ZMOD4410 - Indoor Air Quality Sensor Platform

1. Introduction

The ZMOD4410 Gas Sensor Module is highly configurable to meet various application needs. This document describes the general program flow to set up ZMOD4410 Gas Sensor Modules for gas measurements in a customer environment. It also describes the function of example code provided as C code, which can be executed using the ZMOD4410 evaluation kit (EVK), Arduino and Raspberry Pi hardware.

The corresponding firmware package is provided on the Renesas ZMOD4410 product page under the Software Downloads section. For various Renesas microcontrollers, ready-to-use code (ZMOD4xxx Sample application) is provided on the Sensor Software Modules for Renesas MCU Platforms product page.

For instructions on assembly, connection, and installation of the EVK hardware and software, see the document titled *ZMOD4410 Evaluation Kit User Manual* on the ZMOD4410 EVK product page.

The ZMOD4410 has several modes of operation:

- IAQ 2nd Gen – The embedded artificial intelligence (AI) algorithm (“iaq_2nd_gen”) derived from machine learning outputs total volatile organic compounds (TVOC), equivalent ethanol (EtOH) concentration, estimated carbon dioxide level (eCO2), and a rating for the indoor air quality (IAQ). This method of operation is for highly accurate and consistent sensor readings. **This is the recommended operation mode for IAQ**.
- IAQ 2nd Gen Ultra Low Power – The embedded artificial intelligence (AI) algorithm (“iaq_2nd_gen_ulp”) derived from machine learning outputs total volatile organic compounds (TVOC), equivalent ethanol (EtOH) concentration, estimated carbon dioxide level (eCO2), and a rating for the indoor air quality (IAQ). This method of operation offers a much lower power consumption while keeping accurate and consistent sensor readings.
- Public Building AQ Standard (PBAQ) – The embedded artificial intelligence (AI) algorithm derived from machine learning outputs total volatile organic compounds (TVOC) and equivalent ethanol (EtOH) concentration. This method of operation is for highly accurate and consistent sensor readings to fulfill public building standards.
- Relative IAQ – This lightweight algorithm reacts to air quality changes and outputs a relative IAQ index (Rel IAQ). This method of operation is recommended for threshold-based controls like air ventilation, or in applications with low computational power (lower memory footprint).
- Relative IAQ Ultra Low Power – This lightweight algorithm reacts to air quality changes and outputs a relative IAQ index (Rel IAQ). This method of operation is recommended for applications with power consumption restraints to enable threshold-based controls like air ventilation, or in applications with low computational power (lower memory footprint).
- Sulfur Odor – This semi-selective detection method for gas species allows a discrimination between sulfur odors in the air. Odors are classified as “Acceptable” and “Sulfur” with an intensity level.

The previous Odor Firmware was replaced by the Relative IAQ Firmware.

Recommendation: Before using this document, read the *ZMOD4410 Datasheet* and corresponding documentation on the ZMOD4410 product page.

Contents

Figures

Tables

2. Requirements on Hardware to Operate ZMOD4410

To operate the ZMOD4410, customer-specific hardware with a microcontroller unit (MCU) is needed. Depending on the sensor configuration and the hardware itself, the requirements differ. The following requirements are provided as an orientation only:

- 10 to 30kB program flash for ZMOD4410-related firmware code (MCU architecture and compiler dependent), see Table 1.
- 1kB RAM for ZMOD4410-related operations (see Table 1).
- Capability to perform I²C communication, timing functions (5% precision), and floating-point instructions.
- The algorithm functions work with variables saved in background and need memory retention between each call.

Table 1. Exemplary Memory Footprint of ZMOD4410 Implementation on a Renesas RL78-G13 MCU [1]

	IAQ 2nd Gen	IAQ 2nd Gen ULP	PBAQ	Relative IAQ	Relative IAQ ULP	Sulfur Odor
Program flash usage in kB	15.8	14.6	12.7	10.7	10.7	8.4
RAM usage (required variables) in bytes	496	480	436	400	400	402
RAM usage (stack size for library functions) in bytes	496	336	448	224	224	368

1. This example does not contain hardware-specific I²C and delay functions. CCRL compiler used.

The ZMOD4410 firmware can be downloaded from the ZMOD4410 product page. To get access to the firmware a Software License Agreement has to be accepted. The firmware uses floating-point calculations with various integer and floating-point variables. A part of the firmware are precompiled libraries for many standard targets (microcontrollers), as listed in the following table.

Table 2. Targets and Compilers Supported by Default

Target	Compiler
Arduino (Cortex-M0+)	arm-none-eabi-gcc (Arduino IDE)
Arm Cortex-A	arm-none-eabi-gcc (all others)
	iar-ew-arm (IAR Embedded Workbench)
Arm Cortex-M	armcc (Keil MDK)
	armclang (Arm Developer Studio, Keil MDK)
	arm-none-eabi-gcc (all others)
	iar-ew-arm (IAR Embedded Workbench)
	iar-ew-synergy-arm (IAR Embedded Workbench)
Arm Cortex-R4	arm-none-eabi-gcc (all others)
	iar-ew-arm (IAR Embedded Workbench)
Arm Linux	armv5-gcc
Espressif ESP	xtensa-esp32-elf-gcc
	xtensa-esp32s2-elf-gcc
	xtensa-lx106-elf-gcc
	riscv32-esp-elf-gcc
Intel 8051	iar-ew-8051 (IAR Embedded Workbench)
Linaro Linux	aarch64-linux-gcc
Microchip ATmega32 and AVR	avr-gcc (AVR-Studio, AVR-Eclipse, MPLAB, Atmel Studio)
Microchip PIC	xc8-cc (MPLAB)
	xc16-gcc (MPLAB)
Raspberry Pi	arm-linux-gnueabihf-gcc
Renesas RL78	ccrl (e²studio, CS+)
	iar-ew-rl (IAR Embedded Workbench)
	rl78-elf-gcc
Renesas RX	ccrx (e²studio, CS+)
	iar-ew-rx (IAR Embedded Workbench)
	rx-elf-gcc
Texas Instruments MSP430	msp430-elf-gcc
Windows	mingw32

Note: For other platforms (e.g., other Linux platforms) and other Arduino boards, contact Renesas Technical Support.

3. Structure of ZMOD4410 Firmware

To operate the ZMOD4410 and use its full functionality, five code blocks are required as displayed in Figure 1:

1. The “Target Specific I²C and Low-Level Functions” block is the hardware-specific implementation of the I²C interface. This block contains read and write functions to communicate with the ZMOD4410 and a delay function. If the Renesas EVK is used, files for the EVK HiCom Communication Board are provided with the ZMOD4410 firmware packages. Using the user’s own target hardware requires implementing the user’s target-specific I²C and low-level functions (this is highlighted in light blue in Figure 1).
2. The “Hardware Abstraction Layer (HAL)” block contains hardware-specific initialization and de-initialization functions. Files for the EVK HiCom Communication Board are provided with the ZMOD4410 firmware packages. They need to be adjusted to the target hardware of the user. The HAL is described in the document *ZMOD4xxx-API.pdf*, which is included in the firmware packages.
3. The “Application Programming Interface (API)” block contains the functions needed to operate the ZMOD4410. *The API should not be modified!* A detailed description of the API is located in the document *ZMOD4xxx-API.pdf*, which is included in the firmware packages.
4. The “Programming Example” block provides a code example as *main.*c file that is used to initialize the ZMOD4410, start the cleaning function, perform measurements, and display the data output for each example. Each example contains one configuration file (*zmod4410_config_xxx.h*) that should not be modified! More information is provided in “Description of the Programming Examples”.
5. The “Gas Measurement Libraries” block contains the functions and data structures needed to calculate the firmware-specific results for the Indoor Air Quality related parameters, such as IAQ, TVOC, EtOH, eCO2 (IAQ 2nd Gen and IAQ 2nd Gen ULP); TVOC, EtOH (PBAQ); Relative IAQ index or Sulfur Odor result. Because of the different timing, some algorithms cannot be used together (for timing information, see “Interrupt Usage and Measurement Timing”). The IAQ 2nd Gen, Relative IAQ, and Sulfur Odor libraries can be used together. The ULP algorithms (IAQ 2nd Gen ULP, Relative IAQ ULP) can also be used together. The PBAQ algorithm has another timing and cannot be combined with any other ZMOD4410 algorithm. This block also contains the cleaning library. The libraries are described in more detail in the following documents:
   - *ZMOD4410-IAQ_2nd_Gen-lib.pdf*
   - *ZMOD4410-IAQ_2nd_Gen_ULP-lib.pdf*
   - *ZMOD4410-PBAQ-lib.pdf*
   - *ZMOD4410-Rel_IAQ-lib.pdf*
   - *ZMOD4410-Rel_IAQ_ULP-lib.pdf*
   - *ZMOD4410-Sulfur_Odor-lib.pdf*

   All these files are part of the downloadable firmware packages.

