# SiWaSim

Generated by Doxygen 1.9.3

# Chapter 1

# Class Index

## 1.1 Class List

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

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all files with brief descriptions:

# Chapter 3

# Class Documentation

## 3.1 Configuration Class Reference

`#include <Configuration.hpp>`

### Public Member Functions

- Configuration (std::string path)
- ~Configuration ()
- void loadConfiguration ()

### Public Attributes

- LoadCellMode cellMode = LoadCellMode::NORMAL

  *Loadcell mode to be simulated.*
- SYSTEM_TYPE systemType = SYSTEM_TYPE::DOSING_SCALE

  *System type to be simulated.*
- float exc_voltage = 10.f

  *Nominal EXC voltage ouputted by the SIWAREX module.*
- float load_weight = 20.f

  *Nominal Load Weight of the cell in kg.*
- float initial_weight = 10.f

  *Initial weight (for manual / non-auto mode)*
- float addvol_ratio = 500

  *Inverted OpAmp gain (e.g.: At 10V Aout the added / subtracted voltage is 20mV --> ratio = 10V / 20mV = 500)*
- float max_diff_voltage = 40

  *Maximum Differential Voltage of SIG+-.*
- float cellCharecteristic = 4

  *Characteristic in mV/V.*
- float speedAt100 = 5

  *Belt velocity in m/s at 100% speed.*
- float freqAt100 = 10000

  *Belt encoder frequency at 100% speed.*
- float startVoltage = 2

- float endVoltage = 9
- float a
- float b
- float c
- float d
- MATERIAL_FLOW inputChannel1 = MATERIAL_FLOW::EMPTY
- MATERIAL_FLOW inputChannel2 = MATERIAL_FLOW::FINE
- MATERIAL_FLOW inputChannel3 = MATERIAL_FLOW::COARSE
- MATERIAL_FLOW inputChannel4 = MATERIAL_FLOW::XCOARSE

### 3.1.1 Constructor & Destructor Documentation

#### 3.1.1.1 Configuration()

```
Configuration::Configuration (
            std::string path )
```

Creates a new configuration that stores all configuration settings needed for the Simulator. IMPORTANT: Should only be created once, since there is only one valid configuration for the simulator!

**Parameters**

| | |
|---|---|
| *path* | The path to the configuration file on the filesystem |

#### 3.1.1.2 ~Configuration()

```
Configuration::~Configuration ( )
```

### 3.1.2 Member Function Documentation

#### 3.1.2.1 loadConfiguration()

```
void Configuration::loadConfiguration ( )
```

Loads a configuration file from the file system (specified by path in Configuration(std::string path)) and parses all settings to their respective variables

### 3.1.3 Member Data Documentation

#### 3.1.3.1 a

```
float Configuration::a
```

#### 3.1.3.2 addvol_ratio

```
float Configuration::addvol_ratio = 500
```

Inverted OpAmp gain (e.g.: At 10V Aout the added / subtracted voltage is 20mV --> ratio = 10V / 20mV = 500)

#### 3.1.3.3 b

```
float Configuration::b
```

#### 3.1.3.4 c

```
float Configuration::c
```

#### 3.1.3.5 cellCharecteristic

```
float Configuration::cellCharecteristic = 4
```

Characteristic in mV/V.

#### 3.1.3.6 cellMode

```
LoadCellMode Configuration::cellMode = LoadCellMode::NORMAL
```

Loadcell mode to be simulated.

#### 3.1.3.7 d

```
float Configuration::d
```

#### 3.1.3.8 endVoltage

```
float Configuration::endVoltage = 9
```

#### 3.1.3.9 exc_voltage

```
float Configuration::exc_voltage = 10.f
```

Nominal EXC voltage ouputted by the SIWAREX module.

#### 3.1.3.10 freqAt100

```
float Configuration::freqAt100 = 10000
```

Belt encoder frequency at 100% speed.

#### 3.1.3.11 initial_weight

```
float Configuration::initial_weight = 10.f
```

Initial weight (for manual / non-auto mode)

#### 3.1.3.12 inputChannel1

```
MATERIAL_FLOW Configuration::inputChannel1 = MATERIAL_FLOW::EMPTY
```

#### 3.1.3.13 inputChannel2

```
MATERIAL_FLOW Configuration::inputChannel2 = MATERIAL_FLOW::FINE
```

#### 3.1.3.14 inputChannel3

```
MATERIAL_FLOW Configuration::inputChannel3 = MATERIAL_FLOW::COARSE
```

#### 3.1.3.15 inputChannel4

```
MATERIAL_FLOW Configuration::inputChannel4 = MATERIAL_FLOW::XCOARSE
```

#### 3.1.3.16 load_weight

```
float Configuration::load_weight = 20.f
```

Nominal Load Weight of the cell in kg.

#### 3.1.3.17 max_diff_voltage

```
float Configuration::max_diff_voltage = 40
```

Maximum Differential Voltage of SIG+-.

#### 3.1.3.18 speedAt100

```
float Configuration::speedAt100 = 5
```

Belt velocity in m/s at 100% speed.

#### 3.1.3.19 startVoltage

```
float Configuration::startVoltage = 2
```

#### 3.1.3.20 systemType

```
SYSTEM_TYPE Configuration::systemType = SYSTEM_TYPE::DOSING_SCALE
```

System type to be simulated.

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.hpp
- F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.cpp

## 3.2 CURVE Struct Reference

`#include <MaterialFlow.hpp>`

### Public Attributes

- float startDelay = 0

  *Delay from input high to flow increase start in seconds.*
- float stopDelay = 0

  *Delay from input low to flow decrease start in seconds.*
- float riseTime = 1

  *Time it takes the flow to reach its maximum in seconds.*
- float fallTime = 1

  *Time it takes the flow to reach zero in seconds.*
- float maxFlow = 1

  *Maximal flow after rise time in kg/s.*

### 3.2.1 Member Data Documentation

#### 3.2.1.1 fallTime

`float CURVE::fallTime = 1`

Time it takes the flow to reach zero in seconds.

#### 3.2.1.2 maxFlow

`float CURVE::maxFlow = 1`

Maximal flow after rise time in kg/s.

#### 3.2.1.3 riseTime

`float CURVE::riseTime = 1`

Time it takes the flow to reach its maximum in seconds.

#### 3.2.1.4 startDelay

```
float CURVE::startDelay = 0
```

Delay from input high to flow increase start in seconds.

#### 3.2.1.5 stopDelay

```
float CURVE::stopDelay = 0
```

Delay from input low to flow decrease start in seconds.

