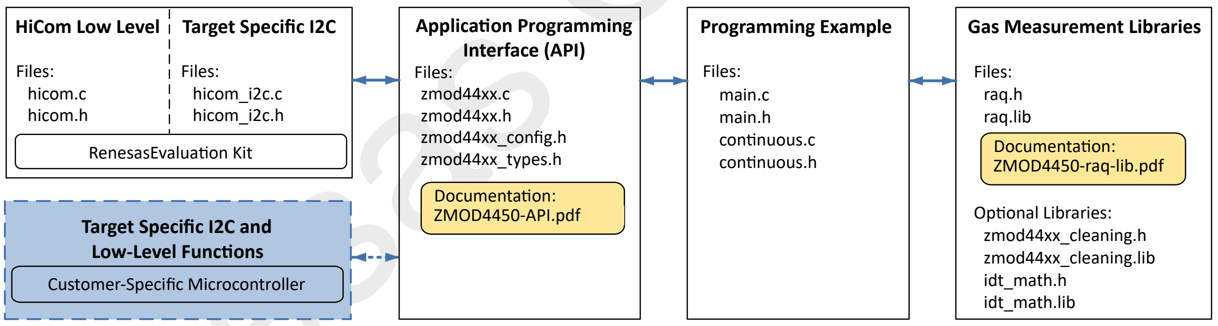
ZMOD44xx Algorithm for RAQ

Contents

Chapter 1

ZMOD4450 Application Programming Interface Overview

This document describes the gas algorithm libraries for the ZMOD44xx Gas Sensor Family for Refrigeration Air Quality (RAQ). The figure below shows an overview of the ZMOD4450 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 sections describe in detail the gas algorithm library for the ZMOD4450 operation, as well as for a recommended cleaning procedure after product assembly. In this mode, the sensor is able to learn the refrigeration environment and report a control signal if a user-defined air quality change threshold is exceeded. This control signal may be used to trigger an action, like turning on an ozone generator, air filter, etc.



Chapter 2

How to Work with the IDT RAQ Algorithm Library

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

Example for RAQ:

```
#include "raq.h"

int main() {
    float r_mox;
    int state;

    raq_results_t raq_results = {.cs_state = OFF, .conc_ratio = 0.0F};

    raq_params raq_par = {.alpha = 0.8,
                          .stop_delay = 24,
                          .threshold = 1.3,
                          .tau = 720,
                          .stabilization_samples = 15};

    // User’s functionality
    // r_mox measurement

    state = calc_raq(r_mox, &raq_par, &raq_results);

    if (!state) {
        // User’s functionality
        // The results:
        // results.cs_state
        // results.conc_ratio
    }

    return 0;
}
```

Example for zmod44xx_cleaning:

```
#include "zmod44xx_cleaning.h"

int main() {
    // initialization of the device structure(dev)
    zmod44xx_dev_t dev;

    // User’s functionality

    zmod44xx_cleaning_run(&dev);

    // User’s functionality

    return 0;
}
```

Chapter 3

IDT Math Library

The **IDT Math Library** is a light weigth math library that implements the math functions used by the gas algorithms. The library is meant to substitute the standard math library of the user's microcontroller in order to achieve reduced code size, therefore inclusion of this library is not necessary unless you seek to reduce the overall code size of your project. The library replaces the functions pow(), log() and exp(), meaning that other standard math functions, not defined in this library, will be unavailable.

How To Use IDT Math

Important to note is that when using the standard math library, some compilers do not require users to explicitlty link it, it is however recommended to explicitly link either the standard or the IDT math library. This prevents potential confusion about which library is being linked.

Compiling A Project

IDEs

For IDE based project builds, the math library should be handled in the same manner as any other externally provided library. The header file should be placed in the include path of your project and the provided `idt_↩ math.lib` file should be placed in the relevant path where your IDE will search for libraries to link. The user should then be able to link the IDT math library to their project.

Command Line

Here a minimal example of how to compile a project with the IDT math library using the gcc compiler for a Linux target is provided.

Assume a project structure like the one below:

```
project
|--- headers
|    |--- idt_math.h
|    |--- tvoc.h
|--- libs
|    |--- libx64_linux_idt_math.a
|    |--- libx64_linux_tvoc.a
|--- src
     |--- main.c
```

Here the user would like to use the tvoc and the IDT math library, which have both been precompiled for the target platform `x64_linux`. In the user's `main.c`, all that is required is to include the file `tvoc.h`, and the user is free to use all the functions in the tvoc library.\ Assuming the current directory is the `project` root directory, the user would compile this project from terminal in the following manner:

```
$ gcc <flags> -o run src/main.c -Iheaders/ -Llibs/ -lx64_linux_tvoc -lx64_linux_idt_math
```

The option `-I` tells the compiler which directories to include while `-L` tells the linker where to look for the libraries.

