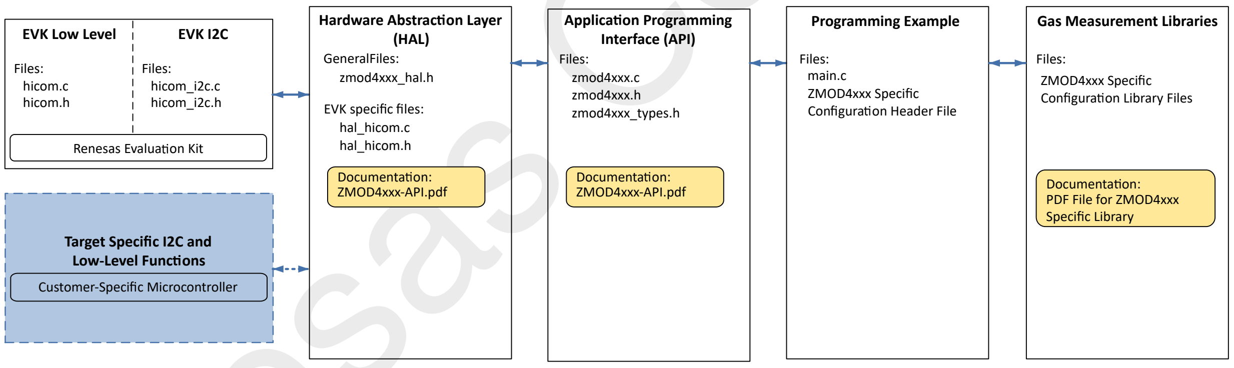
ZMOD4xxx-API Documentation

Contents

Chapter 1

ZMOD4xxx Application Programming Interface Overview

This document refers to the Renesas document *ZMOD4xxx Programming Manual - Read Me*. 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. The following describes in detail the Application Programming Interface (API) of the ZMOD4xxx.


EVK Low Level
Files:
hicom.c
hicom.h
EVK I2C
Files:
hicom_i2c.c
hicom_i2c.h
Renesas Evaluation Kit
Target Specific I2C and Low-Level Functions
Customer-Specific Microcontroller
Hardware Abstraction Layer (HAL)
GeneralFiles:
zmod4xxx_hal.h
EVK specific files:
hal_hicom.c
hal_hicom.h
Documentation: ZMOD4xxx-API.pdf
Application Programming Interface (API)
Files:
zmod4xxx.c
zmod4xxx.h
zmod4xxx_types.h
Documentation: ZMOD4xxx-API.pdf
Programming Example
Files:
main.c
ZMOD4xxx Specific Configuration Header File
Gas Measurement Libraries
Files:
ZMOD4xxx Specific Configuration Library Files
Documentation: PDF File for ZMOD4xxx Specific Library

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

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

Chapter 4

Data Structure Documentation

4.1 zmod4xxx_conf Struct Reference

Structure to hold the gas sensor module configuration.

```
#include <zmod4xxx_types.h>
```

Data Fields

- uint8_t **start**
- zmod4xxx_conf_str **h**
- zmod4xxx_conf_str **d**
- zmod4xxx_conf_str **m**
- zmod4xxx_conf_str **s**
- zmod4xxx_conf_str **r**
- uint8_t **prod_data_len**

4.1.1 Detailed Description

Structure to hold the gas sensor module configuration.

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

- zmod4xxx_types.h

4.2 zmod4xxx_conf_str Struct Reference

A single data set for the configuration.

```
#include <zmod4xxx_types.h>
```

Data Fields

- uint8_t **addr**
- uint8_t **len**
- uint8_t * **data_buf**

4.2.1 Detailed Description

A single data set for the configuration.

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

- zmod4xxx_types.h

4.3 zmod4xxx_dev_t Struct Reference

Device structure ZMOD4xxx.

```
#include <zmod4xxx_types.h>
```

Data Fields

- uint8_t i2c_addr
- uint8_t config [6]
- uint16_t mox_er
- uint16_t mox_lr
- uint16_t pid
- uint8_t * prod_data
- zmod4xxx_i2c_ptr_t read
- zmod4xxx_i2c_ptr_t write
- zmod4xxx_delay_ptr_p delay_ms
- zmod4xxx_conf * init_conf
- zmod4xxx_conf * meas_conf

4.3.1 Detailed Description

Device structure ZMOD4xxx.

