller graph for this function:



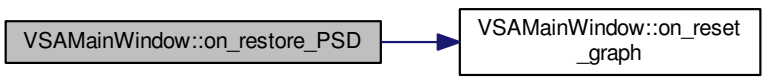
6.67.3.17 void VSAMainWindow::on_restore_PSD ( ) [private], [slot]

Definition at line 628 of file vsamainwindow.cpp.

References on_reset_graph(), and ui.

Referenced by on_PSD_contextMenuRequest().

Here is the call graph for this function:



Here is the caller graph for this function:

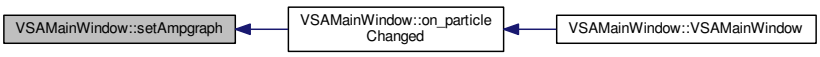
6.67.3.18 void VSAMainWindow::setAmpgraph ( ) [private]

Definition at line 339 of file vsamainwindow.cpp.

References ui.

Referenced by on_particleChanged().

Here is the caller graph for this function:



6.67.3.19 void VSAMainWindow::setAngularityHistogram ( ) [private]

Definition at line 315 of file vsamainwindow.cpp.

References SoilAnalyzer::Sample::Angularity, AngularityBars, AngularityTicks, SoilMath::Stats< T1, T2, T3 >::bins, SoilMath::Stats< T1, T2, T3 >::HighestPDF, SoilMath::Stats< T1, T2, T3 >::Mean, Sample, and ui.

Referenced by on_analyzer_finished().

Here is the caller graph for this function:

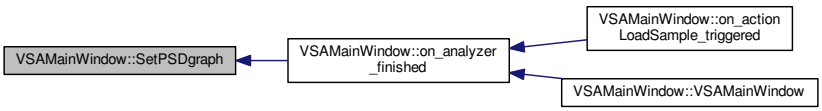
6.67.3.20 void VSAMainWindow::SetPSDgraph ( ) [private]

Definition at line 285 of file vsamainwindow.cpp.

References SoilMath::Stats< T1, T2, T3 >::CFD, SoilAnalyzer::Sample::PSD, PSDTicks, Sample, and ui.

Referenced by on_analyzer_finished().

Here is the caller graph for this function:



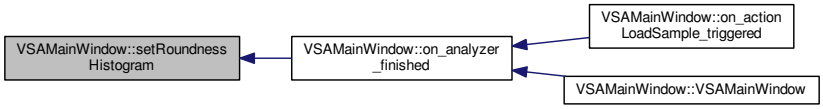
6.67.3.21 void VSAMainWindow::setRoundnessHistogram ( ) [private]

Definition at line 291 of file vsamainwindow.cpp.

References SoilMath::Stats< T1, T2, T3 >::bins, SoilMath::Stats< T1, T2, T3 >::HighestPDF, SoilMath::Stats< T1, T2, T3 >::Mean, SoilAnalyzer::Sample::Roundness, RoundnessBars, RoundnessTicks, Sample, and ui.

Referenced by on_analyzer_finished().

Here is the caller graph for this function:



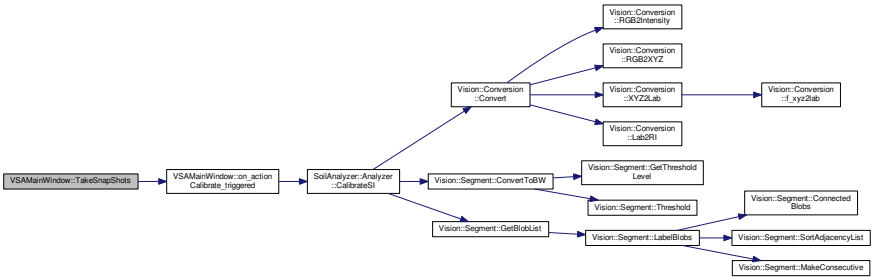
6.67.3.22 void VSAMainWindow::TakeSnapShots ( ) [private]

Definition at line 391 of file vsamainwindow.cpp.

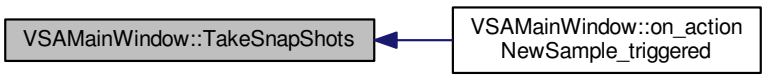
References Analyzer, SoilAnalyzer::Analyzer::Image_t::BackLight, BacklightMessage, SoilAnalyzer::Analyzer::CurrentSIfactor, SoilAnalyzer::Analyzer::Image_t::FrontLight, SoilAnalyzer::SoilSettings::HDRframes, Images, on_actionCalibrate_triggered(), Settings, ShakeItBabyMessage, SoilAnalyzer::Analyzer::SIfactorDet, SoilAnalyzer::Analyzer::Image_t::SIPixelFactor, SoilAnalyzer::SoilSettings::StandardNumberOfShots, SoilAnalyzer::SoilSettings::useBacklightProjection, and SoilAnalyzer::SoilSettings::useHDR.

Referenced by on_actionNewSample_triggered().

