

ZMOD4510 - Outdoor Air Quality Sensor Platform

1. Introduction

This document describes the general program flow to set up the ZMOD4510 Gas Sensor Module 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 ZMOD4510 evaluation kit (EVK), Arduino® and Raspberry Pi® hardware.

The corresponding firmware package is provided on the Renesas [ZMOD4510](#) 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 *Environmental Sensors Evaluation Kit Manual* on the [ZMOD4510 EVK](#) product page.

The ZMOD4510 has two modes of operation:

- NO2 O3 – The embedded artificial intelligence (AI) algorithm derived from machine learning outputs nitrogen dioxide (NO2) and ozone (O3) concentration and an air quality index (AQI). **This is the recommended operation mode.**
- ULP O3 – The embedded artificial intelligence (AI) algorithm derived from machine learning outputs ozone (O3) concentration and an air quality index (AQI). This operation mode offers an ultra-low power consumption (ULP) while keeping accurate and consistent sensor readings.

The previous OAQ 2nd Gen Firmware is renamed to ULP O3 firmware. OAQ 1st Gen firmware has been discontinued and its successor is the NO2 O3 firmware.

Note: This document is generic and valid for several firmware examples. Whenever the placeholder "XXX" is used, replace the name (*no2_o3*, *ulp_o3*) corresponding to the chosen package.

Recommendation: Before using this document, read the *ZMOD4510 Datasheet* and corresponding documentation on the [ZMOD4510](#) product page.

Contents

| | |
|--|----|
| 1. Introduction | 1 |
| 2. Hardware Requirements to Operate ZMOD4510..... | 3 |
| 3. Structure of ZMOD4510 Firmware..... | 5 |
| 4. Description of the Programming Examples..... | 6 |
| 4.1 NO2 O3 Example for EVK..... | 6 |
| 4.2 ULP O3 Example for EVK | 7 |
| 4.3 Compile for EVK Hardware | 7 |
| 4.4 Arduino Examples..... | 8 |
| 4.5 Raspberry Pi Examples | 12 |
| 4.6 Compile for Raspberry Pi Hardware..... | 13 |
| 4.7 Error Codes | 13 |
| 5. Adapting the Firmware Example for Target Hardware | 15 |
| 5.1 System Hierarchy and Implementation Steps | 15 |
| 5.2 Interrupt Usage and Measurement Timing..... | 16 |
| 6. Revision History | 17 |

Figures

| | |
|--|----|
| Figure 1. File Overview for ZMOD4510 Firmware..... | 5 |
| Figure 2. System Hierarchy | 15 |
| Figure 3. Measurement Sequences | 16 |

Tables

| | |
|---|----|
| Table 1. Exemplary Memory Footprint of ZMOD4510 Implementation on a Renesas RL78-G13 MCU ^[1] | 3 |
| Table 2. Targets and Compilers Supported by Default (Cont. on Next Page) | 3 |
| Table 3. NO2 O3 Example Program Flow | 6 |
| Table 4. ULP O3 Example Program Flow | 7 |
| Table 5. Connection of Sensor Board to Raspberry Pi | 12 |
| Table 6. Error Codes (Cont. on Next Page) | 13 |

2. Hardware Requirements to Operate ZMOD4510

To operate the ZMOD4510, customer-specific hardware with a microcontroller unit (MCU) is needed. Depending on the sensor configuration and the hardware itself, the requirements differ. The following minimum requirements are provided as orientation purposes only:

- 10 to 30kB program flash for ZMOD4510-related firmware code (MCU architecture and compiler dependent), see Table 1.
- 1kB RAM for ZMOD4510-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 require memory retention between each call.

Table 1. Exemplary Memory Footprint of ZMOD4510 Implementation on a Renesas RL78-G13 MCU ^[1]

| | NO2 O3 | ULP O3 |
|---|--------|--------|
| Program flash usage in kB | 21.4 | 12.0 |
| RAM usage (required variables) in bytes | 374 | 296 |
| RAM usage (stack size for library functions, worst case) in bytes | 512 | 205 |

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

The ZMOD4510 firmware can be downloaded from the product page. To get access to the firmware a Software License Agreement (SLA) 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 (Cont. on Next Page)

| 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) |
| | aarch64-linux-gnu-gcc |
| Arm Cortex-M | armcc (Keil MDK with Keil Legacy Arm Compiler 5 settings) |
| | 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 | arm-linux-gcc |

| Target | Compiler |
|----------------------------|--|
| Espressif ESP | xtensa-esp32-elf-gcc |
| | xtensa-esp32s2-elf-gcc |
| | xtensa-esp32s3-elf-gcc |
| | xtensa-lx106-elf-gcc |
| | riscv32-esp-elf-gcc |
| Intel 8051 | iar-ew-8051 (IAR Embedded Workbench) |
| Microchip ATmega32 and AVR | avr-gcc (AVR-Studio, AVR-Eclipse, MPLAB, Atmel Studio) |
| Microchip PIC | xc8-cc (MPLAB) |
| | xc16-gcc (MPLAB) |
| | xc32-gcc (MPLAB) |
| Raspberry Pi | arm-linux-gnueabi-hf-gcc |
| | aarch64-linux-gnu-gcc |
| Renesas RL78 | ccrl (e ² studio, CS+) |
| | iar-ew-rl (IAR Embedded Workbench) |
| | rl78-elf-gcc |
| Renesas RX | ccrx (e ² studio, CS+) |
| | iar-ew-rx (IAR Embedded Workbench) |
| | rx-elf-gcc |
| Texas Instruments MSP430 | msp430-elf-gcc |
| Windows | mingw32 |
| | mingw64 |