4.3.2 Field Documentation

4.3.2.1 config

```
uint8_t config[6]
```

configuration parameter set

4.3.2.2 delay_ms

```
zmod4xxx_delay_ptr_p delay_ms
```

function pointer to delay function

4.3.2.3 i2c_addr

```
uint8_t i2c_addr
```

i2c address of the sensor

4.3.2.4 init_conf

```
zmod4xxx_conf* init_conf
```

pointer to the init configuration

4.3.2.5 meas_conf

```
zmod4xxx_conf* meas_conf
```

pointer to the measurement configuration

4.3.2.6 mox_er

```
uint16_t mox_er
```

sensor specific parameter

4.3.2.7 mox_lr

```
uint16_t mox_lr
```

sensor specific parameter

4.3.2.8 pid

```
uint16_t pid
```

product id of the sensor

4.3.2.9 prod_data

```
uint8_t* prod_data
```

production data

4.3.2.10 read

```
zmod4xxx_i2c_ptr_t read
```

function pointer to i2c read

4.3.2.11 write

```
zmod4xxx_i2c_ptr_t write
```

function pointer to i2c write

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

- zmod4xxx_types.h

Chapter 5

File Documentation

5.1 hal_hicom.h File Reference

Hardware abstraction layer for windows.

```
#include "hicom.h"
#include "hicom_i2c.h"
#include "zmod4xxx_types.h"
#include <conio.h>
```

Functions

- int8_t init_hardware (zmod4xxx_dev_t ∗dev)

  Initialize the target hardware.
- int8_t is_key_pressed ()

  Check if any key is pressed.
- int8_t deinit_hardware ()

  deinitialize target hardware

5.1.1 Detailed Description

Hardware abstraction layer for windows.

Version

2.5.1

Author

Renesas Electronics Corporation

5.1.2 Function Documentation

5.1.2.1 deinit_hardware()

```
int8_t deinit_hardware ( )
```

deinitialize target hardware

Returns

error code

Return values

0	success
!= 0	error

5.1.2.2 init_hardware()

```
int8_t init_hardware (
            zmod4xxx_dev_t * dev )
```

Initialize the target hardware.

< Windows Target >

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

5.1.2.3 is_key_pressed()

```
int8_t is_key_pressed ( )
```

Check if any key is pressed.

Return values

1	pressed
0	not pressed

5.2 zmod4xxx.h File Reference

zmod4xxx-API functions

```
#include "zmod4xxx_types.h"
```

Macros

- #define **ZMOD4XXX_ADDR_PID** (0x00)
- #define **ZMOD4XXX_ADDR_CONF** (0x20)
- #define **ZMOD4XXX_ADDR_PROD_DATA** (0x26)
- #define **ZMOD4XXX_ADDR_CMD** (0x93)
- #define **ZMOD4XXX_ADDR_STATUS** (0x94)
- #define **ZMOD4XXX_ADDR_TRACKING** (0x3A)
- #define **ZMOD4XXX_LEN_PID** (2)
- #define **ZMOD4XXX_LEN_CONF** (6)
- #define **ZMOD4XXX_LEN_TRACKING** (6)
- #define **HSP_MAX** (8)
- #define **RSLT_MAX** (32)
- #define STATUS_SEQUENCER_RUNNING_MASK (0x80)
- #define STATUS_SLEEP_TIMER_ENABLED_MASK (0x40)
- #define STATUS_ALARM_MASK (0x20)
- #define STATUS_LAST_SEQ_STEP_MASK (0x1F)
- #define STATUS_POR_EVENT_MASK (0x80)
- #define STATUS_ACCESS_CONFLICT_MASK (0x40)

Functions

- zmod4xxx_err zmod4xxx_calc_factor (zmod4xxx_conf ∗conf, uint8_t ∗hsp, uint8_t ∗config)

  Calculate measurement settings.
- zmod4xxx_err zmod4xxx_calc_rmox (zmod4xxx_dev_t ∗dev, uint8_t ∗adc_result, float ∗rmox)

  Calculate mox resistance.
- zmod4xxx_err zmod4xxx_check_error_event (zmod4xxx_dev_t ∗dev)

  Check the error event of the device.

- zmod4xxx_err zmod4xxx_init_measurement (zmod4xxx_dev_t ∗dev)
  Initialize the sensor for corresponding measurement.
- zmod4xxx_err zmod4xxx_init_sensor (zmod4xxx_dev_t ∗dev)
  Initialize the sensor after power on.
- zmod4xxx_err zmod4xxx_null_ptr_check (zmod4xxx_dev_t ∗dev)
  Check if all function pointers are assinged.
- zmod4xxx_err zmod4xxx_prepare_sensor (zmod4xxx_dev_t ∗dev)
  High-level function to prepare sensor.
- zmod4xxx_err zmod4xxx_read_adc_result (zmod4xxx_dev_t ∗dev, uint8_t ∗adc_result)
  Read adc values from the sensor.
- zmod4xxx_err zmod4xxx_read_rmox (zmod4xxx_dev_t ∗dev, uint8_t ∗adc_result, float ∗rmox)
  High-level function to read rmox.
- zmod4xxx_err zmod4xxx_read_sensor_info (zmod4xxx_dev_t ∗dev)
  Read sensor parameter.
- zmod4xxx_err zmod4xxx_read_status (zmod4xxx_dev_t ∗dev, uint8_t ∗status)
  Read the status of the device.
- zmod4xxx_err zmod4xxx_read_tracking_number (zmod4xxx_dev_t ∗dev, uint8_t ∗track_num)
  Read tracking number of sensor.
- zmod4xxx_err zmod4xxx_start_measurement (zmod4xxx_dev_t ∗dev)
  Start the measurement.

