Sensebox

Generated by Doxygen 1.9.2

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Module Documentation

5.1 Global defines

Macros

- #define **SenseBox_VERSION** "0.0.1"

  Version of the Sensebox software A change in the first version indicator is a major version change which might not be compatible with older major versions A change in the second version indicator is a minor version change which should be compatible with other of the same major versions A change in the third version indicator is a tweak version change, this will include mostly small bug fixes

- #define DEBUGLEVEL 3

  Used by the source code to indicate what to debug to serial monitor.

- #define LOGLEVEL 3

  Used by the source code to indicate what to log to the SD card.

5.1.1 Detailed Description

5.1.2 Macro Definition Documentation

5.1.2.1 DEBUGLEVEL

```
#define DEBUGLEVEL 3
```

Used by the source code to indicate what to debug to serial monitor.

Value	Description
0	Print nothing to serial
1	Print errors to serial
2	Print errors and warnings to serial
3	Print errors, warnings & info to serial
4	Print errors, Warnings, info & data-dumps to serial

5.1.2.2 LOGLEVEL

```
#define LOGLEVEL 3
```

Used by the source code to indicate what to log to the SD card.

Value	Description
0	Log nothing
1	Logs only errors
2	Logs errors & warnings
3	Logs errors, warnings & info
4	Logs errors, warnings, info & data-dumps

5.2 Global Enumerations

Enumerations

- enum ERR_Type {
  SUCCESS = 0 , READ_SUCCESS = 0 , ALREADY_INITIALIZED = 0 , ERROR = 1 ,
  READ_ERROR = 1 , READ_FAIL = 1 , NOT_INITIALIZED , SD_NOT_INIT ,
  MOD_INIT_ERR , RTC_BEGIN_ERR , SD_BEGIN_ERR , NO_SD_CARD ,
  SD_CARDTYPE_NONE , SD_DIR_OPEN_FAIL , SD_NOT_A_DIR , SD_MKDIR_FAIL ,
  SD_RMDIR_FAIL , SD_FILE_OPEN_FAIL , SD_WRITE_FAIL , SD_APP_FAIL ,
  SD_RENAME_FAIL , SD_RM_FAIL , AMBI_I2C_INIT_ERR , AS726X_BEGIN_ERR ,
  AS726X_DATA_NOT_READY , NO_LDS_SENSOR , SCD30_BEGIN_ERR , TSL_BEGIN_ERR }

  List of function return errors.
- enum class LogLevel {
  LogLevel::Error , LogLevel::Warning , LogLevel::Info , LogLevel::DataDump ,
  LogLevel::__PREV }

  Enum values that indicate the log level of the message.
- enum class LogType { LogType::SD = 0b01 , LogType::Serial = 0b10 , LogType::Serial_SD = 0b11 }

  Enum values that indicate to which logging device should be logged.

5.2.1 Detailed Description

5.2.2 Enumeration Type Documentation

5.2.2.1 ERR_Type

```
enum ERR_Type
```

List of function return errors.

Enumerator

SUCCESS	SUCCESS, No error occured.
READ_SUCCESS	READ_SUCCESS, A read operation was succesfull.
ALREADY_INITIALIZED	ALREADY_INITIALIZED, Module's init function was called multiple times.
ERROR	ERROR, generic error
READ_ERROR	READ_ERROR, A generic read error
READ_FAIL	READ_FAIL, A read operation failed.
NOT_INITIALIZED	NOT_INITIALIZED, Indicates that the module has not been initialized correctly.
SD_NOT_INIT	SD_NOT_INIT, indicates that the SD has not been properly initialized.
MOD_INIT_ERR	MOD_INIT_ERR, indicates that a module was not succesfully initialized.
RTC_BEGIN_ERR	RTC_BEGIN_ERR, indicates that the .begin() method has failed.
SD_BEGIN_ERR	SD_BEGIN_ERR, indicates that the .begin() method has failed.
NO_SD_CARD	NO_SD_CARD, indicates that no SD card was inserted.
SD_CARDTYPE_NONE	SD_CARDTYPE_NONE, indicates that the inserted SD card type is not supported.
SD_DIR_OPEN_FAIL	SD_DIR_OPEN_FAIL, failed to open given directory.
SD_NOT_A_DIR	SD_NOT_A_DIR, given directory does not exist
SD_MKDIR_FAIL	SD_MKDIR_FAIL, failed to create given directory
SD_RMDIR_FAIL	SD_RMDIR_FAIL, failed to remove given directory
SD_FILE_OPEN_FAIL	SD_FILE_OPEN_FAIL, failed to open given file
SD_WRITE_FAIL	SD_WRITE_FAIL, failed to write message to file
SD_APP_FAIL	SD_APP_FAIL, failed to append message to file
SD_RENAME_FAIL	SD_RENAME_FAIL, failed to rename file
SD_RM_FAIL	SD_RM_FAIL, failed to remove file
AMBI_I2C_INIT_ERR	AMBI_I2C_INIT_ERR, Aindicates that an I2C error occured in the init function.
AS726X_BEGIN_ERR	AS726X_BEGIN_ERR, indicates that the .begin() method has failed.
AS726X_DATA_NOT_READY	AS726X_DATA_NOT_READY, indicates that a ready was attempted but data was not yet ready
NO_LDS_SENSOR	NO_LDS_SENSOR, indicates that no LDS sensor was found.
SCD30_BEGIN_ERR	SCD30_BEGIN_ERR, indicates that the .begin() method has failed.
TSL_BEGIN_ERR	TSL_BEGIN_ERR, indicates that the .begin() method has failed.

5.2.2.2 LogLevel

```
enum class LogLevel  [strong]
```

Enum values that indicate the log level of the message.

See also

LogType
Logger::print()
Logger::println()
Logger::write()

Enumerator

Error	(0) Error level, This is used for when an error occures that can break the rest of the system.
Warning	(1) Warning level, This is used for when an error occures that does not have a signficiant impact on the system.
Info	(2) Info level, This is used for logging info or debugging info to the SD card.
DataDump	(3) DataDump, This is a special type of logging used for big blocks of data.
__PREV	() __PREV, default value for the logging functions, if this is passed to the function it will use the previous log level.

5.2.2.3 LogType

```
enum class LogType  [strong]
```

Enum values that indicate to which logging device should be logged.

See also

LogLevel

Logger::print()

Logger::println()

Logger::write()

Logger::breakLine()

Logger::dataDumpEnd()

Enumerator

SD	(0) SD card, Flag to only log the passed info to the SD card.
Serial	(1) Serial, Flag to only log the passed info to the Serial.
Serial_SD	(2) Serial_SD, Flag to log to the passed info to both the Serial and SD.

5.3 Global structures

Classes

- struct RTC_DATE_TIME

  Struct containing the date and time of when the read() function was called.
- struct AmbimateData

  struct containing the ambimate data.
- struct ColorSpectrum

  Struct containing the read colorspectrum values.
- struct SCD30_DATA

  Struct containg the SCD30 Data.
- struct TSL2591_DATA

  Struct containing the TSL2591 data.

5.3.1 Detailed Description

Chapter 6

Class Documentation

6.1 __iW_Module Class Reference

Interface for a hardware module hardware module should be implemented as a singleton to ensure that there won't be two processes accessing the same hardware at the same time Singleton format: https://stackoverflow.com/a/1008289.

`#include <__iW_Module.h>`

Inheritance diagram for __iW_Module:

Public Member Functions

- virtual ERR_Type init ()=0

  a standard initialisation method

Protected Member Functions

- virtual bool checkInitialized ()

  Checks if the Init function has been called.

Protected Attributes

- bool **Initialized** = false

  The Initialized flag is used to keep track of the state of the Module.

6.1.1 Detailed Description

Interface for a hardware module hardware module should be implemented as a singleton to ensure that there won't be two processes accessing the same hardware at the same time Singleton format: https://stackoverflow.com/a/1008289.

6.1.2 Member Function Documentation

6.1.2.1 checkInitialized()

```
virtual bool __iW_Module::checkInitialized ( )  [inline], [protected], [virtual]
```

Checks if the Init function has been called.

some sensors will have an initialisation that needs to be completed first. Internally the functions of those sensors should start with a check if the sensor is initialized and if it is not immediatly return.

Returns

true the init function has been called and exited successfully

false the init function has not been called or exited with an error.

6.1.2.2 init()

```
virtual ERR_Type __iW_Module::init ( )  [pure virtual]
```

a standard initialisation method

Returns

ERR_Type returns 0 on success, 1 on failure

See also

ERR_Type

Implemented in Logger, __W_RTC, __W_SD, __W_Ambimate, __W_AS726X, __W_SM_UART_4L, __W_MAX4466, __W_MIX8410, __W_SCD30, and __W_TSL2591.

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

- src/Wrappers/__W_Module/__iW_Module.h

6.2 __W_Ambimate Class Reference

Singleton ambimate module.

```
#include <__W_Ambimate.h>
```

Inheritance diagram for __W_Ambimate:

Public Member Functions

- **__W_Ambimate** (__W_Ambimate const &)=delete
- void **operator=** (__W_Ambimate const &)=delete
- ERR_Type init ()

  Initializes the Ambimate object. Should only be called once. This function initializes the Ambimate object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- AmbimateData read ()

  Reads the Ambimate sensor's data into the AmbimateData struct. eCO2/VOC sensor is only updated in AmbiMate every 60 seconds.
- uint8_t get_opt_sensors ()

  Get the opt sensors flags.

Static Public Member Functions

- static __W_Ambimate & getInstance ()

  Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.2.1 Detailed Description

Singleton ambimate module.

6.2.2 Member Function Documentation

6.2.2.1 get_opt_sensors()

```
uint8_t __W_Ambimate::get_opt_sensors ( )
```

Get the opt sensors flags.

