

# EU045 Evaluation Kit

## Software User's Guide

1. Overview .....	3
2. Purpose of this document .....	4
2.1 Logic View .....	4
2.2 Prerequisites.....	4
2.3 Firmware versions – IMPORTANT NOTE.....	5
2.4 Tools and software versions required .....	5
2.4.1 RA4W1 Microcontroller .....	5
2.4.2 RL78 Microcontroller .....	5
3. RL78 Firmware .....	6
3.1 Firmware Layout.....	6
3.2 Firmware Configuration .....	7
3.3 HS3001 and ISL29020 Drivers.....	8
3.4 Integration of the external ZMOD4XXX libraries .....	8
3.5 I2C Communication on the External Bus .....	8
3.5.1 Repeat Start Signal Support.....	8
3.5.2 Sensor Module Register Map.....	9
3.6 Debug UART Interface .....	12
4. RA4W1 Firmware .....	13
4.1 Firmware Layout.....	13
4.2 Firmware Configuration .....	14
4.3 MAC Address .....	14
4.4 Debug UART Interface .....	15
4.5 RGB LED coding .....	16
5. Bluetooth® Communication .....	17
5.1 Overview.....	17
5.1.1 GATT Server/Client Connection.....	17
5.1.2 Advertising.....	22
5.2 Renesas BLE-puck APP .....	24
5.2.1 Overview.....	24
5.2.2 Install the APP .....	24
5.2.3 Tabs.....	25
6. APPENDIX – Renesas ZMOD Libraries.....	30
6.1.1 ZMOD4410 Library ver. 2.1.1 - 20201001 (used in firmware version “A”).....	30
6.1.1.1 Library Integration Procedure.....	30

---

6.1.1.2	Library Modification Procedure .....	34
6.1.2	ZMOD4410 Library ver. 2.2.0 - 20210827 (used in firmware version "B").....	35
6.1.2.1	Library Integration Procedure.....	35
6.1.2.2	Library Modification Procedure .....	38
6.1.3	ZMOD4510 Library ver. 20191014 (used in firmware version "A") .....	39
6.1.3.1	Library Integration Procedure.....	39
6.1.3.2	Library Modification Procedure .....	42
6.1.4	ZMOD4510 Library ver. 3.0.0 - 20210527 (used in firmware version "B").....	42
6.1.4.1	Library Integration Procedure.....	42
6.1.4.2	Library Modification Procedure .....	45
6.1.5	ZMOD4450 Library ver. 1.2.0 - 20200310 (used in firmware version "A" and "B").....	46
6.1.5.1	Library Integration Procedure.....	46
6.1.5.2	Library Modification Procedure .....	49
7.	References.....	50
	Revision History.....	51

## 1. Overview

This is the Renesas EU045 Air Quality Sensor solution kit.

It demonstrates Air Quality and further environmental sensors to allow quick evaluation of Renesas ZMOD gas sensors, relative humidity / temperature and ambient light sensors. Its Bluetooth® 5.0 communication allows nice and easy visualization of data on a contemporary GUI running on a Smartphone or Tablet, and it comes along with Li-Ion battery, charger and Qi standard wireless power transfer.

The EU045 evaluation kit is available in three variants, basically differing in the assembled air quality sensor:

Kit Name / Order Code	Description	Gas Sensor	Color
Y-EU045-BLUEPUCK / EU045-IAQEV1Z	Indoor Air Quality	ZMOD4410	Blue
Y-EU045-GREENPUCK / EU045-OAQEV1Z	Outdoor Air Quality Sensor	ZMOD4510	Green
Y-EU045-YELLOWPUCK / EU045-RAQEV1Z	Refrigerator Air Quality Sensor	ZMOD4450	Yellow

When closed, the case is somewhat waterproof and contains a membrane to let gas in and out, similar to IP54 (not tested).

All three versions use a common Smartphone App, which is available for Android and Apple iOS and autodetects the type of EU045, see <<https://www.renesas.com/win/eu045>>

The solution kits contain the following Renesas Parts

- HS3001 - relative humidity and ambient temperature sensor
- ISL29020 - ambient light sensor (fitted in lots with S/N: A20\* and A21\* only, component is not recommended for new designs anymore)

one of the following Renesas Air quality sensor:

- ZMOD4410 - indoor air quality sensor  
(gas sensor module for TVOC, eCO2 and IAQ according to German Umweltbundesamt)
- ZMOD4510 - outdoor air quality sensor  
(gas sensor module for OAQ incl. NO2 and O3 according to standard classification from US)
- ZMOD4450 - refrigeration air quality  
(gas sensor module for typical odor in fridge)

plus

- ISL9301 Li-Ion Battery Charger
- ISL9122A Ultra-Low IQ Buck/Boost Regulator
- RA4W1 MCU for Bluetooth® 5.0 LE communication
- RL78/G13 MCU for sensor data evaluation / provision

## 2. Purpose of this document

This Software User manual provides you in-depth details on the Software of this solution kit.

