

WStreamLab

1.3

Generated by Doxygen 1.11.0

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

MD5	??
MeterFlowType	??
QDialog	
HelpAbout	??
Interface	??
License	??
ReportMeasurements	??
TableBoard	??
QMainWindow	
MainWindow	??
PixellImageWidget	??
QWidget	
LedIndicator	??
RS485SettingInfo	??
SelectedInfo	??

Chapter 2

Class Index

2.1 Class List

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

HelpAbout		
	HelpAbout class represents a dialog for displaying Help/About information	??
Interface		
	Dialog window for configuring and managing settings	??
LedIndicator		
	Custom LED indicator widget	??
License		
	Dialog for displaying license information	??
MainWindow		??
MD5		
	Computes MD5 hashes of strings or byte arrays	??
MeterFlowType		
	Structure representing a water flow meter type	??
PixelFormatWidget		
	Custom widget that displays a pixelated image and handles application-specific functionalities .	??
ReportMeasurements		
	Dialog for reporting measurements	??
RS485SettingInfo		
	Structure to hold information about RS485 settings	??
SelectedInfo		
	The SelectedInfo struct holds selected information related to a device or configuration	??
TableBoard		
	Dialog window for managing and displaying water meter test data in a table format	??

Chapter 3

File Index

3.1 File List

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

C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/definitions.h	Header file containing constants and definitions for the project	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/flow-meter-type.h	Header file defining enums for flow meter types	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/helpabout.cpp	Implementation file for HelpAbout dialog functionality	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/helpabout.h	Header file for HelpAbout dialog	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/interface.cpp	Implementation file for Interface dialog functionality	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/interface.h	Header file for the Interface class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/ledindicator.cpp	Implementation file for the LedIndicator class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/ledindicator.h	Header file for the LedIndicator class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/license.cpp	Implementation of the License dialog functionality	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/license.h	Declaration of the License class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/main.cpp	Main entry point of the application	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/mainwindow.cpp	Implementation file for the MainWindow class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/mainwindow.h	Header file for the MainWindow class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/md5.cpp	MD5 class implementation	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/md5.h	Header file for the MD5 class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/report.cpp	Implementation file for the ReportMeasurements class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/report.h	Header file for the ReportMeasurements class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/tableBoard.cpp	Implementation file for the TableBoard class	??

C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/ tableBoard.h	
Header file for the TableBoard class	??
C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/ waterdensity.h	
Header file for water density calculations	??

Chapter 4

Class Documentation

4.1 HelpAbout Class Reference

[HelpAbout](#) class represents a dialog for displaying Help/About information.

```
#include <helpabout.h>
```

Inheritance diagram for HelpAbout:

4.2 Interface Class Reference

The [Interface](#) class represents a dialog window for configuring and managing settings.

```
#include <interface.h>
```

Inheritance diagram for Interface:

Collaboration diagram for Interface:

Public Member Functions

- [Interface](#) (QWidget *parent=nullptr)
Constructs the [Interface](#) dialog.
- [~Interface](#) ()
Destroys the [Interface](#) dialog.
- void [Translate](#) ()
Translates UI elements to the current language.
- bool [checkModbusAddress](#) (qint16 address)
Checks if the Modbus address is valid.

4.2.1 Detailed Description

The [Interface](#) class represents a dialog window for configuring and managing settings.

This class extends QDialog and provides functionalities for interacting with various widgets such as buttons, combo boxes, and LEDs. It manages serial port connections, Modbus configurations, and UI translations.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Interface()

```
Interface::Interface (
    QWidget * parent = nullptr) [explicit]
```

Constructs the [Interface](#) dialog.

Parameters

<i>parent</i>	The parent widget (optional).
---------------	-------------------------------

This constructor initializes the user interface, sets up signal-slot connections, validators for line edits, style sheets for various widgets, and translates text for internationalization.

Parameters

<i>parent</i>	The parent widget, usually a MainWindow .
---------------	---

4.2.2.2 ~Interface()

```
Interface::~~Interface ()
```

Destroys the [Interface](#) dialog.

Destructor for the [Interface](#) class.

This destructor deallocates resources associated with the [Interface](#) object. It specifically deletes the user interface object (ui), which manages the graphical components of the interface.

4.2.3 Member Function Documentation

4.2.3.1 checkModbusAddress()

```
bool Interface::checkModbusAddress (
    quint16 address)
```

Checks if the Modbus address is valid.

Checks the Modbus address.

Parameters

<i>address</i>	The Modbus address to check.
----------------	------------------------------

Returns

true if the address is valid, false otherwise.

This function checks if a given Modbus address is valid by sending a read request to the Modbus device. It uses a lock to ensure thread safety and updates the LED state.

Parameters

<code>address</code>	The Modbus address to check.
----------------------	------------------------------

Returns

true if the request was successfully sent, false otherwise.

4.2.3.2 Translate()

```
void Interface::Translate ()
```

Translates UI elements to the current language.

Translates the UI text to the current language.

This function translates all UI elements to the current language using the current locale settings.

This function sets the translated text for various widgets in the UI based on the current language settings.

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

- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[interface.h](#)
- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[interface.cpp](#)

4.3 LedIndicator Class Reference

The [LedIndicator](#) class represents a custom LED indicator widget.

```
#include <ledindicator.h>
```

Inheritance diagram for LedIndicator:

Collaboration diagram for LedIndicator:

Public Slots

- void [switchLedIndicator](#) ()

Slot function to switch the state of the LED indicator. This slot is triggered to toggle the state of the LED indicator.

Public Member Functions

- [LedIndicator](#) (QWidget *parent=nullptr)
Constructs a [LedIndicator](#) widget.
- void [setState](#) (bool state)
Sets the state of the LED indicator.
- void [toggle](#) ()
Toggles the state of the LED indicator. If the LED is currently on, it will be turned off, and vice versa.
- void [setOnColor](#) (QColor onColor)
Sets the color of the LED when it is turned on.
- void [setOffColor](#) (QColor offColor)
Sets the color of the LED when it is turned off.
- void [setOnPattern](#) (Qt::BrushStyle onPattern)
Sets the pattern of the LED when it is turned on.
- void [setOffPattern](#) (Qt::BrushStyle offPattern)
Sets the pattern of the LED when it is turned off.
- void [setLedSize](#) (int size)
Sets the size of the LED indicator.

Protected Member Functions

- void [paintEvent](#) (QPaintEvent *event) override
Overrides the paint event to render the LED indicator.

4.3.1 Detailed Description

The [LedIndicator](#) class represents a custom LED indicator widget.

This class provides functionality to display an LED-like indicator with customizable colors, patterns, and size. It inherits from QWidget and overrides paintEvent to customize the rendering of the LED indicator.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 LedIndicator()

```
LedIndicator::LedIndicator (
    QWidget * parent = nullptr) [explicit]
```

Constructs a [LedIndicator](#) widget.

Constructs a [LedIndicator](#) widget with default settings.

Parameters

<i>parent</i>	Optional pointer to the parent widget.
<i>parent</i>	The parent widget (optional).

< Color of the LED when it is on

< Color of the LED when it is off

< Pattern of the LED when it is on (solid)

< Pattern of the LED when it is off (solid)

< Neutral pattern of the LED (solid)

4.3.3 Member Function Documentation

4.3.3.1 `paintEvent()`

```
void LedIndicator::paintEvent (
    QPaintEvent * event) [override], [protected]
```

Overrides the paint event to render the LED indicator.

Paints the LED indicator widget.

Parameters

<i>event</i>	The paint event.
--------------	------------------

This function is called automatically whenever the widget needs to be repainted. It draws an ellipse representing the LED, filled with a color and pattern based on the current state (`lit`), and outlines it with a corresponding color.

Parameters

<i>event</i>	A paint event (unused).
--------------	-------------------------

4.3.3.2 `setLedSize()`

```
void LedIndicator::setLedSize (
    int size)
```

Sets the size of the LED indicator.

Sets the size of the LED and triggers a repaint.

Parameters

<i>size</i>	The size (width and height) to set for the LED.
<i>size</i>	The size to set for the LED.

4.3.3.3 `setOffColor()`

```
void LedIndicator::setOffColor (
    QColor offColor)
```

Sets the color of the LED when it is turned off.

Sets the color of the LED when it is off and triggers a repaint.

Parameters

<i>offColor</i>	The color to set when the LED is off.
-----------------	---------------------------------------

4.3.3.4 `setOffPattern()`

```
void LedIndicator::setOffPattern (
    Qt::BrushStyle offPattern)
```

Sets the pattern of the LED when it is turned off.

Sets the pattern of the LED when it is off and triggers a repaint.

Parameters

<i>offPattern</i>	The brush style pattern to set when the LED is off.
<i>offPattern</i>	The pattern to set when the LED is off (e.g., solid, dense, etc.).

4.3.3.5 setOnColor()

```
void LedIndicator::setOnColor (
    QColor onColor)
```

Sets the color of the LED when it is turned on.

Sets the color of the LED when it is on and triggers a repaint.

Parameters

<i>onColor</i>	The color to set when the LED is on.
----------------	--------------------------------------

4.3.3.6 setOnPattern()

```
void LedIndicator::setOnPattern (
    Qt::BrushStyle onPattern)
```

Sets the pattern of the LED when it is turned on.

Sets the pattern of the LED when it is on and triggers a repaint.

Parameters

<i>onPattern</i>	The brush style pattern to set when the LED is on.
<i>onPattern</i>	The pattern to set when the LED is on (e.g., solid, dense, etc.).

4.3.3.7 setState()

```
void LedIndicator::setState (
    bool state)
```

Sets the state of the LED indicator.

Sets the state of the LED indicator and triggers a repaint.

Parameters

<i>state</i>	true to turn the LED on, false to turn it off.
<i>state</i>	The state to set (true for on, false for off).

4.3.3.8 switchLedIndicator

```
void LedIndicator::switchLedIndicator () [slot]
```

Slot function to switch the state of the LED indicator. This slot is triggered to toggle the state of the LED indicator.

Toggles the state of the LED indicator and triggers a repaint.

4.3.3.9 toggle()

```
void LedIndicator::toggle ()
```

Toggles the state of the LED indicator. If the LED is currently on, it will be turned off, and vice versa.

Toggles the state of the LED indicator and triggers a repaint.

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

- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/ledindicator.h
- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/ledindicator.cpp

4.4 License Class Reference

The [License](#) class represents a dialog for displaying license information.

```
#include <license.h>
```

Inheritance diagram for License:

Collaboration diagram for License:

Public Member Functions

- [License](#) (QWidget *parent=nullptr)
Constructs a [License](#) dialog.
- [~License](#) ()
Destroys the [License](#) dialog.
- void [Translate](#) ()
Translates the UI components to the current language.

Public Attributes

- Ui::Licence * [ui](#)
Pointer to the UI object.

4.4.1 Detailed Description

The [License](#) class represents a dialog for displaying license information.

This class inherits from QDialog and provides methods to translate UI components, handle the show event, and manage the dialog's lifecycle.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 License()

```
License::License (
    QWidget * parent = nullptr) [explicit]
```

Constructs a [License](#) dialog.

Parameters

<i>parent</i>	Pointer to the parent widget (optional).
---------------	--

This constructor initializes the [License](#) dialog by setting up the user interface defined in `Ui::License`. It performs translation of UI elements and connects the close button click signal to the `onCloseClicked` slot.

Parameters

<i>parent</i>	Pointer to the parent widget.
---------------	-------------------------------

< Set up the user interface.

< Cast parent to [MainWindow](#) pointer.

< Translate UI elements.

4.4.2.2 ~License()

```
License::~License ()
```

Destroys the [License](#) dialog.

Destructor for the [License](#) dialog.

Deletes the user interface object (ui) associated with the [License](#) dialog. This ensures that resources allocated for the user interface are properly freed when the [License](#) dialog is destroyed. < Delete the user interface object.

4.4.3 Member Function Documentation**4.4.3.1 Translate()**

```
void License::Translate ()
```

Translates the UI components to the current language.

Translates and sets text for UI elements in the [License](#) dialog.

This function sets text for various UI components like labels and buttons based on the current application language settings.

This function translates and sets the text for various `QLabel` and `QPushButton` UI elements in the [License](#) dialog using Qt's translation mechanism (`tr()`). It ensures that all UI elements display text in the appropriate language specified by the application's current locale. < Set the window title.

< Set text for certificate label.

< Set text for water density label.

< Set text for density unit label.

< Set text for archive folder label.

< Set text for company label.

< Set text for maximum entries label.

< Set text for checksum label.

< Set text for close button.

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

- `C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/license.h`
- `C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/license.cpp`

4.5 MainWindow Class Reference

Inheritance diagram for MainWindow:

Collaboration diagram for MainWindow:

Public Types

- typedef const wchar_t *(* [EnumerateSerialPorts](#)) ()

Signals

- void **numberOfWaterMetersChangedSignal** ()
Signal emitted when the number of water meters is changed.
- void **meterTypeChangedSignal** ()
Signal emitted when the meter type is changed.
- void **measurementTypeChangedSignal** ()
Signal emitted when the measurement type is changed.

Public Member Functions

- [MainWindow](#) (QWidget *parent=nullptr)
Constructs a [MainWindow](#) object.
- [~MainWindow](#) ()
Destroys the [MainWindow](#) object.
- void [Translate](#) ()
Translates the UI components to the selected language.
- void [setLabelValue](#) (QLabel *label, double value, int precision)
Sets the value of a QLabel with specified precision.
- void [updateSelectedInfo](#) ()
Updates the selected information based on user inputs.
- void [SelectMeterComboBox](#) ()
Initializes the ComboBox for selecting meter types.
- void [ReadConfiguration](#) ()
Reads the application configuration from settings.
- void [SetDefaultConfiguration](#) ()
Sets the default configuration settings.
- void [CenterToScreen](#) (QWidget *widget)
Centers a given widget to the screen.

