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 = NORMAL
- float exc_voltage = 10.f
- float load_weight = 20.f
- float initial_weight = 10.f
- float addvol_ratio = 500
- float max_diff_voltage = 40
- float cellCharecteristic = 4

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Configuration()

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

3.1.1.2 ~Configuration()

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

3.1.2 Member Function Documentation

3.1.2.1 loadConfiguration()

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

3.1.3 Member Data Documentation

3.1.3.1 addvol_ratio

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

3.1.3.2 cellCharecteristic

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

3.1.3.3 cellMode

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

3.1.3.4 exc_voltage

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

3.1.3.5 initial_weight

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

3.1.3.6 load_weight

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

3.1.3.7 max_diff_voltage

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

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 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)

3.2.1 Constructor & Destructor Documentation

3.2.1.1 GPIO()

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

3.2.1.2 ~GPIO()

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

3.2.2 Member Function Documentation

3.2.2.1 readPin()

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

3.2.2.2 setPinMode()

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

3.2.2.3 setPWM()

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

3.2.2.4 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

3.3 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 ()

3.3.1 Constructor & Destructor Documentation

3.3.1.1 I2C()

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

3.3.1.2 ~I2C()

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

3.3.2 Member Function Documentation

3.3.2.1 begin()

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

3.3.2.2 readData() [1/2]

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

3.3.2.3 readData() [2/2]

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

3.3.2.4 writeData() [1/2]

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

3.3.2.5 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

3.4 IABoard Class Reference

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

Public Member Functions

- IABoard ()
- ~IABoard ()
- bool detectBoard ()
- uint8_t digitalRead ()
- bool digitalRead (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)
- float readAnalogVolIn (uint8_t channel)
- float readAnalogVolInPM (uint8_t channel)
- float readAnalogCurIn (uint8_t channel)

3.4.1 Constructor & Destructor Documentation

3.4.1.1 IABoard()

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

3.4.1.2 ~IABoard()

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

3.4.2 Member Function Documentation

3.4.2.1 detectBoard()

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

3.4.2.2 digitalRead() [1/2]

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

3.4.2.3 digitalRead() [2/2]

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

3.4.2.4 getAnalogCurOut()

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

3.4.2.5 getAnalogVolOut()

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

3.4.2.6 getLED()

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

3.4.2.7 getOpenDrainDOUT() [1/2]

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

3.4.2.8 getOpenDrainDOUT() [2/2]

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

3.4.2.9 getOpenDrainPWM()

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

3.4.2.10 getTransistionType()

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

3.4.2.11 readAnalogCurIn()

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

3.4.2.12 readAnalogVolIn()

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

3.4.2.13 readAnalogVolInPM()

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

3.4.2.14 readTransistions()

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

3.4.2.15 resetTransitions()

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

3.4.2.16 setAnalogCurOut()

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

3.4.2.17 setAnalogVolOut()

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

3.4.2.18 setLED()

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

3.4.2.19 setOpenDrainDOUT()

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

3.4.2.20 setOpenDrainPWM()

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

3.4.2.21 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

3.5 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 setSENVoltage (float voltage)
- float getEXCVoltage ()
- float getSENVoltage ()

3.5.1 Constructor & Destructor Documentation

3.5.1.1 PCB()

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

3.5.1.2 ~PCB()

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

3.5.2 Member Function Documentation

3.5.2.1 getEXCVoltage()

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

3.5.2.2 getSENVoltage()

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

3.5.2.3 ledBusy()

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

3.5.2.4 ledFault()

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

3.5.2.5 ledReady()

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

3.5.2.6 setEXTRASW1()

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

3.5.2.7 setEXTRASW2()

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

3.5.2.8 setImpedance()

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

3.5.2.9 setLoadcellDCVoltage()

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

3.5.2.10 setLoadcellVoltage()

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

3.5.2.11 setPOWERSW1()

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

3.5.2.12 setPOWERSW2()

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

3.5.2.13 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

3.6 Simulator Class Reference

```
#include <Simulator.hpp>
```

Public Member Functions

- Simulator ()
- ~Simulator ()
- void setWeightPER (float percentage)
- void setWeightKG (float kg)

3.6.1 Constructor & Destructor Documentation

3.6.1.1 Simulator()

```
Simulator::Simulator ( )
```

3.6.1.2 ~Simulator()

```
Simulator::~Simulator ( )
```

3.6.2 Member Function Documentation

3.6.2.1 setWeightKG()

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

3.6.2.2 setWeightPER()

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

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

3.7 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 ()

3.7.1 Constructor & Destructor Documentation

3.7.1.1 UART()

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

3.7.1.2 ~UART()

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

3.7.2 Member Function Documentation

3.7.2.1 begin()

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

3.7.2.2 receiveMSG()