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.hpp

## 3.3 GPIO Class Reference

```
#include <GPIO.hpp>
```

### Public Member Functions

- GPIO ()
- ~GPIO ()
- void setPWM (int pin, float dutyCycle, float frequency)
- void setPinMode (uint8_t pin, uint8_t mode)
- void writePin (uint8_t pin, bool state)
- bool readPin (uint8_t pin)

### 3.3.1 Constructor & Destructor Documentation

#### 3.3.1.1 GPIO()

```
GPIO::GPIO ( )
```

#### 3.3.1.2 ~GPIO()

```
GPIO::~GPIO ( )
```

### 3.3.2 Member Function Documentation

#### 3.3.2.1 readPin()

```
bool GPIO::readPin (
            uint8_t pin )
```

#### 3.3.2.2 setPinMode()

```
void GPIO::setPinMode (
            uint8_t pin,
            uint8_t mode )
```

#### 3.3.2.3 setPWM()

```
void GPIO::setPWM (
            int pin,
            float dutyCycle,
            float frequency )
```

#### 3.3.2.4 writePin()

```
void GPIO::writePin (
            uint8_t pin,
            bool state )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.hpp
- F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.cpp

## 3.4 I2C Class Reference

```
#include <I2C.hpp>
```

## Public Member Functions

- I2C (std::string dev, uint16_t address)
- ~I2C ()
- bool begin ()
- bool writeData (uint8_t data)
- bool writeData (uint8_t *data, uint8_t length)
- bool readData (uint8_t *data, uint8_t length)
- uint8_t readData ()

### 3.4.1 Constructor & Destructor Documentation

#### 3.4.1.1 I2C()

```
I2C::I2C (
            std::string dev,
            uint16_t address )
```

#### 3.4.1.2 ~I2C()

```
I2C::~I2C ( )
```

### 3.4.2 Member Function Documentation

#### 3.4.2.1 begin()

```
bool I2C::begin ( )
```

#### 3.4.2.2 readData() [1/2]

```
uint8_t I2C::readData ( )
```

#### 3.4.2.3 readData() [2/2]

```
bool I2C::readData (
            uint8_t * data,
            uint8_t length )
```

#### 3.4.2.4 writeData() [1/2]

```
bool I2C::writeData (
            uint8_t * data,
            uint8_t length )
```

#### 3.4.2.5 writeData() [2/2]

```
bool I2C::writeData (
            uint8_t data )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.hpp
- F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.cpp

## 3.5 IABoard Class Reference

```
#include <IABoard.hpp>
```

### Public Member Functions

- IABoard ()
- ~IABoard ()
- bool detectBoard ()
- uint8_t digitalRead ()
- bool digitalRead (uint8_t channel)
- bool getDigitalRead (uint8_t channel)
- uint16_t readTransistions (uint8_t channel)
- TRANSITION getTransistionType (uint8_t channel)
- void setTransistionType (uint8_t channel, TRANSITION tran)
- void resetTransitions (uint8_t channel)
- float getAnalogVolOut (uint8_t channel)
- void setAnalogVolOut (uint8_t channel, float voltage)
- float getAnalogCurOut (uint8_t channel)
- void setAnalogCurOut (uint8_t channel, float current)
- float getOpenDrainPWM (uint8_t channel)
- void setOpenDrainPWM (uint8_t channel, float dutyCycle)
- uint8_t getOpenDrainDOUT ()

- bool getOpenDrainDOUT (uint8_t channel)
- void setOpenDrainDOUT (uint8_t channel, bool value)
- bool getLED (uint8_t channel)
- void setLED (uint8_t channel, bool value)
- void setAllLED (bool value)
- float readAnalogVolIn (uint8_t channel)
- float readAnalogVolInPM (uint8_t channel)
- float readAnalogCurIn (uint8_t channel)
- void getBoardData ()
- void getBoardData (uint8_t *temp, float *rail24, float *rail5)
- void setAllOFF ()

## 3.5.1 Constructor & Destructor Documentation

### 3.5.1.1 IABoard()

```
IABoard::IABoard ( )
```

### 3.5.1.2 ~IABoard()

```
IABoard::~IABoard ( )
```

## 3.5.2 Member Function Documentation

### 3.5.2.1 detectBoard()

```
bool IABoard::detectBoard ( )
```

### 3.5.2.2 digitalRead() [1/2]

```
uint8_t IABoard::digitalRead ( )
```

#### 3.5.2.3 digitalRead() [2/2]

```
bool IABoard::digitalRead (
            uint8_t channel )
```

#### 3.5.2.4 getAnalogCurOut()

```
float IABoard::getAnalogCurOut (
            uint8_t channel )
```

#### 3.5.2.5 getAnalogVolOut()

```
float IABoard::getAnalogVolOut (
            uint8_t channel )
```

#### 3.5.2.6 getBoardData() [1/2]

```
void IABoard::getBoardData ( )
```

Receives the board data through the command. Board data includes temperature, 24V input rail voltage and 5V rail voltage

#### 3.5.2.7 getBoardData() [2/2]

```
void IABoard::getBoardData (
            uint8_t * temp,
            float * rail24,
            float * rail5 )
```

#### 3.5.2.8 getDigitalRead()

```
bool IABoard::getDigitalRead (
            uint8_t channel )
```

#### 3.5.2.9 getLED()

```
bool IABoard::getLED (
            uint8_t channel )
```

Gets the current state of one of the on board LEDs

**Parameters**

| | |
|---|---|
| *channel* | The LED to be read (1 - 4) |

**Returns**

Returns the state of the LED (0 = OFF, 1 = ON)

#### 3.5.2.10 getOpenDrainDOUT() [1/2]

```
uint8_t IABoard::getOpenDrainDOUT ( )
```

#### 3.5.2.11 getOpenDrainDOUT() [2/2]

```
bool IABoard::getOpenDrainDOUT (
            uint8_t channel )
```

#### 3.5.2.12 getOpenDrainPWM()

```
float IABoard::getOpenDrainPWM (
            uint8_t channel )
```

#### 3.5.2.13 getTransistionType()

```
TRANSITION IABoard::getTransistionType (
            uint8_t channel )
```

#### 3.5.2.14 readAnalogCurIn()

```
float IABoard::readAnalogCurIn (
            uint8_t channel )
```

Reads the Analog Input Current of a channel

**Parameters**

| | |
|---|---|
| *channel* | The channel as marked on the IABoard-PCB (1 - 4) |

**Returns**

Returns the measured current in mA

#### 3.5.2.15 readAnalogVolIn()

```
float IABoard::readAnalogVolIn (
            uint8_t channel )
```

Reads the Analog Input Voltage of a channel if the jumper is not set

**Parameters**

| | |
|---|---|
| *channel* | The channel as marked on the IABoard-PCB (1 - 4) |

**Returns**

Returns the measured voltage in Volts from 0V to 10V

#### 3.5.2.16 readAnalogVolInPM()

```
float IABoard::readAnalogVolInPM (
            uint8_t channel )
```

Reads the Analog Input Voltage of a channel if the jumper is set to measure negative voltages