Public Attributes

- [SelectedInfo](#) selectedInfo
- Ui::MainWindow * ui {nullptr}
- [TableBoard](#) * inputData {nullptr}
- [License](#) * licenseDialog {nullptr}
- [HelpAbout](#) * helpAbout
- [Interface](#) * interfaceDialog {nullptr}
- QActionGroup * alignmentGroup
- [LedIndicator](#) * LED
- unsigned [MAX_NR_WATER_METERS](#) {20}
- unsigned [NUMBER_ENTRIES_METER_FLOW_DB](#) {0}
- std::map< std::string, std::string > [optionsConfiguration](#)
- [EnumerateSerialPorts](#) serialPorts {nullptr}

Protected Member Functions

- void [mousePressEvent](#) (QMouseEvent *event) override
Handles mouse press events on the main window.
- bool [eventFilter](#) (QObject *obj, QEvent *event) override
Filters events for the main window.

4.5.1 Member Typedef Documentation

4.5.1.1 EnumerateSerialPorts

```
typedef const wchar_t *(* MainWindow::EnumerateSerialPorts) ()
```

Function pointer type for serial port enumeration.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 MainWindow()

```
MainWindow::MainWindow (
    QWidget * parent = nullptr) [explicit]
```

Constructs a [MainWindow](#) object.

Constructor for [MainWindow](#) class.

Parameters

<i>parent</i>	The parent widget (default: nullptr).
---------------	---------------------------------------

Initializes the main window of the application, sets up UI elements, loads necessary libraries, handles settings and configurations, connects signals and slots, and initializes dialogs.

Parameters

<i>parent</i>	The parent widget.
---------------	--------------------

4.5.2.2 ~MainWindow()

```
MainWindow::~MainWindow ()
```

Destroys the [MainWindow](#) object.

Destructor for [MainWindow](#) class.

Cleans up resources associated with the [MainWindow](#).

4.5.3 Member Function Documentation

4.5.3.1 CenterToScreen()

```
void MainWindow::CenterToScreen (
    QWidget * widget)
```

Centers a given widget to the screen.

Centers a widget to the screen.

Parameters

<i>widget</i>	Pointer to the widget to center.
---------------	----------------------------------

Moves the given widget to the center of the primary screen.

Parameters

<i>widget</i>	The widget to be centered.
---------------	----------------------------

4.5.3.2 eventFilter()

```
bool MainWindow::eventFilter (  
    QObject * obj,  
    QEvent * event) [inline], [override], [protected]
```

Filters events for the main window.

Parameters

<i>obj</i>	The object that received the event.
<i>event</i>	The event that occurred.

Returns

True if the event was handled, otherwise false.

4.5.3.3 mousePressEvent()

```
void MainWindow::mousePressEvent (  
    QMouseEvent * event) [override], [protected]
```

Handles mouse press events on the main window.

Handles the mouse press event.

Parameters

<i>event</i>	The mouse event.
--------------	------------------

Activates the main window when a mouse press event occurs.

Parameters

<i>event</i>	The mouse event object.
--------------	-------------------------

4.5.3.4 ReadConfiguration()

```
void MainWindow::ReadConfiguration ()
```

Reads the application configuration from settings.

Reads and parses configuration settings from a file.

This function reads the configuration file specified by 'watermeters.conf' located in the application's directory. It parses key-value pairs separated by '=' and terminates each entry with '>'. It then validates the MD5 checksum formed by concatenating 'company' and 'maximum' fields against a stored control checksum ('control'). If validation succeeds, default configuration values are updated.

If the configuration file cannot be opened or if validation fails, default configuration values are set using [SetDefaultConfiguration\(\)](#). e.g. company=Compania de Apa Braila> archive=C:/Stand/Fise> maximum=20> certificate=CE 06.02-355/15> density_20=998> control=f1807e24ccba79a76baa08194b7fa9bf>

company + density_20 => MD5

4.5.3.5 SelectMeterComboBox()

```
void MainWindow::SelectMeterComboBox ()
```

Initializes the ComboBox for selecting meter types.

Updates selectedInfo based on the currently selected water meter, creates necessary directories, and updates UI labels.

This function performs the following tasks:

1. Calls [updateSelectedInfo\(\)](#) to update selectedInfo with current configuration and UI data.
2. Creates directories for results and input data using selectedInfo.pathResults.
3. Updates various QLabel widgets in the UI with values from selectedInfo.

4.5.3.6 SetDefaultConfiguration()

```
void MainWindow::SetDefaultConfiguration ()
```

Sets the default configuration settings.

Sets default configuration values for optionsConfiguration.

This function clears the optionsConfiguration map and initializes default values for various configuration keys:

- "company": Default value is "NONE".
- "archive": Default path is "C:/Stand/Fise".
- "maximum": Default value is "2".
- "certificate": Default value is "NONE".
- "density_20": Default value is "998.2009".
- "control": Default value is "00000000000000000000000000000000".

4.5.3.7 setLabelValue()

```
void MainWindow::setLabelValue (
    QLabel * label,
    double value,
    int precision)
```

Sets the value of a QLabel with specified precision.

Sets the text of a QLabel to display a double value with specified precision.

Parameters

<i>label</i>	Pointer to the QLabel widget.
<i>value</i>	Value to set on the label.
<i>precision</i>	Number of decimal places for the value.

This function converts a double value to a string with fixed precision and sets the text of the specified QLabel to display this string.

Parameters

<i>label</i>	The QLabel widget whose text will be set.
<i>value</i>	The double value to be displayed.
<i>precision</i>	The number of decimal places to display.

4.5.3.8 Translate()

```
void MainWindow::Translate ()
```

Translates the UI components to the selected language.

Translates all UI elements to the current language.

This function translates all visible UI elements to the current language. It sets the window title, menu titles, action texts, labels, group box titles, radio button texts, and push button texts.

4.5.3.9 updateSelectedInfo()

```
void MainWindow::updateSelectedInfo ()
```

Updates the selected information based on user inputs.

Updates the selectedInfo structure with current configuration and UI data.

This function updates the selectedInfo structure with the following information:

- Parameters from optionsConfiguration: density_20, archive path, certificate, and number of water meters.
- Lab conditions from settings: ambient temperature, relative air humidity, and atmospheric pressure.
- Selected water meter information from UI: name, nominal diameter, nominal flow, maximum flow, transition flow, minimum flow, nominal error, and maximum error.

4.5.4 Member Data Documentation**4.5.4.1 alignmentGroup**

```
QActionGroup* MainWindow::alignmentGroup
```

Action group for alignment settings.

4.5.4.2 helpAbout

`HelpAbout*` MainWindow::helpAbout

Pointer to the help/about dialog.

4.5.4.3 inputData

`TableBoard*` MainWindow::inputData {nullptr}

Pointer to the input data board.

4.5.4.4 interfaceDialog

`Interface*` MainWindow::interfaceDialog {nullptr}

Pointer to the interface dialog.

4.5.4.5 LED

`LedIndicator*` MainWindow::LED

Pointer to the LED indicator.

4.5.4.6 licenseDialog

`License*` MainWindow::licenseDialog {nullptr}

Pointer to the license dialog.

4.5.4.7 MAX_NR_WATER_METERS

`unsigned` MainWindow::MAX_NR_WATER_METERS {20}

Maximum number of water meters supported.

4.5.4.8 NUMBER_ENTRIES_METER_FLOW_DB

`unsigned` MainWindow::NUMBER_ENTRIES_METER_FLOW_DB {0}

Number of entries in meter flow database.

4.5.4.9 optionsConfiguration

`std::map<std::string, std::string>` MainWindow::optionsConfiguration

Map for storing configuration options.

4.5.4.10 selectedInfo

`SelectedInfo` MainWindow::selectedInfo

Holds selected information related to the application.

4.5.4.11 serialPorts

`EnumerateSerialPorts` MainWindow::serialPorts {nullptr}

Pointer to the function for serial port enumeration.

4.5.4.12 ui

`Ui::MainWindow*` MainWindow::ui {nullptr}

Pointer to the UI components of the main window.

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

- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[mainwindow.h](#)
- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[mainwindow.cpp](#)

4.6 MD5 Class Reference

The `MD5` class computes `MD5` hashes of strings or byte arrays.

```
#include <md5.h>
```

Public Types

- typedef unsigned int **size_type**

Public Member Functions

- `MD5` ()
Default constructor initializes the `MD5` state.
- `MD5` (const std::string &text)
Constructor that initializes the `MD5` state with an initial string.
- void `update` (const unsigned char *buf, size_type length)
Update the hash with a block of unsigned characters.
- void `update` (const char *buf, size_type length)
Update the hash with a block of characters.
- `MD5` & `finalize` ()
Finalize the `MD5` computation.
- std::string `hexdigest` () const
Get the hexadecimal representation of the `MD5` hash.

Friends

- `std::ostream & operator<< (std::ostream &os, MD5 md5)`

Output operator to stream the MD5 object.

4.6.1 Detailed Description

The MD5 class computes MD5 hashes of strings or byte arrays.

Implementation of the MD5 Message-Digest Algorithm.

It is not designed for speed or security, but rather as a simple implementation for educational purposes.

Usage: 1) Feed it blocks of uchars with `update()` 2) Finalize() 3) Get `hexdigest()` string or use `MD5(std::string).hexdigest()`

Assumes that char is 8-bit and int is 32-bit.

This class provides methods to calculate MD5 hashes for strings or byte arrays. It is based on the reference implementation of RFC 1321 by RSA Data Security, Inc.

Note

This implementation is not intended for speed or security.

Copyright

Copyright (C) 1991-2, RSA Data Security, Inc. All rights reserved. \license [License](#) to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function. [License](#) is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work. RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

Note

These notices must be retained in any copies of any part of this documentation and/or software.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 MD5() [1/2]

```
MD5::MD5 ()
```

Default constructor initializes the MD5 state.

Default constructor for MD5 hash computation.

Initializes the MD5 hashing algorithm by calling the `init()` function. This constructor sets up the initial state of the MD5 calculation.

4.6.2.2 MD5() [2/2]

```
MD5::MD5 (
    const std::string & text) [explicit]
```

Constructor that initializes the MD5 state with an initial string.

Parameters

<i>text</i>	Initial string to hash using MD5.
-------------	-----------------------------------

4.6.3 Member Function Documentation

4.6.3.1 finalize()

```
MD5 & MD5::finalize ()
```

Finalize the MD5 computation.

Finalizes the MD5 message-digest operation.

Returns

Reference to the MD5 object.

Finalizes the current MD5 context, computes the message digest, and clears sensitive information from the context for security. This function should be called once after all updates are done.

Returns

Reference to the current MD5 instance.

4.6.3.2 hexdigest()

```
std::string MD5::hexdigest () const
```

Get the hexadecimal representation of the MD5 hash.

Returns the hexadecimal representation of the MD5 digest.

Returns

Hexadecimal string representing the MD5 hash.

Converts the 16-byte MD5 digest into a hexadecimal string representation. The function returns an empty string if the MD5 context has not been finalized.

Returns

Hexadecimal string representation of the MD5 digest.

4.6.3.3 update() [1/2]

```
void MD5::update (  
    const char * buf,  
    size_type length)
```

Update the hash with a block of characters.

Parameters

<i>buf</i>	Pointer to the buffer of characters.
<i>length</i>	Length of the buffer in <code>size_type</code> units.

4.6.3.4 update() [2/2]

```
void MD5::update (
    const unsigned char * buf,
    size_type length)
```

Update the hash with a block of unsigned characters.

Parameters

<i>buf</i>	Pointer to the buffer of unsigned characters.
<i>length</i>	Length of the buffer in <code>size_type</code> units.

4.6.4 Friends And Related Symbol Documentation**4.6.4.1 operator<<**

```
std::ostream & operator<< (
    std::ostream & os,
    MD5 md5) [friend]
```

Output operator to stream the [MD5](#) object.

Parameters

<i>os</i>	Output stream.
<i>md5</i>	MD5 object to output.

Returns

Reference to the output stream.

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

- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[md5.h](#)
- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[md5.cpp](#)

4.7 MeterFlowType Struct Reference

Structure representing a water flow meter type.

```
#include <flow-meter-type.h>
```

Public Member Functions

- [MeterFlowType](#) (const std::string &_nameWaterMeter, unsigned _nominalDiameter, double _nominalFlow, double _maximumFlow, double _transitionFlow, double _minimumFlow, double _nominalError, double _↵ maximumError)
 Constructor to initialize [MeterFlowType](#) with specific values.
- [MeterFlowType](#) ()
 Default constructor to initialize [MeterFlowType](#) with default values.

Public Attributes

- std::string **nameWaterMeter**
 Name of the water meter.
- unsigned **nominalDiameter**
 Nominal diameter of the water meter.
- double **nominalFlow**
 Nominal flow rate of the water meter.
- double **maximumFlow**
 Maximum flow rate of the water meter.
- double **transitionFlow**
 Transition flow rate of the water meter.
- double **minimumFlow**
 Minimum flow rate of the water meter.
- double **nominalError**
 Nominal error margin of the water meter.
- double **maximumError**
 Maximum error margin of the water meter.

4.7.1 Detailed Description

Structure representing a water flow meter type.

This structure holds various properties of a water flow meter, including its name, nominal diameter, flow rates, and error margins.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 MeterFlowType() [1/2]

```
MeterFlowType::MeterFlowType (
    const std::string & _nameWaterMeter,
    unsigned _nominalDiameter,
    double _nominalFlow,
    double _maximumFlow,
    double _transitionFlow,
    double _minimumFlow,
    double _nominalError,
    double _maximumError) [inline]
```

Constructor to initialize [MeterFlowType](#) with specific values.