Returns

uint8_t byte containing the optional sensor flags.

6.2.2.2 getInstance()

```
static __W_Ambimate & __W_Ambimate::getInstance ( )  [inline], [static]
```

Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Returns

__W_Ambimate& the handle to the singleton instance

6.2.2.3 init()

```
ERR_Type __W_Ambimate::init ( )  [virtual]
```

Initializes the Ambimate object. Should only be called once. This function initializes the Ambimate object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

6.2.2.4 read()

```
AmbimateData __W_Ambimate::read ( )
```

Reads the Ambimate sensor's data into the AmbimateData struct. eCO2/VOC sensor is only updated in AmbiMate every 60 seconds.

Returns

AmbimateData the struct containing the read data.

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

- src/Wrappers/Sensors/Ambimate/__W_Ambimate.h
- src/Wrappers/Sensors/Ambimate/__W_Ambimate.cpp

6.3 __W_AS726X Class Reference

Singleton AS726X module.

```
#include <__W_AS726X.h>
```

Inheritance diagram for __W_AS726X:

Public Member Functions

- **__W_AS726X** (__W_AS726X const &)=delete
- void **operator=** (__W_AS726X const &)=delete
- ERR_Type init ()

  Initializes the AS726X object. Should only be called once. This function initializes the AS726X object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- void setDrvLed (bool B)

  turn the Drv Led on or off.
- void setDrvCurrent (uint8_t V)

  Set the Drv Led current.
- void indicateLED (bool B)

  turn the indicate Led on or off.
- void setIndicateCurrent (uint8_t V)

  Set the indicate Led current.
- void setGain (uint8_t gain)

  Set the Gain value.
- void setIntegrationTime (uint8_t time)

  Set the Integration Time.
- void setConversionType (uint8_t type)

  Set the Conversion Type.
- void **startMeasurement** ()

  Start a measurement.
- bool checkDataReady ()

  Check if the data is ready.
- void getMeasurements (ColorSpectrum *CS)

  Get the Measurements.
- uint8_t getTemperature ()

  Get the Temperature value.

Static Public Member Functions

- static __W_AS726X & getInstance ()

  Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.3.1 Detailed Description

Singleton AS726X module.

6.3.2 Member Function Documentation

6.3.2.1 checkDataReady()

```
bool __W_AS726X::checkDataReady ( )
```

Check if the data is ready.

Returns

true Data is ready.

false Data is not ready.

6.3.2.2 getInstance()

```
static __W_AS726X & __W_AS726X::getInstance ( )  [inline], [static]
```

Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Returns

__W_AS726X& the handle to the singleton instance

6.3.2.3 getMeasurements()

```
void __W_AS726X::getMeasurements (
            ColorSpectrum * CS )
```

Get the Measurements.

Parameters

CS	The object into which the measurements should be read.

6.3.2.4 getTemperature()

```
uint8_t __W_AS726X::getTemperature ( )
```

Get the Temperature value.

Returns

uint8_t The read temperature.

6.3.2.5 indicateLED()

```
void __W_AS726X::indicateLED (
            bool B )
```

turn the indicate Led on or off.

Parameters

B	On or Off.

6.3.2.6 init()

```
ERR_Type __W_AS726X::init ( )  [virtual]
```

Initializes the AS726X object. Should only be called once. This function initializes the AS726X object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

6.3.2.7 setConversionType()

```
void __W_AS726X::setConversionType (
            uint8_t type )
```

Set the Conversion Type.

Parameters

type	The conversion Type.

6.3.2.8 setDrvCurrent()

```
void __W_AS726X::setDrvCurrent (
            uint8_t V )
```

Set the Drv Led current.

Parameters

V	the current value.

6.3.2.9 setDrvLed()

```
void __W_AS726X::setDrvLed (
            bool B )
```

turn the Drv Led on or off.

Parameters

B	On or Off.

6.3.2.10 setGain()

```
void __W_AS726X::setGain (
            uint8_t gain )
```

Set the Gain value.

Parameters

gain	the gain value.

6.3.2.11 setIndicateCurrent()

```
void __W_AS726X::setIndicateCurrent (
            uint8_t V )
```

Set the indicate Led current.

Parameters

V	the current value.

6.3.2.12 setIntegrationTime()

```
void __W_AS726X::setIntegrationTime (
            uint8_t time )
```

Set the Integration Time.

Parameters

time	the integration time.

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

- src/Wrappers/Sensors/AS7262/__W_AS726X.h
- src/Wrappers/Sensors/AS7262/__W_AS726X.cpp

6.4 __W_MAX4466 Class Reference

Singleton MAX4466 module.

```
#include <__W_MAX4466.h>
```

Inheritance diagram for __W_MAX4466:

Public Member Functions

- **__W_MAX4466** (__W_MAX4466 const &)=delete
- void **operator=** (__W_MAX4466 const &)=delete
- ERR_Type init ()

  Initializes the MAX4466 object. Should only be called once. This function initializes the MAX4466 object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- int read ()

  Reads the MAX4466 analog value.

Static Public Member Functions

- static __W_MAX4466 & getInstance ()

  Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.4.1 Detailed Description

Singleton MAX4466 module.

6.4.2 Member Function Documentation

6.4.2.1 getInstance()

```
static __W_MAX4466 & __W_MAX4466::getInstance ( )  [inline], [static]
```

Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Returns

__W_MAX4466& the handle to the singleton instance

6.4.2.2 init()

```
ERR_Type __W_MAX4466::init ( )  [virtual]
```

Initializes the MAX4466 object. Should only be called once. This function initializes the MAX4466 object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

6.4.2.3 read()

```
int __W_MAX4466::read ( )
```

Reads the MAX4466 analog value.

Returns

int The ADC converted value.

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

- src/Wrappers/Sensors/MAX4466/__W_MAX4466.h
- src/Wrappers/Sensors/MAX4466/__W_MAX4466.cpp

6.5 __W_MIX8410 Class Reference

Singleton MIX8410 module.

```
#include <__W_MIX8410.h>
```

Inheritance diagram for __W_MIX8410:

Public Member Functions

- **__W_MIX8410** (__W_MIX8410 const &)=delete
- void **operator=** (__W_MIX8410 const &)=delete
- ERR_Type init ()

  Initializes the MIX8410 object. Should only be called once. This function initializes the MIX8410 object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- float readConcentration ()

  Converted ADC value into O2 percentage.
- float readO2Vout ()

  Reads the ADC value and converts it into a voltage value.

Static Public Member Functions

- static __W_MIX8410 & getInstance ()

  Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.5.1 Detailed Description

Singleton MIX8410 module.

6.5.2 Member Function Documentation

6.5.2.1 getInstance()

```
static __W_MIX8410 & __W_MIX8410::getInstance ( )  [inline], [static]
```

Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Returns

__W_MIX8410& the handle to the singleton instance

6.5.2.2 init()

```
ERR_Type __W_MIX8410::init ( )  [inline], [virtual]
```

Initializes the MIX8410 object. Should only be called once. This function initializes the MIX8410 object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

6.5.2.3 readConcentration()

```
float __W_MIX8410::readConcentration ( )
```

Converted ADC value into O2 percentage.

Returns

float The Percentage of Oxygen in the air.

6.5.2.4 readO2Vout()

```
float __W_MIX8410::readO2Vout ( )
```

Reads the ADC value and converts it into a voltage value.

Returns

float The read ADC value.

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

- src/Wrappers/Sensors/MIX8410/__W_MIX8410.h
- src/Wrappers/Sensors/MIX8410/__W_MIX8410.cpp

6.6 __W_RTC Class Reference

Singleton RTC Module.

```
#include <__W_RTC.h>
```

Inheritance diagram for __W_RTC:

Public Member Functions

- **__W_RTC** (__W_RTC const &)=delete
- void **operator=** (__W_RTC const &)=delete
- ERR_Type init ()

  Initializes the RTC object. Should only be called once. This function initializes the RTC object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- RTC_DATE_TIME read ()

  Returns the current date time.
- String stringDateTime ()

  Returns a datetime in a string format.
- String stringTime ()

  Returns the time in a string format.

Static Public Member Functions

- static __W_RTC & getInstance ()

  Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.6.1 Detailed Description

Singleton RTC Module.

6.6.2 Member Function Documentation

6.6.2.1 getInstance()

```
static __W_RTC & __W_RTC::getInstance ( )  [inline], [static]
```

Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Returns

__W_RTC& the handle to the singleton instance

6.6.2.2 init()

```
ERR_Type __W_RTC::init ( )  [virtual]
```

Initializes the RTC object. Should only be called once. This function initializes the RTC object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

6.6.2.3 read()

```
RTC_DATE_TIME __W_RTC::read ( )
```

Returns the current date time.

Returns

the current date time in a RTC_DATE_TIME struct.

6.6.2.4 stringDateTime()

```
String __W_RTC::stringDateTime ( )
```

Returns a datetime in a string format.

Returns

a datetime in a YYYY-MM-DD_hh.mm.ss format.

6.6.2.5 stringTime()

```
String __W_RTC::stringTime ( )
```

Returns the time in a string format.

Returns

a time string in a hh.mm.ss format.