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

3.7.2.3 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>
```

Classes

- class Configuration

Enumerations

- enum LoadCellMode { NORMAL = 0x00 , OVERLOAD = 0x01 , INVERTED = 0x02 }
- enum IMPEDANCE { OPEN = 0x00 , NOMINAL = 0x01 , SHORT = 0x02 }

4.2.1 Enumeration Type Documentation

4.2.1.1 IMPEDANCE

```
enum IMPEDANCE
```

Enumerator

OPEN	
NOMINAL	
SHORT	

4.2.1.2 LoadCellMode

```
enum LoadCellMode
```

Enumerator

NORMAL	
OVERLOAD	
INVERTED	

4.3 Configuration.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include <string>
3
4 enum LoadCellMode
5 {
6    NORMAL = 0x00,   // Positive differential voltage from 0 - 100% nominal load
7    OVERLOAD = 0x01, // Positive differential voltage from 0 - 120% nominal load
8    INVERTED = 0x02, // Negative differential voltage from 0 - 100% nominal load
9
10 } typedef LoadCellMode;
11
12 enum IMPEDANCE
13 {
14    OPEN = 0x00,
15    NOMINAL = 0x01,
16    SHORT = 0x02,
17
18 } typedef IMPEDANCE;
19
20 class Configuration
21 {
22 public:
23    Configuration(std::string path);
24    ~Configuration();
25
26    void loadConfiguration();
27
28    // SETTING VARIABLES
29    LoadCellMode cellMode = NORMAL; // Loadcell mode to be simulated
30    float exc_voltage = 10.f;       // Nominal EXC voltage
31    float load_weight = 20.f;       // Nominal Load Weight of the cell in kg
32    float initial_weight = 10.f;    // Initial weight (for manual / non-auto mode)
33    float addvol_ratio = 500;       // Inverted OpAmp gain (e.g.: At 10V Aout the added / subtracted
       voltage is 20mV --> ratio = 10V / 20mV = 500)
34    float max_diff_voltage = 40;    // Maximum Differential Voltage of SIG+-
35    float cellCharecteristic = 4;   // Charecteristik in mV/V
36
37 private:
38    void parseJSON();
39 };
```

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

Classes

- class IABoard

Macros

- #define I2C_ADDRESS 0x50

Enumerations

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

4.11.1 Macro Definition Documentation

4.11.1.1 I2C_ADDRESS

```
#define I2C_ADDRESS 0x50
```

4.11.2 Enumeration Type Documentation

4.11.2.1 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 "I2C.hpp"
3 #include "utility.hpp"
4
5 #define I2C_ADDRESS 0x50
6
7 enum TRANSITION
8 {
9    DISABLE = 0x00,
10    RISING = 0x01,
11    FALLING = 0x02,
12    BOTH = 0x03,
13    UNDEFINED = 0x04
14 } typedef TRANSITION;
15
16 class IABoard
17 {
18 public:
19    IABoard();
```

```
20      ~IABoard();
21
22      // Check if the board is responding
23      bool detectBoard();
24
25      // Read all digital inputs
26      uint8_t digitalRead();
27      // Read digital input of certain channel 1 - 4
28      bool digitalRead(uint8_t channel);
29
30      // Reads the number of counted transitions (if enabled)
31      uint16_t readTransistions(uint8_t channel);
32      // Reads th ecurrently set transition type
33      TRANSITION getTransistionType(uint8_t channel);
34      // Sets the type of transistions that should be counted
35      void setTransistionType(uint8_t channel, TRANSITION tran);
36      // Sets the transistion counter of a channel to 0
37      void resetTransitions(uint8_t channel);
38
39      // Get the currently set analog output voltage
40      float getAnalogVolOut(uint8_t channel);
41      // Set the analog output voltage from 0 - 10V, voltage in volts
42      void setAnalogVolOut(uint8_t channel, float voltage);
43
44      // Get the currently set analog output current
45      float getAnalogCurOut(uint8_t channel);
46      // Set the analog output current from 4 - 20mA, current in mA
47      void setAnalogCurOut(uint8_t channel, float current);
48
49      // Get the PWM Duty Cycle for the Open Drain Output (if not used as digital out)
50      float getOpenDrainPWM(uint8_t channel);
51      // Set the PWM Duty Cycle (0 - 100%) for the Open Drain Output
52      void setOpenDrainPWM(uint8_t channel, float dutyCycle);
53
54      // Read all digital open drain outputs
55      uint8_t getOpenDrainDOUT();
56      // Get the currently set open drain digital out value
57      bool getOpenDrainDOUT(uint8_t channel);
58      // Set the digital open drain output
59      void setOpenDrainDOUT(uint8_t channel, bool value);
60
61      // Gets the state of a certain LED
62      bool getLED(uint8_t channel);
63      // Sets a certain LED Low or High
64      void setLED(uint8_t channel, bool value);
65
66      // Reads the analog input voltage of a certain channel (0-10V)
67      float readAnalogVolIn(uint8_t channel);
68      // Reads the analog input voltage of a certain channel (-10-10V, Jumper set)
69      float readAnalogVolInPM(uint8_t channel);
70
71      // Reads the analog input current of a certain channel (4-20mA)
72      float readAnalogCurIn(uint8_t channel);
73
74  private:
75      I2C *_i2c;
76  };
```

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"
```