Parameters

<code>_nameWaterMeter</code>	Name of the water meter.
<code>_nominalDiameter</code>	Nominal diameter of the water meter.
<code>_nominalFlow</code>	Nominal flow rate of the water meter.
<code>_maximumFlow</code>	Maximum flow rate of the water meter.
<code>_trasitionFlow</code>	Transition flow rate of the water meter.
<code>_minimumFlow</code>	Minimum flow rate of the water meter.
<code>_nominalError</code>	Nominal error margin of the water meter.
<code>_maximumError</code>	Maximum error margin of the water meter.

4.7.2.2 MeterFlowType() [2/2]

```
MeterFlowType::MeterFlowType () [inline]
```

Default constructor to initialize [MeterFlowType](#) with default values.

Initializes all properties of the water meter to default values.

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

- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[flow-meter-type.h](#)

4.8 PixellImageWidget Class Reference

The [PixellImageWidget](#) class represents a custom widget that displays a pixelated image and handles application-specific functionalities.

Inheritance diagram for [PixellImageWidget](#):

Collaboration diagram for [PixellImageWidget](#):

Public Member Functions

- [PixellImageWidget](#) (QWidget *parent=nullptr)
Constructor for [PixellImageWidget](#).

Public Attributes

- [MainWindow](#) * **mainWindow** = nullptr

Protected Member Functions

- QRect [centeredRect](#) (const QSize &outer, const QSize &inner)
Calculates the centered rectangle within an outer rectangle.
- std::wstring [ExePath](#) ()
Retrieves the path of the executable file.
- void [paintEvent](#) (QPaintEvent *) override
Paints the widget with a gradient and textual information.

4.8.1 Detailed Description

The [PixelImageWidget](#) class represents a custom widget that displays a pixelated image and handles application-specific functionalities.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 PixelImageWidget()

```
PixelImageWidget::PixelImageWidget (  
    QWidget * parent = nullptr) [inline], [explicit]
```

Constructor for [PixelImageWidget](#).

Constructs a [PixelImageWidget](#) instance with optional parent widget. Initializes the widget with translucent background, frameless window hint, and a fixed size of 400x300 pixels. Sets up a timer to hide the widget after five seconds.

Parameters

<i>parent</i>	Optional parent widget (default is nullptr).
---------------	--

4.8.3 Member Function Documentation

4.8.3.1 centeredRect()

```
QRect PixelImageWidget::centeredRect (  
    const QSize & outer,  
    const QSize & inner) [inline], [protected]
```

Calculates the centered rectangle within an outer rectangle.

Calculates and returns a QRect that represents a centered rectangle within the given outer and inner sizes.

Parameters

<i>outer</i>	Size of the outer rectangle.
<i>inner</i>	Size of the inner rectangle.

Returns

QRect representing the centered rectangle.

4.8.3.2 ExePath()

```
std::wstring PixelImageWidget::ExePath () [inline], [protected]
```

Retrieves the path of the executable file.

Retrieves the path of the current executable file.

Returns

std::wstring containing the path of the executable.

4.8.3.3 paintEvent()

```
void PixelImageWidget::paintEvent (
    QPaintEvent * ) [inline], [override], [protected]
```

Paints the widget with a gradient and textual information.

This method overrides the QWidget::paintEvent and is responsible for painting the widget with a gradient from red to blue across its entire area. It also displays textual information at the top of the widget.

Parameters

<i>event</i>	The paint event that triggered this method.
--------------	---

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

- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[main.cpp](#)

4.9 ReportMeasurements Class Reference

The [ReportMeasurements](#) class provides a dialog for reporting measurements.

```
#include <report.h>
```

Inheritance diagram for ReportMeasurements:

Collaboration diagram for ReportMeasurements:

Signals

- void **pdfGenerationCompleted** ()
Signal emitted when PDF generation is completed.

Public Member Functions

- [ReportMeasurements](#) (QWidget *parent, const std::vector< QCheckBox * > &vectorCheckNumber, const std::vector< QLineEdit * > &vectorSerialNumber, const QString resultAllTests[MAX_ARRAY_SIZE])
Constructs a [ReportMeasurements](#) dialog.
- [~ReportMeasurements](#) ()
Destroys the [ReportMeasurements](#) dialog.
- void [Translate](#) ()
Translates the UI elements to the selected language.

Static Public Member Functions

- static void [printPdfThread](#) (QString report)
Static method to start a thread for PDF generation.

4.9.1 Detailed Description

The [ReportMeasurements](#) class provides a dialog for reporting measurements.

This class allows users to generate and print a report based on selected checkboxes and serial numbers. It supports PDF generation and provides signals upon completion.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 ReportMeasurements()

```
ReportMeasurements::ReportMeasurements (
    QWidget * parent,
    const std::vector< QCheckBox * > & vectorCheckNumber,
    const std::vector< QLineEdit * > & vectorSerialNumber,
    const QString resultAllTests[MAX_ARRAY_SIZE]) [explicit]
```

Constructs a [ReportMeasurements](#) dialog.

Constructor for the [ReportMeasurements](#) class.

Parameters

<i>parent</i>	The parent widget.
<i>vectorCheckNumber</i>	Vector of QCheckBox pointers for measurement selection.
<i>vectorSerialNumber</i>	Vector of QLineEdit pointers for serial numbers.
<i>resultAllTests</i>	Array of QStrings containing test results.

Initializes the [ReportMeasurements](#) dialog with the provided parent widget, sets up the user interface, copies arrays, and initializes timer and connections.

Parameters

<i>parent</i>	Parent widget to which this dialog belongs.
<i>vectorCheckNumber</i>	Vector of QCheckBox pointers used for storing check numbers.
<i>vectorSerialNumber</i>	Vector of QLineEdit pointers used for storing serial numbers.
<i>resultAllTests</i>	Array of QString containing results of all tests.

4.9.2.2 ~ReportMeasurements()

```
ReportMeasurements::~~ReportMeasurements ()
```

Destroys the [ReportMeasurements](#) dialog.

Destructor for the [ReportMeasurements](#) class.

Cleans up the user interface (ui) resources.

4.9.3 Member Function Documentation

4.9.3.1 printPdfThread()

```
void ReportMeasurements::printPdfThread (  
    QString report) [static]
```

Static method to start a thread for PDF generation.

Generates a PDF document from HTML content and opens it using the default PDF viewer.

Parameters

<i>report</i>	The report to generate in PDF format.
---------------	---------------------------------------

This function generates a PDF document from the provided HTML report content. It ensures thread safety when accessing shared resources using a mutex. After generating the PDF, it checks for errors and opens the generated PDF file using the default PDF viewer.

Parameters

<i>report</i>	The HTML content to be printed into the PDF.
---------------	--

4.9.3.2 Translate()

```
void ReportMeasurements::Translate ()
```

Translates the UI elements to the selected language.

Translates UI elements and sets localized text for the [ReportMeasurements](#) dialog.

This function translates various UI elements and sets localized text using the current application's translation system. It is typically called during initialization or when the language/locale is changed to ensure the UI reflects the selected language.

Translated elements include window title, group box title, labels, combo box items, and push button texts.

Example usage:

```
ReportMeasurements report;  
report.Translate();
```

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

- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[report.h](#)
- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[report.cpp](#)

4.10 RS485SettingInfo Struct Reference

Structure to hold information about RS485 settings.

Public Attributes

- `const char * key`
Pointer to a constant character array representing the setting's identifier.
- `const QVariant defaultValue`
QVariant storing the default value associated with the setting.

4.10.1 Detailed Description

Structure to hold information about RS485 settings.

This struct defines a key-value pair where:

- `key` is a pointer to a constant character array representing the setting's identifier.
- `defaultValue` is a QVariant storing the default value associated with the setting.

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

- `C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/mainwindow.cpp`

4.11 SelectedInfo Struct Reference

The [SelectedInfo](#) struct holds selected information related to a device or configuration.

```
#include <mainwindow.h>
```

Public Member Functions

- [SelectedInfo](#) ()
Default constructor initializes all members to default values.

Public Attributes

- float [density_20](#)
- std::string [pathResults](#)
- size_t [entriesNumber](#)
- std::string [certificate](#)
- std::string [nameWaterMeter](#)
- unsigned [nominalDiameter](#)
- double [nominalFlow](#)
- double [maximumFlow](#)
- double [transitionFlow](#)
- double [minimumFlow](#)
- double [nominalError](#)
- double [maximumError](#)
- std::string [ambientTemperature](#)
- std::string [athmosphericPressure](#)
- std::string [relativeAirHumidity](#)
- bool [rbGravimetric_new](#)
- bool [rbVolumetric](#)
- bool [rbManual](#)
- bool [rbInterface](#)
- bool [rbTerminal](#)
- bool [serialPort](#)
- SELECTED_LANGUAGE [selectedLanguage](#)
- QModbusClient * [modbusDevice](#)

4.11.1 Detailed Description

The [SelectedInfo](#) struct holds selected information related to a device or configuration.

This struct encapsulates various parameters and flags that represent selected settings and state for a device or application configuration.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 SelectedInfo()

```
SelectedInfo::SelectedInfo () [inline]
```

Default constructor initializes all members to default values.

< Density at 20 degrees Celsius. < Number of entries. < Nominal diameter of the device. < Nominal flow rate. < Maximum flow rate. < Transition flow rate. < Minimum flow rate. < Nominal error. < Maximum error. < Ambient temperature. < Atmospheric pressure. < Relative air humidity. < Gravimetric measurement flag. < Volumetric measurement flag. < Manual operation mode flag. < [Interface](#) operation mode flag. < Terminal operation mode flag. < Serial port usage flag. < Selected language enumeration. < Pointer to the Modbus client device.

4.11.3 Member Data Documentation

4.11.3.1 ambientTemperature

```
std::string SelectedInfo::ambientTemperature
```

Ambient temperature.

4.11.3.2 atmosphericPressure

```
std::string SelectedInfo::atmosphericPressure
```

Atmospheric pressure.

4.11.3.3 certificate

```
std::string SelectedInfo::certificate
```

Certificate information.

4.11.3.4 density_20

```
float SelectedInfo::density_20
```

Density at 20 degrees Celsius.

4.11.3.5 entriesNumber

```
size_t SelectedInfo::entriesNumber
```

Number of entries.

4.11.3.6 maximumError

```
double SelectedInfo::maximumError
```

Maximum error.

4.11.3.7 maximumFlow

```
double SelectedInfo::maximumFlow
```

Maximum flow rate.

4.11.3.8 minimumFlow

```
double SelectedInfo::minimumFlow
```

Minimum flow rate.

4.11.3.9 modbusDevice

```
QModbusClient* SelectedInfo::modbusDevice
```

Pointer to the Modbus client device.

4.11.3.10 nameWaterMeter

```
std::string SelectedInfo::nameWaterMeter
```

Name of the water meter.

4.11.3.11 nominalDiameter

```
unsigned SelectedInfo::nominalDiameter
```

Nominal diameter of the device.

4.11.3.12 nominalError

```
double SelectedInfo::nominalError
```

Nominal error.

4.11.3.13 nominalFlow

```
double SelectedInfo::nominalFlow
```

Nominal flow rate.

4.11.3.14 pathResults

```
std::string SelectedInfo::pathResults
```

Path for storing results.

4.11.3.15 rbGravimetric_new

```
bool SelectedInfo::rbGravimetric_new
```

Gravimetric measurement flag.

4.11.3.16 rbInterface

```
bool SelectedInfo::rbInterface
```

[Interface](#) operation mode flag.

4.11.3.17 rbManual

```
bool SelectedInfo::rbManual
```

Manual operation mode flag.

4.11.3.18 rbTerminal

```
bool SelectedInfo::rbTerminal
```

Terminal operation mode flag.

4.11.3.19 rbVolumetric

```
bool SelectedInfo::rbVolumetric
```

Volumetric measurement flag.

4.11.3.20 relativeAirHumidity

```
std::string SelectedInfo::relativeAirHumidity
```

Relative air humidity.

4.11.3.21 selectedLanguage

```
SELECTED_LANGUAGE SelectedInfo::selectedLanguage
```

Selected language enumeration.

4.11.3.22 serialPort

```
bool SelectedInfo::serialPort
```

Serial port usage flag.

4.11.3.23 trasiitionFlow

```
double SelectedInfo::trasiitionFlow
```

Transition flow rate.

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

- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[mainwindow.h](#)

4.12 TableBoard Class Reference

Dialog window for managing and displaying water meter test data in a table format.

```
#include <tableBoard.h>
```

Inheritance diagram for TableBoard:

Collaboration diagram for TableBoard:

Public Member Functions

- [TableBoard](#) (QWidget * _parent=nullptr)
Constructs a [TableBoard](#) dialog.
- [~TableBoard](#) ()
Destroys the [TableBoard](#) dialog.
- void [ValidatorInput](#) ()
Validates input data in the table.
- void [PopulateTable](#) ()
Populates the table with initial data.
- void [Translate](#) ()
Translates the UI components to the current language.

Static Public Member Functions

- static void [printPdfThread](#) (QString report)
Initiates a PDF generation process in a separate thread.

Protected Member Functions

- void [showEvent](#) (QShowEvent *event)
Event handler for when the dialog is shown.

4.12.1 Detailed Description

Dialog window for managing and displaying water meter test data in a table format.

This class extends QDialog and provides functionality for input validation, populating the table with data, and managing PDF report generation.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 TableBoard()

```
TableBoard::TableBoard (
    QWidget * _parent = nullptr) [explicit]
```

Constructs a [TableBoard](#) dialog.