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

- src/Wrappers/RTC/__W_RTC.h
- src/Wrappers/RTC/__W_RTC.cpp

6.7 __W_SCD30 Class Reference

Singleton SCD30 module.

```
#include <__W_SCD30.h>
```

Inheritance diagram for __W_SCD30:

Public Member Functions

- **__W_SCD30** (__W_SCD30 const &)=delete
- void **operator=** (__W_SCD30 const &)=delete
- SCD30_DATA read ()

  Reads the SCD30 sensor.
- ERR_Type init (bool autoCalibrate)

  Init function if autoCalibrate is set to true, this will activate the sensors autocalibrate feature. Please see section 1.3.6 of the SCD30 datasheet: "When activated for the first time a period of minimum 7 days is needed so that the algorithm can find its initial parameter set for ASC. The sensor has to be exposed to fresh air for at least 1 hour every day. Also during that period, the sensor may not be disconnected from the power supply, otherwise the procedure to find calibration parameters is aborted and has to be restarted from the beginning. The successfully calculated parameters are stored in non-volatile memory of the SCD30 having the effect that after a restart the previously found parameters for ASC are still present. ".
- ERR_Type init ()

  Initializes the SCD30 object. Should only be called once. This function initializes the SCD30 object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- bool dataAvailable ()

  Checks if the data is available.
- uint16_t getCO2 ()

  Returns the read CO2 ppm value.
- float getTemperature ()

  Get the Temperature value.
- float getHumidity ()

  Get the Humidity level.
- void setMeasurementsInterval (uint16_t seconds)

  Set the Measurements Interval. Change number of seconds between measurements: 2 to 1800 (30 minutes), stored in non-volatile memory of SCD30.
- bool getMeasurementsInterval (uint16_t &seconds)

  Get the Measurements Interval.
- void setAltitudeCompensation (uint16_t altitude)

  Set the Altitude Compensation Set altitude of the sensor in m, stored in non-volatile memory of SCD30.
- bool getAltitudeCompensation (uint16_t &altitude)

  Get the Altitude Compensation object.
- void setAmbientPressure (uint16_t Pressure)

  Set the Ambient Pressure. Current ambient pressure in mBar: 700 to 1200, will overwrite altitude compensation.
- void setTemperatureOffset (float offset)

  Set the Temperature Offset Optionally we can set temperature offset to ℃, stored in non-volatile memory of SCD30.
- bool getTemperatureOffset (float &offset)

  Get the Temperature Offset.
- bool getAutoSelfCalibration ()

  Get the Auto Self Calibration value.
- bool getForcedRecalibration (uint16_t &settingVal)

  Get the Forced Recalibration value.
- bool getFirmwareVersion (uint16_t &settingVal)

  Get the Firmware Version.

Static Public Member Functions

- static __W_SCD30 & getInstance ()

  Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.7.1 Detailed Description

Singleton SCD30 module.

6.7.2 Member Function Documentation

6.7.2.1 dataAvailable()

```
bool __W_SCD30::dataAvailable ( )
```

Checks if the data is available.

Returns

true Data is ready.

false Data is not ready.

6.7.2.2 getAltitudeCompensation()

```
bool __W_SCD30::getAltitudeCompensation (
            uint16_t & altitude )
```

Get the Altitude Compensation object.

Parameters

altitude	object into which the value should be read.

Returns

true get successfull.

false get failure.

6.7.2.3 getAutoSelfCalibration()

```
bool __W_SCD30::getAutoSelfCalibration ( )
```

Get the Auto Self Calibration value.

Returns

true autoselfcalibration is on.

false autoselfcalibration is off.

6.7.2.4 getCO2()

```
uint16_t __W_SCD30::getCO2 ( )
```

Returns the read CO2 ppm value.

Returns

uint16_t the CO2 in ppm.

6.7.2.5 getFirmwareVersion()

```
bool __W_SCD30::getFirmwareVersion (
            uint16_t & settingVal )
```

Get the Firmware Version.

Parameters

settingVal	object into which the value should be read.

Returns

true get successfull.

false get failure.

6.7.2.6 getForcedRecalibration()

```
bool __W_SCD30::getForcedRecalibration (
            uint16_t & settingVal )
```

Get the Forced Recalibration value.

Parameters

settingVal	object into which the value should be read.

Returns

true get successfull.

false get failure.

6.7.2.7 getHumidity()

```
float __W_SCD30::getHumidity ( )
```

Get the Humidity level.

Returns

float The Humidity level.

6.7.2.8 getInstance()

```
static __W_SCD30 & __W_SCD30::getInstance ( )  [inline], [static]
```

Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Returns

__W_SCD30& the handle to the singleton instance

6.7.2.9 getMeasurementsInterval()

```
bool __W_SCD30::getMeasurementsInterval (
            uint16_t & seconds )
```

Get the Measurements Interval.

Parameters

seconds	object into which the value should be read.

Returns

true get successfull.

false get failure.

6.7.2.10 getTemperature()

```
float __W_SCD30::getTemperature ( )
```

Get the Temperature value.

Returns

float The temperature.

6.7.2.11 getTemperatureOffset()

```
bool __W_SCD30::getTemperatureOffset (
            float & offset )
```

Get the Temperature Offset.

Parameters

offset	object into which the value should be read.

Returns

true get successfull.

false get failure.

6.7.2.12 init() [1/2]

```
ERR_Type __W_SCD30::init ( )  [virtual]
```

Initializes the SCD30 object. Should only be called once. This function initializes the SCD30 object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

6.7.2.13 init() [2/2]

```
ERR_Type __W_SCD30::init (
            bool autoCalibrate )
```

Init function if autoCalibrate is set to true, this will activate the sensors autocalibrate feature. Please see section 1.3.6 of the SCD30 datasheet: "When activated for the first time a period of minimum 7 days is needed so that the algorithm can find its initial parameter set for ASC. The sensor has to be exposed to fresh air for at least 1 hour every day. Also during that period, the sensor may not be disconnected from the power supply, otherwise the procedure to find calibration parameters is aborted and has to be restarted from the beginning. The successfully calculated parameters are stored in non-volatile memory of the SCD30 having the effect that after a restart the previously found parameters for ASC are still present. ".

Parameters

autoCalibrate	turns on auto calibrate.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code

6.7.2.14 read()

```
SCD30_DATA __W_SCD30::read ( )
```

Reads the SCD30 sensor.

Returns

SCD30_DATA Struct containing the read SCD30 Data.

6.7.2.15 setAltitudeCompensation()

```
void __W_SCD30::setAltitudeCompensation (
            uint16_t altitude )
```

Set the Altitude Compensation Set altitude of the sensor in m, stored in non-volatile memory of SCD30.

Parameters

altitude	Altitude of the sensor in m.

6.7.2.16 setAmbientPressure()

```
void __W_SCD30::setAmbientPressure (
            uint16_t Pressure )
```

Set the Ambient Pressure. Current ambient pressure in mBar: 700 to 1200, will overwrite altitude compensation.

Parameters

Pressure	Current ambient pressure in mBar: 700 to 1200

6.7.2.17 setMeasurementsInterval()

```
void __W_SCD30::setMeasurementsInterval (
            uint16_t seconds )
```

Set the Measurements Interval. Change number of seconds between measurements: 2 to 1800 (30 minutes), stored in non-volatile memory of SCD30.

Parameters

seconds	

6.7.2.18 setTemperatureOffset()

```
void __W_SCD30::setTemperatureOffset (
            float offset )
```

Set the Temperature Offset Optionally we can set temperature offset to ℃, stored in non-volatile memory of SCD30.

Parameters

offset	temperature offset.

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

- src/Wrappers/Sensors/SCD30/__W_SCD30.h
- src/Wrappers/Sensors/SCD30/__W_SCD30.cpp

6.8 __W_SD Class Reference

Singleton SD module.

`#include <__W_SD.h>`

Inheritance diagram for __W_SD:

Public Member Functions

- **__W_SD** (__W_SD const &)=delete
- void **operator=** (__W_SD const &)=delete
- ERR_Type init ()

  Initializes the SD object. Should only be called once. This function initializes the SD object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- ERR_Type listDir (const char *dirname, uint8_t levels)

  lists the directories in given path.
- ERR_Type createDir (const char *path)

  Create a directory at the given path. /path/to/dir will create directory 'dir' in the folder 'to'.
- ERR_Type removeDir (const char *path)

  Removes the directory at the given path.
- ERR_Type readFile (const char *path, char *buffer)

  Reads the file at the given path into the passed buffer.
- ERR_Type writeFile (const char *path, const char *message)

  rewrites the message to the given file.
- ERR_Type appendFile (const char *path, const char *message)

  Adds the message at the end of the file.
- ERR_Type renameFile (const char *path1, const char *path2)

  renames the file to the given name. e.g. "/path/to/file" "/path/to/file2" rewrites "file" to "file2".
- ERR_Type deleteFile (const char *path)

  deletes the file at the given path.
- ERR_Type testFileIO (const char *path)

  tests the IO functionality of the SDcard.

Static Public Member Functions

- static __W_SD & getInstance ()

  Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.8.1 Detailed Description

Singleton SD module.

6.8.2 Member Function Documentation

6.8.2.1 appendFile()

```
ERR_Type __W_SD::appendFile (
            const char * path,
            const char * message )
```

Adds the message at the end of the file.

Parameters

path	the path to the file.
message	the message to be added to the file.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.8.2.2 createDir()

```
ERR_Type __W_SD::createDir (
            const char * path )
```

Create a directory at the given path. /path/to/dir will create directory 'dir' in the folder 'to'.

Parameters

path	the path to the directory.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.8.2.3 deleteFile()

```
ERR_Type __W_SD::deleteFile (
            const char * path )
```

deletes the file at the given path.

Parameters

path	path to the file.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.8.2.4 getInstance()

```
static __W_SD & __W_SD::getInstance ( )  [inline], [static]
```

Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Returns

__W_SD& the handle to the singleton instance

6.8.2.5 init()

```
ERR_Type __W_SD::init ( )  [virtual]
```

Initializes the SD object. Should only be called once. This function initializes the SD object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

6.8.2.6 listDir()

```
ERR_Type __W_SD::listDir (
            const char * dirname,
            uint8_t levels )
```

lists the directories in given path.