The executable `run` will now run with the tvoc functions using the light weight IDT math library. To use the standard `math.h` library, simply replace `-lx64_linux_idt_math` with `-lm`.

Chapter 4

Module Index

4.1 Modules

Here is a list of all modules:

Chapter 5

Data Structure Index

5.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 6

File Index

6.1 File List

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

Chapter 7

Module Documentation

7.1 status

Macros

- #define ZMOD4450_OK (0)
- #define ZMOD4450_STABILIZING (9)

7.1.1 Detailed Description

Status of the sensor.

7.1.2 Macro Definition Documentation

7.1.2.1 ZMOD4450_OK

```
#define ZMOD4450_OK (0)
```

Sensor stabilized.

7.1.2.2 ZMOD4450_STABILIZING

```
#define ZMOD4450_STABILIZING (9)
```

Sensor not stabilized.

Chapter 8

Data Structure Documentation

8.1 raq_params Struct Reference

Parameters to control the RAQ.

```
#include <raq.h>
```

Data Fields

- float alpha
- uint32_t stop_delay
- float threshold
- uint32_t tau
- uint32_t stabilization_samples

8.1.1 Detailed Description

Parameters to control the RAQ.

8.1.2 Field Documentation

8.1.2.1 alpha

```
float alpha
```

Slope parameter for RAQ.

8.1.2.2 stabilization_samples

```
uint32_t stabilization_samples
```

Ignore number of samples for sensor stabilization.

8.1.2.3 stop_delay

```
uint32_t stop_delay
```

Control signal follow-up time.

8.1.2.4 tau

```
uint32_t tau
```

Time constant for averaging.

8.1.2.5 threshold

```
float threshold
```

Threshold to switch, i.e. 1.3 - corresponds to 30 % rise in concentration.

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

- raq.h

8.2 raq_results_t Struct Reference

RAQ results.

```
#include <raq.h>
```

Data Fields

- control_signal_state_t cs_state
- float conc_ratio

8.2.1 Detailed Description

RAQ results.

8.2.2 Field Documentation

8.2.2.1 conc_ratio

```
float conc_ratio
```

Concentration ratio.

8.2.2.2 cs_state

```
control_signal_state_t cs_state
```

Control signal input.

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

- raq.h

Chapter 9

File Documentation

9.1 raq.h File Reference

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

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

Data Structures

- struct raq_params

  Parameters to control the RAQ.
- struct raq_results_t

  RAQ results.

Macros

- #define ZMOD4450_OK (0)
- #define ZMOD4450_STABILIZING (9)

Enumerations

- enum control_signal_state_t { **OFF** = 0, **ON** = 1 }

  Control signal states.

Functions

- control_signal_state_t calc_raq (float r_mox, raq_params ∗params, raq_results_t ∗results)

  Calculates RAQ from r_mox and raq parameters.

9.1.1 Detailed Description

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

Date

2019-11-26

Author

IDT

Version

2.0.1 - https://semver.org/

This file contains the function definitions for the RAQ algorithm.

9.1.2 Function Documentation

9.1.2.1 calc_raq()

```
control_signal_state_t calc_raq (
             float r_mox,
             raq_params * params,
             raq_results_t * results )
```

Calculates RAQ from r_mox and raq parameters.

Parameters

in	*r_mox*	MOx resistance.
in	*params*	RAQ parameters.
in,out	*results*	RAQ results.

Returns

Status of the sensor.

Return values

0	Sensor stabilized.
9	Sensor not stabilized.

9.2 zmod44xx_cleaning.h File Reference

This file contains the cleaning function definition for ZMOD44xx.

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

Macros

- #define ERROR_CLEANING (9)

Functions

- uint8_t zmod44xx_cleaning_run (zmod44xx_dev_t ∗dev)

    *Start a cleaning procedure (**Around 10 minutes blocking.**).*

9.2.1 Detailed Description

This file contains the cleaning function definition for ZMOD44xx.

Date

2019-10-28

Author

IDT

Version

1.0.1 - `https://semver.org/`

The library contains the function that starts the cleaning procedure. **The procedure takes around 10 minutes.** After successful cleaning, the function return 0. **The procedure can be run only once.**

9.2.2 Macro Definition Documentation

9.2.2.1 ERROR_CLEANING

```
#define ERROR_CLEANING (9)
```

Error cleaning.

9.2.3 Function Documentation

9.2.3.1 zmod44xx_cleaning_run()

```
uint8_t zmod44xx_cleaning_run (
            zmod44xx_dev_t * dev )
```

Start a cleaning procedure (**Around 10 minutes blocking.**).

Parameters

`in`	*dev*	Pointer to the device.

Returns

Error code.

Return values

0	Success.
!= 0	Error.