Constructor for [TableBoard](#) class.

Parameters

<code>_parent</code>	Pointer to the parent widget (default: nullptr).
----------------------	--

Initializes the [TableBoard](#) dialog with the given parent widget, sets up UI, connects signals to slots, and configures window flags.

Parameters

<code>_parent</code>	Pointer to the parent widget.
----------------------	-------------------------------

4.12.2.2 ~TableBoard()

```
TableBoard::~~TableBoard ()
```

Destroys the [TableBoard](#) dialog.

Destructor for [TableBoard](#) class.

Cleans up resources used by the [TableBoard](#) dialog. Deletes the UI instance and resets input data in the [MainWindow](#) if applicable.

4.12.3 Member Function Documentation**4.12.3.1 PopulateTable()**

```
void TableBoard::PopulateTable ()
```

Populates the table with initial data.

Populates the table with data based on selected information from the main window.

Initializes and fills the table with default or stored data.

This function updates the UI elements in the dialog to reflect the selected water meter information from the main window. It adjusts visibility, sets palettes for rows, and updates text fields with relevant data.

It also initializes and sets palettes for alternating row colors to improve readability.

This function assumes that the necessary data (entries, nameWaterMeter, minimumFlowMain, transitoriuFlow↔Main, nominalFlowMain, nominalError, maximumError) have already been set in the main window (mainwindow->selectedInfo).

4.12.3.2 printPdfThread()

```
void TableBoard::printPdfThread (
    QString report) [static]
```

Initiates a PDF generation process in a separate thread.

Generates a PDF document from the provided HTML report and saves it.

Parameters

<code>report</code>	Path or identifier of the report to generate.
---------------------	---

This function generates a PDF document using QTextDocument and QPrinter, based on the provided HTML report. The generated PDF is saved to a file with a timestamped filename in a specified directory.

Parameters

<i>report</i>	The HTML report content to be converted to PDF.
---------------	---

4.12.3.3 showEvent()

```
void TableBoard::showEvent (
    QShowEvent * event) [protected]
```

Event handler for when the dialog is shown.

Overridden function called when the dialog is shown.

Parameters

<i>event</i>	Pointer to the QShowEvent being received.
--------------	---

This function is automatically called when the dialog is shown. It ensures that the table is populated with data when the dialog is displayed.

Parameters

<i>event</i>	A QShowEvent object.
--------------	----------------------

4.12.3.4 Translate()

```
void TableBoard::Translate ()
```

Translates the UI components to the current language.

Translates and updates UI elements with localized texts.

Adjusts all visible UI elements to display text in the chosen language.

This function updates various UI elements such as window title, labels, buttons, and dynamic text with their translated versions. It ensures the user interface reflects the current language settings.

4.12.3.5 ValidatorInput()

```
void TableBoard::ValidatorInput ()
```

Validates input data in the table.

Sets up validators, event filters, read-only states, and check states for UI elements.

Invoked to ensure input data in the table cells meets specified criteria.

This method initializes validators for numeric inputs, installs event filters for certain UI elements, sets read-only states for error fields, and sets check states for checkboxes.

It also sets specific styles and properties for certain QLineEdit elements related to flow rates, masses, and temperatures.

Note

This function assumes the existence of specific UI elements (e.g., `ui->lbN1`, `ui->cbSet1`, `ui->leSN1`).

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

- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[tableBoard.h](#)
- C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/[tableBoard.cpp](#)

Chapter 5

File Documentation

5.1 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/definitions.h File Reference

Header file containing constants and definitions for the project.

```
#include <cstdint>
```

Include dependency graph for definitions.h: This graph shows which files directly or indirectly include this file:

Variables

- constexpr std::size_t **MAX_PATH_LENGTH** = 260
- constexpr std::size_t **MAX_ARRAY_SIZE** = 20
- constexpr size_t **MAX_ENTRIES** {MAX_ARRAY_SIZE}
- constexpr int **MAIN_WINDOW_WIDTH** = 1450
Width of the main window in pixels.
- constexpr int **MAIN_WINDOW_HEIGHT** = 800
Height of the main window in pixels.
- constexpr size_t **MIN_TEMPERATURE** = 0.0
Minimum allowable temperature in Celsius.
- constexpr size_t **MAX_TEMPERATURE** = 100.0
Maximum allowable temperature in Celsius.
- constexpr double **DEFAULT_DENSITY_BELOW_ZERO** = 999.8395
Default density when temperature is below zero.
- constexpr double **DEFAULT_DENSITY_ABOVE_HUNDRED** = 958.3449
Default density when temperature is above one hundred.
- constexpr double **DEFAULT_VOLUME_CORRECTION_BELOW_ZERO** = 1.00116
Default volume correction factor when temperature is below zero.
- constexpr double **DEFAULT_VOLUME_CORRECTION_ABOVE_HUNDRED** = 1.04451
Default volume correction factor when temperature is above one hundred.
- constexpr const char * **VERSION_BUILD** = "1.3 Windows x86_32"
Version and build information of the application.
- constexpr char **CSV_DELIMITER** = ','
Delimiter used in CSV files.
- constexpr std::size_t **MAX_NUMBER_FLOW_METER_TYPES** = 1000

Maximum number of flow meter types supported.

- `constexpr const char * CSV_FLOW_METER_TYPES = "watermeters.csv"`

Filename of the CSV file containing flow meter types.

- `constexpr int S11 = 7`

MD5 transformation constant S11.

- `constexpr int S12 = 12`

MD5 transformation constant S12.

- `constexpr int S13 = 17`

MD5 transformation constant S13.

- `constexpr int S14 = 22`

MD5 transformation constant S14.

- `constexpr int S21 = 5`

MD5 transformation constant S21.

- `constexpr int S22 = 9`

MD5 transformation constant S22.

- `constexpr int S23 = 14`

MD5 transformation constant S23.

- `constexpr int S24 = 20`

MD5 transformation constant S24.

- `constexpr int S31 = 4`

MD5 transformation constant S31.

- `constexpr int S32 = 11`

MD5 transformation constant S32.

- `constexpr int S33 = 16`

MD5 transformation constant S33.

- `constexpr int S34 = 23`

MD5 transformation constant S34.

- `constexpr int S41 = 6`

MD5 transformation constant S41.

- `constexpr int S42 = 10`

MD5 transformation constant S42.

- `constexpr int S43 = 15`

MD5 transformation constant S43.

- `constexpr int S44 = 21`

MD5 transformation constant S44.

- `constexpr const char * MANUAL_RO = "Manual de utilizare WStreamLab V1.2.pdf"`

5.1.1 Detailed Description

Header file containing constants and definitions for the project.

This file defines constants and provides important definitions used throughout the project.

Author

Constantin

5.2 definitions.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef DEFINITIONS_H_INCLUDED
00012 #define DEFINITIONS_H_INCLUDED
00013
00014 #include <cstdint> // For NULL, size_t, ptrdiff_t, and other standard library facilities related to
    sizes and offsets.
00015
00016 // Maximum path length
00017 constexpr std::size_t MAX_PATH_LENGTH = 260;
00018
00019 // Maximum size for arrays
00020 constexpr std::size_t MAX_ARRAY_SIZE = 20;
00021
00022 // Alias for MAX_ARRAY_SIZE, used for clarity
00023 constexpr size_t MAX_ENTRIES{MAX_ARRAY_SIZE};
00024
00025 // Main window dimensions
00026 constexpr int MAIN_WINDOW_WIDTH = 1450;
00027 constexpr int MAIN_WINDOW_HEIGHT = 800;
00028
00029 // Minimum and maximum temperature
00030 constexpr size_t MIN_TEMPERATURE = 0.0;
00031 constexpr size_t MAX_TEMPERATURE = 100.0;
00032
00033 // Default density values
00034 constexpr double DEFAULT_DENSITY_BELOW_ZERO = 999.8395;
00035 constexpr double DEFAULT_DENSITY_ABOVE_HUNDRED = 958.3449;
00036
00037 // Default volume correction values
00038 constexpr double DEFAULT_VOLUME_CORRECTION_BELOW_ZERO = 1.00116;
00039 constexpr double DEFAULT_VOLUME_CORRECTION_ABOVE_HUNDRED = 1.04451;
00040
00041 // Version information
00042 constexpr const char* VERSION_BUILD = "1.3 Windows x86_32";
00043
00044 // CSV delimiter
00045 constexpr char CSV_DELIMITER = ',';
00046
00047 // Maximum number of flow meter types
00048 constexpr std::size_t MAX_NUMBER_FLOW_METER_TYPES = 1000;
00049
00050 // File name for flow meter types CSV
00051 constexpr const char* CSV_FLOW_METER_TYPES = "watermeters.csv";
00052
00053 // Constants for MD5Transform routine
00054 constexpr int S11 = 7;
00055 constexpr int S12 = 12;
00056 constexpr int S13 = 17;
00057 constexpr int S14 = 22;
00058 constexpr int S21 = 5;
00059 constexpr int S22 = 9;
00060 constexpr int S23 = 14;
00061 constexpr int S24 = 20;
00062 constexpr int S31 = 4;
00063 constexpr int S32 = 11;
00064 constexpr int S33 = 16;
00065 constexpr int S34 = 23;
00066 constexpr int S41 = 6;
00067 constexpr int S42 = 10;
00068 constexpr int S43 = 15;
00069 constexpr int S44 = 21;
00070
00071 // Filename of the Romanian language manual
00072 constexpr const char* MANUAL_RO = "Manual de utilizare WStreamLab V1.2.pdf";
00073
00074 #endif // DEFINITIONS_H_INCLUDED

```

5.3 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/flow-meter-type.h File Reference

Header file defining enums for flow meter types.

```

#include <QDir>
#include <QMessageBox>

```

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include "definitions.h"
```

Include dependency graph for flow-meter-type.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [MeterFlowType](#)
Structure representing a water flow meter type.

Functions

- `std::vector< MeterFlowType > readFlowMeterTypesCSV (const std::string &filename)`
Reads meter flow types from a CSV file and returns them as a vector of [MeterFlowType](#) objects.

Variables

- `std::vector< MeterFlowType > meterFlowTypesDefault`
Vector containing default [MeterFlowType](#) objects representing various water flow meters.
- `MeterFlowType MeterFlowDB [MAX_NUMBER_FLOW_METER_TYPES] = {}`
Database array storing water flow meter types.

5.3.1 Detailed Description

Header file defining enums for flow meter types.

This file defines enums that represent different types of flow meters used in the project.

Author

Constantin

5.3.2 Function Documentation

5.3.2.1 readFlowMeterTypesCSV()

```
std::vector< MeterFlowType > readFlowMeterTypesCSV (
    const std::string & filename)
```

Reads meter flow types from a CSV file and returns them as a vector of [MeterFlowType](#) objects.

This function reads meter flow types from the specified CSV file. Each line in the CSV file represents a [MeterFlowType](#) object with fields separated by CSV_DELIMITER. If the file cannot be opened, it displays a warning message using QMessageBox and returns the default meter flow types.

Parameters

<i>filename</i>	The path to the CSV file containing meter flow types.
-----------------	---

Returns

std::vector<MeterFlowType> A vector containing [MeterFlowType](#) objects read from the CSV file. If the file cannot be read or is corrupted, it returns meterFlowTypesDefault.

See also

[MeterFlowType](#), [meterFlowTypesDefault](#), [QMessageBox](#)

5.3.3 Variable Documentation

5.3.3.1 MeterFlowDB

```
MeterFlowType MeterFlowDB[MAX_NUMBER_FLOW_METER_TYPES] = {}
```

Database array storing water flow meter types.

This array stores information about water flow meter types. Each element represents a [MeterFlowType](#) structure, which includes attributes such as name, nominal diameter, flow rates (nominal, maximum, transition, minimum), and error margins.

This array is initialized to store up to MAX_NUMBER_FLOW_METER_TYPES meter types. Ensure that the array size is sufficient for your application's needs to avoid overflow.