Parameters

dirname	the path to the directory.
levels	the depth of the tree search.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.8.2.7 readFile()

```
ERR_Type __W_SD::readFile (
            const char * path,
            char * buffer )
```

Reads the file at the given path into the passed buffer.

Parameters

path	the path to the file.
buffer	the buffer that is read into.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.8.2.8 removeDir()

```
ERR_Type __W_SD::removeDir (
            const char * path )
```

Removes the directory at the given path.

Parameters

path	the path to the directory.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.8.2.9 renameFile()

```
ERR_Type __W_SD::renameFile (
            const char * path1,
            const char * path2 )
```

renames the file to the given name. e.g. "/path/to/file" "/path/to/file2" rewrites "file" to "file2".

Parameters

path1	the original path to the file.
path2	the new path to the file.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.8.2.10 testFileIO()

```
ERR_Type __W_SD::testFileIO (
            const char * path )
```

tests the IO functionality of the SDcard.

Parameters

path	to a file to test.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.8.2.11 writeFile()

```
ERR_Type __W_SD::writeFile (
            const char * path,
            const char * message )
```

rewrites the message to the given file.

Parameters

path	the path to the file.
message	the message to be written to the file.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

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

- src/Wrappers/SD/__W_SD.h
- src/Wrappers/SD/__W_SD.cpp

6.9 __W_SM_UART_4L Class Reference

Singleton LDS module.

```
#include <__W_SMUART_4L.h>
```

Inheritance diagram for __W_SM_UART_4L:

Public Member Functions

- **__W_SM_UART_4L** (__W_SM_UART_4L const &)=delete
- void **operator=** (__W_SM_UART_4L const &)=delete
- ERR_Type init ()

  Initializes the LDS object. Should only be called once. This function initializes the LDS object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- ERR_Type read (PM25_AQI_Data &data)

  Reads the LDS sensor data into the data object.

Static Public Member Functions

- static __W_SM_UART_4L & getInstance ()

  Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.9.1 Detailed Description

Singleton LDS module.

6.9.2 Member Function Documentation

6.9.2.1 getInstance()

```
static __W_SM_UART_4L & __W_SM_UART_4L::getInstance ( )  [inline], [static]
```

Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Returns

__W_SM_UART_4L& the handle to the singleton instance

6.9.2.2 init()

```
ERR_Type __W_SM_UART_4L::init ( )  [virtual]
```

Initializes the LDS object. Should only be called once. This function initializes the LDS object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

6.9.2.3 read()

```
ERR_Type __W_SM_UART_4L::read (
            PM25_AQI_Data & data )
```

Reads the LDS sensor data into the data object.

Parameters

data	The PM25_AQI_Data object into which the data should be read.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

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

- src/Wrappers/Sensors/DustSensor/__W_SMUART_4L.h
- src/Wrappers/Sensors/DustSensor/__W_SMUART_4L.cpp

6.10 __W_TSL2591 Class Reference

Singleton TSL2591 module.

`#include <__W_TSL2591.h>`

Inheritance diagram for __W_TSL2591:

Public Member Functions

- **__W_TSL2591** (__W_TSL2591 const &)=delete
- void **operator=** (__W_TSL2591 const &)=delete
- ERR_Type init ()

  Initializes the TSL2591 object. Should only be called once. This function initializes the TSL2591 object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- void **displaySensorDetails** ()

  Displays the sensor details to the SD card and Serial.
- uint16_t getLuminosity (uint8_t spectrum=TSL2591_VISIBLE)

  Get the Luminosity value.
- TSL2591_DATA getFullLuminosity ()

  Read the full luminosity.
- float getLux (uint16_t full, uint16_t ir)

  Calculates the visible Lux based on the two light sensors.

Static Public Member Functions

- static __W_TSL2591 & getInstance ()

  Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.10.1 Detailed Description

Singleton TSL2591 module.

6.10.2 Member Function Documentation

6.10.2.1 getFullLuminosity()

```
TSL2591_DATA __W_TSL2591::getFullLuminosity ( )
```

Read the full luminosity.

Returns

TSL2591_DATA Struct containing all luminosity data.

6.10.2.2 getInstance()

```
static __W_TSL2591 & __W_TSL2591::getInstance ( )  [inline], [static]
```

Get the singleton instance of the class This function makes sure that only one instance is created and accessible during runtime.

Returns

__W_TSL2591& the handle to the singleton instance

6.10.2.3 getLuminosity()

```
uint16_t __W_TSL2591::getLuminosity (
            uint8_t spectrum = TSL2591_VISIBLE )
```

Get the Luminosity value.

Parameters

spectrum	Spectrum to be read. other values are: TSL2591_FULLSPECTRUM or TSL2591_INFRARED.

Returns

uint16_t raw 16-bit ADC value

6.10.2.4 getLux()

```
float __W_TSL2591::getLux (
            uint16_t full,
            uint16_t ir )
```

Calculates the visible Lux based on the two light sensors.

Parameters

full	Data from channel 0 (IR+Visible)
ir	Data from channel 1 (IR)

Returns

Lux, based on AMS coefficients (or $<$ 0 if overflow)

6.10.2.5 init()

```
ERR_Type __W_TSL2591::init ( )  [virtual]
```

Initializes the TSL2591 object. Should only be called once. This function initializes the TSL2591 object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

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

- src/Wrappers/Sensors/TSL2591/__W_TSL2591.h
- src/Wrappers/Sensors/TSL2591/__W_TSL2591.cpp

6.11 AmbimateData Struct Reference

struct containing the ambimate data.

`#include <__W_Ambimate.h>`

Public Attributes

- union {
  uint8_t status
  struct {
  uint8_t MOT_EVENT: 1
  uint8_t AUD_EVENT: 1
  uint8_t **__pad0__**: 5
  uint8_t PIR_EVENT: 1
  }
  };

- float temperatureC
- float Humidity
- float batVolts
- uint16_t light
- uint16_t audio
- uint16_t eco2_ppm
- uint16_t voc_ppm

6.11.1 Detailed Description

struct containing the ambimate data.

6.11.2 Member Data Documentation

6.11.2.1 AUD_EVENT

`uint8_t AmbimateData::AUD_EVENT`

True when an Audio event has occured.

6.11.2.2 audio

`uint16_t AmbimateData::audio`

Audio level.

6.11.2.3 batVolts

```
float AmbimateData::batVolts
```

Battery voltage.

6.11.2.4 eco2_ppm

```
uint16_t AmbimateData::eco2_ppm
```

equivalent CO2 ppm.

6.11.2.5 Humidity

```
float AmbimateData::Humidity
```

Humidity.

6.11.2.6 light

```
uint16_t AmbimateData::light
```

Light level.

6.11.2.7 MOT_EVENT

```
uint8_t AmbimateData::MOT_EVENT
```

True when an motion event has occured.

6.11.2.8 PIR_EVENT

```
uint8_t AmbimateData::PIR_EVENT
```

True when a PIR event has occured.

6.11.2.9 status

```
uint8_t AmbimateData::status
```

the status full byte.

6.11.2.10 temperatureC

```
float AmbimateData::temperatureC
```

Temperature in C.

6.11.2.11 voc_ppm

```
uint16_t AmbimateData::voc_ppm
```

VOC ppm.

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

- src/Wrappers/Sensors/Ambimate/__W_Ambimate.h

6.12 ColorSpectrum Struct Reference

Struct containing the read colorspectrum values.

```
#include <__W_AS726X.h>
```

Public Attributes

- float **Violet**

  Measured violet color spectrum value.
- float **Blue**

  Measured blue color spectrum value.
- float **Green**

  Measured green color spectrum value.
- float **Yellow**

  Measured yellow color spectrum value.
- float **Orange**

  Measured orange color spectrum value.
- float **Red**

  Measured red color spectrum value.

6.12.1 Detailed Description

Struct containing the read colorspectrum values.

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

- src/Wrappers/Sensors/AS7262/__W_AS726X.h

6.13 iSingleton Class Reference

Singleton template this doesn't actually implement anything, but this is more for keeping track if a class is a singleton. Should assure that only one instance of the class can exist at a time.

`#include <Singleton.h>`

Inheritance diagram for iSingleton:

6.13.1 Detailed Description

Singleton template this doesn't actually implement anything, but this is more for keeping track if a class is a singleton. Should assure that only one instance of the class can exist at a time.

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

- src/Wrappers/Singleton/Singleton.h

6.14 Logger Class Reference

Logger singleton class. Logger is implemented as a Singleton to prevent two simultaneous logger instances from accessing the serial or sd.

`#include <Logger.h>`

Inheritance diagram for Logger:

Public Member Functions

- **Logger** (Logger const &)=delete
- void **operator=** (Logger const &)=delete
- ERR_Type init ()

  Initializes the Logger object. Should only be called once. This function initializes the Logger object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- template<typename T >
  void print (T value, LogLevel LL=LogLevel::__PREV, LogType LT=LogType::Serial_SD)

  Prints the given value to the specified output with the given loglevel. To end the log entry call Logger::println().
- template<typename T >
  void println (T value, LogLevel LL=LogLevel::__PREV, LogType LT=LogType::Serial_SD)

  Prints the given value to the specified output with the given loglevel and ends the line. Call this function when a log entry ends. If you need multiple prints in the same log entry use Logger::print().
- void print (char *s, LogLevel LL=LogLevel::__PREV, LogType LT=LogType::Serial_SD)

  Prints the given value to the specified output with the given loglevel. To end the log entry call Logger::println().
- void println (char *s, LogLevel LL=LogLevel::__PREV, LogType LT=LogType::Serial_SD)

  Prints the given value to the specified output with the given loglevel and ends the line. Call this function when a log entry ends. If you need multiple prints in the same log entry use Logger::print().
- void write (char s, LogLevel LL=LogLevel::__PREV, LogType LT=LogType::Serial_SD)

  writes the given character to the specified output with the given loglevel. Use this with loglevel DataDump and Logger::dataDumpEnd(). To end the log entry call Logger::println().
- void breakLine (LogType LT)

  adds a linebreak to a log entry without ending it.
- void dataDumpEnd (LogType LT)

  End the started datadump entry.

