Contents

List of Figures

List of Tables

1. Introduction

The ZMOD44xx Gas Sensor Family is highly configurable in order to meet various application needs. This document describes a general program flow to set up ZMOD44xx gas sensor modules for gas measurements. The term ZMOD44xx refers to any product in the ZMOD44xx Gas Sensor Family such as the ZMOD4410 Gas Sensor Module for TVOC and Indoor Air Quality. In addition, this document describes the function of an example code provided as a C file by IDT for the ZMOD4410, which can be operated using its evaluation kit (EVK). The ZMOD4410 is designed for the detection of total volatile organic compounds (TVOC) and provides an output for an estimated carbon dioxide level (eCO2) and a rating for the indoor air quality (IAQ).

Recommendation: Read the datasheet for the specific ZMOD44xx product before using this manual.

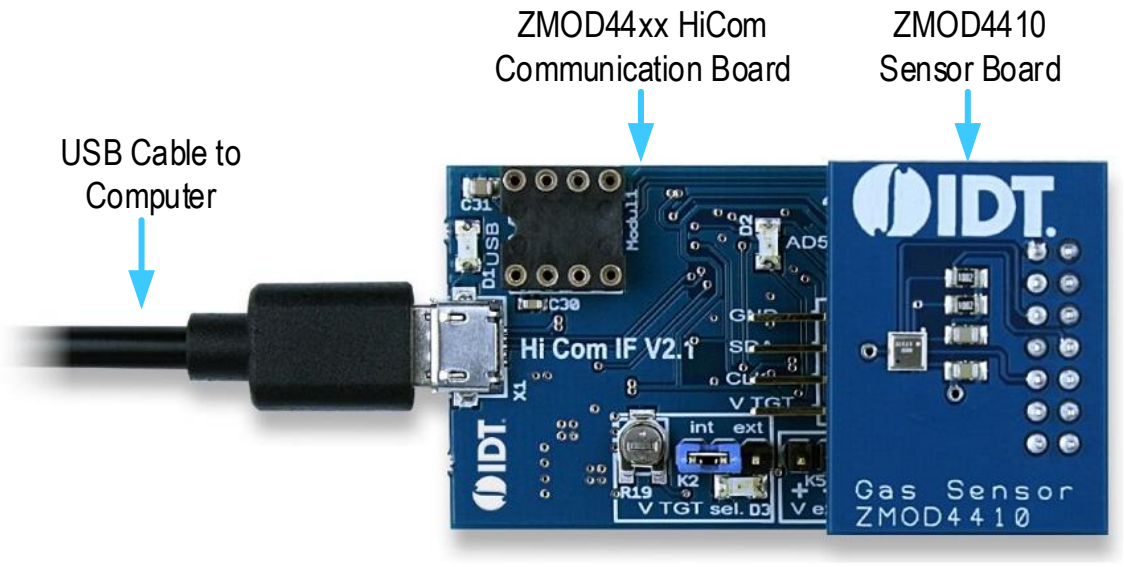
2. Hardware Setup for the ZMOD4410

To implement the ZMOD44xx sensor with the user's program, specific hardware is needed. The example below shows the ZMOD4410 EVK, which consist of three components:

- ZMOD44xx HiCom Communication Board
- ZMOD4410 Sensor Board (Daughter Board) with the ZMOD4410 Gas Sensor Module
- Micro-USB cable

Refer to the *ZMOD4410 Evaluation Kit Description* for instructions on assembly, connections, and installation of hardware and software. Figure 1 shows the assembled evaluation kit.