See also

[MeterFlowType](#), [MAX_NUMBER_FLOW_METER_TYPES](#)

5.4 flow-meter-type.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef FLOWMETERTYPE_H
00012 #define FLOWMETERTYPE_H
00013
00014 #include <QDir>           // Qt class for handling directories and their contents.
00015 #include <QMessageBox>    // Qt class for displaying modal dialog boxes with messages.
00016 #include <iostream>       // Standard C++ stream input/output library.
00017 #include <fstream>        // Standard C++ file stream input/output library.
00018 #include <sstream>        // Standard C++ string stream input/output library.
00019 #include <vector>         // Standard C++ container class for dynamic arrays.
00020 #include <string>         // Standard C++ string class.
00021 #include "definitions.h"  // User-defined header file containing project-specific definitions.
00022
00023 struct MeterFlowType
00024 {
00025     std::string nameWaterMeter;
00026     unsigned nominalDiameter;
00027     double nominalFlow;
00028     double maximumFlow;
00029     double transitionFlow;
00030     double minimumFlow;
00031     double nominalError;
00032     double maximumError;
00033
00034     MeterFlowType(const std::string &_nameWaterMeter,
```

```

00051         unsigned _nominalDiameter,
00052         double _nominalFlow,
00053         double _maximumFlow,
00054         double _transitionFlow,
00055         double _minimumFlow,
00056         double _nominalError,
00057         double _maximumError)
00058     : nameWaterMeter(_nameWaterMeter),
00059     nominalDiameter(_nominalDiameter),
00060     nominalFlow(_nominalFlow),
00061     maximumFlow(_maximumFlow),
00062     transitionFlow(_transitionFlow),
00063     minimumFlow(_minimumFlow),
00064     nominalError(_nominalError),
00065     maximumError(_maximumError) {}
00066
00071     MeterFlowType()
00072     : nameWaterMeter(""),
00073     nominalDiameter(0),
00074     nominalFlow(0),
00075     maximumFlow(0),
00076     transitionFlow(0),
00077     minimumFlow(0),
00078     nominalError(0),
00079     maximumError(0) {}
00080 };
00081
00085 std::vector < MeterFlowType > meterFlowTypesDefault =
00086 {
00087     //

```

ErrNom	ErrMax	Type	DN	QN	Qmax	Qt	Qmin	%
			mm	l/h	l/h	l/h	l/h	%
5)		MeterFlowType("Itron Flodis DN 15",	15,	1500,	3000,	22.5,	15,	2,
5)		MeterFlowType("Itron Flodis DN 20",	20,	2500,	5000,	37.5,	25,	2,
5)		MeterFlowType("Itron Flodis DN 25",	25,	3500,	7000,	52.5,	35,	2,
5)		MeterFlowType("Itron Flostar DN 40",	40,	16000,	20000,	160,	80,	2,
5)		MeterFlowType("Itron Flostar DN 50",	50,	25000,	31250,	127,	79,	2,
5)		MeterFlowType("Itron Flostar DN 65",	65,	40000,	50000,	160,	100,	2,
5)		MeterFlowType("Itron Flostar DN 80",	80,	63000,	78000,	252,	157,	2,
5)		MeterFlowType("Itron Flostar DN 100",	100,	100000,	125000,	400,	250,	2,
5)		MeterFlowType("Itron Flostar DN 150",	150,	160000,	200000,	406,	254,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 15 R80",	15,	2500,	3125,	50,	31.25,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 15 R160",	15,	2500,	3125,	25,	15.63,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 20 R80",	20,	4000,	5000,	100,	62.5,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 20 R160",	20,	4000,	5000,	40,	25.0,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 25 R80",	25,	6300,	7875,	126,	78.75,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 25 R160",	25,	6300,	7875,	63,	39.4,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 25 R80",	25,	8000,	10000,	160,	100,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 25 R160",	25,	8000,	10000,	80.64,	50.4,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 32 R80",	32,	10000,	12000,	200,	125,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 32 R160",	32,	10000,	12000,	100,	62.5,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 40 R80",	40,	16000,	20000,	320,	200,	2,
5)		MeterFlowType("FGH Sentinel 420 PC DN 40 R160",	40,	16000,	20000,	160,	100,	2,
5)		MeterFlowType("FGH MeiStream Plus Cl. C DN 40",	40,	30000,	50000,	130,	80,	2,

```

00116     MeterFlowType("FGH MeiStream Plus Cl. C DN 50",    50,    35000,    55000,    130,    70,    2,
00117 5),
00117     MeterFlowType("FGH MeiStream Plus Cl. C DN 65",    65,    40000,    60000,    160,    100,    2,
00118 5),
00118     MeterFlowType("FGH MeiStream Plus Cl. C DN 80",    80,    63000,    120000,    250,    130,    2,
00119 5),
00119     MeterFlowType("FGH MeiStream Plus Cl. C DN 100",    100,    100000,    160000,    400,    200,    2,
00120 5),
00120     MeterFlowType("FGH MeiStream Plus Cl. C DN 150",    150,    250000,    400000,    630,    350,    2,
00121 5),
00121     //
-----
00122     MeterFlowType("FGH iPERL R800 DN 15",              15,    2500,    3125,    5,    3.13,    2,
00123 5),
00123     MeterFlowType("FGH iPERL R800 DN 20",              20,    4000,    5000,    8,    5,    2,
00124 5),
00124     MeterFlowType("FGH iPERL R800 DN 25",              25,    6300,    7875,    12.6,    7.88,    2,
00125 5),
00125     MeterFlowType("FGH iPERL R800 DN 30",              30,    10000,    12500,    20,    12.5,    2,
00126 5),
00126     MeterFlowType("FGH iPERL R800 DN 40",              40,    16000,    20000,    32,    20,    2,
00127 5),
00127     //
-----
00128     MeterFlowType("FGH MNK DN 15 R80",                15,    2500,    3125,    50,    31.25,    2,
00129 5),
00129     MeterFlowType("FGH MNK DN 15 R160",                15,    2500,    3125,    25,    15.63,    2,
00130 5),
00130     MeterFlowType("FGH MNK DN 20 R80",                20,    4000,    5000,    100,    62.5,    2,
00131 5),
00131     MeterFlowType("FGH MNK DN 20 R160",                20,    4000,    5000,    40,    25,    2,
00132 5),
00132     MeterFlowType("FGH MNK DN 25 R80",                25,    6300,    7875,    126,    78.75,    2,
00133 5),
00133     MeterFlowType("FGH MNK DN 25 A",                  25,    6300,    7875,    63,    39.4,    2,
00134 5),
00134     MeterFlowType("FGH MNK DN 25 B",                  25,    8000,    10000,    160,    100,    2,
00135 5),
00135     MeterFlowType("FGH MNK DN 25 C",                  25,    8000,    10000,    80.64,    50.4,    2,
00136 5),
00136     MeterFlowType("FGH MNK DN 32 R80",                32,    10000,    12000,    200,    125,    2,
00137 5),
00137     MeterFlowType("FGH MNK DN 32 R160",                32,    10000,    12000,    100,    62.5,    2,
00138 5),
00138     MeterFlowType("FGH MNK DN 40 R80",                40,    16000,    20000,    320,    200,    2,
00139 5),
00139     MeterFlowType("FGH MNK DN 40 R160",                40,    16000,    20000,    160,    100,    2,
00140 5),
00140     MeterFlowType("FGH MNK DN 50 R80",                50,    25000,    31250,    500,    312.5,    2,
00141 5),
00141     MeterFlowType("FGH MNK DN 50 R255",                50,    25000,    31250,    160,    100,    2,
00142 5),
00142 };
00143
00158 std::vector < MeterFlowType > readFlowMeterTypesCSV(
00159     const std::string &filename)
00160 {
00161     QString currentPath = QDir::currentPath();
00162     Q_UNUSED(currentPath);
00163     std::ifstream file(filename);
00164     std::vector < MeterFlowType > meterFlowTypes;
00165
00166     if (!file.is_open())
00167     {
00168         QMessageBox warningMessage;
00169         warningMessage.addButton(QMessageBox::Ok);
00170         warningMessage.setWindowTitle(QObject::tr("Warning"));
00171         warningMessage.setText(QObject::tr("Flow Meters DB"));
00172         warningMessage.setInformativeText(
00173             QObject::tr("The watermeters.csv with Flow Meters DB cannot be found.));
00174         warningMessage.setWindowFlags(Qt::Dialog | Qt::CustomizeWindowHint |
00175             Qt::WindowTitleHint |
00176             Qt::WindowCloseButtonHint);
00177         warningMessage.exec();
00178         /* Default Water Flow Meters DB */
00179         return meterFlowTypesDefault;
00180     }
00181
00182     bool dbCorrupted = false;
00183
00184     try
00185     {
00186         std::string line;
00187         size_t expectedFieldCount = 0;
00188
00189         if (std::getline(file, line)) {
00190             std::istringstream iss(line);

```

```

00191         std::string field;
00192
00193         // Count the number of fields in the first line to determine the expected count
00194         while (std::getline(iss, field, CSV_DELIMITER)) {
00195             expectedFieldCount++;
00196         }
00197     }
00198
00199     while (std::getline(file, line)) {
00200         std::string lineCopy = line;
00201         size_t actualFieldCount = 0;
00202
00203         // Count the number of fields in the current line
00204         std::istringstream issCopy(lineCopy);
00205         std::string field;
00206         while (std::getline(issCopy, field, CSV_DELIMITER)) {
00207             actualFieldCount++;
00208         }
00209
00210         // Check if the field count matches the expected count
00211         if (actualFieldCount != expectedFieldCount) {
00212             dbCorrupted = true;
00213             continue;
00214         }
00215
00216         // Parse the line into MeterFlowType structure
00217         MeterFlowType meterFlow;
00218         std::istringstream iss(line);
00219         std::string token;
00220
00221         std::getline(iss, meterFlow.nameWaterMeter, CSV_DELIMITER);
00222         std::getline(iss, token, CSV_DELIMITER);
00223         meterFlow.nominalDiameter = std::stod(token);
00224         std::getline(iss, token, CSV_DELIMITER);
00225         meterFlow.maximumFlow = std::stod(token);
00226         std::getline(iss, token, CSV_DELIMITER);
00227         meterFlow.nominalFlow = std::stod(token);
00228         std::getline(iss, token, CSV_DELIMITER);
00229         meterFlow.transitionFlow = std::stod(token);
00230         std::getline(iss, token, CSV_DELIMITER);
00231         meterFlow.minimumFlow = std::stod(token);
00232         std::getline(iss, token, CSV_DELIMITER);
00233         meterFlow.nominalError = std::stod(token);
00234         std::getline(iss, token);
00235         meterFlow.maximumError = std::stod(token);
00236
00237         // Add the parsed MeterFlowType to the vector
00238         meterFlowTypes.push_back(meterFlow);
00239     }
00240 }
00241 catch (const std::exception &e)
00242 {
00243     file.close();
00244     QMessageBox warningMessage;
00245     warningMessage.addButton(QMessageBox::Ok);
00246     warningMessage.setWindowTitle(QObject::tr("Warning"));
00247     warningMessage.setText(QObject::tr("Flow Meters DB"));
00248     warningMessage.setInformativeText(
00249         QObject::tr("The watermeters.csv with Flow Meters DB is corrupted."));
00250     warningMessage.setWindowFlags(Qt::Dialog | Qt::CustomizeWindowHint |
00251                                   Qt::WindowTitleHint |
00252                                   Qt::WindowCloseButtonHint);
00253     warningMessage.exec();
00254     return meterFlowTypesDefault;
00255 }
00256
00257 file.close();
00258
00259 if (dbCorrupted)
00260 {
00261     QMessageBox warningMessage;
00262     warningMessage.addButton(QMessageBox::Ok);
00263     warningMessage.setWindowTitle(QObject::tr("Warning"));
00264     warningMessage.setText(QObject::tr("Flow Meters DB"));
00265     warningMessage.setInformativeText(
00266         QObject::tr("The watermeters.csv with Flow Meters DB is corrupted."));
00267     warningMessage.setWindowFlags(Qt::Dialog | Qt::CustomizeWindowHint |
00268                                   Qt::WindowTitleHint |
00269                                   Qt::WindowCloseButtonHint);
00270     warningMessage.exec();
00271 }
00272 return meterFlowTypes;
00273 }
00274
00275 MeterFlowType MeterFlowDB[MAX_NUMBER_FLOW_METER_TYPES] = {};
00276
00277
00278
00279

```



```
00290 #endif // FLOWMETERTYPE_H
```

5.5 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/helpabout.cpp File Reference

Implementation file for [HelpAbout](#) dialog functionality.

```
#include "helpabout.h"
#include "ui_helpabout.h"
#include "definitions.h"
Include dependency graph for helpabout.cpp:
```

5.5.1 Detailed Description

Implementation file for [HelpAbout](#) dialog functionality.

This file contains the implementation of the [HelpAbout](#) class, which provides functionality for displaying help and about information in a dialog window.

Author

Constantin

Date

Insert date

5.6 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/helpabout.h File Reference

Header file for [HelpAbout](#) dialog.

```
#include <QDialog>
Include dependency graph for helpabout.h: This graph shows which files directly or indirectly include this file:
```

Classes

- class [HelpAbout](#)
[HelpAbout](#) class represents a dialog for displaying Help/About information.

5.6.1 Detailed Description

Header file for [HelpAbout](#) dialog.

This file defines the [HelpAbout](#) class, which represents a dialog for displaying help and information about the application.

Author

Constantin

5.7 helpabout.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef HELPABOUT_H
00012 #define HELPABOUT_H
00013
00014 #include <QDialog> // Qt class for creating modal or modeless dialogs.
00015
00016 namespace Ui {
00017     class HelpAbout;
00018 }
00019
00027 class HelpAbout : public QDialog
00028 {
00029     Q_OBJECT
00030
00031 public:
00036     explicit HelpAbout(QWidget *parent = nullptr);
00037
00041     ~HelpAbout();
00042
00049     void Translate();
00050
00051 private:
00052     Ui::HelpAbout *ui;
00053
00054 private slots:
00061     void onCloseClicked();
00062 };
00063
00064
00065 #endif // HELPABOUT_H
```

5.8 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/interface.cpp

File Reference

Implementation file for [Interface](#) dialog functionality.

```
#include <mutex>
#include <thread>
#include <unistd.h>
#include <QMessageBox>
#include <QSettings>
#include <QStringList>
#include <QtSerialBus/QModbusClient>
#include <QtSerialBus/QModbusRtuSerialServer>
#include <QtSerialPort/QSerialPort>
#include "interface.h"
#include "mainwindow.h"
#include "ui_interface.h"
#include "ui_mainwindow.h"
```

Include dependency graph for interface.cpp:

Variables

- std::mutex **modbusLock**

5.8.1 Detailed Description

Implementation file for [Interface](#) dialog functionality.

Implementation of the [Interface](#) class.

This file contains the implementation of the [Interface](#) class, which provides functionality for configuring and interacting with a user interface dialog.

Author

Constantin

Date

Insert date

5.9 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/interface.h File Reference

Header file for the [Interface](#) class.

```
#include <QDir>
#include <QMessageBox>
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include "ledindicator.h"
```

Include dependency graph for interface.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Interface](#)

The [Interface](#) class represents a dialog window for configuring and managing settings.