Here is the call graph for this function:



Here is the caller graph for this function:



6.67.4 Member Data Documentation

6.67.4.1 SoilAnalyzer::Analyzer* VSAMainWindow::Analyzer = nullptr [private]

Definition at line 83 of file vsamainwindow.h.

Referenced by on_actionCalibrate_triggered(), on_actionLoadSample_triggered(), on_actionNeuralNet_triggered(), on_actionNewSample_triggered(), on_actionUseLearning_toggled(), on_Classification_changed(), on_particle_deleted(), TakeSnapShots(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.2 QCPBars* VSAMainWindow::AngularityBars = nullptr [private]

Definition at line 86 of file vsamainwindow.h.

Referenced by setAngularityHistogram(), and VSAMainWindow().

6.67.4.3 QVector<QString> VSAMainWindow::AngularityCat [private]

Initial value:

```
= {"Very Angular", "Angular", "Sub Angular",
                                "Sub Rounded",  "Rounded", "Well Rounded"}
```

Definition at line 92 of file vsamainwindow.h.

Referenced by VSAMainWindow().

6.67.4.4 std::vector<double> VSAMainWindow::AngularityTicks = {1, 2, 3, 4, 5, 6} [private]

Definition at line 94 of file vsamainwindow.h.

Referenced by setAngularityHistogram(), and VSAMainWindow().

6.67.4.5 QMessageBox* VSAMainWindow::BacklightMessage = nullptr [private]

Definition at line 76 of file vsamainwindow.h.

Referenced by TakeSnapShots(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.6 QErrorMessage* VSAMainWindow::CamError = nullptr [private]

Definition at line 74 of file vsamainwindow.h.

Referenced by on_actionNewSample_triggered(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.7 SoilAnalyzer::Analyzer::Images_t* VSAMainWindow::Images = nullptr [private]

Definition at line 84 of file vsamainwindow.h.

Referenced by on_actionLoadSample_triggered(), on_actionNewSample_triggered(), TakeSnapShots(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.8 Hardware::Microscope* VSAMainWindow::Microscope = nullptr [private]

Definition at line 81 of file vsamainwindow.h.

Referenced by ~VSAMainWindow().

6.67.4.9 DialogNN* VSAMainWindow::nnWindow = nullptr [private]

Definition at line 72 of file vsamainwindow.h.

Referenced by on_actionNeuralNet_triggered(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.10 bool VSAMainWindow::ParticleDisplayerFilled = false [private]

Definition at line 96 of file vsamainwindow.h.

Referenced by on_actionLoadSample_triggered(), and on_analyzer_finished().

6.67.4.11 QProgressBar* VSAMainWindow::Progress [private]

Definition at line 73 of file vsamainwindow.h.

Referenced by VSAMainWindow().

6.67.4.12 std::vector<double> VSAMainWindow::PSDTicks [private]

Initial value:

```
= {0.0,  0.038, 0.045, 0.063, 0.075,
                               0.09, 0.125, 0.18,  0.25,  0.355,
                               0.5,  0.71,  1.0,   1.4,   2.0}
```

Definition at line 87 of file vsamainwindow.h.

Referenced by on_compare_against(), SetPSDgraph(), and VSAMainWindow().

6.67.4.13 QReportGenerator* VSAMainWindow::ReportGenWindow = nullptr [private]

Definition at line 78 of file vsamainwindow.h.

Referenced by on_actionReport_Generator_triggered().

6.67.4.14 QCPBars* VSAMainWindow::RoundnessBars = nullptr [private]

Definition at line 85 of file vsamainwindow.h.

Referenced by setRoundnessHistogram(), and VSAMainWindow().

6.67.4.15 QVector<QString> VSAMainWindow::RoundnessCat = {"High", "Medium", "Low"} [private]

Definition at line 90 of file vsamainwindow.h.

Referenced by VSAMainWindow().

6.67.4.16 std::vector<double> VSAMainWindow::RoundnessTicks = {1, 2, 3} [private]

Definition at line 91 of file vsamainwindow.h.

Referenced by setRoundnessHistogram(), and VSAMainWindow().

6.67.4.17 SoilAnalyzer::Sample* VSAMainWindow::Sample = nullptr [private]

Definition at line 82 of file vsamainwindow.h.

Referenced by on_actionLoadSample_triggered(), on_actionNewSample_triggered(), on_actionReport_Generator_triggered(), on_action↩SaveSample_triggered(), on_analyzer_finished(), on_Classification_changed(), setAngularityHistogram(), SetPSDgraph(), setRoundness↩Histogram(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.18 QMessageBox* VSAMainWindow::SaveMeMessage = nullptr [private]

Definition at line 75 of file vsamainwindow.h.

Referenced by on_actionLoadSample_triggered(), on_actionNewSample_triggered(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.19 SoilAnalyzer::SoilSettings* VSAMainWindow::Settings = nullptr [private]

Definition at line 80 of file vsamainwindow.h.