To avoid naming conflicts, all API function names start with the prefix “zmod4xxx” in the ZMOD4410 code. This naming applies to all ZMOD4410 operation modes. The Arduino and Raspberry Pi examples have a similar structure but have some other features that facilitate operation with the corresponding hardware (see “Arduino Examples“ and “Raspberry Pi Example”).

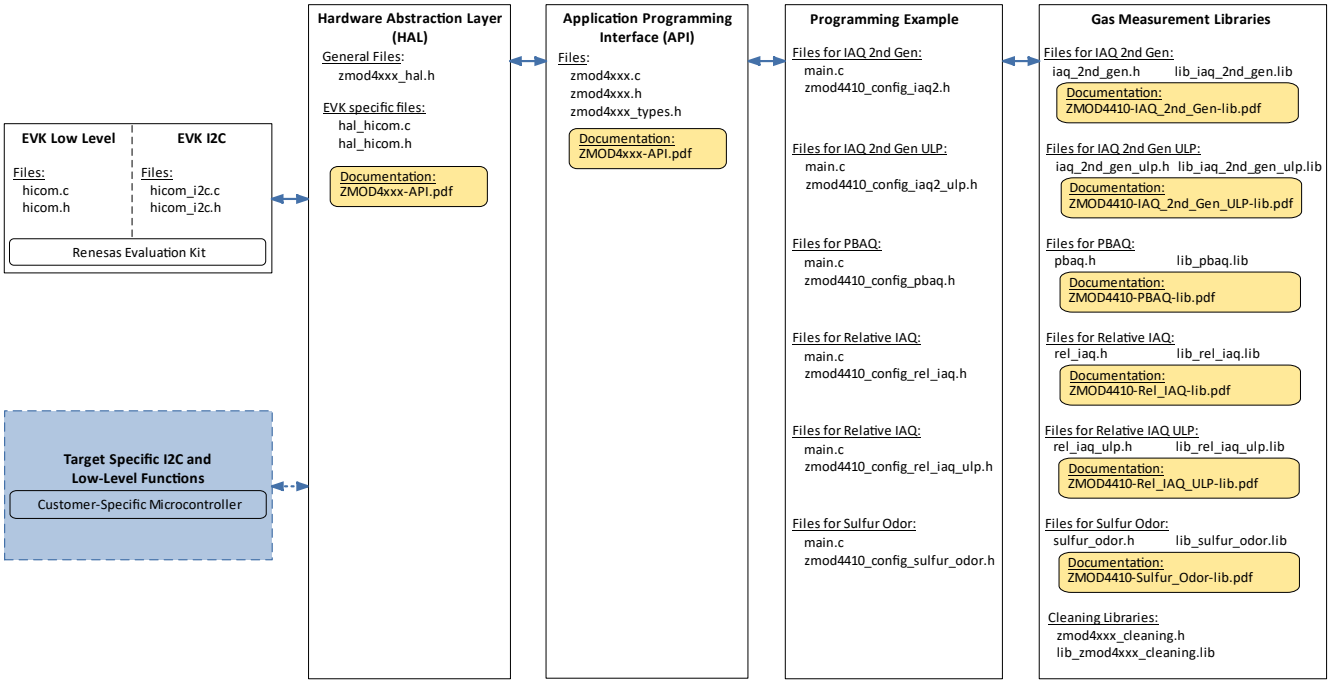


Figure 1. File Overview for ZMOD4410 Firmware

All files are part of zipped firmware packages available on the ZMOD4410 product page under the Software Downloads section. Note that not all configurations and libraries are available for all operation methods; the individual library documentation will provide detailed insight on possible settings.

4. Description of the Programming Examples

This section describes the structure of the programming examples and the steps needed to operate the sensor module. In the examples, the ZMOD4410 is initialized, the measurement is started, and measured values are outputted. They are intended to work on a Windows® computer in combination with the Renesas Gas Sensor EVK but can be easily adjusted to operate on other platforms (see “Adapting the Programming Example for Target Hardware”). To run each example using the EVK without further configuration, start the files *zmod4410_xxx_example.exe,* which are included in the firmware packages. Examples for Arduino and Raspberry Pi hardware are also introduced (see “Arduino Examples“ and “Raspberry Pi Example”).

4.1 IAQ 2nd Gen Example for EVK

The *main.c* file of the example contains the main program flow. First, the target-specific initializations are performed. The ZMOD4410 is configured by reading device parameters as well as Final Module Test parameters from the sensor’s non-volatile memory (NVM), and then initializing it to run a sequence of different operating temperatures. An endless measurement loop continuously checks the status of the ZMOD4410 and reads its data. The raw data is subsequently processed and the TVOC, EtOH, IAQ, and eCO2 algorithm results are calculated with the embedded neural net machine learning algorithm. All values are printed in the command line window. To stop the loop, press any key, which releases the hardware and stops the program. For more information, refer to the example code.

Note: The blue colored lines in the following table can be run in an endless loop.

Table 3. IAQ 2nd Gen Program Flow

Line	Program Actions	Notes	API and Algorithm Functions
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Read product ID and configuration parameters.	This step is required to select the correct configuration for the sensor.	zmod4xxx_read_sensor_info
3	Run the cleaning once.	Cleaning runs during first sensor operation.	zmod4xxx_cleaning_run
4	Calibration parameters are determined and measurement is configured.	This function must be called after every startup of ZMOD4410.	zmod4xxx_prepare_sensor
5	Initialize the IAQ (TVOC, EtOH, eCO2) algorithm.	Gas Algorithm Library function.	init_iaq_2nd_gen
6	Start the measurement.	One measurement is started.	zmod4xxx_start_measurement
7	Delay (3000ms).	This delay is necessary to keep the right measurement timing and to call a measurement every 3 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-
8	Read status register.	Check if the measurement is done. This will also be signaled on the interrupt pin with a falling signal (edge detection needed).	zmod4xxx_read_status
9	Check if an error occurred.	Check for a Power-On Reset.	zmod4xxx_check_error_event
10	Read sensor ADC output.	Result contains raw sensor output.	zmod4xxx_read_adc_result
11	Check if an error occurred.	Check for errors during ADC readout.	zmod4xxx_check_error_event
12	Algorithm calculation and sensor self-check.	Calculate current MOx resistance Rmox, clean dry air resistance Rcda, IAQ, TVOC, EtOH, and eCO2. Relative humidity (in % RH) and temperature values (in °C) and ADC results need to be passed as algo_input. First 100 samples (5 minutes) are used for minimal, hard-coded sensor warm-up. Actual warm-up can take longer (up to 48 hours).	calc_iaq_2nd_gen

4.2 IAQ 2nd Gen ULP Example for EVK

The *main.c* file of the example contains the main program flow. First, the target-specific initializations are performed. The ZMOD4410 is configured by reading device parameters as well as Final Module Test parameters from the sensor's non-volatile memory (NVM) and initializing it to run a sequence of different operating temperatures. An endless measurement loop continuously checks the status of the ZMOD4410 and reads its data. The raw data is subsequently processed and the TVOC, EtOH, IAQ, and eCO2 algorithm results are calculated with the embedded neural net machine learning algorithm. All values are printed in the command line window. To stop the loop, press any key, which releases the hardware and stops the program. For more information, refer to the example code.

Note: The blue colored lines in the following table can be run in an endless loop.

Table 4. IAQ 2nd Gen ULP Program Flow

Line	Program Actions	Notes	API and Algorithm Functions
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Read product ID and configuration parameters.	This step is required to select the correct configuration for the sensor.	zmod4xxx_read_sensor_info
3	Run the cleaning once.	Cleaning runs during first sensor operation.	zmod4xxx_cleaning_run
4	Calibration parameters are determined and measurement is configured.	This function must be called after every startup of ZMOD4410.	zmod4xxx_prepare_sensor
5	Initialize the IAQ (TVOC, EtOH, eCO2) algorithm.	Gas Algorithm Library function.	init_iaq_2nd_gen_ulp
6	Start the measurement.	One measurement is started.	zmod4xxx_start_measurement
7	Delay ().	Wait until the measurement is done. This is the first delay. It should be longer than 1010 ms.	-
8	Read status register.	Check if the measurement is done. This will also be signaled on the interrupt pin with a falling signal (edge detection needed).	zmod4xxx_read_status
9	Check if an error occurred.	Check for a Power-On Reset.	zmod4xxx_check_error_event
10	Read sensor ADC output.	Result contains raw sensor output.	zmod4xxx_read_adc_result
11	Check if an error occurred.	Check for errors during ADC readout.	zmod4xxx_check_error_event
12	Algorithm calculation and sensor self-check.	Calculate current MOx resistance Rmox, clean dry air resistance Rcda, IAQ, TVOC, EtOH, and eCO2. Relative humidity (in % RH) and temperature values (in °C) need to be passed as arguments. First 10 samples (15 minutes) are used for minimal, hard-coded sensor warm-up. Actual warm-up can take longer (up to 48 hours).	calc_iaq_2nd_gen_ulp
13	Delay ().	This second delay is necessary to keep the right measurement timing. The sum of the first and second delay should amount 90 seconds to call a measurement every 90 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-

4.3 PBAQ Example for EVK

The *main.c* file of the example contains the main program flow. First, the target-specific initializations are performed. The ZMOD4410 is configured by reading device parameters as well as Final Module Test parameters from the sensor's nonvolatile memory (NVM), and then initializing it to run a sequence of different operating temperatures. An endless measurement loop continuously checks the status of the ZMOD4410 and reads its data. The raw data is subsequently processed and the TVOC and EtOH algorithm results are calculated with the embedded neural net machine learning algorithm. All values are printed in the command line window. To stop the loop, press any key, which releases the hardware and stops the program. For more information, refer to the example code.