5.9.1 Detailed Description

Header file for the [Interface](#) class.

This file defines the [Interface](#) class, which represents a dialog for configuring and managing interface settings.

Author

Constantin

5.10 interface.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef INTERFACE_H
00012 #define INTERFACE_H
00013
00014 #include <QDir>           // Qt class for handling directories and their contents.
00015 #include <QMessageBox>    // Qt class for displaying modal dialog boxes with messages.
00016 #include <iostream>        // Standard C++ stream input/output library.
00017 #include <fstream>         // Standard C++ file stream input/output library.
00018 #include <sstream>         // Standard C++ string stream input/output library.
00019 #include <vector>          // Standard C++ container class for dynamic arrays.
00020 #include <string>          // Standard C++ string class.
00021 #include "ledindicator.h" // Class LED indicator
00022
00023 namespace Ui
00024 {
00025     class Interface;
00026 }
00027
00036 class Interface : public QDialog
00037 {
00038     Q_OBJECT
00039
00040 public:
00045     explicit Interface(QWidget *parent = nullptr);
00046
00050     ~Interface();
00051
00058     void Translate();
00059
00065     bool checkModbusAddress(qint16 address);
00066
00067 private:
00068     Ui::Interface *ui;
00069     QVector<QString> entries;
00070     bool isOpenModbusPort {false};
00071     LedIndicator *ledStateTable[10];
00079     void DisconnectSerialPort();
00080
00081 private slots:
00085     void onCloseClicked();
00086
00090     void onTestConfigurationClicked();
00091
00095     void onSaveConfigurationClicked();
00096
00100     void onRefreshSerialPortClicked();
00101
00105     void onSelectSerialChanged();
00106
00110     void onBaudRateChanged();
00111
00115     void onSelectDataBitsChanged();
00116
00120     void onSelectParityChanged();
00121
00125     void onSelectStopBitsChanged();
00126
00130     void onReadModbusReady();
00131
00136     void showEvent(QShowEvent *event) override;
00137 };
00138 #endif // INTERFACE_H

```

5.11 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/ledindicator.cpp File Reference

Implementation file for the [LedIndicator](#) class.

```

#include <QPainter>
#include "ledindicator.h"

```

Include dependency graph for ledindicator.cpp:

5.11.1 Detailed Description

Implementation file for the [LedIndicator](#) class.

This file contains the implementation of the [LedIndicator](#) class, which provides functionality for displaying LED indicators in a graphical user interface.

Author

Constantin

Date

Insert date

5.12 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/ledindicator.h File Reference

Header file for the [LedIndicator](#) class.

```
#include <QDialog>
```

Include dependency graph for ledindicator.h: This graph shows which files directly or indirectly include this file:

Classes

- class [LedIndicator](#)

The [LedIndicator](#) class represents a custom LED indicator widget.

5.12.1 Detailed Description

Header file for the [LedIndicator](#) class.

This file defines the [LedIndicator](#) class, which represents an LED indicator widget for displaying states or statuses.

Author

Constantin

5.13 ledindicator.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef LEDINDICATOR_H
00012 #define LEDINDICATOR_H
00013
00014 #include <QDialog> // Qt class for creating modal or modeless dialogs.
00015
00023 class LedIndicator : public QWidget
00024 {
00025     Q_OBJECT
00026
00027 public:
00032     explicit LedIndicator(QWidget *parent = nullptr);
00033
00038     void setState(bool state);
00039
00044     void toggle();
00045
00050     void setOnColor(QColor onColor);
00051
00056     void setOffColor(QColor offColor);
00057
00062     void setOnPattern(Qt::BrushStyle onPattern);
00063
00068     void setOffPattern(Qt::BrushStyle offPattern);
00069
00074     void setLedSize(int size);
00075
00076 public slots:
00081     void switchLedIndicator();
00082
00083 protected:
00088     void paintEvent(QPaintEvent *event) override;
00089
00090 private:
00091     bool lit;
00092     QColor ledOnColor;
00093     QColor ledOffColor;
00094     QColor ledNeutral;
00095     Qt::BrushStyle ledOnPattern;
00096     Qt::BrushStyle ledOffPattern;
00097     Qt::BrushStyle ledNeutralPattern;
00098     int ledSize;
00099 };
00100
00101 #endif // LEDINDICATOR_H

```

5.14 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/license.cpp

File Reference

Implementation of the [License](#) dialog functionality.

```

#include "license.h"
#include "mainwindow.h"
#include "ui_license.h"

```

Include dependency graph for license.cpp:

5.14.1 Detailed Description

Implementation of the [License](#) dialog functionality.

This file contains the implementation of methods for the [License](#) dialog, which displays configuration details retrieved from the main window.

Author

Constantin

5.15 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/license.h File Reference

Declaration of the [License](#) class.

```
#include <QDialog>
```

Include dependency graph for license.h: This graph shows which files directly or indirectly include this file:

Classes

- class [License](#)

The [License](#) class represents a dialog for displaying license information.

5.15.1 Detailed Description

Declaration of the [License](#) class.

This file contains the declaration of the [License](#) class, which provides functionalities related to managing license information.

Author

Constantin

Date

[date]

5.16 license.h

[Go to the documentation of this file.](#)

```
00001
00012 #ifndef LICENSE_H
00013 #define LICENSE_H
00014
00015 #include <QDialog>
00016
00017 namespace Ui {
00018     class Licence;
00019 }
00020
00027 class License : public QDialog
00028 {
00029     Q_OBJECT
00030
00031 public:
00036     explicit License(QWidget *parent = nullptr);
00037
00041     ~License();
00042
00049     void Translate();
00050
00051     Ui::Licence *ui;
00052
00053 private slots:
00061     void showEvent(QShowEvent *event) override;
00062
00068     void onCloseClicked();
00069 };
00070
00071 #endif // LICENSE_H
```

5.17 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/main.cpp File Reference

Main entry point of the application.

```
#include <QApplication>
#include <QMessageBox>
#include <QSharedMemory>
#include <QString>
#include <QTimer>
#include <QPainter>
#include <QEventLoop>
#include <QDir>
#include <QThread>
#include <windows.h>
#include <winnt.h>
#include <fstream>
#include "mainwindow.h"
Include dependency graph for main.cpp:
```

Classes

- class [PixellImageWidget](#)

The [PixellImageWidget](#) class represents a custom widget that displays a pixelated image and handles application-specific functionalities.

Functions

- bool [loadTranslations](#) ()
Lloads the application translations from a specified path.
- bool [checkAndHandleMultipleInstances](#) (QSharedMemory *shared)
Checks and handles multiple instances of the application using shared memory.
- int [main](#) (int argc, char *argv[])
The main entry point of the application.

Variables

- QTranslator * [appTranslator](#) = nullptr
Global pointer to manage application translations.

5.17.1 Detailed Description

Main entry point of the application.

This file contains the main function, which serves as the entry point for the application. It initializes necessary components and starts the main event loop.

Author

Constantin

Date

Insert date

5.17.2 Notes

```
astyle *.cpp,*.h --style=java --indent=spaces=4 --break-blocks --pad-oper --pad-comma --pad-paren --align-pointer=name --add-braces --mode=c --recursive
```

5.17.3 Function Documentation

5.17.3.1 checkAndHandleMultipleInstances()

```
bool checkAndHandleMultipleInstances (
    QSharedMemory * shared)
```

Checks and handles multiple instances of the application using shared memory.

This function attempts to create a shared memory segment with a given key. If the segment cannot be created, it indicates that another instance of the application is already running and displays a warning message.

Parameters

<i>shared</i>	Pointer to a QSharedMemory instance for managing shared memory.
---------------	---

Returns

True if another instance is already running, false if this is the first instance.

5.17.3.2 loadTranslations()

```
bool loadTranslations ()
```

Loads the application translations from a specified path.

This function initializes a global QTranslator object (`appTranslator`) with translations loaded from a specified .qm file located in the application's "translations" directory.

Returns

True if the translation loaded successfully, false otherwise.

5.17.3.3 main()

```
int main (
    int argc,
    char * argv[])
```

The main entry point of the application.

Parameters

<i>argc</i>	Number of command-line arguments.
<i>argv</i>	Array of command-line arguments.

Returns

Application exit status.

5.17.4 Variable Documentation**5.17.4.1 appTranslator**

```
QTranslator* appTranslator = nullptr
```

Global pointer to manage application translations.

< Implements input/output operations on memory-based streams.

This pointer is used globally to manage translations for the application. It is initially set to nullptr and later assigned an instance of QTranslator when loading translations.

5.18 C:/Users/Constantin/Desktop/HERE_WFlowLab/↵ Meter/mainwindow.cpp File Reference

Implementation file for the [MainWindow](#) class.

```
#include <filesystem>
#include <fstream>
#include <iomanip>
#include <map>
#include <sstream>
#include <QDesktopServices>
#include <QDir>
#include <QLibrary>
#include <QLineEdit>
#include <QMessageBox>
#include <QSettings>
#include <QValidator>
#include <windows.h>
#include "definitions.h"
#include "waterdensity.h"
#include "flow-meter-type.h"
#include "mainwindow.h"
#include "md5.h"
#include "ui_mainwindow.h"
#include <QFile>
#include <QUrl>
Include dependency graph for mainwindow.cpp:
```

Classes

- struct [RS485SettingInfo](#)
Structure to hold information about RS485 settings.

Functions

- `std::wstring ExePath ()`

Variables

- `QTranslator * appTranslator`
< Implements input/output operations on memory-based streams.
- `MainWindow * pMainWindow`
Pointer to the main window instance.

5.18.1 Detailed Description

Implementation file for the [MainWindow](#) class.

This file contains the implementations of member functions and slots for the [MainWindow](#) class. It handles the main window of the application, including initialization, event handling, and slot implementations.

Author

Constantin

Date

Insert date

5.18.2 Variable Documentation

5.18.2.1 [appTranslator](#)

```
QTranslator* appTranslator [extern]
```

< Implements input/output operations on memory-based streams.

< Provides facilities to manipulate and query file systems and their components. < Input/output stream class to operate on files. Manipulators for formatting output. < Associative containers that store elements in a mapped fashion. < Access to the desktop services such as opening a URL. < Provides access to directory structures and their contents. < Platform-independent library loading and function resolution. < Single-line text editor widget with input validation and styling. < Modal dialog for informing the user or for asking the user a question and receiving an answer. < Persistent platform-independent application settings. < Base class for all validators that can be easily attached to input widgets. < Main Windows SDK header providing core Windows APIs. < Custom application-specific definitions. < Header for water density calculations. < Header defining flow meter types. < Header for the main application window. < Header for MD5 hashing functionality. < User interface header generated from Qt Designer. < Provides functions to read from and write to files. < Provides access to directory structures and their contents. Represents a URL. < Access to the desktop services such as opening a URL. < Input/output stream class to operate on files. Manipulators for formatting output. Implements input/output operations on memory-based streams.

< Implements input/output operations on memory-based streams.

This pointer is used globally to manage translations for the application. It is initially set to nullptr and later assigned an instance of QTranslator when loading translations.

5.18.2.2 pMainWindow

`MainWindow* pMainWindow`

Pointer to the main window instance.

`\extern MainWindow *pMainWindow`

This global variable holds a pointer to the main window instance, allowing access to its properties and methods from various parts of the application.

5.19 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/mainwindow.h File Reference

Header file for the `MainWindow` class.

```
#include <QMainWindow>
#include <QActionGroup>
#include <QApplication>
#include <QtSerialBus/QModbusRtuSerialClient>
#include <QSerialPort>
#include <QSerialPortInfo>
#include <QTranslator>
#include "tableBoard.h"
#include "license.h"
#include "helpabout.h"
#include "interface.h"
```

Include dependency graph for mainwindow.h: This graph shows which files directly or indirectly include this file:

Classes

- struct `SelectedInfo`
The `SelectedInfo` struct holds selected information related to a device or configuration.
- class `MainWindow`

Enumerations

- enum `SELECTED_LANGUAGE` { `ROMANIAN` , `ENGLISH` , `DEFAULT` = `ENGLISH` }

5.19.1 Detailed Description

Header file for the `MainWindow` class.

This file defines the `MainWindow` class, which represents the main window of the application.