5.2.1 Detailed Description

zmod4xxx-API functions

Version

2.5.1

Author

Renesas Electronics Corporation

5.2.2 Macro Definition Documentation

5.2.2.1 STATUS_ACCESS_CONFLICT_MASK

```
#define STATUS_ACCESS_CONFLICT_MASK (0x40)
```

AccessConflict

5.2.2.2 STATUS_ALARM_MASK

```
#define STATUS_ALARM_MASK (0x20)
```

Alarm

5.2.2.3 STATUS_LAST_SEQ_STEP_MASK

```
#define STATUS_LAST_SEQ_STEP_MASK (0x1F)
```

Last executed sequencer step

5.2.2.4 STATUS_POR_EVENT_MASK

```
#define STATUS_POR_EVENT_MASK (0x80)
```

POR_event

5.2.2.5 STATUS_SEQUENCER_RUNNING_MASK

```
#define STATUS_SEQUENCER_RUNNING_MASK (0x80)
```

Sequencer is running

5.2.2.6 STATUS_SLEEP_TIMER_ENABLED_MASK

```
#define STATUS_SLEEP_TIMER_ENABLED_MASK (0x40)
```

SleepTimer_enabled

5.2.3 Function Documentation

5.2.3.1 zmod4xxx_calc_factor()

```
zmod4xxx_err zmod4xxx_calc_factor (
             zmod4xxx_conf * conf,
             uint8_t * hsp,
             uint8_t * config )
```

Calculate measurement settings.

Parameters

in	*conf*	measurement configuration data
in	*hsp*	heater set point pointer
in	*config*	sensor configuration data pointer

Returns

error code

Return values

0	success

5.2.3.2 zmod4xxx_calc_rmox()

```
zmod4xxx_err zmod4xxx_calc_rmox (
              zmod4xxx_dev_t * dev,
              uint8_t * adc_result,
              float * rmox )
```

Calculate mox resistance.

Note

This is not a generic function. Only use it if indicated in your example program flow.

Parameters

in	*dev*	pointer to the device
in,out	*adc_result*	pointer to the adc results
in,out	*rmox*	pointer to the rmox values

Returns

error code

Return values

0	success
!= 0	error

5.2.3.3 zmod4xxx_check_error_event()

```
zmod4xxx_err zmod4xxx_check_error_event (
            zmod4xxx_dev_t * dev )
```

Check the error event of the device.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

5.2.3.4 zmod4xxx_init_measurement()

```
zmod4xxx_err zmod4xxx_init_measurement (
            zmod4xxx_dev_t * dev )
```

Initialize the sensor for corresponding measurement.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

Note

Before calling function, measurement data set has to be passed the dev->meas_conf

5.2.3.5 zmod4xxx_init_sensor()

```
zmod4xxx_err zmod4xxx_init_sensor (
            zmod4xxx_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

Note

Before calling function, initialization data set has to be passed the dev->init_conf

5.2.3.6 zmod4xxx_null_ptr_check()

```
zmod4xxx_err zmod4xxx_null_ptr_check (
            zmod4xxx_dev_t * dev )
```

Check if all function pointers are assinged.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

5.2.3.7 zmod4xxx_prepare_sensor()

```
zmod4xxx_err zmod4xxx_prepare_sensor (
            zmod4xxx_dev_t * dev )
```

High-level function to prepare sensor.

Parameters

in	*dev*	pointer to the device

Returns

error code

Return values

0	success
!=0	error

5.2.3.8 zmod4xxx_read_adc_result()

```
zmod4xxx_err zmod4xxx_read_adc_result (
            zmod4xxx_dev_t * dev,
            uint8_t * adc_result )
```

Read adc values from the sensor.

Parameters

in	*dev*	pointer to the device
in,out	*adc_result*	pointer to the adc results

Returns

error code

Return values

0	success
!= 0	error

5.2.3.9 zmod4xxx_read_rmox()

```
zmod4xxx_err zmod4xxx_read_rmox (
             zmod4xxx_dev_t * dev,
             uint8_t * adc_result,
             float * rmox )
```

High-level function to read rmox.

Note

This is not a generic function. Only use it if indicated in your example program flow.

Parameters

`in`	*dev*	pointer to the device
`in,out`	*adc_result*	pointer to the adc results
`in,out`	*rmox*	pointer to the rmox values

Returns

error code

Return values

0	success
!= 0	error

5.2.3.10 zmod4xxx_read_sensor_info()

```
zmod4xxx_err zmod4xxx_read_sensor_info (
             zmod4xxx_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.