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

3. Structure of ZMOD4510 Firmware

To operate the ZMOD4510 and use its full functionality, the following five code blocks are required (see 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 ZMOD4510 and a delay function. For the use with Renesas EVKs, there are implementations for ESCOM and HiCom boards provided with the ZMOD4510 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. For the use with Renesas EVKs the Communication Board HAL is provided, which supports ESCOM and HiCom boards. The HAL is described in the document *ZMOD4510-XXX-Firmware-Documentation.pdf / .html*, which is included in the firmware packages.
3. The “Application Programming Interface (API)” block contains the functions needed to operate the ZMOD4510. The API should not be modified! A detailed description of the API is located in the document *ZMOD4510-XXX-Firmware-Documentation.pdf / .html*, which is included in the firmware packages.
4. The example.c file in the “Programming Example” block demonstrates the sensor initialization, sensor cleaning, raw sensor data readout and algorithm results computation. For more information, see “Description of the Programming Examples”.
5. The “Gas Measurement Libraries” block contains one configuration file (*zmod4510_config_xxx.h*) that should not be modified! Furthermore, a library needed to calculate the nitrogen oxide (NO₂, only NO₂ O₃ mode) and ozone (O₃) concentrations, and the resulting Air Quality Index (AQI) is provided. Because of the different timing, some algorithms cannot be used together (for timing information, see “Interrupt Usage and Measurement Timing”). This block also contains the cleaning library. The libraries are described in more detail in *ZMOD4510-XXX-Firmware-Documentation.pdf / .html*.

To avoid naming conflicts, all API function names start with the prefix “zmod4xxx” in the ZMOD4510 code. The Arduino and Raspberry Pi examples have a similar structure but have some other features that facilitate operation with the corresponding hardware (see “Arduino Example” and “Raspberry Pi Example”).

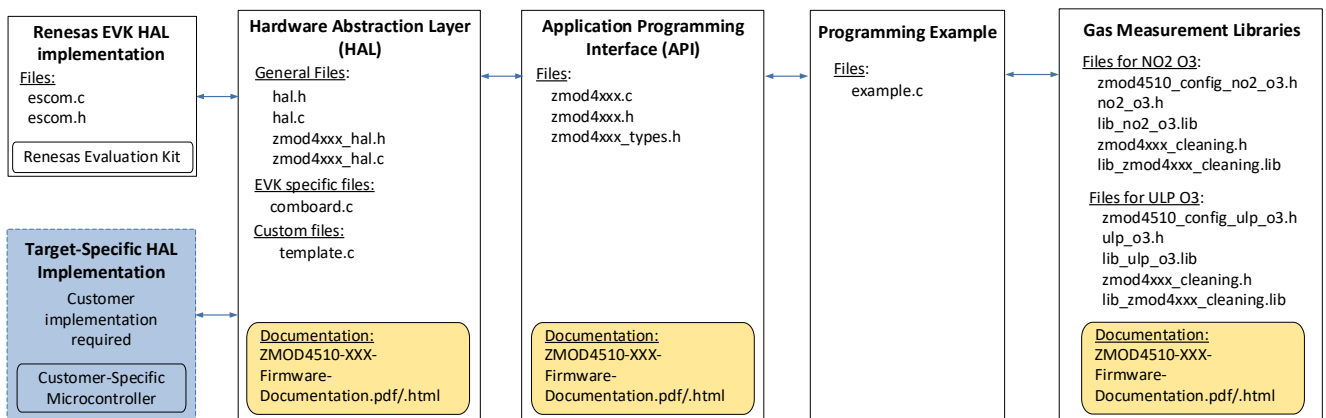


Figure 1. File Overview for ZMOD4510 Firmware

All files are part of a zipped firmware packages available on the [ZMOD4510](#) product page under the Software Downloads section. Note that not all configurations and libraries are available for all operation modes; the individual library documentation will provide detailed insight on possible settings.

4. Description of the Programming Examples

This section describes the structure of the firmware example and the steps needed to operate the sensor module. In the example, the ZMOD4510 is initialized, the measurement is started, and measured values are outputted. The example is intended to work on a Windows® computer in combination with the Renesas Environmental Sensor EVK, however, it can be easily adjusted to operate on other platforms (see “Adapting the Firmware Example for Target Hardware”). To run the example using the EVK without further configuration, start the file *XXX-example.exe*, which is included in the firmware packages. Examples for Arduino and Raspberry Pi hardware are also introduced (see “Arduino Example” and “Raspberry Pi Example”).

Note: Running the executable with legacy EVKs (HiCom) requires an FTDI driver to be available on the host computer.

4.1 NO2 O3 Example for EVK

The *example.c* file of the example contains the main program flow. First, the target-specific initializations are performed. The ZMOD4510 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. A measurement loop runs a sequence of changing operating temperatures, continuously checks the status and errors of the ZMOD4510, and reads its data. The raw data is subsequently processed. The nitrogen dioxide and ozone gas concentration, the fast output for AQI and standardized AQI 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 Ctrl + C, which releases the hardware and stops the program. For more information, refer to the example code.