Author

Constantin

5.20 mainwindow.h

[Go to the documentation of this file.](#)

```

00001
00011 #ifndef MAINWINDOW_H
00012 #define MAINWINDOW_H
00013
00014 #include <QMainWindow> // Qt class for main application window.
00015 #include <QActionGroup> // Qt class for grouping actions together.
00016 #include <QApplication> // Qt class for managing the application's control flow.
00017 #include <QtSerialBus/QModbusRtuSerialClient> // Qt class for Modbus RTU serial client communication.
00018 #include <QSerialPort> // Qt class for accessing serial port hardware.
00019 #include <QSerialPortInfo> // Qt class for retrieving information about available
    serial ports.
00020 #include <QTranslator> // Qt class for providing translations in the application.
00021 #include "tableBoard.h" // Custom header for TableBoard class.
00022 #include "license.h" // Custom header for License class.
00023 #include "helpabout.h" // Custom header for HelpAbout class.
00024 #include "interface.h" // Custom header for Interface class.
00025
00026 enum SELECTED_LANGUAGE
00027 {
00028     ROMANIAN,
00029     ENGLISH,
00030     DEFAULT = ENGLISH
00031 };
00032
00033 QT_BEGIN_NAMESPACE
00034 namespace Ui
00035 {
00036     class MainWindow;
00037 }
00038 QT_END_NAMESPACE
00039
00046 struct SelectedInfo
00047 {
00051     SelectedInfo():
00052         density_20{0.0f},
00053         entriesNumber{0},
00054         nominalDiameter{0},
00055         nominalFlow{0.0f},
00056         maximumFlow{0.0f},
00057         trasitionFlow{0.0f},
00058         minimumFlow{0.0f},
00059         nominalError{0.0f},
00060         maximumError{0.0f},
00061         ambientTemperature{0},
00062         athmosphericPressure{0},
00063         relativeAirHumidity{0},
00064         rbGravimetric_new{true},
00065         rbVolumetric{false},
00066         rbManual{true},
00067         rbInterface{false},
00068         rbTerminal{false},
00069         serialPort{false},
00070         selectedLanguage{ROMANIAN},
00071         modbusDevice{nullptr}
00072     {
00073     }
00074
00075     float density_20;
00076     std::string pathResults;
00077     size_t entriesNumber;
00078     std::string certificate;
00079     std::string nameWaterMeter;
00080     unsigned nominalDiameter;
00081     double nominalFlow;
00082     double maximumFlow;
00083     double trasitionFlow;
00084     double minimumFlow;
00085     double nominalError;
00086     double maximumError;
00087     std::string ambientTemperature;
00088     std::string athmosphericPressure;
00089     std::string relativeAirHumidity;
00090     bool rbGravimetric_new;
00091     bool rbVolumetric;
00092     bool rbManual;
00093     bool rbInterface;
00094     bool rbTerminal;
00095     bool serialPort;
00096     SELECTED_LANGUAGE selectedLanguage;
00097     QModbusClient *modbusDevice;
00105 };
00106

```

```

00107 class MainWindow : public QMainWindow
00108 {
00109     Q_OBJECT
00110
00111 public:
00116     explicit MainWindow(QWidget *parent = nullptr);
00117
00121     ~MainWindow();
00122
00123     SelectedInfo selectedInfo;
00125     Ui::MainWindow *ui {nullptr};
00126     TableBoard *inputData{nullptr};
00127     License *licenseDialog{nullptr};
00128     HelpAbout *helpAbout;
00129     Interface *interfaceDialog{nullptr};
00130     QActionGroup *alignmentGroup;
00131     LedIndicator *LED;
00136     void Translate();
00137
00144     void setLabelValue(QLabel* label, double value, int precision);
00145
00149     void updateSelectedInfo();
00150
00154     void SelectMeterComboBox();
00155
00159     void ReadConfiguration();
00160
00164     void SetDefaultConfiguration();
00165
00170     void CenterToScreen(QWidget *widget);
00171
00172     unsigned MAX_NR_WATER_METERS {20};
00173     unsigned NUMBER_ENTRIES_METER_FLOW_DB {0};
00174     std::map<std::string, std::string> optionsConfiguration;
00176     typedef const wchar_t * (*EnumerateSerialPorts)();
00177     EnumerateSerialPorts serialPorts{nullptr};
00179 protected:
00184     void mousePressEvent(QMouseEvent *event) override;
00185
00192     bool eventFilter(QObject *obj, QEvent *event) override
00193     {
00194         if (event->type() == QEvent::MouseButtonPress)
00195         {
00196             // Check if the event occurred on this window
00197             QMouseEvent *mouseEvent = static_cast<QMouseEvent *>(event);
00198             if (rect().contains(mouseEvent->pos()))
00199             {
00200                 activateWindow();
00201             }
00202         }
00203         return QMainWindow::eventFilter(obj, event);
00204     }
00205
00206 private slots:
00211     void onMeterTypeChanged(int index);
00212
00217     void onNumberOfWaterMetersChanged(int index);
00218
00222     void onNewSessionClicked();
00223
00227     void onExitApplication();
00228
00232     void onRbGavritmetricClicked();
00233
00237     void onRbVolumeClicked();
00238
00242     void onRbManualClicked();
00243
00247     void onRbInterfaceClicked();
00248
00252     void onAmbientTemperatureTextChanged();
00253
00257     void onRelativeAirHumidityTextChanged();
00258
00262     void onAthmosphericPressureTextChanged();
00263
00267     void onSetRomanian();
00268
00272     void onSetEnglish();
00273
00277     void onGeneralDescription();
00278
00282     void onShowLicense();
00283
00287     void onWaterDensityPage();
00288
00292     void onHelpAbout();

```

```

00293
00297     void onPortSettings();
00298
00303     void closeEvent(QCloseEvent *event) override;
00304
00305
00306 signals:
00310     void numberOfWaterMetersChangedSignal();
00311
00315     void meterTypeChangedSignal();
00316
00320     void measurementTypeChangedSignal();
00321
00322 };
00323 #endif // MAINWINDOW_H

```

5.21 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/md5.cpp File Reference

[MD5](#) class implementation.

```

#include "md5.h"
#include <stdio.h>
#include <cstdio>
#include "definitions.h"

```

Include dependency graph for md5.cpp:

Functions

- `std::ostream & operator<< (std::ostream &out, const MD5 &md5)`
Overloaded operator to output the hexadecimal representation of the [MD5](#) digest to an output stream.
- `std::string md5 (const std::string &str)`
Computes the [MD5](#) hash of a given string and returns its hexadecimal representation.

5.21.1 Detailed Description

[MD5](#) class implementation.

Author

Frank Thilo (thilo @unix-ag.org)

Date

Created: 1991

Last modified: Insert modification date

This file contains the implementation of the [MD5](#) class, converted from the reference implementation of RFC 1321 by RSA Data Security, Inc.

The original implementation was based on [md5.h](#) and `md5.c`.

See also

<http://www.bzflag.org>

5.21.2 Function Documentation

5.21.2.1 md5()

```
std::string md5 (
    const std::string & str)
```

Computes the [MD5](#) hash of a given string and returns its hexadecimal representation.

This function computes the [MD5](#) hash of the input string using the [MD5](#) class. It then converts the computed digest into a hexadecimal string representation using the `hexdigest()` method and returns it.

Parameters

<i>str</i>	The input string for which the MD5 hash will be computed.
------------	---

Returns

Hexadecimal string representation of the [MD5](#) hash of the input string.

5.21.2.2 operator<<()

```
std::ostream & operator<< (
    std::ostream & out,
    const MD5 & md5)
```

Overloaded operator to output the hexadecimal representation of the [MD5](#) digest to an output stream.

This operator allows the [MD5](#) digest to be output directly to an ostream. It calls the `hexdigest()` method of the [MD5](#) object to obtain the hexadecimal representation and outputs it to the specified ostream.

Parameters

<i>out</i>	The output stream to which the MD5 digest will be written.
<i>md5</i>	The MD5 object whose digest will be written to the output stream.

Returns

Reference to the output stream after writing the [MD5](#) digest.

5.22 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/md5.h File Reference

Header file for the [MD5](#) class.

```
#include <cstring>
#include <iostream>
```

Include dependency graph for `md5.h`: This graph shows which files directly or indirectly include this file:

Classes

- class [MD5](#)

The [MD5](#) class computes [MD5](#) hashes of strings or byte arrays.

Functions

- `std::string md5 (const std::string &str)`

Computes the [MD5](#) hash of a given string and returns its hexadecimal representation.

5.22.1 Detailed Description

Header file for the [MD5](#) class.

This file defines the [MD5](#) class, which provides functionality for calculating [MD5](#) hashes of strings or byte arrays.

Author

Frank Thilo (thilo@unix-ag.org)

Date

Converted to C++ class on unknown date (based on [md5.h](#) and [md5.c](#))

Copyright

Copyright (C) 1991-2, RSA Data Security, Inc. All rights reserved.

This class is based on the reference implementation of the [MD5](#) Message-Digest Algorithm, as specified in RFC 1321.

[License](#) to copy and use this software is granted provided that it is identified as the "RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing this software or this function.

[License](#) is also granted to make and use derivative works provided that such works are identified as "derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm" in all material mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either the merchantability of this software or the suitability of this software for any particular purpose. It is provided "as is" without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this documentation and/or software.

5.22.2 Function Documentation

5.22.2.1 md5()

```
std::string md5 (  
    const std::string & str)
```

Computes the [MD5](#) hash of a given string and returns its hexadecimal representation.

This function computes the [MD5](#) hash of the input string using the [MD5](#) class. It then converts the computed digest into a hexadecimal string representation using the `hexdigest()` method and returns it.

Parameters

<code>str</code>	The input string for which the MD5 hash will be computed.
------------------	---

Returns

Hexadecimal string representation of the [MD5](#) hash of the input string.

5.23 md5.h

[Go to the documentation of this file.](#)

```

00001
00036 #ifndef BZF_MD5_H
00037 #define BZF_MD5_H
00038
00039 #include <cstring>      // C-style string manipulation functions
00040 #include <iostream>    // Standard input/output stream objects
00041
00055 class MD5
00056 {
00057     public:
00058         typedef unsigned int size_type; // must be 32bit
00059
00063         MD5();
00064
00069         explicit MD5(const std::string &text);
00070
00076         void update(const unsigned char *buf, size_type length);
00077
00083         void update(const char *buf, size_type length);
00084
00089         MD5 &finalize();
00090
00095         std::string hexdigest() const;
00096
00103         friend std::ostream &operator<< (std::ostream &os, MD5 md5);
00104
00105     private:
00109         void init();
00110
00111         typedef unsigned char uint1; // 8bit
00112         typedef unsigned int uint4;  // 32bit
00113         enum { blocksize = 64 }; // VC6 won't eat a const static int here
00114
00119         void transform(const uint1 block[blocksize]);
00120
00127         static void decode(uint4 output[], const uint1 input[], size_type len);
00128
00135         static void encode(uint1 output[], const uint4 input[], size_type len);
00136
00137         // low level logic operations
00138         static inline uint4 F(uint4 x, uint4 y, uint4 z);
00139         static inline uint4 G(uint4 x, uint4 y, uint4 z);
00140         static inline uint4 H(uint4 x, uint4 y, uint4 z);
00141         static inline uint4 I(uint4 x, uint4 y, uint4 z);
00142         static inline uint4 rotate_left(uint4 x, int n);
00143         static inline void FF(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x,
00144                               uint4 s, uint4 ac);
00145         static inline void GG(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x,
00146                               uint4 s, uint4 ac);
00147         static inline void HH(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x,
00148                               uint4 s, uint4 ac);
00149         static inline void II(uint4 &a, uint4 b, uint4 c, uint4 d, uint4 x,
00150                               uint4 s, uint4 ac);
00151
00152         bool finalized;
00153         uint1 buffer[blocksize];
00154         uint4 count[2];
00155         uint4 state[4];
00156         uint1 digest[16];
00157 };
00158
00159 std::string md5(const std::string &str);
00160
00161 #endif

```

5.24 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/report.cpp File Reference

Implementation file for the [ReportMeasurements](#) class.

```
#include <algorithm>
#include <chrono>
#include <ctime>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <string>
#include <thread>
#include <mutex>
#include <QCheckBox>
#include <QCoreApplication>
#include <QDateTime>
#include <QDialog>
#include <QDesktopServices>
#include <QDoubleValidator>
#include <QFileDialog>
#include <QKeyEvent>
#include <QLabel>
#include <QLineEdit>
#include <QMainWindow>
#include <QMessageBox>
#include <QPageSize>
#include <QPainter>
#include <QPrintDialog>
#include <QPrinter>
#include <QSettings>
#include <QString>
#include <QTimer>
#include <QValidator>
#include "mainwindow.h"
#include "report.h"
#include "ui_mainwindow.h"
#include "ui_report.h"
```

Include dependency graph for report.cpp:

Functions

- `std::string convertNumberToWords (int num, bool addSuffix=false)`
Converts an integer number into its Romanian words representation.

Variables

- `MainWindow * pMainWindow`
Pointer to the main window instance.
- `std::mutex printReportPdfThreadMutex`
Mutex for thread-safe PDF printing.

5.24.1 Detailed Description

Implementation file for the [ReportMeasurements](#) class.

This file contains the implementation of methods for the [ReportMeasurements](#) class, which is responsible for generating and handling reports related to measurements.

Author

Constantin

Date

Insert creation date

5.24.2 Function Documentation

5.24.2.1 `convertNumberToWords()`

```
std::string convertNumberToWords (  
    int num,  
    bool addSuffix = false)
```

Converts an integer number into its Romanian words representation.

This function converts a given integer number into its equivalent words in Romanian. It handles numbers from -999,999 to 999,999.

Parameters

<i>num</i>	The integer number to convert.
<i>addSuffix</i>	Flag indicating whether to add suffixes like "mii", "milion", etc. Default is false.

Returns

A string containing the Romanian words representation of the number.

< 0

< 1

< 2

< 3

< 4

< 5

< 6

< 7

< 8

< 9

< 0

< 11

< 12

< 13

< 14

< 15

< 16

< 17

< 18

< 19

< 0

< 1

< 20

< 30

< 40

< 50

< 60

< 70

< 80

< 90

5.24.3 Variable Documentation

5.24.3.1 pMainWindow

```
MainWindow* pMainWindow [extern]
```

Pointer to the main window instance.

```
\extern MainWindow *pMainWindow
```

This global variable holds a pointer to the main window instance, allowing access to its properties and methods from various parts of the application.

5.24.3.2 printReportPdfThreadMutex

```
std::mutex printReportPdfThreadMutex
```

Mutex for thread-safe PDF printing.

```
\extern std::mutex printReportPdfThreadMutex
```

This global mutex ensures thread safety when printing PDF documents from multiple threads. It protects critical sections where file paths are manipulated and directories are created.

5.25 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/report.h File Reference

Header file for the [ReportMeasurements](#) class.

```
#include <QDialog>
#include <QCheckBox>
#include <QLineEdit>
#include <QString>
#include <QTimer>
#include <vector>
#include "definitions.h"
```

Include dependency graph for report.h: This graph shows which files directly or indirectly include this file:

Classes

- class [ReportMeasurements](#)

The [ReportMeasurements](#) class provides a dialog for reporting measurements.