**Parameters**

| | |
|---|---|
| *channel* | The channel as marked on the IABoard-PCB (1 - 4) |

**Returns**

Returns the measured voltage in Volts from -10V to 10V

#### 3.5.2.17 readTransistions()

```
uint16_t IABoard::readTransistions (
            uint8_t channel )
```

#### 3.5.2.18 resetTransitions()

```
void IABoard::resetTransitions (
            uint8_t channel )
```

#### 3.5.2.19 setAllLED()

```
void IABoard::setAllLED (
            bool value )
```

Sets all IABoard-LEDs to the same state

**Parameters**

| | |
|---|---|
| *value* | The wanted state of all the LEDs (0 = OFF, 1 = ON) |

#### 3.5.2.20 setAllOFF()

```
void IABoard::setAllOFF ( )
```

Sets all digital and analog outputs to OFF / 0V

#### 3.5.2.21 setAnalogCurOut()

```
void IABoard::setAnalogCurOut (
            uint8_t channel,
            float current )
```

#### 3.5.2.22 setAnalogVolOut()

```
void IABoard::setAnalogVolOut (
            uint8_t channel,
            float voltage )
```

#### 3.5.2.23 setLED()

```
void IABoard::setLED (
            uint8_t channel,
            bool value )
```

Sets on of the four on board LEDs to a certain state

**Parameters**

| | |
|---|---|
| *channel* | The LED to be toggled (1 - 4) |
| *value* | The wanted state of the LED (0 = OFF, 1 = ON) |

#### 3.5.2.24 setOpenDrainDOUT()

```
void IABoard::setOpenDrainDOUT (
            uint8_t channel,
            bool value )
```

Sets on of the four digital outputs

**Parameters**

| | |
|---|---|
| *channel* | The Open Drain Pin to be toggled (1 - 4) |
| *value* | The wanted state of the channel |

#### 3.5.2.25 setOpenDrainPWM()

```
void IABoard::setOpenDrainPWM (
            uint8_t channel,
            float dutyCycle )
```

#### 3.5.2.26 setTransistionType()

```
void IABoard::setTransistionType (
            uint8_t channel,
            TRANSITION tran )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.hpp
- F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.cpp

## 3.6 MaterialFlow Class Reference

```
#include <MaterialFlow.hpp>
```

## Public Member Functions

- MaterialFlow (uint8_t channel)
- MaterialFlow (uint8_t channel, MATERIAL_FLOW flowType)
- ~MaterialFlow ()
- void setFlowCurve (CURVE curve)
- void setFlowType (MATERIAL_FLOW flowType)
- float update (float *currentWeight, float dt, bool pinState)

### 3.6.1 Constructor & Destructor Documentation

#### 3.6.1.1 MaterialFlow() [1/2]

```
MaterialFlow::MaterialFlow (
            uint8_t channel )
```

#### 3.6.1.2 MaterialFlow() [2/2]

```
MaterialFlow::MaterialFlow (
            uint8_t channel,
            MATERIAL_FLOW flowType )
```

#### 3.6.1.3 ~MaterialFlow()

```
MaterialFlow::~MaterialFlow ( )
```

### 3.6.2 Member Function Documentation

#### 3.6.2.1 setFlowCurve()

```
void MaterialFlow::setFlowCurve (
            CURVE curve )
```

#### 3.6.2.2 setFlowType()

```
void MaterialFlow::setFlowType (
            MATERIAL_FLOW flowType )
```

#### 3.6.2.3 update()

```
float MaterialFlow::update (
            float * currentWeight,
            float dt,
            bool pinState )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.hpp
- F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.cpp

## 3.7 PCB Class Reference

```
#include <PCB.hpp>
```

### Public Member Functions

- PCB (Configuration *config)
- ~PCB ()
- void ledFault (bool state)
- void ledBusy (bool state)
- void ledReady (bool state)
- void setImpedance (IMPEDANCE impedance)
- void setEXTRASW1 (bool state)
- void setEXTRASW2 (bool state)
- void setPOWERSW1 (bool state)
- void setPOWERSW2 (bool state)
- void setLoadcellVoltage (float voltage)
- void setLoadcellDCVoltage (float voltage)
- void setCellAddvol (float voltage)
- void setCellSubvol (float voltage)
- void setSENVoltage (float voltage)
- float getEXCVoltage ()
- float getSENVoltage ()
- void setPWM (float frequency, float dutyCycle)
- void getBoardStatus ()
- void reloadConfig ()

### 3.7.1 Constructor & Destructor Documentation

#### 3.7.1.1 PCB()

```
PCB::PCB (
            Configuration * config )
```

#### 3.7.1.2 ~PCB()

```
PCB::~PCB ( )
```

### 3.7.2 Member Function Documentation

#### 3.7.2.1 getBoardStatus()

```
void PCB::getBoardStatus ( )
```

#### 3.7.2.2 getEXCVoltage()

```
float PCB::getEXCVoltage ( )
```

#### 3.7.2.3 getSENVoltage()

```
float PCB::getSENVoltage ( )
```

#### 3.7.2.4 ledBusy()

```
void PCB::ledBusy (
            bool state )
```

#### 3.7.2.5 ledFault()

```
void PCB::ledFault (
            bool state )
```

#### 3.7.2.6 ledReady()

```
void PCB::ledReady (
            bool state )
```

#### 3.7.2.7 reloadConfig()

```
void PCB::reloadConfig ( )
```

#### 3.7.2.8 setCellAddvol()

```
void PCB::setCellAddvol (
            float voltage )
```

#### 3.7.2.9 setCellSubvol()

```
void PCB::setCellSubvol (
            float voltage )
```

#### 3.7.2.10 setEXTRASW1()

```
void PCB::setEXTRASW1 (
            bool state )
```

#### 3.7.2.11 setEXTRASW2()

```
void PCB::setEXTRASW2 (
            bool state )
```

#### 3.7.2.12 setImpedance()

```
void PCB::setImpedance (
            IMPEDANCE impedance )
```

#### 3.7.2.13 setLoadcellDCVoltage()

```
void PCB::setLoadcellDCVoltage (
            float voltage )
```

#### 3.7.2.14 setLoadcellVoltage()

```
void PCB::setLoadcellVoltage (
            float voltage )
```

#### 3.7.2.15 setPOWERSW1()

```
void PCB::setPOWERSW1 (
            bool state )
```

#### 3.7.2.16 setPOWERSW2()

```
void PCB::setPOWERSW2 (
            bool state )
```

#### 3.7.2.17 setPWM()

```
void PCB::setPWM (
            float frequency,
            float dutyCycle )
```

#### 3.7.2.18 setSENVoltage()

```
void PCB::setSENVoltage (
            float voltage )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.hpp
- F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.cpp

## 3.8 Simulator Class Reference

`#include <Simulator.hpp>`

### Public Member Functions

- Simulator ()
- ~Simulator ()
- void setWeightPER (float percentage)
- void setWeightKG (float kg)
- void setVelocity (float meterspersecond)
- void setVelocityPER (float percentage)
- void setVelocityFRQ (float frequency)
- void setImpedance (IMPEDANCE impedance)
- void bootupAnimation ()
- void reloadConfig ()
- void testFunction ()
- float run (RUN_MODE runMode, float timestep, float *weight)
- float runPassive (float timestep, float *weight)
- void calibrateLCVoltage ()

### 3.8.1 Constructor & Destructor Documentation

#### 3.8.1.1 Simulator()

`Simulator::Simulator ( )`

#### 3.8.1.2 ~Simulator()

`Simulator::~Simulator ( )`

### 3.8.2 Member Function Documentation

#### 3.8.2.1 bootupAnimation()

`void Simulator::bootupAnimation ( )`

Starts an animation with the on board LEDs

#### 3.8.2.2 calibrateLCVoltage()