Static Public Member Functions

- static Logger & getInstance ()

  Get the Instance of the Logger object. This function makes sure that only one instance is created and accessible during runtime.

Additional Inherited Members

6.14.1 Detailed Description

Logger singleton class. Logger is implemented as a Singleton to prevent two simultaneous logger instances from accessing the serial or sd.

6.14.2 Member Function Documentation

6.14.2.1 breakLine()

```
void Logger::breakLine (
            LogType LT )
```

adds a linebreak to a log entry without ending it.

Parameters

LT	The Loglevel.

6.14.2.2 dataDumpEnd()

```
void Logger::dataDumpEnd (
            LogType LT )
```

End the started datadump entry.

Parameters

LT	The Loglevel.

6.14.2.3 getInstance()

```
static Logger & Logger::getInstance ( )  [inline], [static]
```

Get the Instance of the Logger object. This function makes sure that only one instance is created and accessible during runtime.

Returns

Logger& The singleton instance of the object.

6.14.2.4 init()

```
ERR_Type Logger::init ( )  [virtual]
```

Initializes the Logger object. Should only be called once. This function initializes the Logger object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

Implements __iW_Module.

6.14.2.5 print() [1/2]

```
void Logger::print (
            char * s,
            LogLevel LL = LogLevel::__PREV,
            LogType LT = LogType::Serial_SD )
```

Prints the given value to the specified output with the given loglevel. To end the log entry call Logger::println().

Template Parameters

T	the datatype of the passed value. Should be convertable to a string.

Parameters

s	the string to be printed.
LL	the log level.
LT	the output to log to.

See also

Logger::println()
Logger::breakLine()

6.14.2.6 print() [2/2]

```
template<typename T >
void Logger::print (
```

```
            T value,
            LogLevel LL = LogLevel::__PREV,
            LogType LT = LogType::Serial_SD )  [inline]
```

Prints the given value to the specified output with the given loglevel. To end the log entry call Logger::println().

Template Parameters

T	the datatype of the passed value. Should be convertable to a string.

Parameters

value	the value to be printed.
LL	the log level.
LT	the output to log to.

See also

Logger::println()

Logger::breakLine()

6.14.2.7 println() [1/2]

```
void Logger::println (
            char * s,
            LogLevel LL = LogLevel::__PREV,
            LogType LT = LogType::Serial_SD )
```

Prints the given value to the specified output with the given loglevel and ends the line. Call this function when a log entry ends. If you need multiple prints in the same log entry use Logger::print().

Template Parameters

T	the datatype of the passed value. Should be convertable to a string.

Parameters

s	the value to be printed.
LL	the log level.
LT	the output to log to.

See also

Logger::print()

Logger::breakLine()

6.14.2.8 println() [2/2]

```
template<typename T >
void Logger::println (
            T value,
            LogLevel LL = LogLevel::__PREV,
            LogType LT = LogType::Serial_SD )  [inline]
```

Prints the given value to the specified output with the given loglevel and ends the line. Call this function when a log entry ends. If you need multiple prints in the same log entry use Logger::print().

Template Parameters

T	the datatype of the passed value. Should be convertable to a string.

Parameters

value	the value to be printed.
LL	the log level.
LT	the output to log to.

See also

Logger::print()

Logger::breakLine()

6.14.2.9 write()

```
void Logger::write (
            char s,
            LogLevel LL = LogLevel::__PREV,
            LogType LT = LogType::Serial_SD )
```

writes the given character to the specified output with the given loglevel. Use this with loglevel DataDump and Logger::dataDumpEnd(). To end the log entry call Logger::println().

Template Parameters

T	the datatype of the passed value. Should be convertable to a string.

Parameters

s	the value to be printed.
LL	the log level.
LT	the output to log to.

See also

Logger::print()

Logger::println()

Logger::breakLine()

Logger::dataDumpEnd()

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

- src/Logger/Logger.h
- src/Logger/Logger.cpp

6.15 RTC_DATE_TIME Struct Reference

Struct containing the date and time of when the read() function was called.

`#include <__W_RTC.h>`

Public Attributes

- uint8_t Day
- uint8_t Month
- uint16_t Year
- uint8_t Hour
- uint8_t Minute
- uint8_t Second

6.15.1 Detailed Description

Struct containing the date and time of when the read() function was called.

6.15.2 Member Data Documentation

6.15.2.1 Day

`uint8_t RTC_DATE_TIME::Day`

Current Day.

6.15.2.2 Hour

`uint8_t RTC_DATE_TIME::Hour`

Current Hour.

6.15.2.3 Minute

`uint8_t RTC_DATE_TIME::Minute`

Current Minute.

6.15.2.4 Month

`uint8_t RTC_DATE_TIME::Month`

Current Month.

6.15.2.5 Second

`uint8_t RTC_DATE_TIME::Second`

Current Second.

6.15.2.6 Year

`uint16_t RTC_DATE_TIME::Year`

Current Year.

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

- src/Wrappers/RTC/__W_RTC.h

6.16 SBox Class Reference

SBox class containing handles to every sensor on the PCB.

`#include <Sbox.h>`

Public Member Functions

- ERR_Type init ()

  Initializes the SBox object. Should only be called once. This function initializes the SBox object. It has a check build in to see if this function has already been called before. If so it will just return 0.
- RTC_DATE_TIME getTime ()

  Get the a date_time struct.
- AmbimateData getAmbimateData ()

  Get the Ambimate Data.
- ERR_Type getColorSpectrum (ColorSpectrum &CS)

  Get the Color Spectrum data.
- ERR_Type getLDSData (PM25_AQI_Data &Buffer)

  Get the Laser Dust Sensor data.
- int getMax4466 ()

  Get the Max4466 data.
- float getO2 ()

  Get the O2 value.
- SCD30_DATA getSCD30Data ()

  Get the SCD30 data.
- TSL2591_DATA getTSL2591Data ()

  Get the TSL2591 data.

6.16.1 Detailed Description

SBox class containing handles to every sensor on the PCB.

6.16.2 Member Function Documentation

6.16.2.1 getAmbimateData()

```
AmbimateData SBox::getAmbimateData ( )
```

Get the Ambimate Data.

Returns

AmbimateData struct containing the read ambimate data.

6.16.2.2 getColorSpectrum()

```
ERR_Type SBox::getColorSpectrum (
            ColorSpectrum & CS )
```

Get the Color Spectrum data.

Parameters

CS	ColorSpectrum struct buffer.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.16.2.3 getLDSData()

```
ERR_Type SBox::getLDSData (
            PM25_AQI_Data & Buffer )
```

Get the Laser Dust Sensor data.

Parameters

Buffer	PM25_AQI_Data struct buffer.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

6.16.2.4 getMax4466()

```
int SBox::getMax4466 ( )
```

Get the Max4466 data.

Returns

int ADC value of the Max4466 data.

6.16.2.5 getO2()

```
float SBox::getO2 ( )
```

Get the O2 value.

Returns

float the read O2 value.

6.16.2.6 getSCD30Data()

```
SCD30_DATA SBox::getSCD30Data ( )
```

Get the SCD30 data.

Returns

SCD30_DATA struct containing the read SCD30 data.

6.16.2.7 getTime()

```
RTC_DATE_TIME SBox::getTime ( )
```

Get the a date_time struct.

Returns

RTC_DATE_TIME struct containing current date and time.

6.16.2.8 getTSL2591Data()

```
TSL2591_DATA SBox::getTSL2591Data ( )
```

Get the TSL2591 data.

Returns

TSL2591_DATA struct containing the TSL2591 data.

6.16.2.9 init()

```
ERR_Type SBox::init ( )
```

Initializes the SBox object. Should only be called once. This function initializes the SBox object. It has a check build in to see if this function has already been called before. If so it will just return 0.

Returns

ERR_Type returns SUCCESS on succesfull exit. Else it will return an error code.

See also

ERR_Type

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

- src/Sbox/Sbox.h
- src/Sbox/Sbox.cpp

6.17 SCD30_DATA Struct Reference

Struct containg the SCD30 Data.

```
#include <__W_SCD30.h>
```

Public Attributes

- uint16_t CO2
- float Temperature
- float Humidity

6.17.1 Detailed Description

Struct containg the SCD30 Data.

6.17.2 Member Data Documentation

6.17.2.1 CO2

```
uint16_t SCD30_DATA::CO2
```

CO2 ppm value

6.17.2.2 Humidity

```
float SCD30_DATA::Humidity
```

Humidity value

6.17.2.3 Temperature

```
float SCD30_DATA::Temperature
```

Temperature value

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

- src/Wrappers/Sensors/SCD30/__W_SCD30.h

6.18 TSL2591_DATA Struct Reference

Struct containing the TSL2591 data.

```
#include <__W_TSL2591.h>
```

Public Attributes

- uint16_t **visible**

  Lux values of visible spectrum range.
- uint16_t **ir**

  Lux values of infra red spectrum range.
- uint16_t **full**

  Lux values of full spectrum range.

6.18.1 Detailed Description

Struct containing the TSL2591 data.

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

- src/Wrappers/Sensors/TSL2591/__W_TSL2591.h

Chapter 7

File Documentation

7.1 src/Defines/Defines.h File Reference

Defines file used for global precompile settings.