5.25.1 Detailed Description

Header file for the [ReportMeasurements](#) class.

This file defines the [ReportMeasurements](#) class, which provides functionality for generating and managing reports of measurements.

Author

Constantin

Date

Insert date

5.26 report.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef REPORT_H
00013 #define REPORT_H
00014
00015 // Qt headers for various UI components and utilities
00016 #include <QDialog>           // Modal or modeless dialog window
00017 #include <QCheckBox>         // Checkbox UI element
00018 #include <QLineEdit>         // Single-line text input widget
00019 #include <QString>           // Qt's string class with Unicode support
00020 #include <QTimer>            // Timer for delayed or periodic execution
00021 #include <vector>            // Standard C++ vector container
00022
00023 // Project-specific definitions and constants
00024 #include "definitions.h"
00025
00026
00027 namespace Ui
00028 {
00029     class report;
00030 }
00031
00032 class ReportMeasurements : public QDialog
00033 {
00034     Q_OBJECT
00035
00036 public:
00037     explicit ReportMeasurements(QWidget *parent,
00038                                 const std::vector<QCheckBox *> &vectorCheckNumber,
00039                                 const std::vector<QLineEdit *> &vectorSerialNumber,
00040                                 const QString resultAllTests[MAX_ARRAY_SIZE]);
00041
00042     ~ReportMeasurements();
00043
00044     void Translate();
00045
00046     static void printPdfThread(QString report);
00047
00048 signals:
00049     void pdfGenerationCompleted();
00050
00051 private slots:
00052     void onPrintClicked();
00053
00054     void onCloseClicked();
00055
00056     void enableGenerareBvButton();
00057
00058 private:
00059     Ui::report *ui;
00060     std::vector<QCheckBox *> vectorCheckNumberCopy;
00061     std::vector<QLineEdit *> vectorSerialNumberCopy;
00062     QString resultAllTestsCopy[MAX_ARRAY_SIZE];
00063     QTimer *QTimerGenerareBv;
00064     QWidget *parentWidget;
00065 };
00066
00067 #endif // REPORT_H

```

5.27 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/table↵ Board.cpp File Reference

Implementation file for the [TableBoard](#) class.

```

#include <algorithm>
#include <fstream>
#include <iomanip>
#include <iostream>
#include <sstream>
#include <thread>
#include <mutex>

```

```
#include <QDesktopServices>
#include <QDoubleValidator>
#include <QFileDialog>
#include <QMainWindow>
#include <QMessageBox>
#include <QPainter>
#include <QPrinter>
#include <QString>
#include <QTimer>
#include <QValidator>
#include <QtPrintSupport/QPrinter>
#include "definitions.h"
#include "waterdensity.h"
#include "mainwindow.h"
#include "tableBoard.h"
#include "ui_mainwindow.h"
#include "ui_tableBoard.h"
```

Include dependency graph for tableBoard.cpp:

Functions

- `std::string precision_4` (double number)
Converts a double number to a string with 4 decimal precision.
- `bool XOR` (bool a, bool b)
Performs exclusive OR (XOR) operation between two boolean values.

Variables

- `MainWindow * pMainWindow`
Pointer to the main window instance.
- `std::mutex printTablePdfThreadMutex`
- `QString resultAllTests` [20]

5.27.1 Detailed Description

Implementation file for the [TableBoard](#) class.

This file contains the implementation of methods for the [TableBoard](#) class, which is responsible for managing and displaying tabular data related to water meters.

Author

Constantin

Date

Insert creation date

5.27.2 Function Documentation

5.27.2.1 `precision_4()`

```
std::string precision_4 (
    double number)
```

Converts a double number to a string with 4 decimal precision.

This function converts a given double number into a string representation with exactly 4 decimal places.

Parameters

<i>number</i>	The double number to convert.
---------------	-------------------------------

Returns

A string representation of the number with 4 decimal precision.

< Integer part of the number.

< Decimal part of the number.

< Return the formatted string.

< Return the formatted string with leading zero.

5.27.2.2 XOR()

```
bool XOR (
    bool a,
    bool b)
```

Performs exclusive OR (XOR) operation between two boolean values.

This function computes the result of the XOR operation between two boolean values. XOR returns true if one and only one of the boolean operands is true; otherwise, it returns false.

Parameters

<i>a</i>	First boolean operand.
<i>b</i>	Second boolean operand.

Returns

The result of the XOR operation between *a* and *b*.

< Return true if *a* and *b* are different; otherwise, false.

5.27.3 Variable Documentation**5.27.3.1 pMainWindow**

```
MainWindow* pMainWindow [extern]
```

Pointer to the main window instance.

```
\extern MainWindow *pMainWindow
```

This global variable holds a pointer to the main window instance, allowing access to its properties and methods from various parts of the application.

5.28 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/tableBoard.h File Reference

Header file for the [TableBoard](#) class.

```
#include <QDialog>
#include <QLineEdit>
#include <QLabel>
#include <QCheckBox>
#include <QKeyEvent>
#include <QTimer>
#include <sstream>
#include "report.h"
```

Include dependency graph for tableBoard.h: This graph shows which files directly or indirectly include this file:

Classes

- class [TableBoard](#)

Dialog window for managing and displaying water meter test data in a table format.

Functions

- `template<typename T >`
`std::string to_string_with_precision (const T a_value, const int n=6)`
Converts a numeric value to a string with a specified precision.

5.28.1 Detailed Description

Header file for the [TableBoard](#) class.

This file defines the [TableBoard](#) class, which represents a dialog for managing and validating input data related to a table board.

Author

Constantin

Date

Insert date

5.28.2 Function Documentation

5.28.2.1 to_string_with_precision()

```
template<typename T >
std::string to_string_with_precision (
    const T a_value,
    const int n = 6)
```

Converts a numeric value to a string with a specified precision.

Template Parameters

<i>T</i>	Type of the numeric value (e.g., float, double, int).
----------	---

Parameters

<i>a_value</i>	The numeric value to convert to a string.
<i>n</i>	Precision (number of digits after the decimal point).

Returns

std::string The string representation of the numeric value with specified precision.

5.29 tableBoard.h

[Go to the documentation of this file.](#)

```

00001
00012 #ifndef TABLEBOARD_H
00013 #define TABLEBOARD_H
00014
00015 // Qt headers for various UI components and utilities
00016 #include <QDialog> // Modal or modeless dialog window
00017 #include <QLineEdit> // Single-line text input widget
00018 #include <QLabel> // Text or image display widget
00019 #include <QCheckBox> // Checkbox UI element
00020 #include <QKeyEvent> // Keyboard event handling
00021 #include <QTimer> // Timer for delayed or periodic execution
00022
00023 // Standard C++ header for string stream operations
00024 #include <sstream>
00025
00026 // Project-specific header for report generation
00027 #include "report.h"
00028
00029 namespace Ui
00030 {
00031     class TableBoard;
00032 }
00033
00042 template <typename T>
00043 std::string to_string_with_precision(const T a_value, const int n = 6)
00044 {
00045     std::ostringstream out;
00046     out.precision(n);
00047     out << std::fixed << a_value;
00048     return out.str();
00049 }
00050
00051
00058 class TableBoard : public QDialog
00059 {
00060     Q_OBJECT
00061
00062 public:
00068     explicit TableBoard(QWidget *_parent = nullptr);
00069
00073     ~TableBoard();
00074
00080     void ValidatorInput();
00081
00087     void PopulateTable();
00088
00094     static void printPdfThread(QString report);
00095
00101     void Translate();
00102
00103 private:
00104     QWidget *parent;
00105     Ui::TableBoard *ui;
00106
00107     // Member variables grouped by functionality
00108     size_t entries {0};

```

```

00109         ReportMeasurements *reportMeasurementsDialog {nullptr};
00110         std::string nameWaterMeter;
00111         double minimumFlowMain {0};
00112         double transitoriuFlowMain {0};
00113         double nominalFlowMain {0};
00114         double nominalError {0};
00115         double maximumError {0};
00116
00117         // Vectors for managing table widgets
00118         std::vector<QLabel *> vectorNumber;
00119         std::vector<QCheckBox *> vectorCheckNumber;
00120         std::vector<QLineEdit *> vectorSerialNumber;
00121         std::vector<QLineEdit *> vectorFirstIndexStart;
00122         std::vector<QLineEdit *> vectorFirstIndexStop;
00123         std::vector<QLineEdit *> vectorFirstError;
00124         std::vector<QLineEdit *> vectorSecondIndexStart;
00125         std::vector<QLineEdit *> vectorSecondIndexStop;
00126         std::vector<QLineEdit *> vectorSecondError;
00127         std::vector<QLineEdit *> vectorThirdIndexStart;
00128         std::vector<QLineEdit *> vectorThirdIndexStop;
00129         std::vector<QLineEdit *> vectorThirdError;
00130
00131         static QString report;
00132         QTimer *QTimerGenerareFM;
00133
00141         bool eventFilter(QObject *, QEvent *);
00142
00143     private slots:
00147         void onTypeMeterChanged();
00148
00152         void onNumberOfWaterMetersChanged();
00153
00157         void onMeasurementTypeChanged();
00158
00162         void onSelectAllChanged();
00163
00169         void onCbClicked(bool status);
00170
00174         void onCalculateClicked();
00175
00179         void onCleanClicked();
00180
00184         void onCloseClicked();
00185
00189         void onSaveCurrentInputDataClicked();
00190
00194         void onOpenInputDataClicked();
00195
00199         void onPrintPdfDocClicked();
00200
00206         void focusInEvent(QFocusEvent *event);
00207
00213         void focusOutEvent(QFocusEvent *event);
00214
00221         void copyTextBetweenWidgets(const QString& startRegex, const QString& stopRegex);
00222
00226         void onCopy12Clicked();
00227
00231         void onCopy23Clicked();
00232
00236         void onReportClicked();
00237
00241         void enableGenerareFmButton();
00242
00243     protected:
00249         void showEvent(QShowEvent *event);
00250     };
00251
00252
00253 #endif // TABLEBOARD_H

```

5.30 C:/Users/Constantin/Desktop/HERE_WFlowLab/Meter/waterdensity.h File Reference

Header file for water density calculations.

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

Functions

- double [linearInterpolationTemperature](#) (double temperature, double correction)
Performs linear interpolation for temperature correction.
- double [quadraticInterpolationTemperature](#) (double temperature, double correction)
Performs quadratic interpolation for temperature correction.
- double [quadraticInterpolationVolumeCorrection](#) (double temperature)
Calculates the volume correction using quadratic interpolation.

5.30.1 Detailed Description

Header file for water density calculations.

This file defines functions for calculating water density and related corrections.

Author

Constantin

Date

Insert date

5.30.2 Function Documentation

5.30.2.1 [linearInterpolationTemperature\(\)](#)

```
double linearInterpolationTemperature (  
    double temperature,  
    double correction)
```

Performs linear interpolation for temperature correction.

Given a temperature and a correction factor, this function performs linear interpolation to adjust the correction factor based on the temperature.

Parameters

<i>temperature</i>	The temperature for correction.
<i>correction</i>	The correction factor to be adjusted.

Returns

Adjusted correction factor based on linear interpolation.

Performs linear interpolation for temperature correction.

Parameters

<i>temperature</i>	Temperature (in Celsius) for interpolation.
<i>correction</i>	Correction factor applied to the density calculation.

Returns

Interpolated water density.

The function calculates water density using linear interpolation based on temperature points and corresponding density values. It handles out-of-range temperatures with default values.

5.30.2.2 quadraticInterpolationTemperature()

```
double quadraticInterpolationTemperature (  
    double temperature,  
    double correction)
```

Performs quadratic interpolation for temperature correction.

Given a temperature and a correction factor, this function performs quadratic interpolation to adjust the correction factor based on the temperature.

Parameters

<i>temperature</i>	The temperature for correction.
<i>correction</i>	The correction factor to be adjusted.

Returns

Adjusted correction factor based on quadratic interpolation.

Performs quadratic interpolation for temperature correction.

Parameters

<i>temperature</i>	Temperature (in Celsius) for interpolation.
<i>correction</i>	Correction factor applied to the density calculation.

Returns

Interpolated water density.

The function calculates water density using quadratic interpolation based on temperature points and corresponding density values. It handles out-of-range temperatures with default values.

5.30.2.3 quadraticInterpolationVolumeCorrection()

```
double quadraticInterpolationVolumeCorrection (  
    double temperature)
```

Calculates the volume correction using quadratic interpolation.

This function determines the volume correction factor based on the temperature using quadratic interpolation. The exact formula and parameters are specific to the application's requirements.

Parameters

<i>temperature</i>	The temperature for which the volume correction is calculated.
--------------------	--

Returns

Volume correction factor based on quadratic interpolation.

Calculates the volume correction using quadratic interpolation.

Parameters

<i>temperature</i>	Temperature (in Celsius) for interpolation.
--------------------	---

Returns

Interpolated volume correction factor.

The function calculates volume correction factor using quadratic interpolation based on temperature points and corresponding correction values. It handles out-of-range temperatures with default values.

5.31 waterdensity.h

[Go to the documentation of this file.](#)

```
00001
00011 #ifndef WATERDENSITY_H
00012 #define WATERDENSITY_H
00013
00024 double linearInterpolationTemperature(double temperature,
00025                                     double correction);
00026
00037 double quadraticInterpolationTemperature(double temperature,
00038                                     double correction);
00039
00050 double quadraticInterpolationVolumeCorrection(double temperature);
00051
00052 #endif // WATERDENSITY_H
```