Figure 1. ZMOD4410 Evaluation Kit



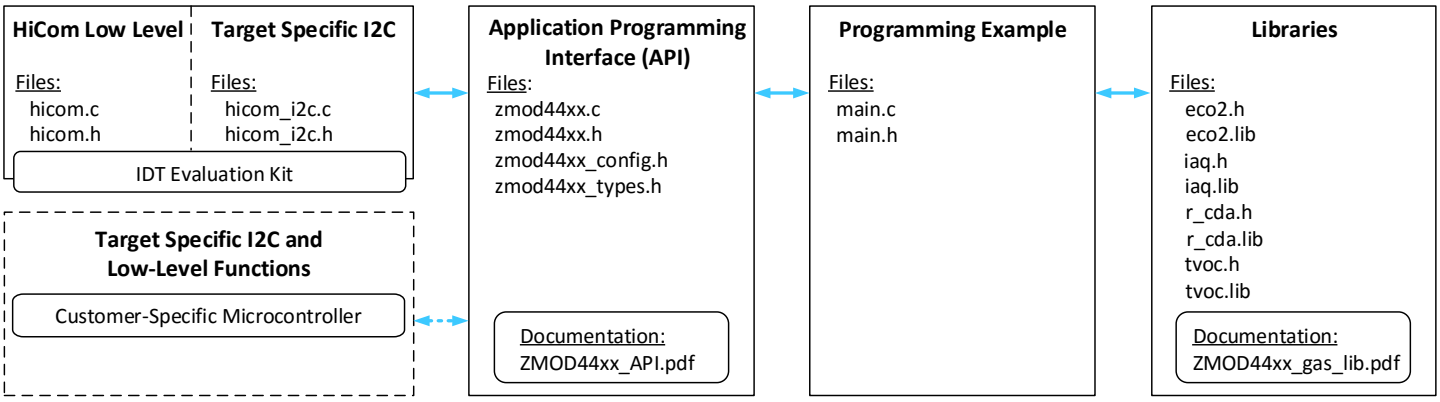
3. General Program Flow for Setting up ZMOD44xx Gas Measurements

To operate the ZMOD44xx in the hardware and use its full functionality, four code blocks are needed as illustrated in Figure 2:

- The "Target Specific I2C and Low-Level Functions" block is the hardware-specific implementation of the I2C interface, it contains read and write functions to communicate with the ZMOD44xx. If the IDT EVK is used, files for the Hi Com Communication Board are provided with this manual. Custom microcontrollers can be used to establish I2C communication.
- The "Application Programming Interface (API)" block contains the functions needed to work with the ZMOD44xx. A detailed description of the API can be found in the document: *ZMOD44xx_API.pdf*.
- The "Programming Example" block provides a code example that is used to initialize the ZMOD4410 gas sensor module, operate it at a constant temperature of 300°C, and display the data for TVOC, IAQ, and eCO2. Further details can be found in section 4 of this document.
- The "Libraries" block contains the functions and data structures needed to calculate eCO2, IAQ, and TVOC. The libraries are described in more detail in the document *ZMOD44xx_gas_lib.pdf*.

To avoid naming conflicts, all function names start with the prefix "`zmod44xx_`" in the ZMOD44xx code. This naming applies to all products in the ZMOD44xx product family.

Figure 2. File Overview for ZMOD44xx Gas Measurements



4. Description of the Programming Example Code

The following section describes the example code and how to use the ZMOD4410 Gas Sensor Module. In the example, the ZMOD4410 is initialized and configured for continuous operation at 300°C, and then it displays measured values. The example is intended to work on a Windows® (trademark of Microsoft, Inc.) computer in combination with the IDT gas sensor EVK. However, the example can be adjusted to operate on other platforms such as ARM® (trademark of ARM, Ltd.) and Linux® (trademark of Linus Torvalds).

To run the example using the EVK without further configuration, run *ZMOD44xx_example.exe,* which is included in the kit software.

4.1 *main* Files

The *main.c / main.h* files contain the main program flow.

The TVOC, IAQ, and eCO2 algorithm is configured by setting the parameters according to the TVOC (IAQ, eCO2) library. Then the target specific initializations are performed. The ZMOD4410 is configured by reading the Final Module Test parameters from the sensor's nonvolatile memory (NVM) and initializing it to run continuously at 300 °C.

An endless loop continuously checks the status of the ZMOD4410 and reads its data. The raw data is subsequently processed, and the TVOC, IAQ, and eCO2 values are calculated. All values are shown in a command line window. To stop the loop, press any key, which releases the hardware and stops the program.

4.2 Program Flow to Run the Sensor

Refer to the example code for more details.

Table 1. Program Flow

Note: In the following table, lines that are shaded blue can be run in an endless loop with polling or interrupt usage.

Line	Program Actions	Notes	API Functions
1	Reset the sensor.	Before configuring the sensor, it is necessary to reset the sensor by powering it off/on or toggling the reset pin.	-
2	Read device parameters from the nonvolatile memory (NVM).	This step is necessary to choose the correct configuration for the sensor.	`zmod44xx_read_sensor_info`
3	Initial initialization.	This function must be called after every startup.	`zmod44xx_init_sensor`
4	Initialize the sensor for TVOC (IAQ, eCO2) measurements.	Initialize the sensor to run at 300°C.	`zmod44xx_init_measurement`
5	Start the measurement.	Start the measurement.	`zmod44xx_start_measurement`
6	Read status register.	Wait until the initialization is ready.	`zmod44xx_read_status`
7	Read status register.	Wait until the measurement is done. This will also be signaled by an interrupt.	`zmod44xx_read_status`
8	Get the MOx resistance value.		`zmod44xx_read_rmox`
9	Read status register.	Wait until the sensor is ready.	`zmod44xx_read_status`