```
void Simulator::calibrateLCVoltage ( )
```

#### 3.8.2.3 reloadConfig()

```
void Simulator::reloadConfig ( )
```

Reloads the configuration from the disk an stores the settings

#### 3.8.2.4 run()

```
float Simulator::run (
            RUN_MODE runMode,
            float timestep,
            float * weight )
```

#### 3.8.2.5 runPassive()

```
float Simulator::runPassive (
            float timestep,
            float * weight )
```

#### 3.8.2.6 setImpedance()

```
void Simulator::setImpedance (
            IMPEDANCE impedance )
```

#### 3.8.2.7 setVelocity()

```
void Simulator::setVelocity (
            float meterspersecond )
```

Sets the simulated belt velocity in meters per second

**Parameters**

| | |
|---|---|
| *meterspersecond* | Velocity in meters / second |

#### 3.8.2.8 setVelocityFRQ()

```
void Simulator::setVelocityFRQ (
            float frequency )
```

Sets the PWM output to a certain frequency to represent belt movement

**Parameters**

| | |
|---|---|
| *frequency* | The frequency of the PWM signal |

#### 3.8.2.9 setVelocityPER()

```
void Simulator::setVelocityPER (
            float percentage )
```

Sets the simulated belt velocity from 0 - 100% of the maximal speed

**Parameters**

| | |
|---|---|
| *percentage* | Percentage of the maximal speed from 0 to 1 |

#### 3.8.2.10 setWeightKG()

```
void Simulator::setWeightKG (
            float kg )
```

Set the output weight of the simulated load cell in kg

**Parameters**

| | |
|---|---|
| *kg* | Output weight in kilograms |

#### 3.8.2.11 setWeightPER()

```
void Simulator::setWeightPER (
            float percentage )
```

Set the output weight as a percentage of the nominal load

**Parameters**

| | |
|---|---|
| *percentage* | Percentage from 0 - 1 where 1 represents the nominal load as specified |

#### 3.8.2.12 testFunction()

```
void Simulator::testFunction ( )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.hpp
- F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.cpp

## 3.9 UART Class Reference

```
#include <UART.hpp>
```

### Public Member Functions

- UART ()
- ~UART ()
- bool begin ()
- bool transmitMSG (uint8_t *msg, uint16_t length)
- std::vector< uint8_t > receiveMSG ()

### 3.9.1 Constructor & Destructor Documentation

#### 3.9.1.1 UART()

```
UART::UART ( )
```

#### 3.9.1.2 ~UART()

```
UART::~UART ( )
```

### 3.9.2 Member Function Documentation

#### 3.9.2.1 begin()

```
bool UART::begin ( )
```

#### 3.9.2.2 receiveMSG()

```
std::vector< uint8_t > UART::receiveMSG ( )
```

#### 3.9.2.3 transmitMSG()

```
bool UART::transmitMSG (
            uint8_t * msg,
            uint16_t length )
```

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

- F:/GITHUB/SiWaSIM-PiSoftware/src/UART.hpp
- F:/GITHUB/SiWaSIM-PiSoftware/src/UART.cpp

# Chapter 4

# File Documentation

## 4.1 F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.cpp File Reference

```
#include "Configuration.hpp"
```

## 4.2 F:/GITHUB/SiWaSIM-PiSoftware/src/Configuration.hpp File Reference

```
#include <string>
#include <iostream>
#include <fstream>
#include "nlohmann/json.hpp"
```

### Classes

- class Configuration

### Macros

- #define CONFIG_PATH "/home/siwasim/SiWaSIM-PiSoftware/Konfiguration/config.json"
- #define I2C_ADDRESS 0x50
- #define I2C_DEVICE "/dev/i2c-1"
- #define PIN_LED_READY 23
- #define PIN_LED_BUSY 24
- #define PIN_LED_FAULT 25
- #define PWM_PIN 13
- #define PIN_POWERSW1 4
- #define PIN_POWERSW2 26
- #define PIN_IMPEDANCE1 5
- #define PIN_IMPEDANCE2 6
- #define PIN_EXTRASW1 27
- #define PIN_EXTRASW2 22
- #define ADDVOL_CHANNEL 2
- #define SUBVOL_CHANNEL 3
- #define CELL_DC 1
- #define SEN_OUT 4
- #define EXC_IN 1
- #define SEN_IN 2

### Typedefs

- using json = nlohmann::json

### Enumerations

- enum LoadCellMode { NORMAL = 0x00 , OVERLOAD = 0x01 , INVERTED = 0x02 }
- enum IMPEDANCE { OPEN = 0x00 , NOMINAL = 0x01 , SHORT = 0x02 }
- enum SYSTEM_TYPE { DOSING_SCALE = 0x01 , BELT_SCALE = 0x02 }
- enum MATERIAL_FLOW {
  NONE = 0x00 , EMPTY , FINE , COARSE ,
  XCOARSE }
- enum RUN_MODE {
  AUTO , PASSIVE , MANUAL , IDLE ,
  OFF }

