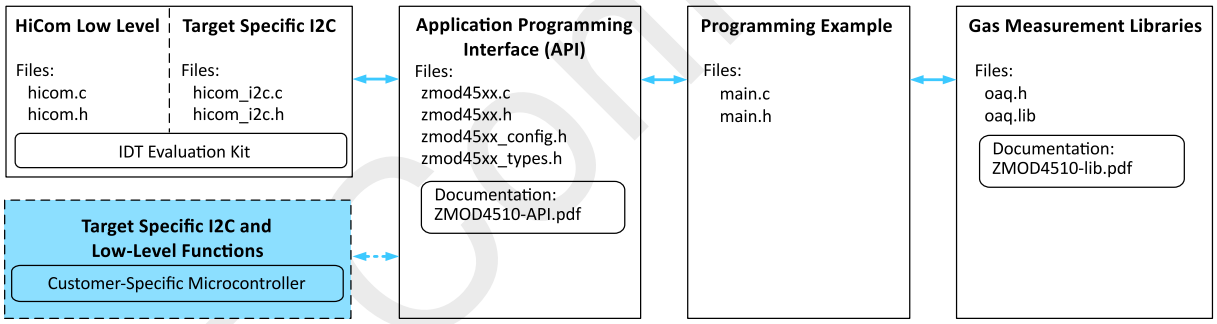
ZMOD45xx Algorithm for OAQ

Contents

Chapter 1

ZMOD4510 Application Programming Interface Overview

This document describes the gas algorithm libraries for the ZMOD45xx Gas Sensor Module for Outdoor Air Quality(OAQ). The figure below shows an overview of the ZMOD4510 API, programming example and libraries. Custom microcontrollers can be used to establish I2C communication. Using the user's own microcontroller requires implementing the user's own target-specific I2C and low-level functions (highlighted in light blue). The following describes in detail the gas measurement libraries of the ZMOD4510 for calculation of Air Quality Index(AQI)∗.



∗The AQI is defined by the US Environmental Protection Agency (EPA).

More information can be found at: https://airnow.gov/index.cfm?action=aqibasics.aqi

Chapter 2

How to Work with the IDT ZMOD4510 Algorithm Library

- Include the header file in the user's program for gas sensor module control; for example:
  `#include "oaq.h"`
- Copy the library file into user's project folder
- Call the intended function in the user's program

Example for zmod4510:

```
#include "oaq.h"

#define D_RISING_M1 0.0
#define D_FALLING_M1 0.13533528

int main() {
    oaq_base_handle_t handle;
    float rmox[15];
    float aqi;
    uint8_t general_purpose[9] = {185, 16, 134, 113, 114, 74, 124, 97, 0}; // 9 bytes trim information read
      from address 0x26
    uint8_t stabilization_samples = 10; // Number of samples to be ignored for stabilization

    // Call oaq_base_init only once at the beginning of measurements
    oaq_base_init(&handle, general_purpose, stabilization_samples);

    // your functionality
    // r_mox measurement

    aqi = calc_oaq(&handle, rmox, RCDA_STRATEGY_ADJ,
      GAS_DETECTION_STRATEGY_AUTO, D_RISING_M1, D_FALLING_M1);

    // your functionality

    return 0;
}
```

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 Handles of the Rcda

Handles of the Rcda.

Macros

- #define RCDA_STRATEGY_FIX (0)
- #define RCDA_STRATEGY_SET (1)
- #define RCDA_STRATEGY_ADJ (2)

6.1.1 Detailed Description

Handles of the Rcda.

6.1.2 Macro Definition Documentation

6.1.2.1 RCDA_STRATEGY_ADJ

```
#define RCDA_STRATEGY_ADJ (2)
```

adjust rcda to follow slow drifts

6.1.2.2 RCDA_STRATEGY_FIX

```
#define RCDA_STRATEGY_FIX (0)
```

hold rcda fixed

6.1.2.3 RCDA_STRATEGY_SET

```
#define RCDA_STRATEGY_SET (1)
```

set the rcda to the actal measurement

6.2 Discriminations of the gases

Discriminations of the gases.

Data Structures

- struct oaq_base_handle_t

  Variables that describe the sensor or the algorithm state.

Macros

- #define GAS_DETECTION_STRATEGY_AUTO (0)
- #define GAS_DETECTION_STRATEGY_FORCEO3 (1)
- #define GAS_DETECTION_STRATEGY_FORCENO2 (2)

Typedefs

- typedef float oaq_prec_t

  internal precision for gas classification, must be at least 32bit floating point!

Functions

- void oaq_base_init (oaq_base_handle_t *handle, const uint8_t *general_purpose, const uint8_t stabilization_samples)

  initializes the OAQ algorithm
- float calc_oaq (oaq_base_handle_t *handle, const float *rmox, const uint8_t rcda_strategy, const uint8_t gas_detection_strategy, const float d_rising_m1, const float d_falling_m1, const float d_class_m1)

  calculates AQI from present sample

Variables

- float trim_beta2
- float trim_b
- float rcda_22
- float rcda_42
- uint8_t trim_data_version
- uint8_t stabilization_sample
- float prob_no2
- float conc_no2
- float conc_o3
- float aqi_no2
- float aqi_o3

6.2.1 Detailed Description

Discriminations of the gases.