5. Using the Example on a Different Hardware Platform

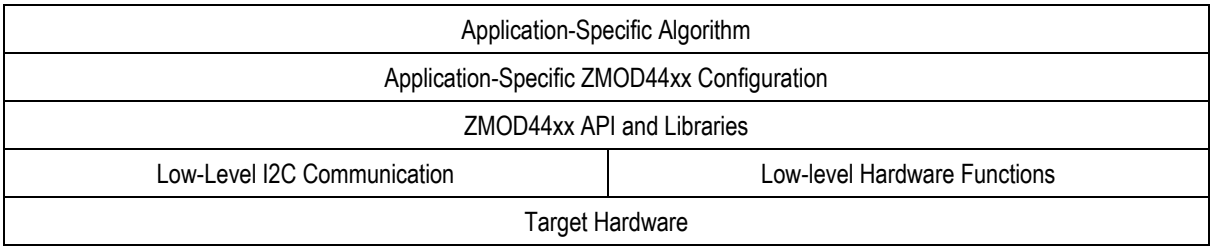
To incorporate this programming example into a different hardware platform, it is necessary to set the device's struct pointers *read*, *write* and *delay_ms*. The type definitions of the function pointers can be found in *zmod44xx_types.h* (see Figure 2). The functions *read* and *write* should point to the I2C implementation of the hardware used.

In addition, IDT provides precompiled algorithm libraries that are hardware-platform dependent. IDT offers a download of these libraries for x86/x64 (Linux/Windows) as well as for ARM and MSP microcontrollers.

5.1 Adaptation for the Target System

IDT's ZMOD44xx C API is located between the application and the target hardware.

Figure 3. System Hierarchy



The low-level I2C functions are implemented in the file *hicom_i2c.c* (see Figure 2) for the EVK hardware running on a Windows-based computer and the HiCom Communication Board.

5.2 Error Codes

Most of the API functions return a code to indicate the success of the operation. If no error occurred, the return code is 0. In the event of an error, a number not equal to zero is returned. The API has predefined symbols `ZMOD44XX_ERROR_*` for the error codes defined in *zmod44xx_types.h*.

6. Revision History

Revision Date	Description of Change
June 11, 2018	Initial release.

Corporate Headquarters
6024 Silver Creek Valley Road
San Jose, CA 95138
www.IDT.com

Sales
1-800-345-7015 or 408-284-8200
Fax: 408-284-2775
www.IDT.com/go/sales

Tech Support
www.IDT.com/go/support

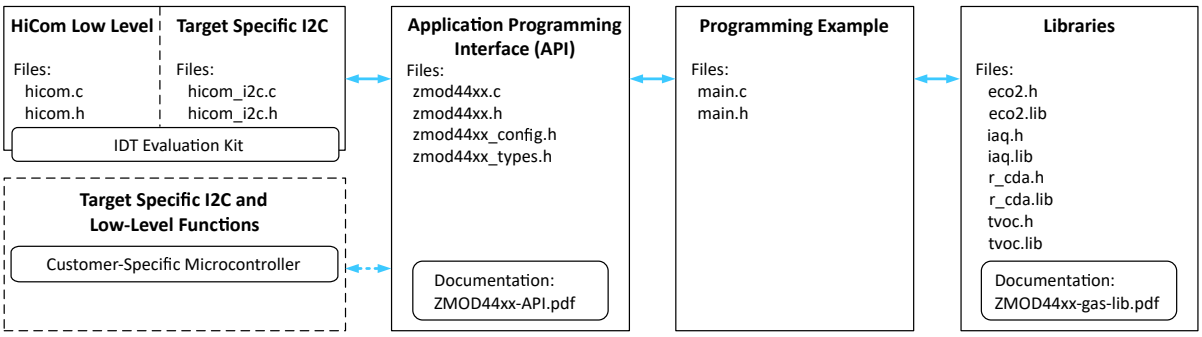
ZMOD44xx-API Documentation

Contents

Chapter 1

ZMOD44xx Application Programming Interface Overview

This document refers to the IDT document *ZMOD44xx Programming Manual with ZMOD4410 Example*. The figure below shows an overview of the ZMOD44xx example, API and libraries. The following describes in detail the Application Programming Interface (API) of the ZMOD44xx.



Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures 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 Gas sensor IDs

Macros

- #define **ZMOD4410_PID** (0x2310)

5.1.1 Detailed Description

The gas sensor product IDs.

5.2 Error codes

Macros

- #define ERROR_INIT_OUT_OF_RANGE (1)
- #define ERROR_GAS_TIMEOUT (2)
- #define ERROR_I2C (3)
- #define ERROR_SENSOR_UNSUPPORTED (4)
- #define ERROR_CONFIG_MISSING (5)
- #define ERROR_SENSOR (6)

5.2.1 Detailed Description

The gas sensor and API error codes.

5.2.2 Macro Definition Documentation

5.2.2.1 #define ERROR_CONFIG_MISSING (5)

There is no pointer to a valid configuration.

5.2.2.2 #define ERROR_GAS_TIMEOUT (2)

The operation took too long.

5.2.2.3 #define ERROR_I2C (3)

Failure in i2c communication.

5.2.2.4 #define ERROR_INIT_OUT_OF_RANGE (1)

The initialize value is out of range.

5.2.2.5 #define ERROR_SENSOR (6)

Sensor malfunction.

5.2.2.6 #define ERROR_SENSOR_UNSUPPORTED (4)

Sensor is not supported with this firmware.

Chapter 6

Data Structure Documentation

6.1 zmod44xx_conf Struct Reference

Structure to hold the gas sensor module configuration.

```
#include <zmod44xx_types.h>
```

Data Fields

- char **name** [ZMOD44XX_NAME_LEN]
- uint8_t **start**
- zmod44xx_conf_str **h**
- zmod44xx_conf_str **d**
- zmod44xx_conf_str **m**
- zmod44xx_conf_str **s**
- zmod44xx_conf_str **r**

6.1.1 Detailed Description

Structure to hold the gas sensor module configuration.

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

- zmod44xx_types.h

6.2 zmod44xx_conf_str Struct Reference

A single data set for the configuration.

```
#include <zmod44xx_types.h>
```

Data Fields

- uint8_t **addr**
- uint8_t **len**
- uint8_t * **data**

6.2.1 Detailed Description

A single data set for the configuration.

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

- zmod44xx_types.h

6.3 zmod44xx_dev_t Struct Reference

Device structure ZMOD44xx.

```
#include <zmod44xx_types.h>
```

Data Fields

- uint8_t i2c_addr
- uint16_t pid
- uint8_t config [6]
- uint16_t mox_lr
- uint16_t mox_er
- zmod44xx_i2c_ptr_t read
- zmod44xx_i2c_ptr_t write
- zmod44xx_delay_ptr_p delay_ms

6.3.1 Detailed Description

Device structure ZMOD44xx.

6.3.2 Field Documentation

6.3.2.1 uint8_t config[6]

configuration parameter set

6.3.2.2 zmod44xx_delay_ptr_p delay_ms

function pointer to delay function

6.3.2.3 uint8_t i2c_addr

i2c address of the sensor

6.3.2.4 uint16_t mox_er

sensor specific parameter

6.3.2.5 uint16_t mox_lr

sensor specific parameter

6.3.2.6 uint16_t pid

product id of the sensor

6.3.2.7 zmod44xx_i2c_ptr_t read

function pointer to i2c read

6.3.2.8 zmod44xx_i2c_ptr_t write

function pointer to i2c write

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

- zmod44xx_types.h

Chapter 7

File Documentation

7.1 zmod44xx.c File Reference

ZMOD44xx functions.

```
#include "zmod44xx.h"
```

Functions

- int8_t zmod44xx_read_sensor_info (zmod44xx_dev_t *dev)
  Read sensor parameter.
- int8_t zmod44xx_init_sensor (zmod44xx_dev_t *dev)
  Initialize the sensor after power on.
- int8_t zmod44xx_init_measurement (zmod44xx_dev_t *dev)
  Initialize the sensor for IAQ, TVOC and eCO2 measurement.
- int8_t zmod44xx_start_measurement (zmod44xx_dev_t *dev)
  Start the measurement.
- int8_t zmod44xx_read_status (zmod44xx_dev_t *dev, uint8_t *status)
  Read the status of the device.
- int8_t zmod44xx_read_rmox (zmod44xx_dev_t *dev, float *rmox)
  Read adc values from sensor and calculate RMOX.

7.1.1 Detailed Description

ZMOD44xx functions.