Note: The blue colored lines in the following table can be run in an endless loop.

Table 5. PBAQ Program Flow

Line	Program Actions	Notes	API and Algorithm Functions
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Read product ID and configuration parameters.	This step is required to select the correct configuration for the sensor.	zmod4xxx_read_sensor_info
3	Run the cleaning once.	Cleaning runs during first sensor operation.	zmod4xxx_cleaning_run
4	Calibration parameters are determined and measurement is configured.	This function must be called after every startup of ZMOD4410.	zmod4xxx_prepare_sensor
5	Initialize the PBAQ (TVOC, EtOH) algorithm.	Gas Algorithm Library function.	init_pbaq
6	Start the measurement.	One measurement is started.	zmod4xxx_start_measurement
7	Delay (5000ms).	This delay is necessary to keep the right measurement timing and to call a measurement every 5 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-
8	Read status register.	Check if the measurement is done. This will also be signaled on the interrupt pin with a falling signal (edge detection needed).	zmod4xxx_read_status
9	Check if an error occurred.	Check for a Power-On Reset.	zmod4xxx_check_error_event
10	Read sensor ADC output.	Result contains raw sensor output.	zmod4xxx_read_adc_result
11	Check if an error occurred.	Check for errors during ADC readout.	zmod4xxx_check_error_event
12	Algorithm calculation and sensor self-check.	Calculate current MOx resistance Rmox, clean dry air resistance Rcda, TVOC and EtOH. Relative humidity (in % RH) and temperature values (in °C) and ADC results need to be passed as algo_input. First 60 samples (5 minutes) are used for minimal, hard-coded sensor warm-up. Actual warm-up can take longer (up to 48 hours).	calc_pbaq

4.4 Relative IAQ Example for EVK

The *main.c* file of the example contains the main program flow. First, the target-specific initializations are performed in the example. The ZMOD4410 is configured by reading device parameters as well as Final Module Test parameters from the sensor's non-volatile memory (NVM) and initializing it to run at its operating temperature. An endless measurement loop continuously checks the status of the ZMOD4410 and reads its data. The raw data is subsequently processed and the Rel IAQ algorithm results are calculated. All values are printed in the command line window. To stop the loop, press any key, which releases the hardware and stops the program. For more information, refer to the example code.

Note: The blue colored lines in the following table can be run in an endless loop.

Table 6. Relative IAQ Program Flow

Line	Program Actions	Notes	API and Algorithm Functions
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Read product ID and configuration parameters.	This step is required to select the correct configuration for the sensor.	zmod4xxx_read_sensor_info
3	Run the cleaning once.	Cleaning runs during first sensor operation.	zmod4xxx_cleaning_run
4	Calibration parameters are determined and measurement is configured.	This function must be called after every startup of ZMOD4410.	zmod4xxx_prepare_sensor
5	Initialize the Relative IAQ algorithm.	Gas Algorithm Library function.	init_rel_iaq
6	Start the measurement.	One measurement is started.	zmod4xxx_start_measurement
7	Delay (3000ms).	This delay is necessary to keep the right measurement timing and to call a measurement every 3 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-
8	Read status register.	Check if the measurement is done. This will also be signaled on the interrupt pin with a falling signal (edge detection needed).	zmod4xxx_read_status
9	Check if an error occurred.	Check for a Power-On Reset.	zmod4xxx_check_error_event
10	Read sensor ADC output.	Result contains raw sensor output.	zmod4xxx_read_adc_result
11	Check if an error occurred.	Check for errors during ADC readout.	zmod4xxx_check_error_event
12	Algorithm calculation and sensor self-check.	Calculate current MOx resistance Rmox and Rel IAQ. ADC results need to be passed as algo_input. First 100 samples (5 minutes) are used for minimal, hard-coded sensor warm-up. Actual warm-up can take longer (up to 48 hours).	calc_rel_iaq

4.5 Relative IAQ ULP Example for EVK

The *main.c* file of the example contains the main program flow. First, the target-specific initializations are performed in the example. The ZMOD4410 is configured by reading device parameters as well as Final Module Test parameters from the sensor's non-volatile memory (NVM) and initializing it to run at its operating temperature. An endless measurement loop continuously checks the status of the ZMOD4410 and reads its data. The raw data is subsequently processed and the Rel IAQ algorithm results are calculated. All values are printed in the command line window. To stop the loop, press any key, which releases the hardware and stops the program. For more information, refer to the example code.

Note: The blue colored lines in the following table can be run in an endless loop.

Table 7. Relative IAQ ULP Program Flow

Line	Program Actions	Notes	API and Algorithm Functions
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Read product ID and configuration parameters.	This step is required to select the correct configuration for the sensor.	zmod4xxx_read_sensor_info
3	Run the cleaning once.	Cleaning runs during first sensor operation.	zmod4xxx_cleaning_run
4	Calibration parameters are determined and measurement is configured.	This function must be called after every startup of ZMOD4410.	zmod4xxx_prepare_sensor
5	Initialize the Relative IAQ algorithm.	Gas Algorithm Library function.	init_rel_iaq_ulp
6	Start the measurement.	One measurement is started.	zmod4xxx_start_measurement
7	Delay ().	Wait until the measurement is done. This is the first delay. It should be longer than 1010 ms.	-
8	Read status register.	Check if the measurement is done. This will also be signaled on the interrupt pin with a falling signal (edge detection needed).	zmod4xxx_read_status
9	Check if an error occurred.	Check for a Power-On Reset.	zmod4xxx_check_error_event
10	Read sensor ADC output.	Result contains raw sensor output.	zmod4xxx_read_adc_result
11	Check if an error occurred.	Check for errors during ADC readout.	zmod4xxx_check_error_event
12	Algorithm calculation and sensor self-check.	Calculate current MOx resistance Rmox and Rel IAQ. ADC results need to be passed as algo_input. First 10 samples (15 minutes) are used for minimal, hard-coded sensor warm-up. Actual warm-up can take longer (up to 48 hours).	calc_rel_iaq_ulp
13	Delay ().	This second delay is necessary to keep the right measurement timing. The sum of the first and second delay should amount 90 seconds to call a measurement every 90 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-

4.6 Sulfur Odor Example for EVK

The *main.c* file of the example contains the main program flow. First, the target-specific initializations are performed. The ZMOD4410 is configured by reading device parameters as well as Final Module Test parameters from the sensor's non-volatile memory (NVM) and initializing it to run a sequence of different operating temperatures. An endless measurement loop continuously checks the status of the ZMOD4410 and reads its data. The raw data is subsequently processed, and the Sulfur Odor intensity and classification results are calculated with the embedded neural net machine-learning algorithm. All values are printed in the command line window. To stop the loop, press any key, which releases the hardware and stops the program. For more information, refer to the example code.

Note: The blue colored lines in the following table can be run in an endless loop.

Table 8. Sulfur Odor Program Flow

Line	Program Actions	Notes	API and Algorithm Functions
1	Reset the sensor.	Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin.	-
2	Read product ID and configuration parameters.	This step is required to select the correct configuration for the sensor.	zmod4xxx_read_sensor_info
3	Calibration parameters are determined and measurement is configured.	This function must be called after every startup of ZMOD4410.	zmod4xxx_prepare_sensor
4	Initialize the IAQ (TVOC, EtOH, eCO2) algorithm.	Gas Algorithm Library function. Initialize again after 48 hours of sensor operation.	init_sulfur_odor
5	Start the measurement.	One measurement is started.	zmod4xxx_start_measurement
6	Read status register.	Wait until the measurement is done. This will also be signaled on the interrupt pin with a falling signal (edge detection needed).	zmod4xxx_read_status
7	Read sensor ADC output.	Result contains raw sensor output.	zmod4xxx_read_adc_result
8	Algorithm calculation.	Calculate current MOx resistance Rmox, Sulfur Odor intensity, and classification. First 60 samples (3 minutes) are used for minimal, hard-coded sensor warm-up. Actual warm-up can take longer (up to 48 hours).	calc_sulfur_odor
9	Delay (1990ms).	This delay is necessary to keep the right measurement timing and call a measurement every 3 seconds with a maximum deviation of 5% to keep the algorithm accuracy.	-
10	Start next measurement.	One measurement is started.	zmod4xxx_start_measurement