## 4.2.1 Macro Definition Documentation

#### 4.2.1.1 ADDVOL_CHANNEL

```
#define ADDVOL_CHANNEL 2
```

#### 4.2.1.2 CELL_DC

```
#define CELL_DC 1
```

#### 4.2.1.3 CONFIG_PATH

```
#define CONFIG_PATH "/home/siwasim/SiWaSIM-PiSoftware/Konfiguration/config.json"
```

#### 4.2.1.4 EXC_IN

```
#define EXC_IN 1
```

#### 4.2.1.5 I2C_ADDRESS

```
#define I2C_ADDRESS 0x50
```

#### 4.2.1.6 I2C_DEVICE

```
#define I2C_DEVICE "/dev/i2c-1"
```

#### 4.2.1.7 PIN_EXTRASW1

```
#define PIN_EXTRASW1 27
```

#### 4.2.1.8 PIN_EXTRASW2

```
#define PIN_EXTRASW2 22
```

#### 4.2.1.9 PIN_IMPEDANCE1

```
#define PIN_IMPEDANCE1 5
```

#### 4.2.1.10 PIN_IMPEDANCE2

```
#define PIN_IMPEDANCE2 6
```

#### 4.2.1.11 PIN_LED_BUSY

```
#define PIN_LED_BUSY 24
```

#### 4.2.1.12 PIN_LED_FAULT

```
#define PIN_LED_FAULT 25
```

#### 4.2.1.13 PIN_LED_READY

```
#define PIN_LED_READY 23
```

#### 4.2.1.14 PIN_POWERSW1

```
#define PIN_POWERSW1 4
```

#### 4.2.1.15 PIN_POWERSW2

```
#define PIN_POWERSW2 26
```

#### 4.2.1.16 PWM_PIN

```
#define PWM_PIN 13
```

#### 4.2.1.17 SEN_IN

```
#define SEN_IN 2
```

#### 4.2.1.18 SEN_OUT

```
#define SEN_OUT 4
```

#### 4.2.1.19 SUBVOL_CHANNEL

```
#define SUBVOL_CHANNEL 3
```

### 4.2.2 Typedef Documentation

#### 4.2.2.1 json

```
using json = nlohmann::json
```

### 4.2.3 Enumeration Type Documentation

#### 4.2.3.1 IMPEDANCE

```
enum IMPEDANCE
```

Types of impedances of the load cell that can be simulated. Is equivilant with the impedance between EXC+ and EXC-

**Enumerator**

| | |
|---|---|
| OPEN | Open circuit, high impedance. |
| NOMINAL | Nominal impedance of approx. 350 ohms. |
| SHORT | Short circuit, approx. zero impedance. |

#### 4.2.3.2 LoadCellMode

```
enum LoadCellMode
```

**Enumerator**

| | |
|---|---|
| NORMAL | Positive differential voltage from 0 - 100% nominal load. |
| OVERLOAD | Positive differential voltage from 0 - 120% nominal load. |
| INVERTED | Negative differential voltage from 0 - 100% nominal load. |

#### 4.2.3.3 MATERIAL_FLOW

```
enum MATERIAL_FLOW
```

Types of different material flows

**Enumerator**

| | |
|---|---|
| NONE | |
| EMPTY | |
| FINE | |
| COARSE | |
| XCOARSE | |

#### 4.2.3.4 RUN_MODE

enum RUN_MODE

**Enumerator**

| | |
|---|---|
| AUTO | |
| PASSIVE | |
| MANUAL | |
| IDLE | |
| OFF | |

#### 4.2.3.5 SYSTEM_TYPE

enum SYSTEM_TYPE

Type of the system represented by the simulator

**Enumerator**

| | |
|---|---|
| DOSING_SCALE | Dosing Scale. |
| BELT_SCALE | Belt Scale. |

## 4.3 Configuration.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include <string>
3 #include <iostream>
4 #include <fstream>
5 #include "nlohmann/json.hpp"
6
7 using json = nlohmann::json;
8
9 #define CONFIG_PATH "/home/siwasim/SiWaSIM-PiSoftware/Konfiguration/config.json"
10
11 // I2C
12 #define I2C_ADDRESS 0x50
13 #define I2C_DEVICE "/dev/i2c-1"
14
15 // LED Pins
16 #define PIN_LED_READY 23
17 #define PIN_LED_BUSY 24
18 #define PIN_LED_FAULT 25
19
20 // PWM Pin
21 #define PWM_PIN 13
22
23 // 24V Power Switch Pins
24 #define PIN_POWERSW1 4
25 #define PIN_POWERSW2 26
26
27 // Pins for Impedance switching
28 #define PIN_IMPEDANCE1 5
```

```
29 #define PIN_IMPEDANCE2 6
30
31 // Pins for extra switches (e.g. WebServer, WriteProtect)
32 #define PIN_EXTRASW1 27
33 #define PIN_EXTRASW2 22
34
35 // Analog Channels
36 #define ADDVOL_CHANNEL 2
37 #define SUBVOL_CHANNEL 3
38 #define CELL_DC 1
39 #define SEN_OUT 4
40 #define EXC_IN 1
41 #define SEN_IN 2
42
43 enum LoadCellMode
44 {
46    NORMAL = 0x00,
48    OVERLOAD = 0x01,
50    INVERTED = 0x02,
51
52 } typedef LoadCellMode;
53
58 enum IMPEDANCE
59 {
61    OPEN = 0x00,
63    NOMINAL = 0x01,
65    SHORT = 0x02,
66
67 } typedef IMPEDANCE;
68
72 enum SYSTEM_TYPE
73 {
75    DOSING_SCALE = 0x01,
77    BELT_SCALE = 0x02,
78 } typedef SYSTEM_TYPE;
79
83 enum MATERIAL_FLOW
84 {
85    NONE = 0x00,
86    EMPTY,
87    FINE,
88    COARSE,
89    XCOARSE,
90
91 } typedef MATERIAL_FLOW;
92
93 enum RUN_MODE
94 {
95    AUTO,
96    PASSIVE,
97    MANUAL,
98    IDLE,
99    OFF
100
101 } typedef RUN_MODE;
102
103 class Configuration
104 {
105 public:
106    Configuration(std::string path);
107    ~Configuration();
108
109    void loadConfiguration();
110
111    // SETTING VARIABLES
113    LoadCellMode cellMode = LoadCellMode::NORMAL;
115    SYSTEM_TYPE systemType = SYSTEM_TYPE::DOSING_SCALE;
117    float exc_voltage = 10.f;
119    float load_weight = 20.f;
121    float initial_weight = 10.f;
123    float addvol_ratio = 500;
125    float max_diff_voltage = 40;
127    float cellCharecteristic = 4;
129    float speedAt100 = 5;
131    float freqAt100 = 10000;
132
133    float startVoltage = 2;
134    float endVoltage = 9;
135
136    float a, b, c, d;
137
138    // Input channel assignment
139    MATERIAL_FLOW inputChannel1 = MATERIAL_FLOW::EMPTY;
140    MATERIAL_FLOW inputChannel2 = MATERIAL_FLOW::FINE;
141    MATERIAL_FLOW inputChannel3 = MATERIAL_FLOW::COARSE;
142    MATERIAL_FLOW inputChannel4 = MATERIAL_FLOW::XCOARSE;
143
```

```
144 private:
145     void parseJSON();
146     std::string _path;
147 };
```

## 4.4 F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.cpp File Reference

```
#include "GPIO.hpp"
```

## 4.5 F:/GITHUB/SiWaSIM-PiSoftware/src/GPIO.hpp File Reference

```
#include <signal.h>
#include <pigpio.h>
#include <stdint.h>
#include <cstdio>
```

### Classes

- class GPIO

## 4.6 GPIO.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include <signal.h>
3 #include <pigpio.h>
4 #include <stdint.h>
5 #include <cstdio>
6
7 class GPIO
8 {
9 public:
10     GPIO();
11     ~GPIO();
12     void setPWM(int pin, float dutyCycle, float frequency);
13
14     void setPinMode(uint8_t pin, uint8_t mode);
15
16     void writePin(uint8_t pin, bool state);
17     bool readPin(uint8_t pin);
18
19 private:
20 };
```