Referenced by on_actionAutomatic_Shape_Pediction_triggered(), on_actionLoadSample_triggered(), on_actionNeuralNet_triggered(), on↩_actionNewSample_triggered(), on_actionReport_Generator_triggered(), on_actionSaveSample_triggered(), on_compare_against(), Take↩SnapShots(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.20 DialogSettings* VSAMainWindow::settingsWindow = nullptr [private]

Definition at line 71 of file vsamainwindow.h.

Referenced by on_actionNeuralNet_triggered(), on_actionSettings_triggered(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.21 QMessageBox* VSAMainWindow::ShakeItBabyMessage = nullptr [private]

Definition at line 77 of file vsamainwindow.h.

Referenced by TakeSnapShots(), VSAMainWindow(), and ~VSAMainWindow().

6.67.4.22 Ui::VSAMainWindow* VSAMainWindow::ui [private]

Definition at line 70 of file vsamainwindow.h.

Referenced by on_actionLoadSample_triggered(), on_actionNewSample_triggered(), on_actionReport_Generator_triggered(), on_analyzer_↩finished(), on_Classification_changed(), on_compare_against(), on_PSD_contextMenuRequest(), on_reset_graph(), on_restore_PSD(), set↩Ampgraph(), setAngularityHistogram(), SetPSDgraph(), setRoundnessHistogram(), VSAMainWindow(), and ~VSAMainWindow().

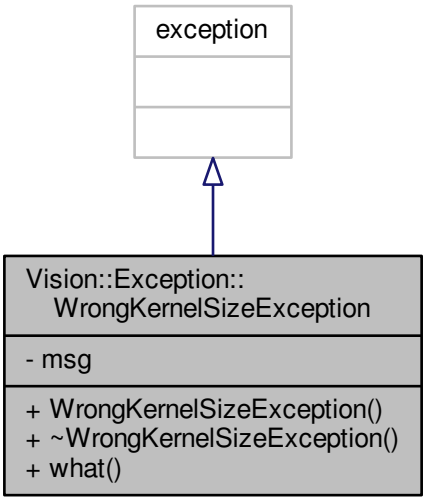
The documentation for this class was generated from the following files:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/vsamainwindow.h
- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/VSA/vsamainwindow.cpp

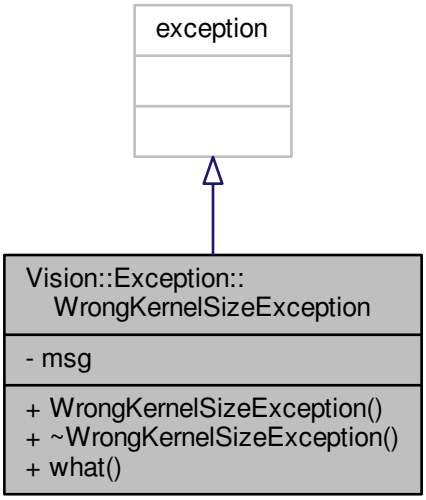
6.68 Vision::Exception::WrongKernelSizeException Class Reference

`#include <WrongKernelSizeException.h>`

Inheritance diagram for Vision::Exception::WrongKernelSizeException:



Collaboration diagram for Vision::Exception::WrongKernelSizeException:



Public Member Functions

- WrongKernelSizeException (string m="Wrong kernel dimensions!")
- ~WrongKernelSizeException () _GLIBCXX_USE_NOEXCEPT
- const char * what () const _GLIBCXX_USE_NOEXCEPT

Private Attributes

- string msg

6.68.1 Detailed Description

Definition at line 20 of file WrongKernelSizeException.h.

6.68.2 Constructor & Destructor Documentation

6.68.2.1 Vision::Exception::WrongKernelSizeException::WrongKernelSizeException (string *m* = "Wrong kernel dimensions!") [inline]

Definition at line 22 of file WrongKernelSizeException.h.

6.68.2.2 Vision::Exception::WrongKernelSizeException::~WrongKernelSizeException ( ) [inline]

Definition at line 23 of file WrongKernelSizeException.h.

6.68.3 Member Function Documentation

6.68.3.1 const char* Vision::Exception::WrongKernelSizeException::what ( ) const [inline]

Definition at line 24 of file WrongKernelSizeException.h.

6.68.4 Member Data Documentation

6.68.4.1 string Vision::Exception::WrongKernelSizeException::msg [private]

Definition at line 24 of file WrongKernelSizeException.h.

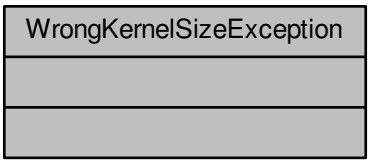
The documentation for this class was generated from the following file:

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/WrongKernelSizeException.h

6.69 WrongKernelSizeException Class Reference

`#include <WrongKernelSizeException.h>`

Collaboration diagram for WrongKernelSizeException:



6.69.1 Detailed Description

Exception class which is thrown when a wrong kernelsize is requested

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

- /home/peer23peer/programmingspace/VSA/VisionSoilAnalyzer/src/SoilVision/WrongKernelSizeException.h