6.2.2 Macro Definition Documentation

6.2.2.1 GAS_DETECTION_STRATEGY_AUTO

```
#define GAS_DETECTION_STRATEGY_AUTO (0)
```

use automatic gas discrimination

6.2.2.2 GAS_DETECTION_STRATEGY_FORCENO2

```
#define GAS_DETECTION_STRATEGY_FORCENO2 (2)
```

handle the measurements as NO2

6.2.2.3 GAS_DETECTION_STRATEGY_FORCEO3

```
#define GAS_DETECTION_STRATEGY_FORCEO3 (1)
```

handle the measurements as O3

6.2.3 Function Documentation

6.2.3.1 calc_oaq()

```
float calc_oaq (
            oaq_base_handle_t * handle,
            const float * rmox,
            const uint8_t rcda_strategy,
            const uint8_t gas_detection_strategy,
            const float d_rising_m1,
            const float d_falling_m1,
            const float d_class_m1 )
```

calculates AQI from present sample

Parameters

in	*handle*	pointer to algorithm state variable
in	*rmox*	pointer to array of the 15 sequencer rmox measurements
in	*rcda_strategy*	how to handle Rcda
in	*gas_detection_strategy*	do automatic gas discrimination or force O3 or NO2
in	*d_rising_m1*	rcda damping factor for rising rmox
in	*d_falling_m1*	rcda damping factor for falling rmox
in	*d_class_m1*	damping factor for gas classification

Returns

AQI value

6.2.3.2 oaq_base_init()

```
void oaq_base_init (
            oaq_base_handle_t * handle,
            const uint8_t * general_purpose,
            const uint8_t stabilization_samples )
```

initializes the OAQ algorithm

Parameters

out	*handle*	pointer to algorithm state variable
in	*general_purpose*	pointer to the 9 bytes trim information
in	*stabilization_samples*	number of samples to be ignored for stabilization

6.2.4 Variable Documentation

6.2.4.1 aqi_no2

```
float aqi_no2
```

equivalent NO2 AQI

6.2.4.2 aqi_o3

```
float aqi_o3
```

equivalent O3 AQI

6.2.4.3 conc_no2

`float conc_no2`

equivalent NO2 concentration [ppb]

6.2.4.4 conc_o3

`float conc_o3`

equivalent O3 concentration [ppb]

6.2.4.5 prob_no2

`float prob_no2`

NO2 probability

6.2.4.6 rcda_22

`float rcda_22`

Rcda estimation for Rmox_22

6.2.4.7 rcda_42

`float rcda_42`

Rcda estimation for Rmox_42

6.2.4.8 stabilization_sample

`uint8_t stabilization_sample`

number of samples still needed for stabilization

6.2.4.9 trim_b

`float trim_b`

NO2 intercept from trim data

6.2.4.10 trim_beta2

`float trim_beta2`

slope from trim data

6.2.4.11 trim_data_version

`uint8_t trim_data_version`

version of trim data in NVM

Chapter 7

Data Structure Documentation

7.1 oaq_base_handle_t Struct Reference

Variables that describe the sensor or the algorithm state.

`#include <oaq.h>`

Data Fields

- float trim_beta2
- float trim_b
- float rcda_22
- float rcda_42
- uint8_t trim_data_version
- uint8_t stabilization_sample
- float prob_no2
- float conc_no2
- float conc_o3
- float aqi_no2
- float aqi_o3

7.1.1 Detailed Description

Variables that describe the sensor or the algorithm state.

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

- oaq.h

Chapter 8

File Documentation

8.1 oaq.h File Reference

This file contains the data structure definitions and the function definitions for the OAQ base algorithm.

```
#include <stdint.h>
#include <math.h>
```

Data Structures

- struct oaq_base_handle_t

  Variables that describe the sensor or the algorithm state.

Macros

- #define RCDA_STRATEGY_FIX (0)
- #define RCDA_STRATEGY_SET (1)
- #define RCDA_STRATEGY_ADJ (2)
- #define GAS_DETECTION_STRATEGY_AUTO (0)
- #define GAS_DETECTION_STRATEGY_FORCEO3 (1)
- #define GAS_DETECTION_STRATEGY_FORCENO2 (2)

Typedefs

- typedef float oaq_prec_t

  internal precision for gas classification, must be at least 32bit floating point!

Functions

- void oaq_base_init (oaq_base_handle_t *handle, const uint8_t *general_purpose, const uint8_t stabilization_samples)

  initializes the OAQ algorithm
- float calc_oaq (oaq_base_handle_t *handle, const float *rmox, const uint8_t rcda_strategy, const uint8_t gas_detection_strategy, const float d_rising_m1, const float d_falling_m1, const float d_class_m1)

  calculates AQI from present sample

8.1.1 Detailed Description

This file contains the data structure definitions and the function definitions for the OAQ base algorithm.

Date

2019-09-24

Author

IDT

Version

4.0.0 - https://semver.org/

The library contains an algorithm to calculate an OAQ index from ZMOD4510 measurements. The implementation is made to allow more than one sensor.