Version

1.0.0

Date

2018-05-17

Author

IDT

7.1.2 Function Documentation

7.1.2.1 int8_t zmod44xx_init_measurement (zmod44xx_dev_t * dev)

Initialize the sensor for IAQ, TVOC and eCO2 measurement.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

7.1.2.2 int8_t zmod44xx_init_sensor (zmod44xx_dev_t * dev)

Initialize the sensor after power on.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

7.1.2.3 int8_t zmod44xx_read_rmox (zmod44xx_dev_t * dev, float * rmox)

Read adc values from sensor and calculate RMOX.

Parameters

in	*dev*	pointer to the device
in,out	*rmox*	pointer to the resulting Rmox value

Returns

error code

Return values

0	success
!= 0	error

7.1.2.4 int8_t zmod44xx_read_sensor_info (zmod44xx_dev_t * *dev*)

Read sensor parameter.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

Note

This function must be called once before running other sensor functions.

7.1.2.5 int8_t zmod44xx_read_status (zmod44xx_dev_t * *dev,* uint8_t * *status*)

Read the status of the device.

Parameters

in	*dev*	pointer to the device
in,out	*status*	pointer to the status variable

Returns

error code

Return values

0	success
!= 0	error

7.1.2.6 int8_t zmod44xx_start_measurement (zmod44xx_dev_t * *dev*)

Start the measurement.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

7.2 zmod44xx.h File Reference

ZMOD44xx functions.

```
#include "zmod44xx_config.h"
#include "zmod44xx_types.h"
```

Macros

- #define **ZMOD4410_I2C_ADDRESS** (0x32)
- #define **ZMOD44XX_ADDR_PID** (0x00)
- #define **ZMOD44XX_ADDR_CONF** (0x20)
- #define **ZMOD44XX_ADDR_CMD** (0x93)
- #define **ZMOD44XX_ADDR_STATUS** (0x94)
- #define **ZMOD44XX_LEN_PID** (2)
- #define **ZMOD44XX_LEN_CONF** (6)

Functions

- int8_t zmod44xx_read_sensor_info (zmod44xx_dev_t *dev)
  Read sensor parameter.
- int8_t zmod44xx_init_sensor (zmod44xx_dev_t *dev)
  Initialize the sensor after power on.
- int8_t zmod44xx_init_measurement (zmod44xx_dev_t *dev)
  Initialize the sensor for IAQ, TVOC and eCO2 measurement.
- int8_t zmod44xx_start_measurement (zmod44xx_dev_t *dev)
  Start the measurement.
- int8_t zmod44xx_read_status (zmod44xx_dev_t *dev, uint8_t *status)
  Read the status of the device.
- int8_t zmod44xx_read_rmox (zmod44xx_dev_t *dev, float *rmox)
  Read adc values from sensor and calculate RMOX.

7.2.1 Detailed Description

ZMOD44xx functions.

Version

1.0.0

Date

2018-05-17

Author

IDT

7.2.2 Function Documentation

7.2.2.1 int8_t zmod44xx_init_measurement (zmod44xx_dev_t * *dev*)

Initialize the sensor for IAQ, TVOC and eCO2 measurement.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

7.2.2.2 int8_t zmod44xx_init_sensor (zmod44xx_dev_t * *dev*)

Initialize the sensor after power on.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

7.2.2.3 int8_t zmod44xx_read_rmox (zmod44xx_dev_t * *dev*, float * *rmox*)

Read adc values from sensor and calculate RMOX.

Parameters

in	*dev*	pointer to the device
in,out	*rmox*	pointer to the resulting Rmox value

Returns

error code

Return values

0	success
!= 0	error

7.2.2.4 int8_t zmod44xx_read_sensor_info (zmod44xx_dev_t * *dev*)

Read sensor parameter.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

Note

This function must be called once before running other sensor functions.

7.2.2.5 int8_t zmod44xx_read_status (zmod44xx_dev_t * *dev*, uint8_t * *status*)

Read the status of the device.

Parameters

in	*dev*	pointer to the device
in, out	*status*	pointer to the status variable

Returns

error code

Return values

0	success
!= 0	error

7.2.2.6 int8_t zmod44xx_start_measurement (zmod44xx_dev_t * *dev*)

Start the measurement.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

7.3 zmod44xx_config.h File Reference

ZMOD44xx configuration.

```
#include <stdint.h>
#include "zmod44xx_types.h"
```

7.3.1 Detailed Description

ZMOD44xx configuration.

Version

1.0.0

Date

2018-05-25

Author

IDT

7.4 zmod44xx_types.h File Reference

ZMOD44xx types.