Functions

- int main ()

4.13.1 Function Documentation

4.13.1.1 main()

```
int main ( )
```

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

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

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

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

Classes

- class PCB

Macros

- #define PIN_LED_READY 23
- #define PIN_LED_BUSY 24
- #define PIN_LED_FAULT 25
- #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

4.15.1 Macro Definition Documentation

4.15.1.1 ADDVOL_CHANNEL

```
#define ADDVOL_CHANNEL 2
```

4.15.1.2 PIN_EXTRASW1

```
#define PIN_EXTRASW1 27
```

4.15.1.3 PIN_EXTRASW2

```
#define PIN_EXTRASW2 22
```

4.15.1.4 PIN_IMPEDANCE1

```
#define PIN_IMPEDANCE1 5
```

4.15.1.5 PIN_IMPEDANCE2

```
#define PIN_IMPEDANCE2 6
```

4.15.1.6 PIN_LED_BUSY

```
#define PIN_LED_BUSY 24
```

4.15.1.7 PIN_LED_FAULT

```
#define PIN_LED_FAULT 25
```

4.15.1.8 PIN_LED_READY

```
#define PIN_LED_READY 23
```

4.15.1.9 PIN_POWERSW1

```
#define PIN_POWERSW1 4
```

4.15.1.10 PIN_POWERSW2

```
#define PIN_POWERSW2 26
```

4.15.1.11 SUBVOL_CHANNEL

```
#define SUBVOL_CHANNEL 3
```

4.16 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 // LED Pins
8 #define PIN_LED_READY 23
9 #define PIN_LED_BUSY 24
10 #define PIN_LED_FAULT 25
11
12 // 24V Power Switch Pins
13 #define PIN_POWERSW1 4
14 #define PIN_POWERSW2 26
15
16 // Pins for Impedance switching
17 #define PIN_IMPEDANCE1 5
18 #define PIN_IMPEDANCE2 6
19
20 // Pins for extra switches (e.g. WebServer, WriteProtect)
21 #define PIN_EXTRASW1 27
22 #define PIN_EXTRASW2 22
23
24 #define ADDVOL_CHANNEL 2
25 #define SUBVOL_CHANNEL 3
26
27 class PCB
28 {
29 public:
30    PCB(Configuration *config);
31    ~PCB();
32
33    void ledFault(bool state);
34    void ledBusy(bool state);
35    void ledReady(bool state);
36
37    void setImpedance(IMPEDANCE impedance);
38
```

```
39     void setEXTRASW1(bool state);
40     void setEXTRASW2(bool state);
41
42     void setPOWERSW1(bool state);
43     void setPOWERSW2(bool state);
44
45     void setLoadcellVoltage(float voltage);
46     void setLoadcellDCVoltage(float voltage);
47     void setSENVoltage(float voltage);
48
49     float getEXCVoltage();
50     float getSENVoltage();
51
52 private:
53     GPIO *_gpio;
54     IABoard *_ia;
55     Configuration *_config;
56 };
```

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

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

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

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

Classes

- class Simulator

4.19 Simulator.hpp

Go to the documentation of this file.

```
1 #pragma once
2 #include "PCB.hpp"
3 #include "Configuration.hpp"
4
5 class Simulator
6 {
7 public:
8     Simulator();
9     ~Simulator();
10
11     void setWeightPER(float percentage); // Set the weight from 0 - 100% of nominal Load
12     void setWeightKG(float kg);          // Set the weight in kg
13
14 private:
15     Configuration *_config;
16     PCB *_pcb;
17 };
```

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

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

4.21 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

4.22 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 };
```

4.23 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)

4.23.1 Function Documentation

4.23.1.1 constrainMax()

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

4.23.1.2 constrainMin()

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

4.23.1.3 constrainMinMax()

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

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

Functions

- float constrainMinMax (float value, float min, float max)
- float constrainMin (float value, float min)
- float constrainMax (float value, float max)

4.24.1 Function Documentation

4.24.1.1 constrainMax()

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

4.24.1.2 constrainMin()

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

4.24.1.3 constrainMinMax()

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

4.25 utility.hpp

Go to the documentation of this file.

```
1 #pragma once
2
3 float constrainMinMax(float value, float min, float max);
4 float constrainMin(float value, float min);
5 float constrainMax(float value, float max);
```

Index