Macros

- #define **SenseBox_VERSION** "0.0.1"

  Version of the Sensebox software A change in the first version indicator is a major version change which might not be compatible with older major versions A change in the second version indicator is a minor version change which should be compatible with other of the same major versions A change in the third version indicator is a tweak version change, this will include mostly small bug fixes

- #define DEBUGLEVEL 3

  Used by the source code to indicate what to debug to serial monitor.

- #define LOGLEVEL 3

  Used by the source code to indicate what to log to the SD card.

Enumerations

- enum ERR_Type {
  SUCCESS = 0 , READ_SUCCESS = 0 , ALREADY_INITIALIZED = 0 , ERROR = 1 ,
  READ_ERROR = 1 , READ_FAIL = 1 , NOT_INITIALIZED , SD_NOT_INIT ,
  MOD_INIT_ERR , RTC_BEGIN_ERR , SD_BEGIN_ERR , NO_SD_CARD ,
  SD_CARDTYPE_NONE , SD_DIR_OPEN_FAIL , SD_NOT_A_DIR , SD_MKDIR_FAIL ,
  SD_RMDIR_FAIL , SD_FILE_OPEN_FAIL , SD_WRITE_FAIL , SD_APP_FAIL ,
  SD_RENAME_FAIL , SD_RM_FAIL , AMBI_I2C_INIT_ERR , AS726X_BEGIN_ERR ,
  AS726X_DATA_NOT_READY , NO_LDS_SENSOR , SCD30_BEGIN_ERR , TSL_BEGIN_ERR }

  List of function return errors.

7.1.1 Detailed Description

Defines file used for global precompile settings.

Author

Imre Korf

Version

0.1

Date

2021-11-29

Copyright

Copyright (c) 2021

7.2 Defines.h

Go to the documentation of this file.

```
1
11 #pragma once
12
24 #define SenseBox_VERSION "0.0.1"
25
36 #define DEBUGLEVEL 3
48 #define LOGLEVEL 3
58 enum ERR_Type {
59     //==== Successful exits ====//
60
62     SUCCESS = 0,
64     READ_SUCCESS = 0,
66     ALREADY_INITIALIZED = 0,
67
68     //==== Errors ====//
69
71     ERROR = 1,
73     READ_ERROR = 1,
75     READ_FAIL = 1,
77     NOT_INITIALIZED,
78
79     // Logger Errors
81     SD_NOT_INIT,
82
83     // SBOX Errors
85     MOD_INIT_ERR,
86
87     // RTC Errors
89     RTC_BEGIN_ERR,
90
91     // SD Errors
93     SD_BEGIN_ERR,
95     NO_SD_CARD,
97     SD_CARDTYPE_NONE,
99     SD_DIR_OPEN_FAIL,
101     SD_NOT_A_DIR,
103     SD_MKDIR_FAIL,
105     SD_RMDIR_FAIL,
107     SD_FILE_OPEN_FAIL,
109     SD_WRITE_FAIL,
111     SD_APP_FAIL,
113     SD_RENAME_FAIL,
115     SD_RM_FAIL,
116
117     // Ambimate Errors
```

```
119     AMBI_I2C_INIT_ERR,
120
121     // AS7226X Errors
123     AS726X_BEGIN_ERR,
125     AS726X_DATA_NOT_READY,
126
127     // LDS Errors
129     NO_LDS_SENSOR,
130
131     // SCD30 Errors
133     SCD30_BEGIN_ERR,
134
135     // TSL2591 Errors
137     TSL_BEGIN_ERR,
138
139 };
```

7.3 src/Logger/Logger.h File Reference

Logger class that handles all the types of logging.

```
#include <Arduino.h>
#include "../Wrappers/SD/__W_SD.h"
#include "../Wrappers/Singleton/Singleton.h"
#include "../Defines/Defines.h"
```

Classes

- class Logger

  Logger singleton class. Logger is implemented as a Singleton to prevent two simultaneous logger instances from accessing the serial or sd.

Enumerations

- enum class LogLevel {
  LogLevel::Error , LogLevel::Warning , LogLevel::Info , LogLevel::DataDump ,
  LogLevel::__PREV }

  Enum values that indicate the log level of the message.
- enum class LogType { LogType::SD = 0b01 , LogType::Serial = 0b10 , LogType::Serial_SD = 0b11 }

  Enum values that indicate to which logging device should be logged.

7.3.1 Detailed Description

Logger class that handles all the types of logging.

Author

Imre Korf

Version

0.1

Date

2021-11-29

Copyright

7.4 Logger.h

Go to the documentation of this file.

```
1
11 #pragma once
12
13 #include <Arduino.h>
14 #include "../Wrappers/SD/__W_SD.h"
15 #include "../Wrappers/Singleton/Singleton.h"
16 #include "../Defines/Defines.h"
17
29 enum class LogLevel {
31     Error,
33     Warning,
35     Info,
37     DataDump,
39     __PREV
40 };
41
51 enum class LogType {
53     SD            = 0b01,
55     Serial    = 0b10,
57     Serial_SD  = 0b11
58 };
61 // Logger is the master of Serial1 aswell as the SD card.
62
67 class Logger : public __iW_Module, public iSingleton{
68 private:
72     __W_SD* sdhandle;
73
77     bool SER_line_ended = false;
81     bool SD_line_ended  = false;
82
86     LogLevel PREV_LL;
90     const char* filepath = "";
94     uint8_t curr_day = 32;
95
105     void SER_print_LL_type(LogLevel LL = LogLevel::__PREV);
114     void SD_print_LL_type(LogLevel LL = LogLevel::__PREV);
115
122     virtual bool checkInitialized();
123
124 public:
130     static Logger& getInstance(){
131         static Logger Instance;   // will only be destroyed on program exit.
132         return Instance;
133     }
134
135 private:
136     // remove access to the constructor of logger.
137     Logger(){}
138 public:
139     // Assure that only one instance can exist by removing copy and assign functions.
140     Logger(Logger const&)           = delete;   // delete copy constructor.
141     void operator=(Logger const&) = delete;   // remove assignment operator.
142
149     ERR_Type init();
150
151     // print statements.
152     // ----------
153
164     template<typename T>
165     void print(T value, LogLevel LL = LogLevel::__PREV, LogType LT = LogType::Serial_SD){
166         print(String(value).c_str(), LL, LT);
167     }
168
179     template<typename T>
180     void println(T value, LogLevel LL = LogLevel::__PREV, LogType LT = LogType::Serial_SD){
181         println(String(value).c_str(), LL, LT);
182     }
183
194     void print(char* s, LogLevel LL = LogLevel::__PREV, LogType LT = LogType::Serial_SD);
205     void println(char* s, LogLevel LL = LogLevel::__PREV, LogType LT = LogType::Serial_SD);
218     void write(char s, LogLevel LL = LogLevel::__PREV, LogType LT = LogType::Serial_SD);
223     void breakLine(LogType LT);
228     void dataDumpEnd(LogType LT);
229 };
```

7.5 src/Sbox/Sbox.h File Reference

Contains the highest level SenseBox wrapper class.

```
#include <Arduino.h>
#include "../Wrappers/Sensors/Ambimate/__W_Ambimate.h"
#include "../Wrappers/Sensors/AS7262/__W_AS726X.h"
#include "../Wrappers/Sensors/DustSensor/__W_SMUART_4L.h"
#include "../Wrappers/Sensors/MAX4466/__W_MAX4466.h"
#include "../Wrappers/Sensors/MIX8410/__W_MIX8410.h"
#include "../Wrappers/Sensors/SCD30/__W_SCD30.h"
#include "../Wrappers/Sensors/TSL2591/__W_TSL2591.h"
#include "../Wrappers/RTC/__W_RTC.h"
```

Classes

- class SBox

  SBox class containing handles to every sensor on the PCB.

7.5.1 Detailed Description

Contains the highest level SenseBox wrapper class.

Author

Imre Korf

Version

0.1

Date

2021-11-23

Copyright

Copyright (c) 2021

7.6 Sbox.h

Go to the documentation of this file.

```
1
11 #pragma once
12
13 /*
14 SCD30: CO2
15 MIX8410: O2
16 TSL2591: LI
17 AS7262: Spectrum
18 2314291-2: VOC + extra
19 MAX4466: Sound
20 SM-UART-04L: Fijnstof
21 */
22
23
24 /*
25 All properties:
26 O2
```

```
27 CO2
28 Humidity
29 Temp
30 Light Intensity (LUX)
31 Light Spectrum (6band and intensity)
32 VOC
33 eCO2
34 dB + Hz?
35 Movement (flag)
36 Dust
37 */
38
39 #include <Arduino.h>
40
41 #include "../Wrappers/Sensors/Ambimate/__W_Ambimate.h"
42 #include "../Wrappers/Sensors/AS7262/__W_AS726X.h"
43 #include "../Wrappers/Sensors/DustSensor/__W_SMUART_4L.h"
44 #include "../Wrappers/Sensors/MAX4466/__W_MAX4466.h"
45 #include "../Wrappers/Sensors/MIX8410/__W_MIX8410.h"
46 #include "../Wrappers/Sensors/SCD30/__W_SCD30.h"
47 #include "../Wrappers/Sensors/TSL2591/__W_TSL2591.h"
48 #include "../Wrappers/RTC/__W_RTC.h"
49
53 class SBox {
54 private:
55
57     __W_Ambimate    *Ambimate;
59     __W_AS726X       *AS7262;
61     __W_SM_UART_4L   *LDS;
63     __W_MAX4466      *MAX4466;
65     __W_MIX8410      *MIX8410;
67     __W_SCD30       *SCD30;
69     __W_TSL2591      *TSL2591;
71     __W_RTC          *RTC;
72
73 public:
74
81     ERR_Type init();
82
88     RTC_DATE_TIME getTime();
89
95     AmbimateData getAmbimateData();
103     ERR_Type     getColorSpectrum(ColorSpectrum& CS);
111     ERR_Type    getLDSData(PM25_AQI_Data& Buffer);
117     int          getMax4466();
123     float        getO2();
129     SCD30_DATA     getSCD30Data();
135     TSL2591_DATA getTSL2591Data();
136 };
```

7.7 src/Wrappers/__W_Module/__iW_Module.h File Reference

Module interface for intergrating sensors and other modules.

```
#include <Arduino.h>
#include "../../Defines/Defines.h"
```

Classes

- class __iW_Module

  Interface for a hardware module hardware module should be implemented as a singleton to ensure that there won't be two processes accessing the same hardware at the same time Singleton format: https://stackoverflow.com/a/1008289.