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

Data Structures

- struct zmod44xx_dev_t

  Device structure ZMOD44xx.
- struct zmod44xx_conf_str

  A single data set for the configuration.
- struct zmod44xx_conf

  Structure to hold the gas sensor module configuration.

Macros

- #define **ZMOD4410_PID** (0x2310)
- #define ZMOD44XX_OK (0)
- #define ERROR_INIT_OUT_OF_RANGE (1)
- #define ERROR_GAS_TIMEOUT (2)
- #define ERROR_I2C (3)
- #define ERROR_SENSOR_UNSUPPORTED (4)
- #define ERROR_CONFIG_MISSING (5)
- #define ERROR_SENSOR (6)
- #define **ZMOD44XX_NAME_LEN** (5)

Typedefs

- typedef int8_t(∗ zmod44xx_i2c_ptr_t) (uint8_t addr, uint8_t reg_addr, uint8_t ∗data, uint8_t len)

  function pointer type for i2c access
- typedef void(∗ zmod44xx_delay_ptr_p) (uint32_t ms)

  function pointer to hardware dependent delay function

7.4.1 Detailed Description

ZMOD44xx types.

Version

1.0.0

Date

2018-05-17

Author

IDT

7.4.2 Macro Definition Documentation

7.4.2.1 #define ZMOD44XX_OK (0)

Return value if no fault has been found.

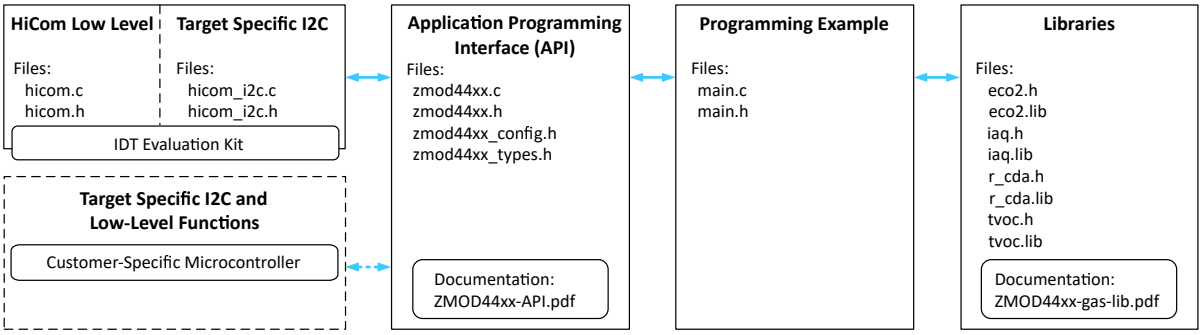
ZMOD44xx Gas Algorithm Documentation

Contents

Chapter 1

ZMOD44xx Application Programming Interface Overview

This document describes the gas algorithm libraries for the ZMOD44xx Gas Sensor Family for Indoor Air Quality. Refer to the *ZMOD44xx Programming Manual with ZMOD4410 Example* for further information regarding sample code. The figure below shows an overview of the ZMOD44xx example, API and libraries. The following sections describe in detail the gas algorithm libraries for clean dry air resistance, total volatile organic compounds, indoor air quality, and estimated carbon dioxide (R_CDA, TVOC, IAQ, and eCO2, respectively).



Chapter 2

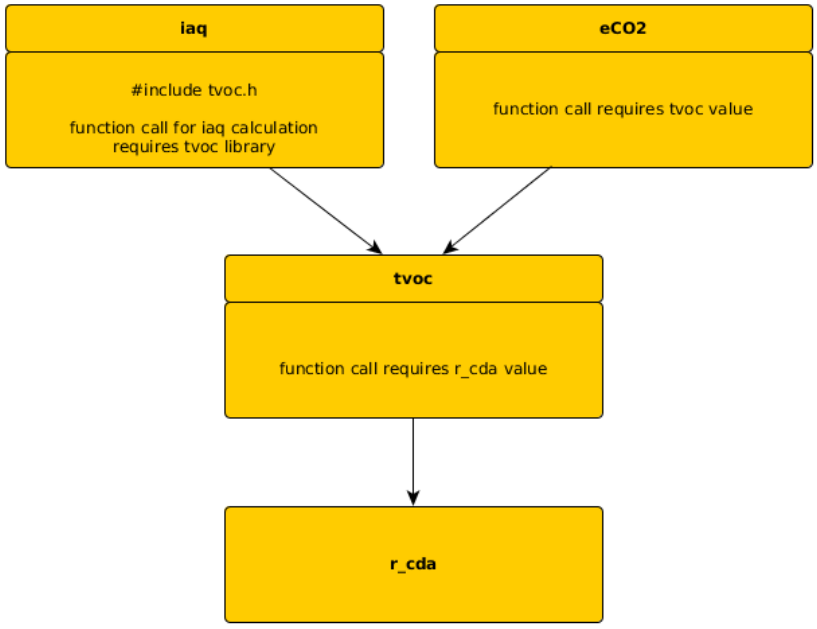
How to Work with the IDT Gas Algorithm Libraries