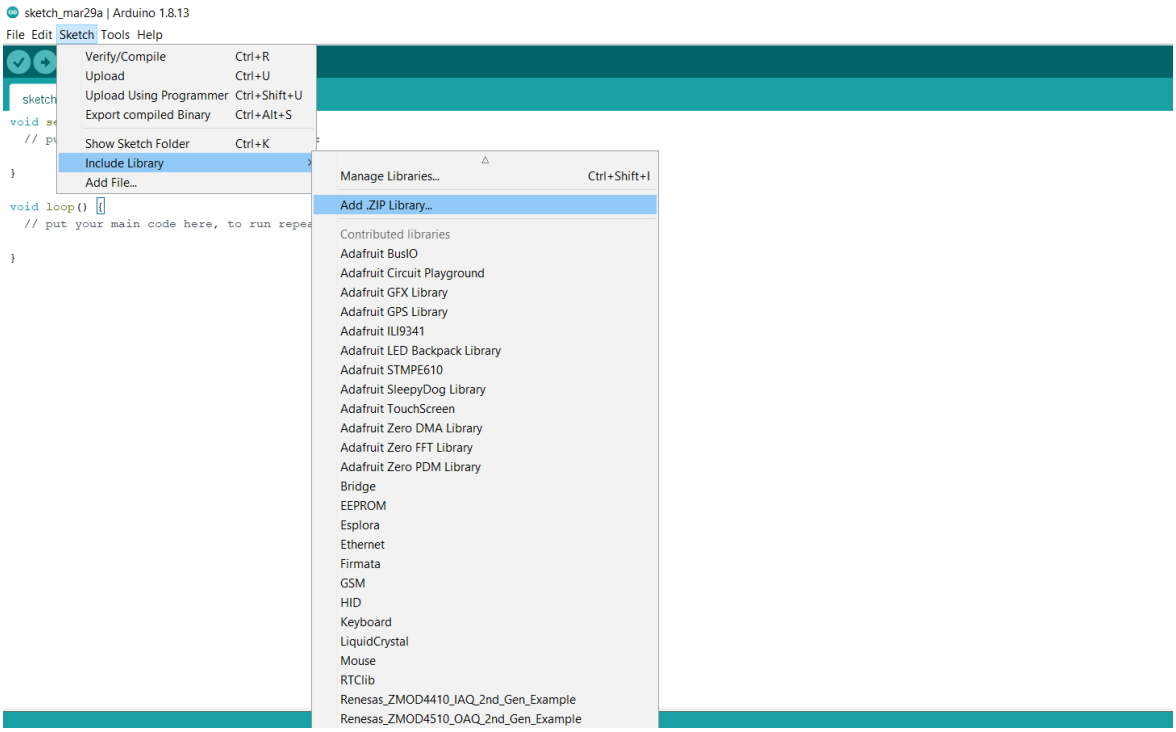
4.7 Arduino Examples

To set up a firmware for an Arduino target, Renesas provides the above-mentioned EVK examples also as Arduino example. These examples have a similar structure as shown in Figure 1 but with a HAL dedicated for Arduino, an Arduino-compatible structure, and Arduino-specific files. An Arduino IDE with version 1.8.13 and higher is needed. The example supports SAMD 32-bit ARM Cortex-M0+ based Arduino-Hardware included in the SAMD Boards library. For example:

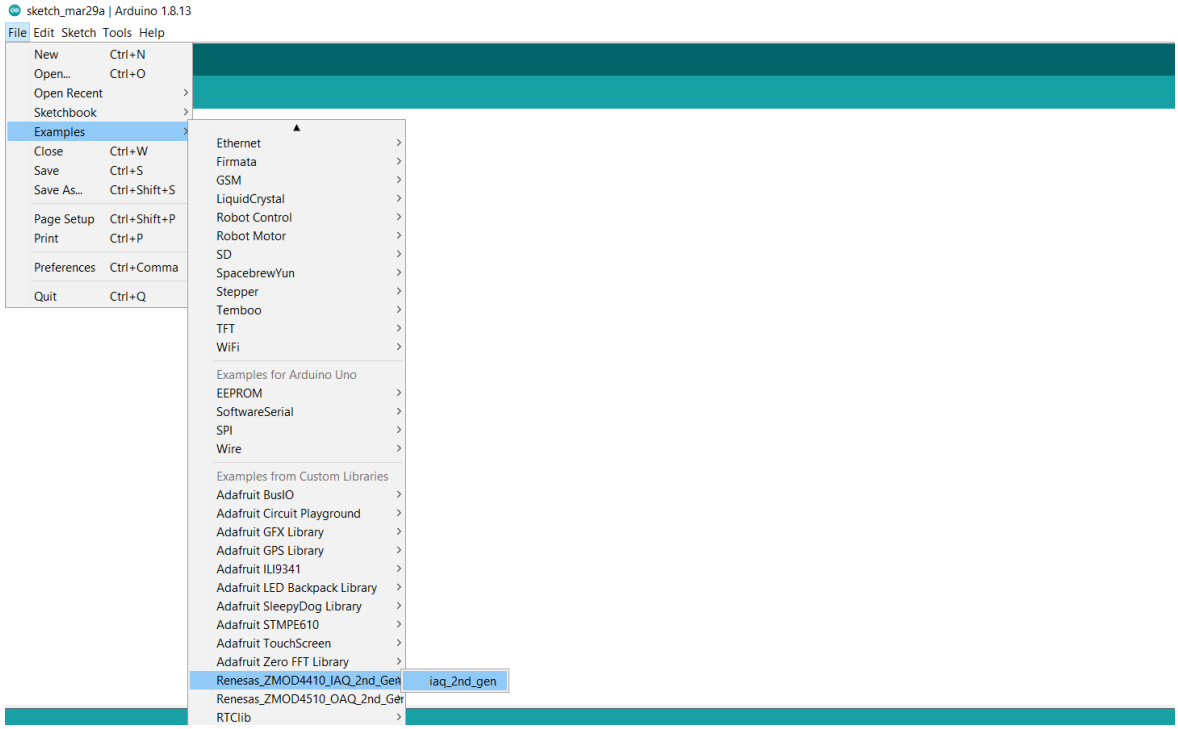
- Arduino ZERO/MKR ZERO
- Arduino MKR1000
- Arduino NANO 33 IoT
- Arduino M0
- Etc.

The Program Flows correspond to those depicted in the according EVK examples. To get the Arduino example started, complete the following steps:

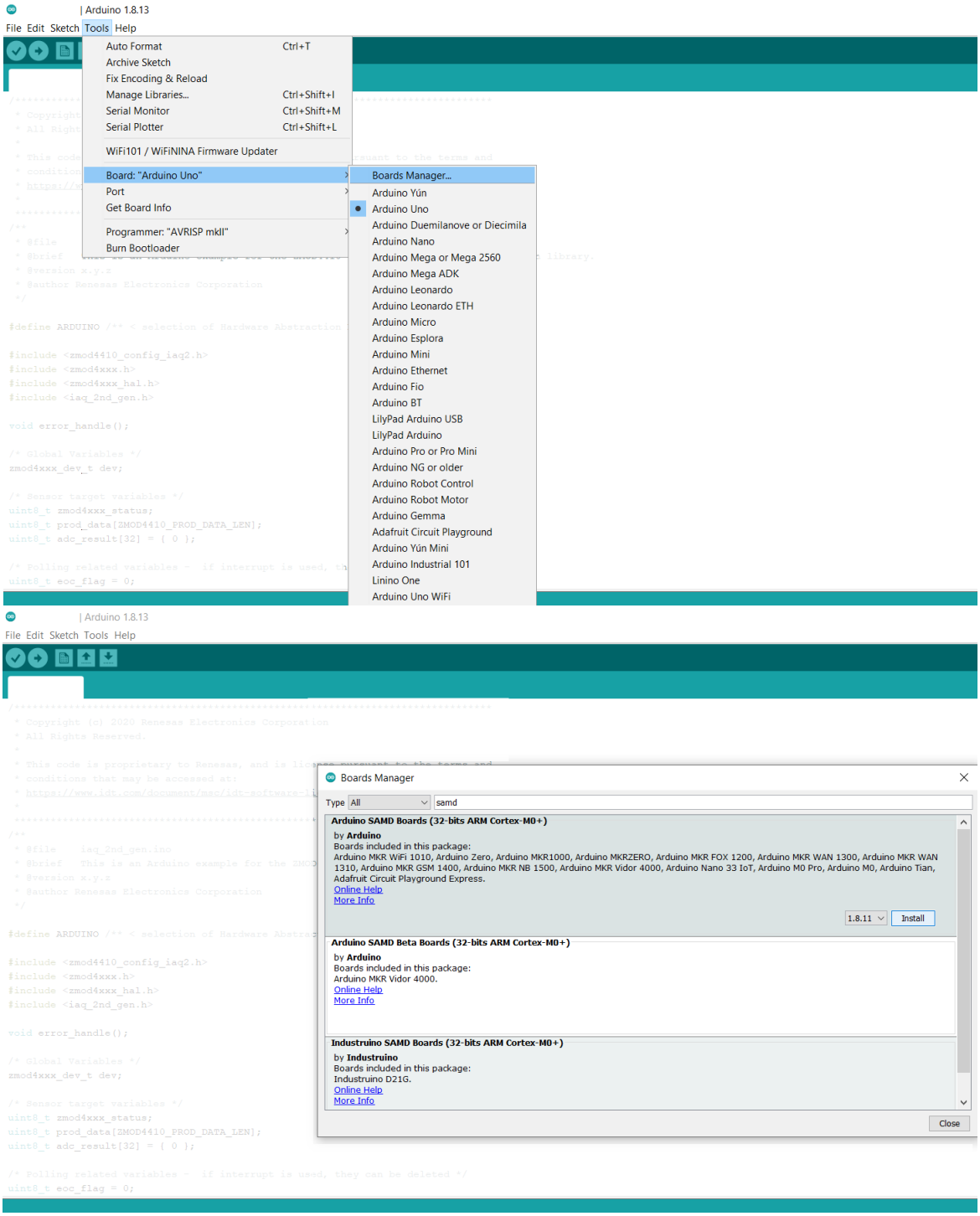
1. Connect the ZMOD4410 to the Arduino board. To connect the EVK Sensor Board, check the pin configuration on connector X1 in the *ZMOD4410 EVK User Manual* on the ZMOD4410 EVK product page.
2. Go to the Arduino example path (for example, […]\Documents\Arduino\libraries) and check if a ZMOD4410 example is existing. Old example folders must be deleted.
3. Open Arduino IDE. Select “Sketch > Include Library > Add .ZIP library”.