7.7.1 Detailed Description

Module interface for intergrating sensors and other modules.

Author

Imre Korf

Version

0.1

Date

2021-11-29

Copyright

7.8 __iW_Module.h

Go to the documentation of this file.

```
1
11 #pragma once
12
13 #include <Arduino.h>
14 #include "../../Defines/Defines.h"
15
22 class __iW_Module {
23 protected:
28     bool Initialized = false;
37     virtual bool checkInitialized(){return Initialized;}
38
39 public:
46     virtual ERR_Type init() = 0;
47 };
```

7.9 src/Wrappers/RTC/__W_RTC.h File Reference

Wrapper for the PC8563 RTC.

```
#include "../__W_Module/__iW_Module.h"
#include "../Singleton/Singleton.h"
#include <RTC.h>
```

Classes

- struct RTC_DATE_TIME

  Struct containing the date and time of when the read() function was called.
- class __W_RTC

  Singleton RTC Module.

7.9.1 Detailed Description

Wrapper for the PC8563 RTC.

Author

Imre Korf

Version

0.1

Date

2021-11-30

Copyright

7.10 __W_RTC.h

Go to the documentation of this file.

```
1
11 #pragma once
12
13 #include "../__W_Module/__iW_Module.h"
14 #include "../Singleton/Singleton.h"
15
16 #include <RTC.h>
17
25 struct RTC_DATE_TIME{
27     uint8_t     Day;
29     uint8_t     Month;
31     uint16_t    Year;
33     uint8_t     Hour;
35     uint8_t     Minute;
37     uint8_t     Second;
38 };
44 class __W_RTC : public __iW_Module, public iSingleton{
45 private:
49     PCF8563 RTC;
50
57     virtual bool checkInitialized();
58
59     // remove access to the constructor of __W_RTC.
60     __W_RTC(){}
61
62 public:
68     static __W_RTC& getInstance(){
69         static __W_RTC Instance; // will only be destroyed on program exit.
70         return Instance;
71     }
72 public:
73     // Assure that only one instance can exist by removing copy and assign functions.
74     __W_RTC(__W_RTC const&)           = delete;   // delete copy constructor.
75     void operator=(__W_RTC const&)    = delete;   // remove assignment operator.
76
83     ERR_Type init();
84
89     RTC_DATE_TIME read();
90
95     String stringDateTime();
96
101     String stringTime();
102 };
```

7.11 src/Wrappers/SD/__W_SD.h File Reference

Wrapper for the SD card hardware.

```
#include "../__W_Module/__iW_Module.h"
#include "../Singleton/Singleton.h"
#include <FS.h>
#include <SD.h>
```

Classes

- class __W_SD

  Singleton SD module.

7.11.1 Detailed Description

Wrapper for the SD card hardware.

Author

Imre Korf

Version

0.1

Date

2021-11-30

Copyright

Copyright (c) 2021

7.12 __W_SD.h

Go to the documentation of this file.

```
1
11 #pragma once
12
13
14 #include "../__W_Module/__iW_Module.h"
15 #include "../Singleton/Singleton.h"
16
17 #include <FS.h>
18 #include <SD.h>
19
24 class __W_SD : public __iW_Module, public iSingleton {
25 private:
27     SDFS* ESP_SD;
28
35     virtual bool checkInitialized();
36
37     // remove access to the constructor of __W_SD.
```

```
38     __W_SD(){}
39
40 public:
46     static __W_SD& getInstance(){
47         static __W_SD Instance;   // will only be destroyed on program exit.
48         return Instance;
49     }
50 public:
51     // Assure that only one instance can exist by removing copy and assign functions.
52     __W_SD(__W_SD const&)           = delete;   // delete copy constructor.
53     void operator=(__W_SD const&) = delete;   // remove assignment operator.
54
61     ERR_Type init();
62
71     ERR_Type listDir(const char* dirname, uint8_t levels);
80     ERR_Type createDir(const char* path);
88     ERR_Type removeDir(const char* path);
97     ERR_Type readFile(const char * path, char* buffer);
106     ERR_Type writeFile(const char * path, const char * message);
115     ERR_Type appendFile(const char * path, const char * message);
124     ERR_Type renameFile(const char * path1, const char * path2);
132     ERR_Type deleteFile(const char * path);
133
141     ERR_Type testFileIO(const char * path);
142 };
```

7.13 src/Wrappers/Sensors/Ambimate/__W_Ambimate.h File Reference

Ambimate wrapper for the Ambimate sensor.

```
#include <Wire.h>
#include "../../__W_Module/__iW_Module.h"
```

Classes

- struct AmbimateData

  struct containing the ambimate data.
- class __W_Ambimate

  Singleton ambimate module.

7.13.1 Detailed Description

Ambimate wrapper for the Ambimate sensor.

Author

Imre Korf

Version

0.1

Date

2021-11-29

source: https://maker.pro/esp8266/tutorial/how-to-program-esp32-with-arduino-ide-and-conn

Copyright

7.14 __W_Ambimate.h

Go to the documentation of this file.

```
1
13 #pragma once
14
15 #include <Wire.h>
16 #include "../../__W_Module/__iW_Module.h"
17
24 struct AmbimateData {
25     union {
27         uint8_t status;
28         struct {
30             uint8_t MOT_EVENT : 1; // motion event.
32             uint8_t AUD_EVENT : 1; // audio event.
33             uint8_t           : 5; // padding.
35             uint8_t PIR_EVENT : 1; // PIR event.
36         };
37     };
39     float temperatureC;
41     float Humidity;
43     float batVolts;
45     uint16_t light;
47     uint16_t audio;
49     uint16_t eco2_ppm;
51     uint16_t voc_ppm;
52 };
58 class __W_Ambimate : public __iW_Module, public iSingleton {
59 private:
63     uint8_t opt_sensors;
64
71     virtual bool checkInitialized();
72
73     // remove access to the constructor of __W_Ambimate.
74     __W_Ambimate(){}
75
76 public:
82     static __W_Ambimate& getInstance(){
83         static __W_Ambimate Instance;   // will only be destroyed on program exit.
84         return Instance;
85     }
86     // Assure that only one instance can exist by removing copy and assign functions.
87     __W_Ambimate(__W_Ambimate const&)           = delete;   // delete copy constructor.
88     void operator=(__W_Ambimate const&) = delete;   // remove assignment operator.
89
96     ERR_Type init();
97
103     AmbimateData read();
104
110     uint8_t get_opt_sensors();
111 };
```

7.15 src/Wrappers/Sensors/AS7262/__W_AS726X.cpp File Reference

TSL2591 wrapper for the TSL2591 adafruit library.

```
#include <Arduino.h>
#include "__W_AS726X.h"
#include "../../../Logger/Logger.h"
```

7.15.1 Detailed Description

TSL2591 wrapper for the TSL2591 adafruit library.

Author

Imre Korf

Version

0.1

Date

2021-11-23

Copyright

7.16 __W_AS726X.h

```
1
11 #pragma once
12
13 #include <Adafruit_AS726x.h>
14 #include "../../__W_Module/__iW_Module.h"
15
22 struct ColorSpectrum {
24     float Violet;
26     float Blue;
28     float Green;
30     float Yellow;
32     float Orange;
34     float Red;
35 };
41 class __W_AS726X : public __iW_Module, public iSingleton {
42 private:
44     Adafruit_AS726x AS7262;
45
52     virtual bool checkInitialized();
53
54     // remove access to the constructor of __W_AS726X.
55     __W_AS726X(){}
56 public:
62     static __W_AS726X& getInstance(){
63         static __W_AS726X Instance;   // will only be destroyed on program exit.
64         return Instance;
65     }
66     // Assure that only one instance can exist by removing copy and assign functions.
67     __W_AS726X(__W_AS726X const&)       = delete;   // delete copy constructor.
68     void operator=(__W_AS726X const&) = delete;   // remove assignment operator.
69
76     ERR_Type init();
77
81     void setDrvLed(bool B);
85     void setDrvCurrent(uint8_t V);
89     void indicateLED(bool B);
93     void setIndicateCurrent(uint8_t V);
94
99     void setGain(uint8_t gain);
104     void setIntegrationTime(uint8_t time);
110     void setConversionType(uint8_t type);
111
115     void startMeasurement();
121     bool checkDataReady();
122
127     void getMeasurements(ColorSpectrum* CS);
132     uint8_t getTemperature();
133 };
```

7.17 src/Wrappers/Sensors/DustSensor/__W_SMUART_4L.h File Reference

SM_UART_4L wrapper for the PM25AQI adafruit library.

```
#include <Adafruit_PM25AQI.h>
#include "../../__W_Module/__iW_Module.h"
```

Classes

- class __W_SM_UART_4L

  Singleton LDS module.