Table 3. NO2 O3 Example Program Flow

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

| Line | Program Actions | Notes | Function |
|------|--|---|----------------------------|
| 1 | Reset the sensor. | Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin. | - |
| 2 | Initialize the ZMOD4510 sensor. | This function must be used after each start-up. | detect_and_configure |
| 3 | Initialize the algorithm. | Gas Algorithm Library function. | init_no2_o3 |
| 4 | Start the measurement. | One measurement is started every 6 seconds and takes 4020ms. ^[1] | zmod4xxx_start_measurement |
| 5 | Delay (6000ms). | This delay is necessary to keep the right measurement timing and to call a measurement every 6 seconds with a maximum deviation of 5% to keep the algorithm accuracy. ^[1] | - |
| 6 | Read the measurement. | Read every 6 seconds. ^[1] | read_and_verify |
| 7 | Algorithm calculation and sensor self-check. | Calculate current MOx resistance Rmox, ozone and nitrogen dioxide gas concentration in ppb, fast output for AQI and 1h/8h AQI rating according to EPA standard ^[1] . Relative humidity (in % RH) and temperature values (in °C) need to be passed as arguments. The first 50 samples (5 minutes) are used for minimal, hard-coded sensor stabilization. Actual stabilization can take longer (e.g., up to 48 hours). | calc_no2_o3 |

1. For a detailed timing diagram, see section 5.2.

¹ Source: <https://www.airnow.gov/sites/default/files/2020-05/aqi-technical-assistance-document-sept2018.pdf>

4.2 ULP O3 Example for EVK

The *example.c* file of the example contains the main program flow. First, the target-specific initializations are performed. The ZMOD4510 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. A measurement loop runs a sequence of changing operating temperatures, continuously checks the status and errors of the ZMOD4510, and then reads its data. The raw data is subsequently processed. The ozone gas concentration, the fast output for AQI, and standardized AQI 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 Ctrl + C, which releases the hardware and stops the program. For more information, refer to the example code.

Table 4. ULP O3 Example Program Flow

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

| Line | Program Actions | Notes | Function |
|------|--|--|----------------------------|
| 1 | Reset the sensor. | Before configuring the sensor, reset the sensor by powering it off/on or toggling the reset pin. | - |
| 2 | Initialize the ZMOD4510 sensor. | This function must be used after each start-up. | detect_and_configure |
| 3 | Initialize the algorithm. | Gas Algorithm Library function. | init_ulp_o3 |
| 4 | Start the measurement. | One measurement is started every 2 seconds and takes 18ms. ^[1] | zmod4xxx_start_measurement |
| 5 | Delay (6000ms). | This delay is necessary to keep the right measurement timing and to call a measurement every 2 seconds with a maximum deviation of 5% to keep the algorithm accuracy. ^[1] | - |
| 6 | Read the measurement. | Read every 2 seconds. ^[1] | read_and_verify |
| 7 | Algorithm calculation and sensor self-check. | Calculate current MOx resistance Rmox, ozone gas concentration in ppb, fast output for AQI and 1h/8h AQI rating according to EPA standard ^[2] . Relative humidity (in % RH) and temperature values (in °C) need to be passed as arguments. The first 900 samples (30 minutes) are used for minimal, hard-coded sensor stabilization. Actual stabilization can take longer (e.g., up to 48 hours). | calc_ulp_o3 |

1. For a detailed timing diagram, see section 5.2.

4.3 Compile for EVK Hardware

The EVK firmware example is 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 describes how to compile the adapted source code into an executable file, which can then be used with the EVK on a Windows platform. The firmware folder structure should be identical to that in the download package.

To compile for 64-bit Windows, "[skeeto/w64devkit](#)" must be downloaded and unzipped.

Install skeeto/w64devkit:

1. Download the latest version of w64devkit-<VERSION>.zip and unzip it.

² Source: <https://www.airnow.gov/sites/default/files/2020-05/aqi-technical-assistance-document-sept2018.pdf>

Compiling:

1. Go to the command prompt and add skeeto/w64devkit to the system path in command line by using:
`set PATH=<PATH_TO_W64DEVKIT>\bin;%PATH%`
2. Change to the following directory of the firmware folder:
`[...]\Renesas-ZMOD4510-XXX-Firmware\src`
3. Execute the following command:
`make`
An executable file called *XXX-example.exe* will be created in folder *build*.

To compile for 32-bit Windows, “MinGW” must be installed.

1. Install MinGW:
 - a. MinGW (32 bit) must be used.
 - b. Download *mingw-get-setup.exe* from <https://osdn.net/projects/mingw/releases/>.
 - c. The downloaded executable file installs “Install MinGW Installation Manager”.
 - d. Select the required packages:
 - 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”.
 - f. Finish installation.

Compiling:

1. Go to the command prompt and add MinGW to the system path in command line by using:
`set PATH=<PATH_TO_MinGW>\bin;<PATH_TO_MinGW>\msys\1.0\bin;%PATH%`
2. Change to the following directory of the example folder: `[...]\Renesas-ZMOD4510-XXX-Firmware\src`
3. Execute the following command:
`make ARCH=32 GCC=mingw32-gcc`
An executable file called *XXX-example.exe* will be created in folder *build*.