- Include the intended header file in the user's program for gas sensor module control; for example:
```
#include "r_cda.h"
#include "tvoc.h"
#include "iaq.h"
#include "eco2.h"
```

- Copy the library file into user's project folder
- Call the intended function in the user's program

File overview and dependencies

The following figure shows an overview of the file dependencies.



Example for clean dry air resistance:

```
#include "r_cda.h"

int main() {
    float r_cda;
    float r_mox;

    // your functionality
    // r_mox measurement

    r_cda = r_cda_tracker(r_mox);

    // your functionality

    return 0;
}
```

Example for TVOC:

```
#include "r_cda.h"
#include "tvoc.h"

int main() {
    float r_cda;
    float r_mox;
    float tvoc;
    tvoc_params tvoc_par = {.A = 600000, .alpha = 0.7};

    // your functionality
    // r_mox measurement

    r_cda = r_cda_tracker(r_mox);
    tvoc = calc_tvoc(r_mox, r_cda, &tvoc_par);

    // your functionality

    return 0;
}
```

Example for IAQ:

```
#include "r_cda.h"
#include "iaq.h"

int main() {
    float r_cda;
    float r_mox;
    uint8_t iaq;
    tvoc_params tvoc_par = {.A = 600000, .alpha = 0.7};

    // your functionality
    // r_mox measurement

    r_cda = r_cda_tracker(r_mox);
    iaq = calc_iaq(r_mox, r_cda, &tvoc_par);

    // your functionality

    return 0;
}
```

Example for eCO2:

```
#include "eco2.h"
#include "r_cda.h"
#include "tvoc.h"

#define DELAY   10  # 10 seconds between measurements

int main() {
    float r_cda;
    float r_mox;
    float tvoc;
    float eco2;
    tvoc_params tvoc_par = {.A = 600000, .alpha = 0.7};
    eco2_params eco2_par = {.min_co2 = 400, .max_co2 = 5000, .tvoc_to_eco2 = 800.0,
                            .hot_wine_rate = 0.3, .open_window_rate = -0.05};

    // your functionality
    // r_mox measurement

    r_cda = r_cda_tracker(r_mox);
    tvoc = calc_tvoc(r_mox, r_cda, &tvoc_par);
    eco2 = calc_eco2(tvoc, DELAY, &eco2_par);

    // your functionality

    return 0;
}
```

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

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

Chapter 5

Data Structure Documentation

5.1 eco2_params Struct Reference

Parameters to control the eCO2 algorithm.

`#include <eco2.h>`

Data Fields

- uint16_t min_co2
- uint16_t max_co2
- float tvoc_to_eco2
- float hot_wine_rate
- float open_window_rate

5.1.1 Detailed Description

Parameters to control the eCO2 algorithm.

5.1.2 Field Documentation

5.1.2.1 float hot_wine_rate

rate for fast positive changes

5.1.2.2 uint16_t max_co2

highest possible CO2 value in ppm

5.1.2.3 uint16_t min_co2

lowest possible CO2 value in ppm

5.1.2.4 float open_window_rate

rate for fast negative changes

5.1.2.5 float tvoc_to_eco2

how much ppm CO2 are generated per mg/m^3 TVOC

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

- eco2.h

5.2 tvoc_params Struct Reference

Parameters to control the TVOC algorithm.

```
#include <tvoc.h>
```

Data Fields

- float A
- float alpha

5.2.1 Detailed Description

Parameters to control the TVOC algorithm.

5.2.2 Field Documentation

5.2.2.1 float A

intercept parameter for TVOC algorithm

5.2.2.2 float alpha

slope parameter for TVOC algorithm

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

- tvoc.h

Chapter 6

File Documentation

6.1 eco2.h File Reference

This file contains the data structure definition and the function definition for the eCO2 algorithm.

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

Data Structures

- struct eco2_params

  Parameters to control the eCO2 algorithm.