5.2.3.11 zmod4xxx_read_status()

```
zmod4xxx_err zmod4xxx_read_status (
            zmod4xxx_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

5.2.3.12 zmod4xxx_read_tracking_number()

```
zmod4xxx_err zmod4xxx_read_tracking_number (
            zmod4xxx_dev_t * dev,
            uint8_t * track_num )
```

Read tracking number of sensor.

This function needs a pointer as a parameter and return tracking number. The tracking number is uint8_t type and 6 dimension array. Ex: uint8_t track_number[6]; zmod_read_tracking_number(dev, track_number); If function return success, the variable is filled with tracking number of sensor

Parameters

`in`	*dev*	pointer to the device
`in,out`	*track_num*	number pointer

Returns

error code

Return values

0	success
!= 0	error

5.2.3.13 zmod4xxx_start_measurement()

```
zmod4xxx_err zmod4xxx_start_measurement (
             zmod4xxx_dev_t * dev )
```

Start the measurement.

Parameters

`in`	*dev*	pointer to the device

Returns

error code

Return values

0	success
!= 0	error

5.3 zmod4xxx_hal.h File Reference

zmod4xxx hardware abstraction layer (HAL)

5.3.1 Detailed Description

zmod4xxx hardware abstraction layer (HAL)

Version

2.5.1

Author

Renesas Electronics Corporation

5.4 zmod4xxx_types.h File Reference

zmod4xxx types

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

Data Structures

- struct zmod4xxx_conf_str

  A single data set for the configuration.
- struct zmod4xxx_conf

  Structure to hold the gas sensor module configuration.
- struct zmod4xxx_dev_t

  Device structure ZMOD4xxx.

Typedefs

- typedef int8_t(∗ zmod4xxx_i2c_ptr_t) (uint8_t addr, uint8_t reg_addr, uint8_t ∗data_buf, uint8_t len)

  function pointer type for i2c access
- typedef void(∗ zmod4xxx_delay_ptr_p) (uint32_t ms)

  function pointer to hardware dependent delay function

Enumerations

- enum zmod4xxx_err {
  ZMOD4XXX_OK = 0, ERROR_INIT_OUT_OF_RANGE, ERROR_GAS_TIMEOUT, ERROR_I2C = -3,
  ERROR_SENSOR_UNSUPPORTED, ERROR_CONFIG_MISSING, ERROR_ACCESS_CONFLICT, ERR↩
  OR_POR_EVENT,
  ERROR_CLEANING, ERROR_NULL_PTR }

  error_codes Error codes

5.4.1 Detailed Description

zmod4xxx types

Version

2.5.1

Author

Renesas Electronics Corporation

5.4.2 Typedef Documentation

5.4.2.1 zmod4xxx_delay_ptr_p

```
typedef void(* zmod4xxx_delay_ptr_p) (uint32_t ms)
```

function pointer to hardware dependent delay function

Parameters

in	*delay*	in milliseconds

Returns

none

5.4.2.2 zmod4xxx_i2c_ptr_t

```
typedef int8_t(* zmod4xxx_i2c_ptr_t) (uint8_t addr, uint8_t reg_addr, uint8_t *data_buf, uint8←
_t len)
```

function pointer type for i2c access

Parameters

in	*addr*	7-bit I2C slave address of the ZMOD4xxx
in	*reg_addr*	address of internal register to read/write
in,out	*data*	pointer to the read/write data value
in	*len*	number of bytes to read/write

Returns

error code

Return values

0	success
!= 0	error

5.4.3 Enumeration Type Documentation

5.4.3.1 zmod4xxx_err

```
enum zmod4xxx_err
```

error_codes Error codes

Enumerator

ERROR_INIT_OUT_OF_RANGE	The initialization value is out of range.
ERROR_GAS_TIMEOUT	A previous measurement is running that could not be stopped or sensor does not respond.
ERROR_I2C	I2C communication was not successful.
ERROR_SENSOR_UNSUPPORTED	The Firmware configuration used does not match the sensor module.
ERROR_CONFIG_MISSING	There is no pointer to a valid configuration.
ERROR_ACCESS_CONFLICT	Invalid ADC results due to a still running measurement while results readout.
ERROR_POR_EVENT	Power-on reset event. Check power supply and reset pin.
ERROR_CLEANING	The maximum numbers of cleaning cycles ran on this sensor. Cleaning function has no effect anymore.
ERROR_NULL_PTR	The dev structure did not receive the pointers for I2C read, write and/or delay.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

Trademarks