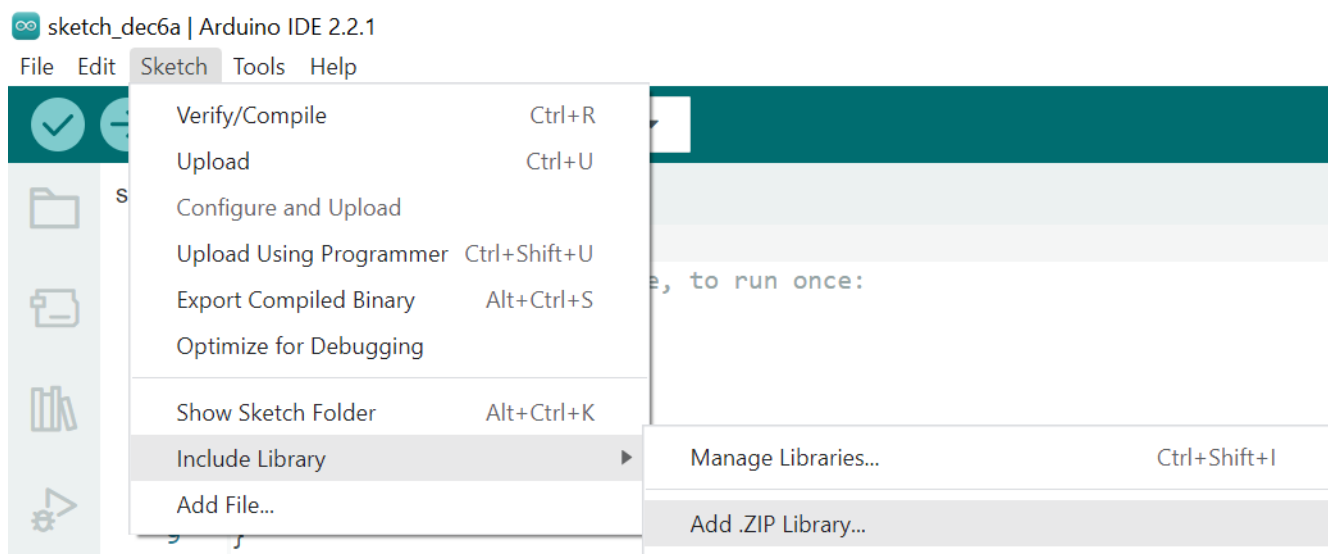
Note: Running compiled executable with legacy EVKs (HiCom) requires an FTDI driver to be available on the host computer.

4.4 Arduino Examples

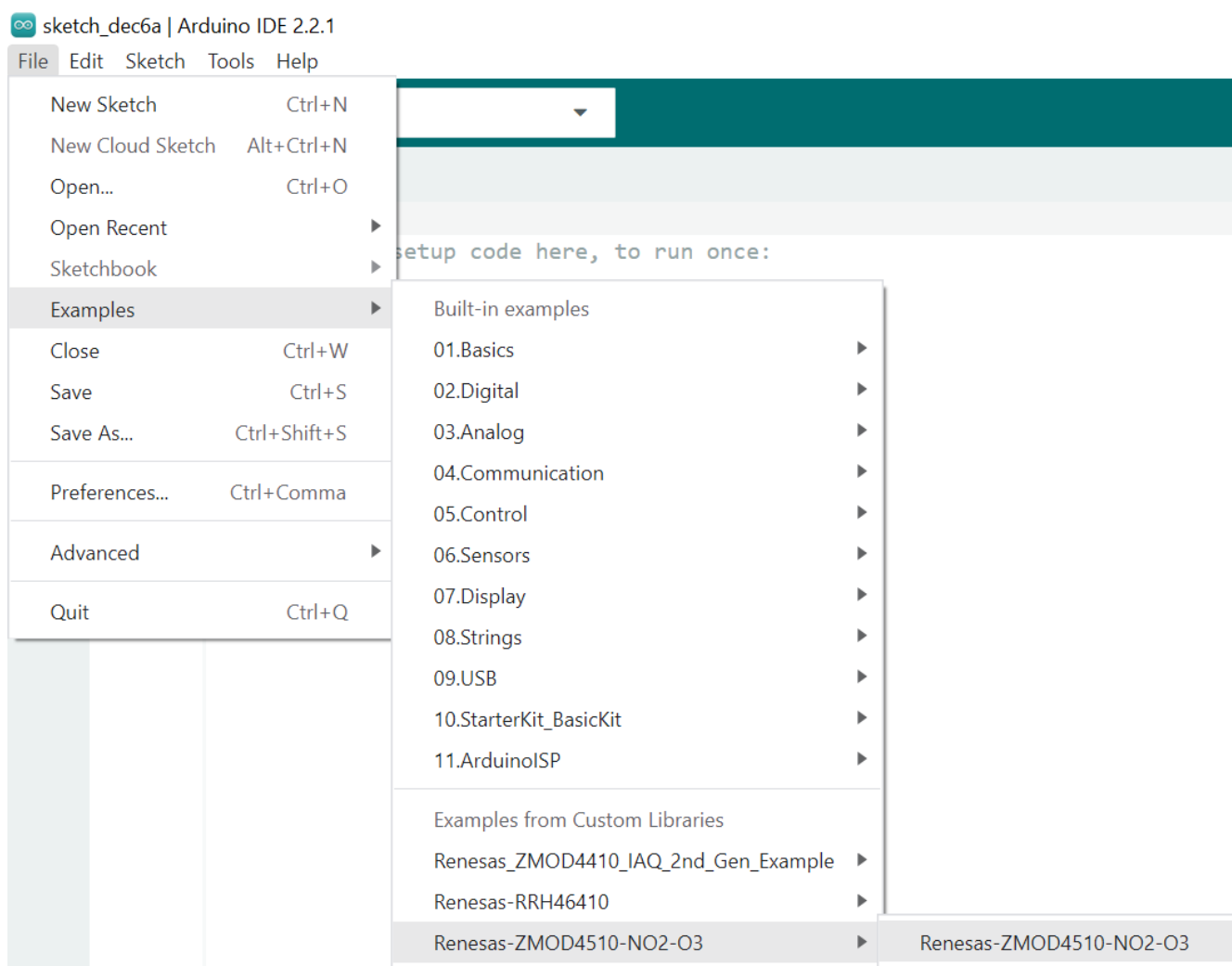
To set up a firmware for an Arduino target, Renesas provides the above-mentioned EVK example also as an Arduino example. This example is a high-level Arduino library and has a similar structure as shown in Figure 1 but with a HAL dedicated for Arduino, an Arduino-compatible structure, and Arduino-specific files. An Arduino IDE with version 2.0.0 or higher is needed.