## 4.7 F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.cpp File Reference

```
#include "I2C.hpp"
```

## 4.8 F:/GITHUB/SiWaSIM-PiSoftware/src/I2C.hpp File Reference

```
#include <stdio.h>
#include <unistd.h>
#include <string>
#include <stdint.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/i2c.h>
#include <linux/i2c-dev.h>
```

### Classes

- class I2C

## 4.9 I2C.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include <stdio.h>
3 #include <unistd.h>
4 #include <string>
5 #include <stdint.h>
6 #include <sys/stat.h>
7 #include <fcntl.h>
8 #include <sys/ioctl.h>
9 #include <linux/i2c.h>
10 #include <linux/i2c-dev.h>
11
12 class I2C
13 {
14 public:
15    I2C(std::string dev, uint16_t address);
16    ~I2C();
17    bool begin();
18    bool writeData(uint8_t data);
19    bool writeData(uint8_t *data, uint8_t length);
20    bool readData(uint8_t *data, uint8_t length);
21    uint8_t readData();
22
23 private:
24    std::string _dev;
25    uint16_t _address;
26    int i2c0 = -1;
27 };
```

## 4.10 F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.cpp File Reference

```
#include "IABoard.hpp"
```

## 4.11 F:/GITHUB/SiWaSIM-PiSoftware/src/IABoard.hpp File Reference

```
#include "Configuration.hpp"
#include "I2C.hpp"
#include "utility.hpp"
#include <chrono>
#include <thread>
#include <iostream>
```

## Classes

- class IABoard

## Enumerations

- enum TRANSITION {
  DISABLE = 0x00 , RISING = 0x01 , FALLING = 0x02 , BOTH = 0x03 ,
  UNDEFINED = 0x04 }

### 4.11.1 Enumeration Type Documentation

#### 4.11.1.1 TRANSITION

```
enum TRANSITION
```

**Enumerator**

| | |
|---|---|
| DISABLE | |
| RISING | |
| FALLING | |
| BOTH | |
| UNDEFINED | |

## 4.12 IABoard.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include "Configuration.hpp"
3 #include "I2C.hpp"
4 #include "utility.hpp"
5
6 #include <chrono>
7 #include <thread>
8 #include <iostream>
9 using namespace std::chrono_literals;
10
11 enum TRANSITION
12 {
13     DISABLE = 0x00,
14     RISING = 0x01,
15     FALLING = 0x02,
16     BOTH = 0x03,
17     UNDEFINED = 0x04
18 } typedef TRANSITION;
19
20 class IABoard
21 {
22 public:
23     IABoard();
24     ~IABoard();
25
26     // Check if the board is responding
27     bool detectBoard();
28
29     // Read all digital inputs
```

```
30     uint8_t digitalRead();
31     // Read digital input of certain channel 1 - 4
32     bool digitalRead(uint8_t channel);
33     bool getDigitalRead(uint8_t channel);
34 
35     // Reads the number of counted transitions (if enabled)
36     uint16_t readTransistions(uint8_t channel);
37     // Reads th ecurrently set transition type
38     TRANSITION getTransistionType(uint8_t channel);
39     // Sets the type of transistions that should be counted
40     void setTransistionType(uint8_t channel, TRANSITION tran);
41     // Sets the transistion counter of a channel to 0
42     void resetTransitions(uint8_t channel);
43 
44     // Get the currently set analog output voltage
45     float getAnalogVolOut(uint8_t channel);
46     // Set the analog output voltage from 0 - 10V, voltage in volts
47     void setAnalogVolOut(uint8_t channel, float voltage);
48 
49     // Get the currently set analog output current
50     float getAnalogCurOut(uint8_t channel);
51     // Set the analog output current from 4 - 20mA, current in mA
52     void setAnalogCurOut(uint8_t channel, float current);
53 
54     // Get the PWM Duty Cycle for the Open Drain Output (if not used as digital out)
55     float getOpenDrainPWM(uint8_t channel);
56     // Set the PWM Duty Cycle (0 - 100%) for the Open Drain Output
57     void setOpenDrainPWM(uint8_t channel, float dutyCycle);
58 
59     // Read all digital open drain outputs
60     uint8_t getOpenDrainDOUT();
61     // Get the currently set open drain digital out value
62     bool getOpenDrainDOUT(uint8_t channel);
63     // Set the digital open drain output
64     void setOpenDrainDOUT(uint8_t channel, bool value);
65 
66     // Gets the state of a certain LED
67     bool getLED(uint8_t channel);
68     // Sets a certain LED Low or High
69     void setLED(uint8_t channel, bool value);
70     // Sets all LEDs ON or OFF
71     void setAllLED(bool value);
72 
73     // Reads the analog input voltage of a certain channel (0-10V)
74     float readAnalogVolIn(uint8_t channel);
75     // Reads the analog input voltage of a certain channel (-10-10V, Jumper set)
76     float readAnalogVolInPM(uint8_t channel);
77 
78     // Reads the analog input current of a certain channel (4-20mA)
79     float readAnalogCurIn(uint8_t channel);
80 
81     void getBoardData();
82     void getBoardData(uint8_t *temp, float *rail24, float *rail5);
83 
84     // Turn all digital and analog outputs off
85     void setAllOFF();
86 
87 private:
88     I2C *_i2c;
89 
90     bool _digitalRead[4] = {0, 0, 0, 0};
91 
92     uint8_t _fwVersion[2] = {0x00, 0x00};
93     uint8_t _boardTemperature = 0;
94     float _24Vrail = 0.f;
95     float _5Vrail = 0.f;
96 
97     // Delay because the IA-Board can only handle commands every few ms
98     const std::chrono::milliseconds _delayBetweenCommands = 2ms;
99     std::chrono::time_point<std::chrono::system_clock, std::chrono::duration<double» _lastCommand;
100 
101     // Wait till the minimum time between commands has elapsed
102     void waitForIA();
103 };
```

## 4.13 F:/GITHUB/SiWaSIM-PiSoftware/src/main.cpp File Reference

```
#include <iostream>
#include <string>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <vector>
#include "I2C.hpp"
#include "UART.hpp"
#include "GPIO.hpp"
#include "IABoard.hpp"
#include "PCB.hpp"
#include "Simulator.hpp"
#include "matplotlib/matplotlibcpp.h"
```

### Functions

- int main ()

#### 4.13.1 Function Documentation

##### 4.13.1.1 main()

```
int main ( )
```

## 4.14 F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.cpp File Reference

```
#include "MaterialFlow.hpp"
```

## 4.15 F:/GITHUB/SiWaSIM-PiSoftware/src/MaterialFlow.hpp File Reference

```
#include <iostream>
#include "Configuration.hpp"
#include "IABoard.hpp"
```