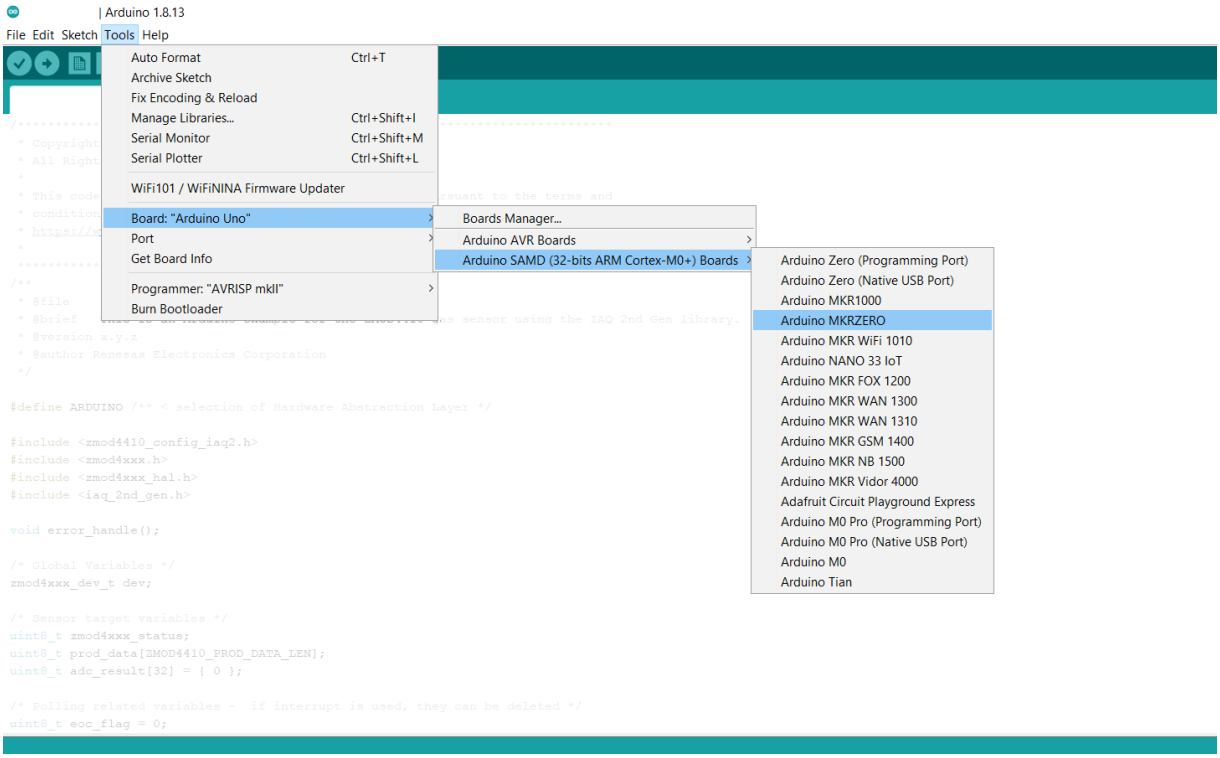
4. Select the Renesas_ZMOD4410- xxx_ Example_Arduino.zip file.
5. Select “File > Examples > Corresponding examples (Renesas_ZMOD4410_ xxx _Example_Arduino).” A new Arduino IDE window opens automatically with examples main file.



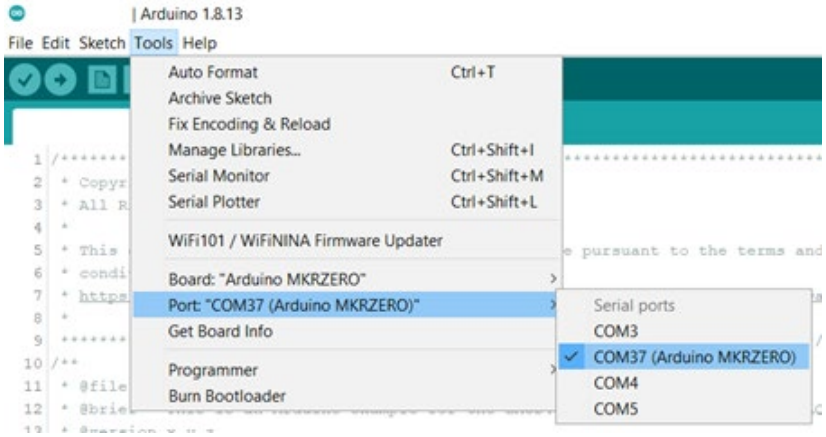
6. Install the “Arduino SAMD (32-bit ARM Cortex-M0+)” Boards library under “Tools > Board > Board Manager”. If it already exists, skip this step. Type “Arduino SAMD Boards” in the search field and click “Install” button in “Arduino SAMD (32-bit ARM Cortex-M0+)” field.



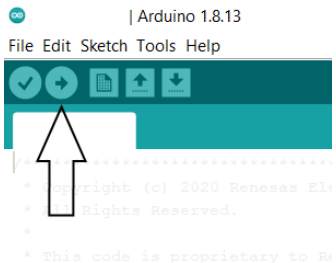
7. Select the target board with e.g. “Tools->Board > Arduino SAMD (32-bits ARM Cortex-M0+) > Arduino MKRZERO”



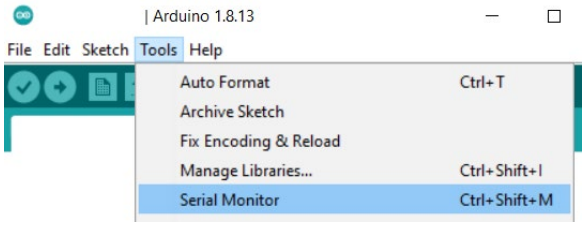
8. Compile the example with the “Verify” icon.
9. Select the connected port with “Tools -> Port -> (Connected Port)”. The correct COM-Port should show your Arduinos board name.



10. Load the program into target hardware with “Upload” icon.



11. Check results with the Serial Monitor (Tools -> Serial Monitor).



4.8 Raspberry Pi Examples

To set up a firmware for a Raspberry Pi based target, Renesas provides the above-mentioned EVK examples also as Raspberry Pi examples. These examples have a similar structure as shown in Figure 1 but with a HAL dedicated for Raspberry Pi and a Makefile to compile the code easily. The example is based on the pigpio library and Raspberry Pi OS (previously called Raspbian).

The example is tested on the following Raspberry Pi models on Raspberry Pi OS (32-bit):

- Raspberry Pi 3 B, B+
- Raspberry Pi 4 B

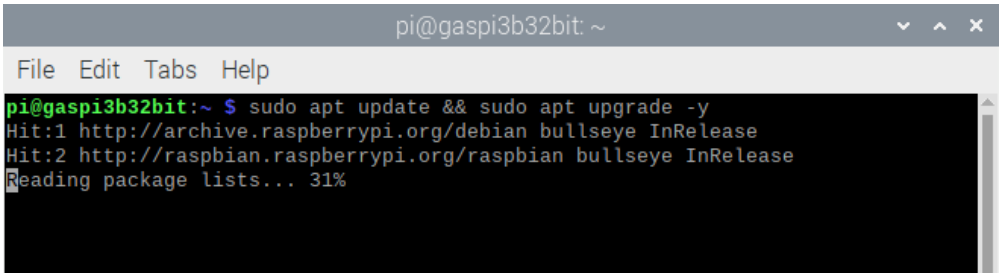
The following table describes the connection of the Sensor Board connector “X1” and the Raspberry Pi GPIO Connector. Documentation of X1 can be found in the *ZMOD4410 EVK User Manual*. Documentation of Raspberry Pi GPIO can be found on command line typing “pinout” or online.