The Program Flow corresponds to that depicted in the EVK example. To get the Arduino example started, complete the following steps (example shown for SAMD 32-bit ARM Cortex-M0+ based Arduino hardware):

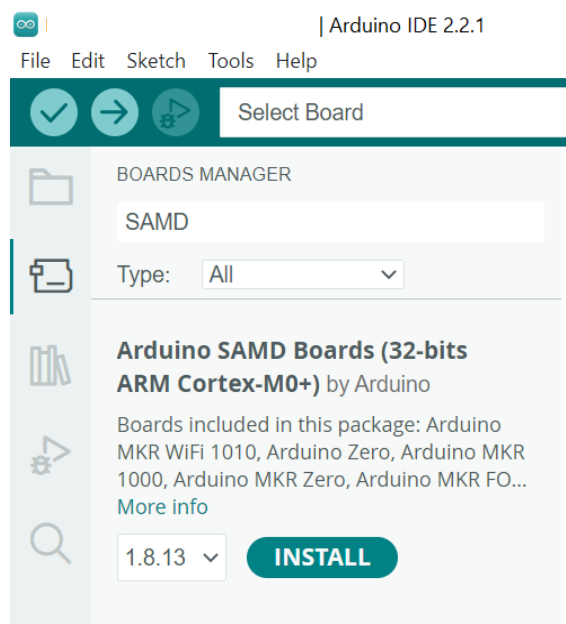
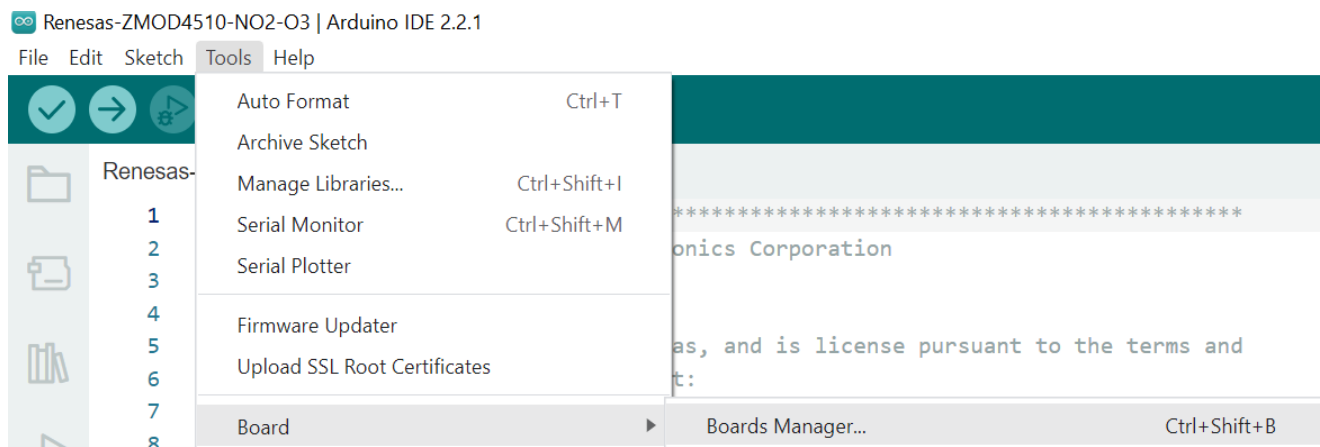
1. Connect the ZMOD4510 to the Arduino board. To connect the EVK Sensor Board, check the pin configuration on connector “X1” in the *Environmental Sensors Evaluation Kit Manual* on [ZMOD4510 EVK](#) product page.
2. Go to the Arduino example path (for example `[...]\Documents\Arduino\libraries`) and check if a ZMOD4510 example exists. Old example folders must be deleted.
3. Open Arduino IDE. Select “Sketch > Include Library > Add .ZIP Library”.



4. Select the Renesas-ZMOD4510-XXX-Firmware-Arduino.zip file.
5. Select “File > Examples > Renesas-ZMOD4510-XXX.” A new Arduino IDE window opens automatically with examples main file.