7.17.1 Detailed Description

SM_UART_4L wrapper for the PM25AQI adafruit library.

Author

Imre Korf

Version

0.1

Date

2021-11-29

Copyright

Copyright (c) 2021

7.18 __W_SMUART_4L.h

Go to the documentation of this file.

```
1
11 #pragma once
12
13 #include <Adafruit_PM25AQI.h>
14 #include "../../__W_Module/__iW_Module.h"
15
19 class __W_SM_UART_4L : public __iW_Module, public iSingleton {
20 private:
22     Adafruit_PM25AQI aqi;
23
30     virtual bool checkInitialized();
31
32     // remove access to the constructor of __W_SM_UART_4L.
33     __W_SM_UART_4L(){}
34 public:
40     static __W_SM_UART_4L& getInstance(){
41         static __W_SM_UART_4L Instance;   // will only be destroyed on program exit.
42         return Instance;
43     }
44     // Assure that only one instance can exist by removing copy and assign functions.
45     __W_SM_UART_4L(__W_SM_UART_4L const&)    = delete;   // delete copy constructor.
46     void operator=(__W_SM_UART_4L const&) = delete;   // remove assignment operator.
47
54     ERR_Type init();
55
63     ERR_Type read(PM25_AQI_Data& data);
64 };
```

7.19 src/Wrappers/Sensors/MAX4466/__W_MAX4466.h File Reference

MAX4466 wrapper for the MAX4466 sensor.

```
#include "../../__W_Module/__iW_Module.h"
```

Classes

- class __W_MAX4466

  Singleton MAX4466 module.

7.19.1 Detailed Description

MAX4466 wrapper for the MAX4466 sensor.

Author

Imre Korf

Version

0.1

Date

2021-11-23

source: https://blog.yavilevich.com/2016/08/arduino-sound-level-meter-and-spectrum-analy

Copyright

7.20 __W_MAX4466.h

Go to the documentation of this file.

```
1
13 #pragma once
14
15 #include "../../__W_Module/__iW_Module.h"
16
20 class __W_MAX4466 : public __iW_Module, public iSingleton {
21 private:
28     virtual bool checkInitialized();
29
30     // remove access to the constructor of __W_MAX4466.
31     __W_MAX4466(){}
32 public:
38     static __W_MAX4466& getInstance(){
39         static __W_MAX4466 Instance; // will only be destroyed on program exit.
40         return Instance;
41     }
42     // Assure that only one instance can exist by removing copy and assign functions.
43     __W_MAX4466(__W_MAX4466 const&)        = delete;   // delete copy constructor.
44     void operator=(__W_MAX4466 const&)    = delete;   // remove assignment operator.
45
52     ERR_Type init();
53
59     int read();
60
61 };
```

7.21 src/Wrappers/Sensors/MIX8410/__W_MIX8410.h File Reference

MIX8410 wrapper for the MIX8410 sensor.

```
#include "../../__W_Module/__iW_Module.h"
```

Classes

- class __W_MIX8410
  Singleton MIX8410 module.

7.21.1 Detailed Description

MIX8410 wrapper for the MIX8410 sensor.

Author

Imre Korf

Version

0.1

Date

2021-11-23

Copyright

Copyright (c) 2021

7.22 __W_MIX8410.h

Go to the documentation of this file.

```
1
11 #pragma once
12
13 #include "../../__W_Module/__iW_Module.h"
14
18 class __W_MIX8410 : public __iW_Module, public iSingleton {
19
20 private:
24     const float VRefer = 3.3;
28     const int pinAdc   = A0;
29
30     // remove access to the constructor of __W_MIX8410.
31     __W_MIX8410(){}
32 public:
38     static __W_MIX8410& getInstance(){
39         static __W_MIX8410 Instance; // will only be destroyed on program exit.
40         return Instance;
41     }
42     // Assure that only one instance can exist by removing copy and assign functions.
43     __W_MIX8410(__W_MIX8410 const&)       = delete;   // delete copy constructor.
44     void operator=(__W_MIX8410 const&)    = delete;   // remove assignment operator.
45
52     ERR_Type init(){return SUCCESS;}
58     float readConcentration();
64     float readO2Vout();
65 };
```

7.23 src/Wrappers/Sensors/SCD30/__W_SCD30.h File Reference

SCD30 wrapper for the SCD30 sensor.

```
#include <Wire.h>
#include <SparkFun_SCD30_Arduino_Library.h>
#include "../../__W_Module/__iW_Module.h"
```

Classes

- struct SCD30_DATA

  Struct containg the SCD30 Data.
- class __W_SCD30

  Singleton SCD30 module.

7.23.1 Detailed Description

SCD30 wrapper for the SCD30 sensor.

Author

Imre Korf

Version

0.1

Date

2021-11-29

Copyright

7.24 __W_SCD30.h

Go to the documentation of this file.

```
1
11 #pragma once
12
13 #include <Wire.h>
14 #include <SparkFun_SCD30_Arduino_Library.h>
15 #include "../../__W_Module/__iW_Module.h"
16
22 struct SCD30_DATA {
24     uint16_t CO2;
26     float Temperature;
28     float Humidity;
29 };
35 class __W_SCD30 : public __iW_Module, public iSingleton {
36 private:
38     SCD30 airSensor;
```

```
39
43      bool autoCalibrate = false;
50      virtual bool checkInitialized();
51
52      // remove access to the constructor of __W_SCD30
53      __W_SCD30(){}
54  public:
60      static __W_SCD30& getInstance(){
61          static __W_SCD30 Instance; // will only be destroyed on program exit.
62          return Instance;
63      }
64      // Assure that only one instance can exist by removing copy and assign functions.
65      __W_SCD30(__W_SCD30 const&)           = delete;   // delete copy constructor.
66      void operator=(__W_SCD30 const&)    = delete;   // remove assignment operator.
67
73      SCD30_DATA read();
74
88      ERR_Type init(bool autoCalibrate);
89
96      ERR_Type init();
97
104      bool dataAvailable();
105
111      uint16_t getCO2();
117      float getTemperature();
123      float getHumidity();
124
125      // Options:
131      void setMeasurementsInterval(uint16_t seconds);
132
140      bool getMeasurementsInterval(uint16_t& seconds);
141
147      void setAltitudeCompensation(uint16_t altitude);
148
156      bool getAltitudeCompensation(uint16_t& altitude);
157
163      void setAmbientPressure(uint16_t Pressure);
164
170      void setTemperatureOffset(float offset);
171
179      bool getTemperatureOffset(float& offset);
180
187      bool getAutoSelfCalibration();
188
196      bool getForcedRecalibration(uint16_t& settingVal);
197
205      bool getFirmwareVersion(uint16_t& settingVal);
206  };
```

7.25 src/Wrappers/Sensors/TSL2591/__W_TSL2591.h File Reference

TSL2591 wrapper for the TSL2591 adafruit library.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_TSL2591.h>
#include "../../__W_Module/__iW_Module.h"
```

Classes

- struct TSL2591_DATA

  Struct containing the TSL2591 data.
- class __W_TSL2591

  Singleton TSL2591 module.

7.25.1 Detailed Description

TSL2591 wrapper for the TSL2591 adafruit library.

Author

Imre Korf

Version

0.1

Date

2021-11-23

Copyright

7.26 __W_TSL2591.h

Go to the documentation of this file.

```
1
11 #pragma once
12
13 #include <Wire.h>
14 #include <Adafruit_Sensor.h>
15 #include <Adafruit_TSL2591.h>
16
17 #include "../../__W_Module/__iW_Module.h"
18
25 struct TSL2591_DATA {
27     uint16_t visible;
29     uint16_t ir;
31     uint16_t full;
32 };
38 class __W_TSL2591 : public __iW_Module, public iSingleton {
39 private:
41     Adafruit_TSL2591 tsl;
42
49     virtual bool checkInitialized();
50     // remove access to the constructor of __W_TSL2591.
51     __W_TSL2591() : tsl(2591){} // pass in a number for the sensor identifier (for your use later).
52 public:
58     static __W_TSL2591& getInstance(){
59         static __W_TSL2591 Instance; // will only be destroyed on program exit.
60         return Instance;
61     }
62     // Assure that only one instance can exist by removing copy and assign functions.
63     __W_TSL2591(__W_TSL2591 const&)           = delete;   // delete copy constructor.
64     void operator=(__W_TSL2591 const&)   = delete;   // remove assignment operator.
65
72     ERR_Type init();
73
78     void displaySensorDetails();
79
86     uint16_t getLuminosity(uint8_t spectrum = TSL2591_VISIBLE);
92     TSL2591_DATA getFullLuminosity();
93
100     float getLux(uint16_t full, uint16_t ir);
101 };
```

7.27 src/Wrappers/Singleton/Singleton.h File Reference

Singleton template interface.

Classes

- class iSingleton

  Singleton template this doesn't actually implement anything, but this is more for keeping track if a class is a singleton. Should assure that only one instance of the class can exist at a time.

7.27.1 Detailed Description

Singleton template interface.

Author

Imre Korf

Version

0.1

Date

2021-12-06

Copyright

Copyright (c) 2021

7.28 Singleton.h

Go to the documentation of this file.

```
1
12 #pragma once
13
19 class iSingleton{
20 /*
21 public:
22     // make sure that only one instance is created and accessible during runtime
23     static Derived& getInstance(){
24         static Derived Instance;    // will only be destroyed on program exit
25         return Instance;
26     }
27 private:
28     // remove access to the constructor of derived
29     Derived(){}
30 public:
31     // Assure that only one instance can exist by removing copy and assign functions
32     Derived(Derived const&)         = delete;   // delete copy constructor
33     void operator=(Derived const&)  = delete;   // remove assignment operator
34 */
35 };
```

Index