Table 9. Connection of Sensor Board to Raspberry Pi

Sensor Board Pin (X1)	Sensor Board Description	Raspberry Pi Pin	Raspberry Pi Description
1	VDD	1, 17	3V3 power
5	SCL	3	GPIO 2 (SDA)
7	SDA	5	GPIO 3 (SCL)
14	GND	6, 9, 14, 20, 25, 30, 34, 39	Ground

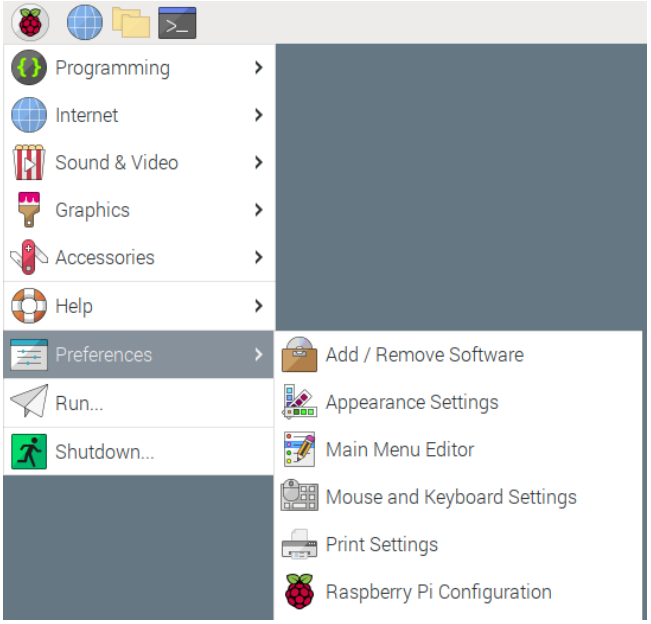
The Program Flows correspond to those displayed in the EVK examples. To get the Raspberry Pi example started, complete the following steps. The procedure is described for the IAQ 2nd Gen Example (*iaq_2nd_gen*). To adapt it for the other example, replace the corresponding name (*iaq_2nd_gen_ulp, rel_iaq, rel_iaq_ulp, sulfur_odor*).

1. Install the Raspberry Pi operating system on the Raspberry Pi. An imager tool is available to easily flash the operating system to SD card.
2. Install updates on the Raspberry Pi using the command “sudo apt update && sudo apt upgrade -y” on the Terminal.

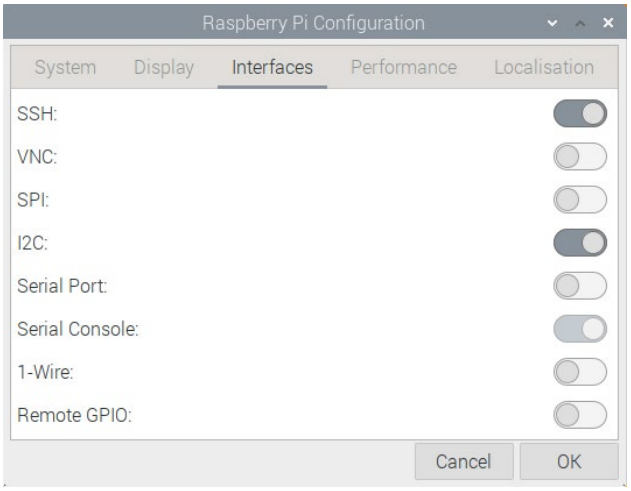


Note: The updates may take some time to finish.

3. Open the Raspberry Pi Configuration via “Start” -> ”Preferences” -> ”Raspberry Pi Configuration”.



4. Select the “Interfaces” menu and ensure that “I2C” is enabled.

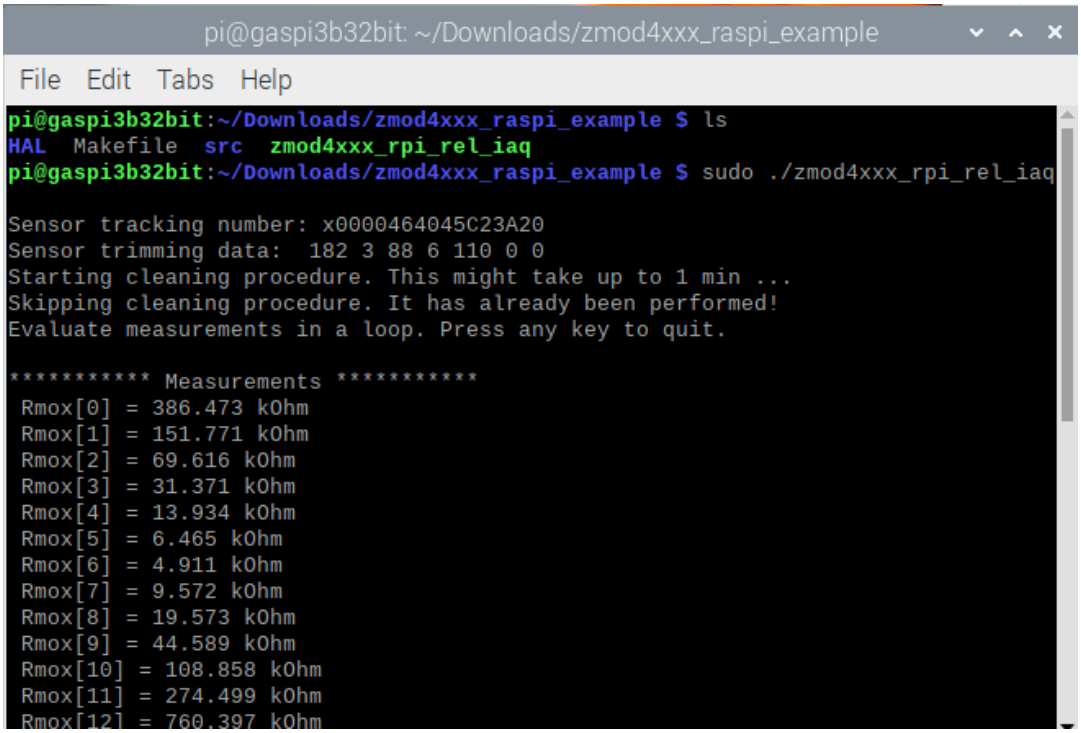


5. Reboot the Raspberry Pi to complete the initial setup. Once done, the example code can be started.
6. Copy the whole Renesas firmware package to your Raspberry Pi and extract it to your preferred location (e.g., “Downloads”).
7. Open the Terminal and go to the directory containing the example code, e.g.:

```
cd /home/pi/Downloads/Renesas_ZMOD4410_Rel_IAQ_Example/
zmod4xxx_raspi_example
```

8. Start the example with the following command (sudo is required for pigpio package):

```
sudo ./zmod4xxx_rpi_rel_iaq
```



Note: If you get an error “Can't lock /var/run/pigpio.pid”, run the command, “sudo killall pigpiod”.

4.9 Cleaning Library

The cleaning is conditioning the sensor material and helps to eliminate contaminations (e.g., solder fluxes and vapors). The cleaning process takes 1 minute and the host microcontroller is blocked during this time. Use *zmod4xxx_cleaning* library for this purpose. The example code in the firmware package shows how to use the cleaning function. The cleaning can be executed only once during sensor lifetime and is recommended to take place after PCB assembly (e.g., during final production test). The cleaning library automatically ensures that the cleaning runs only one time and all further cleanings are skipped. For this, a control bit is written into the sensor module’s NVM. Make sure to not interrupt the cleaning procedure while it is running. If issues with high contamination resulting in low sensitivity or a damage is indicated (see “Troubleshoot Sensor Damage (Sensor Self-Check)) in early product life persist, consider the package option with assembly protection sticker as additional measure (for more information, see “Package Options” in the *ZMOD4410 Datasheet*).

4.10 Troubleshoot Sensor Damage (Sensor Self-Check)

The sensor self-check helps to find issues with the sensor. It indicates that the heater or metal oxide resistance are out of range and the sensor module is probably damaged. The self-check is included in each example library and the return code of each library’s algorithm calculation function (function starts with *calc_*) is then -102. Although rare, sensor damage may be indicated during start-up. This event will usually clear up after a few measurement samples and is not problematic. However, action should be taken if the sensor is permanently in this error state (i.e., > 1 hour). The following are possible reasons for the error:

- Improper or open solder contacts
- High contamination on sensor material surface
- Wrong driver setting (wrong heating temperature)
- I²C issues
- Broken MEMS membrane

- MOx material delamination
- Oxygen depletion on sensor surface

The following measures can solve this issue:

1. Check your I²C wrapper functions for I²C read and write. It is best is to analyze the voltage levels of the SDA/SCL line and check if they match the pattern described in figure “I2C Data Transmission Protocol” in the datasheet. Complete a register check as requested in “I2C Interface and Data Transmission Protocol” in the datasheet. Check also multiple register write and read out.
2. Check to use the right sensor configuration files (*_config.h*) according to the example (see “File Overview for ZMOD4410 Firmware”) with its original content.
3. Check your production to ensure proper soldering. Use a proper soldering paste with correct temperature profile.
4. Find the source of sensor surface contamination during production (e.g., solder vapors, cleaning chemicals) and try to avoid them or use an assembly sticker. Make sure to have executed the cleaning procedure (see “Cleaning Library”).
5. Make sure the sensor module’s vent hole is not blocked because of protection paints or high dust/dirt accumulation. Remove assembly stickers if used.
6. Do not use or store the sensor in an atmosphere without oxygen (e.g., pure nitrogen).