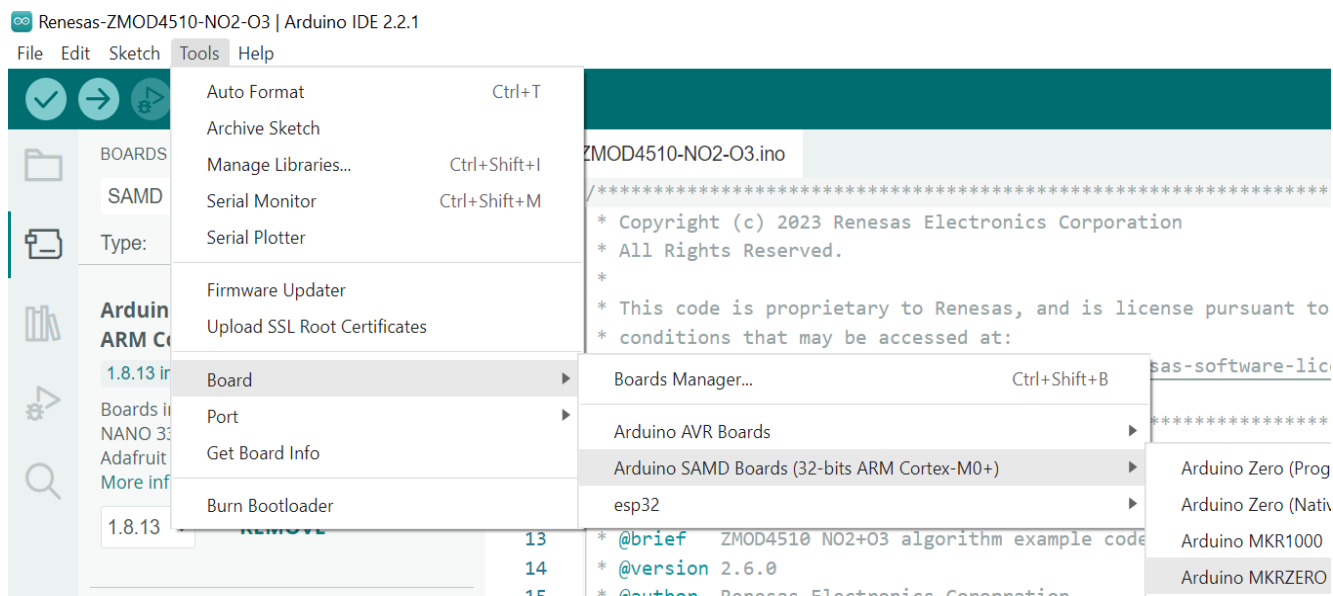


6. (This step may not be required for other Arduino hardware.) Install “Arduino SAMD (32-bit ARM Cortex-M0+)” Boards library under “Tools > Board > Board Manager”.

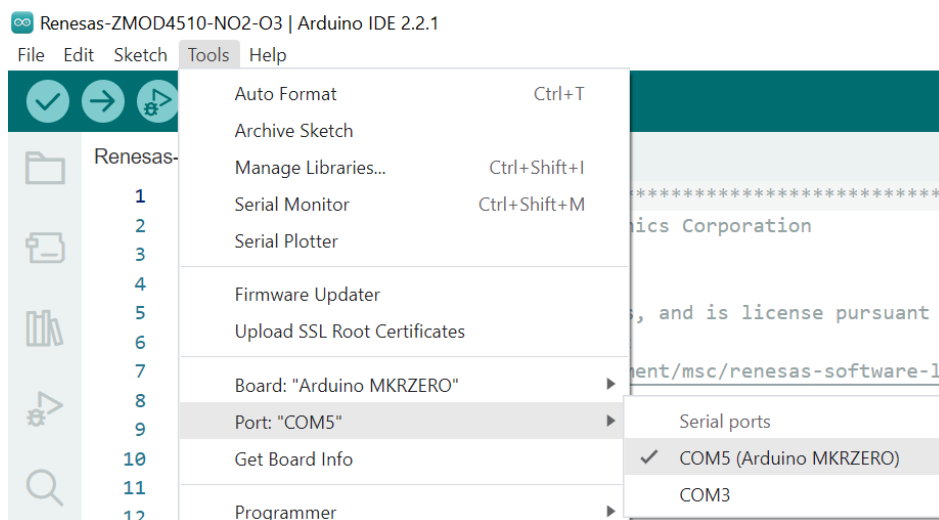


If it already exists, skip this step. Type “Arduino SAMD Boards” in search field and click the “Install” button in “Arduino SAMD (32-bits ARM Cortex-M0+)” field.

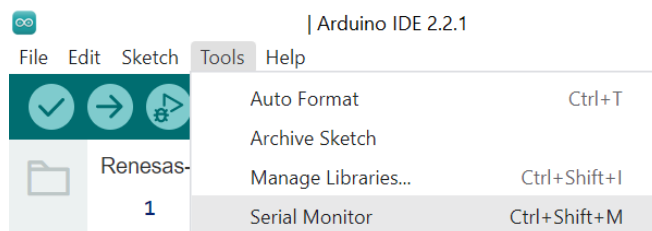
7. Select the target board under “Tools > Board > Arduino SAMD (32-bits ARM Cortex-M0+) > Arduino MKRZERO”.



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 Arduino's board name.



10. Load the program into the target hardware with the “Upload” icon.
11. Check the results with the Serial Monitor (Tools > Serial Monitor).



Note: The sensor may execute the cleaning procedure during the first operation, which takes one minute to complete. Afterwards, the results are shown.