### Classes

- struct CURVE
- class MaterialFlow

## 4.16 MaterialFlow.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include <iostream>
3 #include "Configuration.hpp"
4 #include "IABoard.hpp"
5
6 struct CURVE
7 {
9    float startDelay = 0;
11    float stopDelay = 0;
13    float riseTime = 1;
15    float fallTime = 1;
17    float maxFlow = 1;
18 } typedef CURVE;
19
20 class MaterialFlow
21 {
22 public:
23    MaterialFlow(uint8_t channel);
24    MaterialFlow(uint8_t channel, MATERIAL_FLOW flowType);
25    ~MaterialFlow();
26
27    void setFlowCurve(CURVE curve);
28    void setFlowType(MATERIAL_FLOW flowType);
29    float update(float *currentWeight, float dt, bool pinState);
30
31 private:
32    uint8_t _channel;
33    MATERIAL_FLOW _flowType;
34
35    bool _lastPinState = 0;
36    float _currentFlow = 0;
37    float _lastPinStateTime = 0;
38
39    CURVE _curve;
40
41    IABoard *_ia;
42 };
```

## 4.17 F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.cpp File Reference

```
#include "PCB.hpp"
```

## 4.18 F:/GITHUB/SiWaSIM-PiSoftware/src/PCB.hpp File Reference

```
#include "utility.hpp"
#include "GPIO.hpp"
#include "IABoard.hpp"
#include "Configuration.hpp"
```

### Classes

- class PCB

## 4.19 PCB.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include "utility.hpp"
3 #include "GPIO.hpp"
4 #include "IABoard.hpp"
5 #include "Configuration.hpp"
6
7 class PCB
8 {
9 public:
10    PCB(Configuration *config);
11    ~PCB();
12
13    void ledFault(bool state);
14    void ledBusy(bool state);
15    void ledReady(bool state);
16
17    void setImpedance(IMPEDANCE impedance);
18
19    void setEXTRASW1(bool state);
20    void setEXTRASW2(bool state);
21
22    void setPOWERSW1(bool state);
23    void setPOWERSW2(bool state);
24
25    void setLoadcellVoltage(float voltage);
26    void setLoadcellDCVoltage(float voltage);
27    void setCellAddvol(float voltage);
28    void setCellSubvol(float voltage);
29
30    void setSENVoltage(float voltage);
31
32    float getEXCVoltage();
33    float getSENVoltage();
34
35    void setPWM(float frequency, float dutyCycle);
36    void getBoardStatus();
37
38    void reloadConfig();
39
40 private:
41    GPIO *_gpio;
42    IABoard *_ia;
43    Configuration *_config;
44 };
```

## 4.20 F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.cpp File Reference

```
#include "Simulator.hpp"
```

## 4.21 F:/GITHUB/SiWaSIM-PiSoftware/src/Simulator.hpp File Reference

```
#include "PCB.hpp"
#include "Configuration.hpp"
#include "IABoard.hpp"
#include "MaterialFlow.hpp"
#include <chrono>
#include <thread>
```

### Classes

- class Simulator

## 4.22 Simulator.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include "PCB.hpp"
3 #include "Configuration.hpp"
4 #include "IABoard.hpp"
5 #include "MaterialFlow.hpp"
6
7 #include <chrono>
8 #include <thread>
9
10 using namespace std::chrono_literals;
11
12 class Simulator
13 {
14 public:
15    Simulator();
16    ~Simulator();
17
18    void setWeightPER(float percentage); // Set the weight from 0 - 100% of nominal Load
19    void setWeightKG(float kg);          // Set the weight in kg
20
21    void setVelocity(float meterspersecond);
22    void setVelocityPER(float percentage);
23    void setVelocityFRQ(float frequency);
24
25    void setImpedance(IMPEDANCE impedance);
26
27    void bootupAnimation();
28    void reloadConfig();
29
30    void testFunction();
31
32    float run(RUN_MODE runMode, float timestep, float *weight);
33    float runPassive(float timestep, float *weight);
34
35    void calibrateLCVoltage();
36
37 private:
38    Configuration *_config;
39    PCB *_pcb;
40    IABoard *_ia;
41
42    MaterialFlow *_materialFlows[4];
43 };
```

## 4.23 F:/GITHUB/SiWaSIM-PiSoftware/src/UART.cpp File Reference

```
#include "UART.hpp"
```

## 4.24 F:/GITHUB/SiWaSIM-PiSoftware/src/UART.hpp File Reference

```
#include <stdint.h>
#include <fcntl.h>
#include <iostream>
#include <sstream>
#include <termios.h>
#include <unistd.h>
#include <vector>
```

### Classes

- class UART

## 4.25 UART.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include <stdint.h>
3 #include <fcntl.h>
4 #include <iostream>
5 #include <sstream>
6 #include <termios.h>
7 #include <unistd.h>
8 #include <vector>
9
10 class UART
11 {
12 public:
13     UART();
14     ~UART();
15     bool begin();
16     bool transmitMSG(uint8_t *msg, uint16_t length);
17     std::vector<uint8_t> receiveMSG();
18
19 private:
20     int uart0 = -1;
21     // std::string _dev;
22     const uint8_t _messageSizeRX = 0;      // Number of bytes to wait for
23     const uint8_t _messageTimeoutRX = 50; // Read Timeout in 0.1s steps
24 };
```

## 4.26 F:/GITHUB/SiWaSIM-PiSoftware/src/utility.cpp File Reference

```
#include "utility.hpp"
```

### Functions

- float constrainMinMax (float value, float min, float max)
- float constrainMin (float value, float min)
- float constrainMax (float value, float max)
- void linearRegression (std::vector< float > x, std::vector< float > y, float *m, float *b)
- float calculateAverage (std::vector< float > values)
- void cubicRegression (std::vector< float > x, std::vector< float > y, float *a, float *b, float *c, float *d)
- float solveCubicForVoltage (float a, float b, float c, float d, float value)
- float calculateCubic (float a, float b, float c, float d, float x)
- float calculateCubicDeriv (float a, float b, float c, float x)

### 4.26.1 Function Documentation

#### 4.26.1.1 calculateAverage()

```
float calculateAverage (
            std::vector< float > values )
```

Calculates the average of values of a vector

**Parameters**

| | |
|---|---|
| *values* | The vector that contains the values |

**Returns**

Returns the average of the values in the vector

#### 4.26.1.2 calculateCubic()

```
float calculateCubic (
            float a,
            float b,
            float c,
            float d,
            float x )
```

#### 4.26.1.3 calculateCubicDeriv()

```
float calculateCubicDeriv (
            float a,
            float b,
            float c,
            float x )
```

#### 4.26.1.4 constrainMax()

```
float constrainMax (
            float value,
            float max )
```

Constrain a value to an upper limit if the value is above that limit

**Parameters**

| | |
|---|---|
| *value* | The value to be clipped |
| *max* | The upper limit |