5. Adapting the Programming Example for Target Hardware

5.1 System Hierarchy and Implementation Steps

The Renesas ZMOD4410 C API is located between the application and the hardware level.

<table>
<tr><td colspan="2">Customer Application</td></tr>
<tr><td colspan="2">Application-Specific Configuration of the Programming Example</td></tr>
<tr><td colspan="2">ZMOD4410 API and Libraries (Algorithms)</td></tr>
<tr><td colspan="2">Hardware Abstraction Layer (HAL)</td></tr>
<tr><td>Low-Level I²C Communication</td><td>Low-Level Hardware Functions</td></tr>
<tr><td colspan="2">Hardware Level (ZMOD4410 and Target)</td></tr>
</table>

Figure 2. System Hierarchy

The low-level I²C functions are implemented in the file *hicom_i2c.c* and are allocated in the *hal_hicom.c* (see Figure 1) for the EVK hardware running on a Windows-based computer and the HiCom Communication Board. To incorporate this programming example into a different hardware platform, the following steps are recommended:

1. Establish I²C communication and conduct a register test. For more information, see the “I2C Interface and Data Transmission Protocol” section in the *ZMOD4410 Datasheet*.
2. Adjust the ZMOD4410s HAL files and hardware-specific *init_hardware* and *deinit_hardware* functions to the user’s target hardware (compare with *hal_hicom.c* file). Set the device’s struct pointers *read*, *write,* and *delay_ms* in the hardware initialization by using wrapper functions. The type definitions of the function pointers can be found in *zmod4xxx_types.h* (see Figure 1) and an implementation example for the EVK in the *hicom_i2c.c*. The functions *read* and *write* should point to the I²C implementation of the hardware used. Test the *delay_ms* function with a scope plot.
3. Copy all unmodified ZMOD4410 API files and the *zmod4410_config_xxx.h* file into your project. Copy the main.c content into your code and use the example code without the algorithm library functions first. Therefore, comment out all library related code (*zmod4xxx_cleaning_run* and functions starting with *init_* and

*calc_*). Test if the adapted example runs and *zmod4xxx_read_adc_results()* function outputs changing ADC values in your main measurement loop.

4. To apply the cleaning and get the algorithm output, include the corresponding library and its header file in the extra *gas-algorithm-libraries* folder. Use precompiled libraries according to the target hardware-platform and IDE/compiler (see Table 2). Check your compiler's or IDE's manual on steps to include precompiled libraries. Uncomment the corresponding library functions (*zmod4xxx_cleaning_run* and functions starting with *init_* and *calc_*) in your code.

5.2 Error Codes

All API functions return a code to indicate the success of the operation. If no error occurred, the return code is zero. If an error occurs, a negative number is returned. The API has predefined symbols *zmod4xxx_err* for the error codes defined in *zmod4xxx_types.h*. If an error occurs, check the following table for solutions. Note that the ZMOD API cannot detect an incorrect I²C implementation. Each error may occur also with an incorrect I²C implementation.

Table 10. Error Codes (Cont. on Next Page)

Error Code	Error	Description	Solution
0	ZMOD4XXX_OK	No error.	
-1	ERROR_INIT_OUT_OF_RANGE	The initialization value is out of range.	Not used.
-2	ERROR_GAS_TIMEOUT	A previous measurement is running that could not be stopped or sensor does not respond.	1. Try to reset the sensor by powering it off/on or toggling the reset pin. Then, start the usual Program Flow as shown in "Description of the Programming Examples". 2. Check your I²C wrapper functions for I²C read and write. It is best is to analyze the voltage levels of the SDA/SCL line and check if they match the pattern described in figure "I2C Data Transmission Protocol" in the datasheet. Do a register check as requested in "I2C Interface and Data Transmission Protocol" in the datasheet. Check also multiple register write and read out.
-3	ERROR_I2C	I²C communication was not successful.	1. If available, check the error code of your parent I²C functions used in the ZMOD HAL for I2C_write/I2C_read implementation. 2. Check your I²C wrapper functions for I²C read and write. It is best is to analyze the voltage levels of the SDA/SCL line and check if they match the pattern described in figure "I2C Data Transmission Protocol" in the datasheet. Do a register check as requested in "I2C Interface and Data Transmission Protocol" in the datasheet. Check also multiple register write and read out.
-4	ERROR_SENSOR_UNSUPPORTED	The Firmware configuration used does not match the sensor module.	1. Check the part number of your device. Go to the product page at www.renesas.com/zmod4410. Under the "Downloads", you will find the right firmware for ZMOD4410. Replace it. 2. Check your I²C wrapper functions for I²C read and write. It is best is to analyze the voltage levels of the SDA/SCL line and check if they match the pattern described in figure "I2C Data Transmission Protocol" in the datasheet. Do a register check as requested in "I2C Interface and Data Transmission Protocol" in the datasheet. Check also multiple register write and read out.
-5	ERROR_CONFIG_MISSING	There is no pointer to a valid configuration.	Not used.

Error Code	Error	Description	Solution
-6	ERROR_ACCESS_CONFLICT	Invalid ADC results due to a still running measurement while results readout.	1. Check if the delay function is correctly implemented. You can use a scope plot of a GPIO pin that is switched on and off. The delay function must introduce delays in milliseconds. 2. Check measurement timing by comparing your flow with the Program Flow as shown in “Description of the Programming Examples”. Figure 3 shows graphically the right timing. Make sure to start a result readout after active measurement phase finished. See hints on measurement timing in “Interrupt Usage and Measurement Timing”. 3. Check your I²C wrapper functions for I²C read and write. It is best is to analyze the voltage levels of the SDA/SCL line and check if they match the pattern described in figure “I2C Data Transmission Protocol” in the datasheet. Do a register check as requested in “I2C Interface and Data Transmission Protocol” in the datasheet. Check also multiple register write and read out.
-7	ERROR_POR_EVENT	An unexpected reset of the sensor occurred.	1. Check stability of power supply and power/reset lines (e.g., for cross-talk). After a Power-On reset the sensor lost its configuration and must be reconfigured. If the host-controller did not lose its memory due to a restart, start with *zmod4xxx_prepare_sensor* function and continue in the Program flow as shown in “Description of the Programming Examples”.
-8	ERROR_CLEANING	The maximum numbers of cleaning cycles ran on this sensor. *zmod4xxx_cleaning_run* function has no effect anymore.	1. Using cleaning to often can harm the sensor module. The cleaning cannot be used anymore on this sensor module. Comment out the function *zmod4xxx_cleaning_run* if not needed.

-9	ERROR_NULL_PTR	The dev structure did not receive the pointers for I²C read, write and/or delay.	1. The init_hardware function (located in dependencies/zmod4xxx_api/HAL directory) contains assigning the variable of dev *read, *write and delay function pointers. These three I²C functions must be generated for the corresponding hardware and assigned in the *init_hardware* function. This is exemplary shown in *hal_hicom.c*:
			`dev->read = hicom_i2c_read;`
			`dev->write = hicom_i2c_write;`
			`dev->delay_ms = hicom_sleep;`
			Check if the assignment was done.

5.3 Interrupt Usage and Measurement Timing

The programming examples are written with delays. The microcontroller is blocked during these time periods. Depending on target hardware and the application, this can be avoided by using interrupts. The measurement sequences for each example are displayed in the following figure.