Macros

- #define FILTER_TAU (20.0)
- #define CNT_PREV_SAMPLES_TVOC (4)

Functions

- float calc_eco2 (float conc_tvoc, float sample_rate, eco2_params *params)

  calculates estimated CO2 value from TVOC

6.1.1 Detailed Description

This file contains the data structure definition and the function definition for the eCO2 algorithm.

Date

2017-11-08

Author

IDT

Version

1.1.2 - https://semver.org/

The library contains an algorithm to calculate an estimated CO2 value in ppm from a given TVOC (Total Volatile Organic Compound) value in mg/m^3. Refer to the Application Note *ZMOD4410 Estimating Carbon Dioxide* for further information.

6.1.2 Macro Definition Documentation

6.1.2.1 #define CNT_PREV_SAMPLES_TVOC (4)

jump over first x samples

6.1.2.2 #define FILTER_TAU (20.0)

filter coefficient

6.1.3 Function Documentation

6.1.3.1 float calc_eco2 (float *conc_tvoc,* float *sample_rate,* eco2_params * *params*)

calculates estimated CO2 value from TVOC

Parameters

in	*conc_tvoc*	TVOC concentration
in	*sample_rate*	sample rate in seconds
in	*params*	algorithm control parameters

Returns

eCO2 value

6.2 iaq.h File Reference

This file contains the IAQ algorithm function definition.

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

Enumerations

- enum {
  VERY_GOOD = 1, **GOOD** = 2, **MEDIUM** = 3, **POOR** = 4,
  BAD = 5 }

  IAQ ratings (German Federal Environmental Agency)

Functions

- uint8_t calc_iaq (float r_mox, float r_cda, tvoc_params *params)

  calculates IAQ from r_mox, r_cda and settings

6.2.1 Detailed Description

This file contains the IAQ algorithm function definition.

Date

2018-03-23

Author

IDT

Version

1.0.3 - https://semver.org/

The library contains an algorithm that calculates the Indoor Air Quality based on the scale introduced from the German Federal Environment Agency, which is divided into five levels from "1" for very good to "5" for bad indoor air quality. For further details see Umweltbundesamt, Beurteilung von Innenraumluftkontaminationen mittels Referenz- und Richtwerten, (Bundesgesundheitsblatt - Gesundheitsforschung - Gesundheitsschutz, 2007).

6.2.2 Function Documentation

6.2.2.1 uint8_t calc_iaq (float *r_mox*, float *r_cda*, tvoc_params * *params*)

calculates IAQ from r_mox, r_cda and settings

Parameters

in	*r_mox*	MOx resistance
in	*r_cda*	clean dry air resistance
in	*params*	TVOC algorithm parameters

Returns

iaq rating

6.3 r_cda.h File Reference

This file contains the clean dry air resistance tracker function definition.

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

Functions

- float r_cda_tracker (float r_mox)
  tracks r_cda

6.3.1 Detailed Description

This file contains the clean dry air resistance tracker function definition.

Date

2017-08-04

Author

IDT

Version

1.0.5 - https://semver.org/

The library contains an algorithm to calculate and track the clean dry air resistance from a given MOx resistance value. The r_cda is the MOx resistance in ohms at clean dry air, and it is used as the reference value for further gas measurements.

6.3.2 Function Documentation

6.3.2.1 float r_cda_tracker (float *r_mox*)

tracks r_cda

Parameters

in	*r_mox*	current MOx resistance

Returns

new r_cda

6.4 tvoc.h File Reference

This file contains the data structure definition and the TVOC algorithm function definition.

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

Data Structures

- struct tvoc_params

  Parameters to control the TVOC algorithm.

Functions

- float calc_tvoc (float r_mox, float r_cda, tvoc_params *params)

  calculates TVOC from r_mox and r_cda

6.4.1 Detailed Description

This file contains the data structure definition and the TVOC algorithm function definition.

Date

2017-11-07

Author

IDT

Version

1.0.9 - https://semver.org/

The library contains an algorithm to calculate the TVOC concentration (Total Volatile Organic Compound) from a given MOx and clean dry air resistance value.

6.4.2 Function Documentation

6.4.2.1 float calc_tvoc (float *r_mox*, float *r_cda*, tvoc_params * *params*)

calculates TVOC from r_mox and r_cda

Parameters

in	*r_mox*	MOx resistance
in	*r_cda*	clean dry air resistance
in	*params*	TVOC algorithm parameters

Returns

tvoc concentration in mg/m^3