**Returns**

Returns the clipped / constrained value

#### 4.26.1.5 constrainMin()

```
float constrainMin (
            float value,
            float min )
```

Constrain a value to a lower limit if the value is below that limit

**Parameters**

| | |
|---|---|
| *value* | The value to be clipped |
| *min* | The lower limit |

**Returns**

Returns the clipped / constrained value

#### 4.26.1.6 constrainMinMax()

```
float constrainMinMax (
            float value,
            float min,
            float max )
```

Constrain a value between an upper and a lower limit to clip the value

**Parameters**

| | |
|---|---|
| *value* | The value to be clipped |
| *min* | The lower limit |
| *max* | The upper limit |

**Returns**

Returns the clipped / constrained value

#### 4.26.1.7 cubicRegression()

```
void cubicRegression (
            std::vector< float > x,
            std::vector< float > y,
            float * a,
            float * b,
            float * c,
            float * d )
```

Calculates a cubic regression f(x)=ax^3+bx^2+cx+d for a dataset of x and y values

**Parameters**

| | |
|---|---|
| *x* | Vector of x-values of the dataset |
| *y* | Vector of y-values of the dataset |
| *a* | Coefficient in front of x^3 |
| *b* | Coefficient in front of x^2 |
| *c* | Coefficient in front of x^1 |
| *d* | Coefficient in front of x^0 |

#### 4.26.1.8 linearRegression()

```
void linearRegression (
            std::vector< float > x,
            std::vector< float > y,
            float * m,
            float * b )
```

Calculates a linear regression f(x)=mx+b for a dataset of x and y values

#### 4.26.1.9 solveCubicForVoltage()

```
float solveCubicForVoltage (
            float a,
            float b,
            float c,
            float d,
            float value )
```

Newton-Raphson Method for finding the x-value that corresponds to a y-value of a cubic function ax^3+x^2+cx+d=value, only for the range 0 - 10

**Parameters**

| | |
|---|---|
| *a* | Coefficient in front of x^3 |
| *b* | Coefficient in front of x^2 |
| *c* | Coefficient in front of x^1 |
| *d* | Coefficient in front of x^0 |
| *value* | y-value of the cubic function that corresponds to the wanted x-value |

**Returns**

Returns the x-value that corresponds to the y-value

## 4.27 F:/GITHUB/SiWaSIM-PiSoftware/src/utility.hpp File Reference

```
#include <vector>
#include <iostream>
```

```
#include <cmath>
#include "Eigen/Dense"
```

## Functions

- float constrainMinMax (float value, float min, float max)
- float constrainMin (float value, float min)
- float constrainMax (float value, float max)
- void linearRegression (std::vector< float > x, std::vector< float > y, float *m, float *b)
- float calculateAverage (std::vector< float > values)
- void cubicRegression (std::vector< float > x, std::vector< float > y, float *a, float *b, float *c, float *d)
- float solveCubicForVoltage (float a, float b, float c, float d, float value)
- float calculateCubic (float a, float b, float c, float d, float x)
- float calculateCubicDeriv (float a, float b, float c, float x)

### 4.27.1 Function Documentation

#### 4.27.1.1 calculateAverage()

```
float calculateAverage (
            std::vector< float > values )
```

Calculates the average of values of a vector

**Parameters**

| | |
|---|---|
| *values* | The vector that contains the values |

**Returns**

Returns the average of the values in the vector

#### 4.27.1.2 calculateCubic()

```
float calculateCubic (
            float a,
            float b,
            float c,
            float d,
            float x )
```

#### 4.27.1.3 calculateCubicDeriv()

```
float calculateCubicDeriv (
            float a,
            float b,
            float c,
            float x )
```

#### 4.27.1.4 constrainMax()

```
float constrainMax (
            float value,
            float max )
```

Constrain a value to an upper limit if the value is above that limit

**Parameters**

| | |
|---|---|
| *value* | The value to be clipped |
| *max* | The upper limit |

**Returns**

Returns the clipped / constrained value

#### 4.27.1.5 constrainMin()

```
float constrainMin (
            float value,
            float min )
```

Constrain a value to a lower limit if the value is below that limit

**Parameters**

| | |
|---|---|
| *value* | The value to be clipped |
| *min* | The lower limit |

**Returns**

Returns the clipped / constrained value

#### 4.27.1.6 constrainMinMax()

```
float constrainMinMax (
            float value,
            float min,
            float max )
```

Constrain a value between an upper and a lower limit to clip the value

**Parameters**

| | |
|---|---|
| *value* | The value to be clipped |
| *min* | The lower limit |
| *max* | The upper limit |

**Returns**

Returns the clipped / constrained value

#### 4.27.1.7 cubicRegression()

```
void cubicRegression (
            std::vector< float > x,
            std::vector< float > y,
            float * a,
            float * b,
            float * c,
            float * d )
```

Calculates a cubic regression f(x)=ax^3+bx^2+cx+d for a dataset of x and y values

**Parameters**

| | |
|---|---|
| *x* | Vector of x-values of the dataset |
| *y* | Vector of y-values of the dataset |
| *a* | Coefficient in front of x^3 |
| *b* | Coefficient in front of x^2 |
| *c* | Coefficient in front of x^1 |
| *d* | Coefficient in front of x^0 |

#### 4.27.1.8 linearRegression()

```
void linearRegression (
            std::vector< float > x,
            std::vector< float > y,
```

```
            float * m,
            float * b )
```

Calculates a linear regression f(x)=mx+b for a dataset of x and y values

#### 4.27.1.9 solveCubicForVoltage()

```
float solveCubicForVoltage (
            float a,
            float b,
            float c,
            float d,
            float value )
```

Newton-Raphson Method for finding the x-value that corresponds to a y-value of a cubic function ax^3+x^2+cx+d=value, only for the range 0 - 10

**Parameters**

| | |
|---|---|
| *a* | Coefficient in front of x^3 |
| *b* | Coefficient in front of x^2 |
| *c* | Coefficient in front of x^1 |
| *d* | Coefficient in front of x^0 |
| *value* | y-value of the cubic function that corresponds to the wanted x-value |

**Returns**

Returns the x-value that corresponds to the y-value

## 4.28 utility.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include <vector>
3 #include <iostream>
4 #include <cmath>
5 #include "Eigen/Dense"
6
7 using Eigen::MatrixXd;
8 using Eigen::VectorXd;
9
10 float constrainMinMax(float value, float min, float max);
11 float constrainMin(float value, float min);
12 float constrainMax(float value, float max);
13
14 void linearRegression(std::vector<float> x, std::vector<float> y, float *m, float *b);
15
16 float calculateAverage(std::vector<float> values);
17
18 void cubicRegression(std::vector<float> x, std::vector<float> y, float *a, float *b, float *c, float *d);
19
20 float solveCubicForVoltage(float a, float b, float c, float d, float value);
21
22 float calculateCubic(float a, float b, float c, float d, float x);
23 float calculateCubicDeriv(float a, float b, float c, float x);
```

# Index