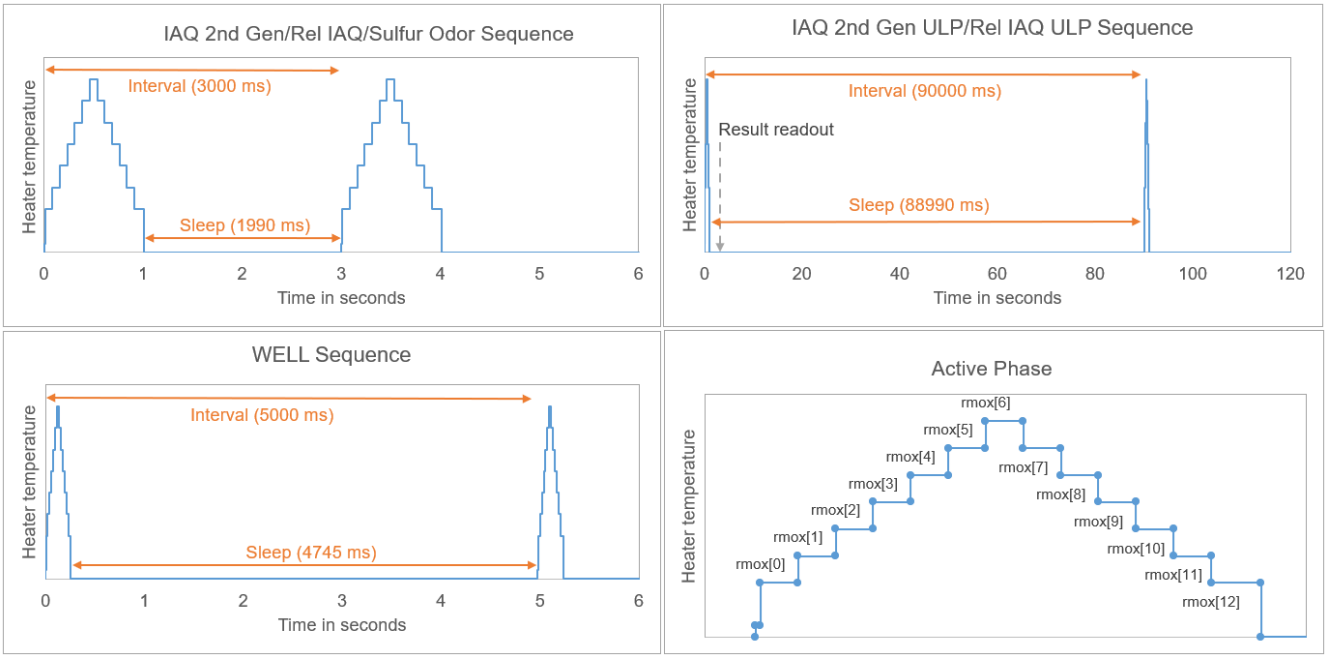


Figure 3. Measurement Sequences

An active measurement is indicated with a heater target temperature greater than zero. To lower power consumption a Sleep phase follows. This timing must be kept exactly with a maximum deviation of 5% to keep the algorithm accuracy. Each measurement must be restarted with the API command *zmod4xxx_start_measurement*.

The following interrupt usages are possible to detect the end of an active measurement:

- Using ZMODs Interrupt pin (INT) – This pin indicates the end of a measurement with a falling edge and stays LOW until the next measurement is restarted regardless if the results are read or not.
- Using Timer-based interrupts – As an alternative a timer interrupt can be used to wait until the end of the active measurement phase. Note that an ADC read-out just before the end of the active measurement phase when results are written to the registers will lead to an error. When replacing the delay make sure that the measurement is completed before the ADC readout by checking with API function *zmod4xxx_read_status* and compare the output variable *zmod4xxx_status* with *STATUS_SEQUENCER_RUNNING_MASK*. An AND link of both should give zero, otherwise the measurement was not completed.

Instead of reading out the results directly after the measurement, another option is to use just one timer interrupt with the measurement interval. Then, the ADC result read-out, error check, and algorithm calculation is done just before starting the next measurement (default for IAQ 2nd Gen, PBAQ, and Rel IAQ).

Note: This option will increase the sensor response time with the measurement interval length.

5.4 How to Compile for EVK Hardware

The EVK programming examples are designed to work with the EVK hardware. To evaluate the impact of code changes on sensor performance, it is possible to use the EVK as reference. This section provides guidelines for compiling the adapted source code into an executable file. This executable can be used with the EVK on a Windows platform. For compiling, MinGW must be installed. The folder structure should be identical to that in the

download package. The procedure is described on the IAQ 2nd Gen Example (*iaq_2nd_gen*). To adapt it for the other example, just replace the corresponding name (*iaq_2nd_gen_ulp, pbaq, rel_iaq, rel_iaq_ulp, sulfur_odor*).

1. Install MinGW:
   a. MinGW (32 bit) must be used. Mingw64 will not work due to the 32-bit FTDI library for the EVK HiCom board.
   b. Download mingw-get-setup.exe from https://osdn.net/projects/mingw/releases/.
   c. The downloaded executable file installs “MinGW Installation Manager Setup Tool”.
   d. Select required packages and mark them for installation:
      i. mingw-developer-toolkit-bin
      ii. mingw32-base-bin
      iii. mingw32-gcc-g++-bin
      iv. msys-base-bin
   e. Click “Installation” from the top-left corner and select “Update Catalogue”. You may be asked to review and apply changes.
   f. Finish the installation.
2. Add the *mingw-gcc* in system path:
   a. Open “Control Panel”, select “System”, select “Advanced System Settings”, then select “Environment Variables”.
   b. Find “Path” in System Variables then add *C:\MinGW\bin* (change the path in case MinGW is installed in different location).
3. Compiling:
   a. Go to the Command Prompt and change to the following directory of the example folder: [...]\Renesas_ZMOD4410_IAQ_2nd_Gen_Example\ zmod4xxx_evk_example
   b. Execute the following command in one line:
      gcc src*.c HAL*.c -o zmod4410_iaq_2nd_gen_example_custom.exe -DHICOM -Isrc -IHAL -I..\gas-algorithm-libraries\iaq_2nd_gen\Windows\x86\mingw32 -L. -l:HAL\RSRFTCI2C.lib -l:..\gas-algorithm-libraries\iaq_2nd_gen\Windows\x86\mingw32\lib_iaq_2nd_gen.lib -l:..\gas-algorithm-libraries\iaq_2nd_gen\Windows\x86\mingw32\lib_zmod4xxx_cleaning.lib
      Note: gcc command may need admin rights! For troubleshooting, you can try using the path to mingw gcc like “C:\MinGW\bin\gcc.exe” instead of gcc in the above command.
   c. An executable file called *zmod4410_iaq_2nd_gen_example_custom.exe* will be created.

5.5 How to Compile for Raspberry Pi Hardware

This section provides guidelines for compiling the adapted source code into an executable file. This executable can be used on the Raspberry Pi like the original provided executable file. For compiling, Make must be installed, which is a standard package in Raspberry Pi OS. The folder structure should be identical to that in the downloaded package. The procedure is described for the Relative IAQ Example (*rel_iaq*). To adapt it for another example, replace the corresponding name (*iaq_2nd _gen, iaq_2nd_gen_ulp, pbaq, rel_iaq_ulp, sulfur_odor*).

1. Complete your code changes in the source code of the Programming Example package.
2. Open the Terminal and go to the directory containing the example code. For example,

   “cd /home/pi/Downloads/Renesas_ZMOD4410_Rel_IAQ_Example/zmod4xxx_raspi_example”.
3. Type “make”, and a file called zmod4xxx_rel_iaq_rpi will be generated in the same folder.
4. Start the example with the following command (sudo is required for pigpio package). Make sure to have the I²C interface enabled (for instructions, see "Raspberry Pi Examples”).

   ```
   sudo ./zmod4xxx_rel_iaq_rpi
   ```

6. Revision History

Revision	Date	Description
1.13	Mar 7, 2023	▪ Updated to include a broader definition of Public Building Air Quality (PBAQ)
1.12	Jan 30, 2023	▪ Added WELL example description ▪ Removed Arduino ATmega32 support ▪ Completed other minor changes
1.11	Nov 7, 2022	▪ Completed minor edits.
1.10	Oct 12, 2022	▪ Added Rel IAQ and Rel IAQ ULP description ▪ Added Raspberry Pi description ▪ Updated cleaning functionality ▪ Added sensor damage self-check functionality ▪ Added RH/T compensation functionality for IAQ 2nd Gen ▪ Removed C90 support ▪ Removed Odor descriptions (legacy) ▪ Completed other minor changes
1.09	Dec 2, 2021	▪ Added IAQ 2nd Gen ULP Operation Mode description ▪ Added error code description ▪ Removed IAQ 1st Gen descriptions (legacy) ▪ Completed other minor changes
1.08	Aug 19, 2021	▪ Added Arduino description and updated target and compiler list. ▪ Added stack RAM usage. ▪ Corrected and reworked Program Flows. ▪ Extended “Interrupt Usage” description. ▪ Completed other minor changes
1.07	Sep 24, 2020	▪ Add sections “Interrupt Usage”, “Adaptions to Follow C90 Standard”, “How to Compile for EVK Hardware”. ▪ Add example for memory footprint and update target and compiler list. ▪ Refined implementation steps. ▪ Minor edits in text.
1.06	May 27, 2020	▪ Completed many changes throughout the document.
1.05	Nov 14, 2019	▪ Cleaning procedure added and explained. ▪ Figure for file overview updated.
1.04	Feb 12, 2019	▪ Update for change in the program flow for Continuous (skip the first 10 samples) and Low Power (skip the first 5 samples) Operation Modes. ▪ Implementation of plain trim value calibration. ▪ Minor edits in text.
1.03	Dec 5, 2018	▪ Update for Low Power Operation. ▪ Minor edits.
1.02	Sep 27, 2018	▪ Revision of document title from ZMOD44xx Programming Manual with ZMOD4410 Example to ZMOD4410 Programming Manual – Read Me. ▪ Full update for Odor Operation Mode 2. ▪ Minor edits.
1.00	Jun 11, 2018	Initial release.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use o any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.0 Mar 2020)

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

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.