4.5 Raspberry Pi Examples

To set up firmware for a Raspberry Pi based target, Renesas provides the above-mentioned EVK examples also as a 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 easily compile the code. 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 *Environmental Sensors Evaluation Kit Manual*. Documentation of Raspberry Pi GPIO can be found on command line typing “pinout” or [online](#).

Table 5. 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 |
| 7 | SDA | 3 | GPIO 2 (SDA) |
| 5 | SCL | 5 | GPIO 3 (SCL) |
| 14 | GND | 6, 9, 14, 20, 25, 30, 34, 39 | Ground |

The Program flows corresponds to those displayed in the EVK example. To get the Raspberry Pi example started, complete the following steps:

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. To configure the I²C interface go to `/boot` directory and open the configuration file:

```
sudo nano config.txt
```
3. Enable the I²C interface and change the baud rate by uncommenting the *line* `#dtoverlay=i2c-arms` and changing it to:

```
dtoverlay=i2c-arms,i2c-arms-baudrate=200000
```
4. Reboot the Raspberry Pi to complete the initial setup. Once done, the example code can be started.
5. Copy the whole Renesas firmware package to your Raspberry Pi and extract it to your preferred location (e.g., “Downloads”).
6. Open the Terminal and go to the directory containing the executable, e.g.:

```
cd ~/Downloads/Renesas-ZMOD4510-XXX-Firmware/raspberrypi/
```
7. Start the example with the following command (sudo is required for pigpio package):

```
sudo ./XXX-example
```

You may have to assign yourself execute permissions with `chmod 544 XXX-example`. If you get an error “Can't lock /var/run/pigpio.pid”, run the following command:

```
sudo killall pigpiod
```

Also you can try to establish an internet connection via Wi-Fi or LAN and install updates on the Raspberry Pi using the command `sudo apt update && sudo apt upgrade -y` on the Terminal. The updates may take some time to finish.

4.6 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.

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

```
cd ~/Downloads/Renesas-ZMOD4510-XXX-Firmware/src
```
3. Type `make`, and a file called `XXX-example` will be generated in the folder named `build`.
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 Example”).

```
sudo build/XXX-example
```

4.7 Error Codes

All API functions return a code to indicate the success of the operation. If an error does not occur, 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 6. Error Codes (Cont. on Next Page)

| Error Code | Error | Description | Solution |
|------------|-------------------------|---|---|
| 1 | XXX_STABILIZATION | Sensor is in stabilization. | Algorithm results obtained during this period should not be considered as valid outputs. Ignore the outputs. |
| 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. | <ol style="list-style-type: none"> 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 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. Also check multiple register write and read out. |
| -3 | ERROR_I2C | I ² C communication was not successful. | <ol style="list-style-type: none"> 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 if the sensor correctly wired and if the voltage levels are appropriate. 3. Check your I²C wrapper functions for I²C read and write. It is best 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. Also check multiple register write and read out. |

| Error Code | Error | Description | Solution |
|------------|--------------------------|--|---|
| -4 | ERROR_SENSOR_UNSUPPORTED | The Firmware configuration used does not match the sensor module. | <ol style="list-style-type: none"> 1. Check the part number of your device. Go to the product page at www.renesas.com/zmod4510. Under "Downloads", you will find the right firmware for ZMOD4510. Replace it. 2. Check your I²C wrapper functions for I²C read and write. It is best 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. |
| -6 | ERROR_ACCESS_CONFLICT | Invalid ADC results due to a still running measurement while results readout. | <ol style="list-style-type: none"> 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 correct 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 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. | <ol style="list-style-type: none"> 1. Check stability of power supply and power/reset lines (e.g. for crosstalk). 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 <i>zmod4xxx_prepare_sensor</i> 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. <i>zmod4xxx_cleaning_run</i> function has no effect anymore. | Using cleaning too often can harm the sensor module. The cleaning cannot be used anymore on this sensor module. Comment out the function <i>zmod4xxx_cleaning_run</i> if not needed. |
| -9 | ERROR_NULL_PTR | The <i>hal</i> structure did not receive the pointers for I ² C read, write, delay and/or reset. | <ol style="list-style-type: none"> 1. The HAL_Init function contains assigning the variable of <i>hal</i> i2cRead, i2cWrite and msSleep function pointers. These three functions have to be generated for the corresponding hardware and assigned in the <i>init_hardware</i> function. This is exemplary shown in template.c: <pre> hal -> msSleep = _Sleep; hal -> i2cRead = _I2CRead; hal -> i2cWrite = _I2CWrite; </pre> Check if the assignment was done. |
| -102 | XXX_DAMAGE | Sensor is probably damaged. | If an error occurs at the beginning of a sensor's lifetime, wait for 60 minutes and check if the error disappears. For more information, see "Conditioning and Sensor Self-Check Status" in the <i>ZMOD4510 Datasheet</i> . |

5. Adapting the Firmware Example for Target Hardware

5.1 System Hierarchy and Implementation Steps

The Renesas ZMOD4510 C API is located between the application and the hardware level.

| |
|--|
| Customer Application |
| Application-Specific Configuration of the Firmware Example |
| ZMOD4510 API and Libraries (Algorithms) |
| Hardware Abstraction Layer (HAL) |
| Hardware Level (ZMOD4510 and Target) |

Figure 2. System Hierarchy

The ZMOD4510 example code uses a Hardware Abstraction Layer to separate target hardware implementation details from the actual sensor interface. To transfer the example to a different hardware platform, the following steps are recommended:

1. Recursively copy everything from *src* folder into your project.
2. Remove all subdirectories from *src/hal* except *custom*.
3. Adapt the file *src/hal/custom/template.c* to work with your hardware. This requires customized implementations of the `_I2CRead`, `_I2CWrite` and `_Sleep` functions. For more information, see the “I²C Interface and Data Transmission Protocol” section of the *ZMOD4510 Datasheet* and the *ZMOD4510-XXX-Firmware-Documentation.pdf* and *.html*. If possible, try to verify your implementation with a scope or logic analyzer. Conduct a register test as described in the datasheet.
4. Modify the file *src/example.c* as required or copy its functionality to your own application code. Make sure to leave the order of operations and timing unchanged.
Before continuing to the next step, you can also try compiling and running the application without the algorithm libraries. Comment out calls to `zmod4xxx_cleaning_run`, `init_*` and `calc_*` functions and check if `zmod4xxx_read_adc_results()` function outputs changing ADC values in main measurement loop.
5. Link the application code with the algorithm and cleaning libraries matching your microcontroller and compiler. A list of supported MCUs and compilers is provided in Table 2.

5.2 Interrupt Usage and Measurement Timing

The firmware example is 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 sequence for the algorithm is displayed in the following figure.

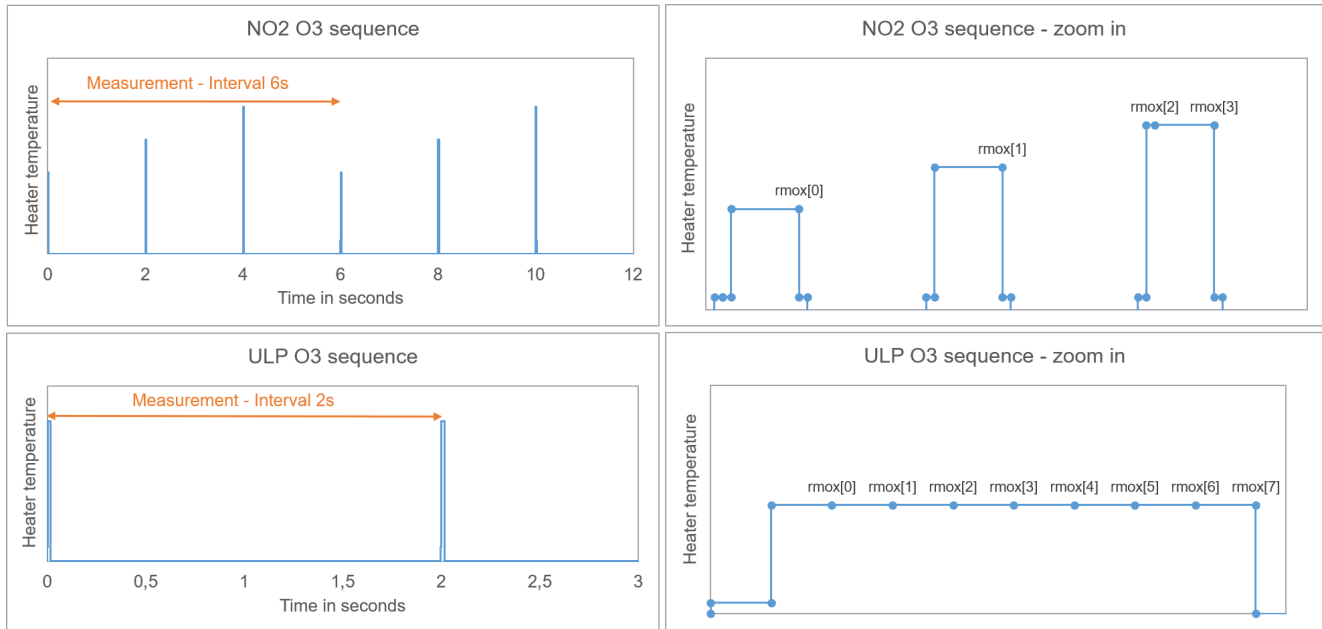


Figure 3. Measurement Sequences

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`. A timer interrupt can be used to start each measurement sequence. 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 delays, 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.

6. Revision History

| Revision | Date | Description |
|----------|--------------|---|
| 1.03 | May 29, 2024 | <ul style="list-style-type: none">▪ Introduced NO2 O3 Firmware example and renamed OAQ 2nd Gen to ULP O3 Firmware.▪ Removed OAQ 1st Gen.▪ Completed numerous changes according to updated EVK Hardware and new firmware package structure.▪ Added Raspberry Pi description.▪ Moved cleaning description to the <i>ZMOD4510 Datasheet</i>.▪ Changed compiling strategy for EVK hardware.▪ Removed Arduino ATmega32 and C90 support.▪ Completed other minor changes. |
| 1.02 | Jun 9, 2022 | <ul style="list-style-type: none">▪ Changed OAQ 2nd Gen Program Flow▪ Added error code description▪ Extended “Interrupt Usage” description▪ Completed other minor changes |
| 1.01 | Apr 30, 2021 | <ul style="list-style-type: none">▪ Added Operation Mode OAQ 2nd Gen and OAQ 1st Gen (old algorithm version)▪ Added and explained Cleaning procedure▪ Added and explained Arduino example▪ Completed other minor improvements |
| 1.00 | Sep 4, 2019 | 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 who are 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 to develop 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 from 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 of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

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-us/.

Trademarks

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