### 2.1 Logic View

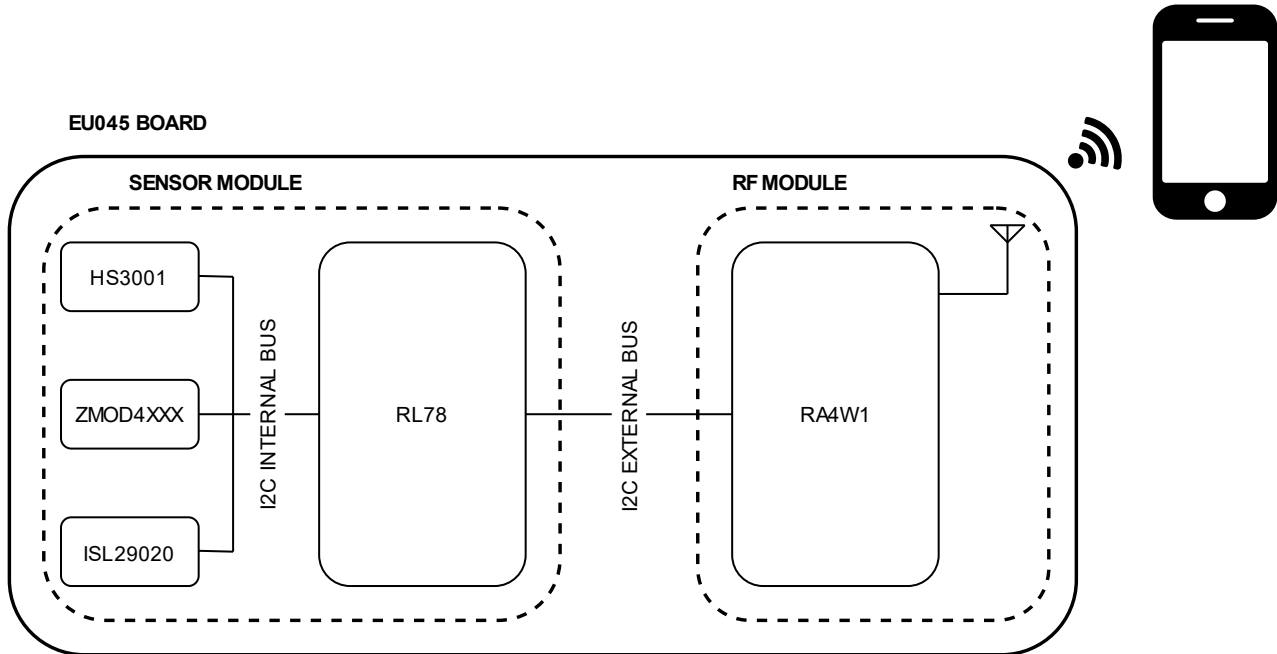


Figure 1: EU045 Logic View

The PCB board is logically divided in two Modules (Figure 1): the Sensor Module and the RF Module. In the Sensor Module an RL78 microcontroller and the sensors are present. These devices communicate via I2C serial bus, where the RL78 has the master role while the Sensors are I2C slaves. At power-up the RL78 initializes the sensors and starts to collect their output data. Sensor configuration parameters and output data are placed in a virtual register bank (present on the RL78). These data can be read and write through the I2C external bus. On this bus the RL78 has the slave role and an external device (i.e. the I2C master, on this board, this is the RF module) accesses the RL78 virtual register bank and loads the sensor configuration and/or reads the sensor output data. Furthermore, the RL78 contains Renesas AI-based algorithms to evaluate air quality data from the raw sensor values.

In the RF Module a RA4W1 microcontroller is used. The role of this device is to collect the sensor data (i.e. retrieving them from the RL78) and make their values available through a Bluetooth® LE connection by using an advertising message (unconnected mode) and/or GATT Server/Client (connected mode).

### 2.2 Prerequisites

This guide assumes you have some experience with the *Renesas e<sup>2</sup> studio* IDE together with the *Flexible Software Package (FSP)*, the *QE for BLE* plug-in (both used on the RA4W1 microcontroller), the *CC-RL compiler* (recommended for the RL78), *GCC ARM* compiler (for RA MCU) and the *Code Generator* plugin (used to configure the RL78 peripherals). If not, please familiarize yourself by referring to our website using the smart search function on these *keywords* and/or details in the next section.

## 2.3 Firmware versions – IMPORTANT NOTE

After a first EU045 production lot, a major firmware upgrade has been released. Users must pay attention which firmware version is running on their boards. By default, users receive only working combinations.

In the upgrade, the ZMOD libraries have been updated, introducing the support for Ultra Low Power (ULP) mode. Moreover, an autodetection of the presence of light sensor (ISL29020) has been implemented to allow running the hardware with or without light sensor.

In this document, firmware version “A” and version “B” are used to distinguish the original from the upgraded releases, as indicated in Table 1.

Version	Firmware Versions		Boards Compatibility		Notes
	RL78	RA4W1	ISL29020 light sensor fitted	ISL29020 light sensor NOT assembled	
<b>A</b>	< 0.2.0	< 0.4.0	OK	NO	Pre-upgrade versions
<b>B</b>	≥ 0.2.0	≥ 0.4.0	OK	OK	Upgraded versions • Updated ZMOD4410 libraries • Updated ZMOD4510 libraries • Autodetect ISL29020

Table 1: Firmware versions differences.

## 2.4 Tools and software versions required

### 2.4.1 RA4W1 Microcontroller

- Firmware version **A**
  - Renesas e<sup>2</sup> studio Integrated Solution Development Environment (IDE) v 2020-10 or greater (required to use FSP 2.2.1)
  - Renesas Flexible Software Package (FSP) v.2.2.1 or greater using the GCC ARM Embedded Toolchain 9.2.1.20191025
  - QE for BLE: Development Assistance Tool for Bluetooth® Low Energy
- Firmware version **B**
  - Renesas e2 studio Integrated Solution Development Environment (IDE) v 2021-10 or greater (required to use FSP 3.4.0)
  - Renesas Flexible Software Package (FSP) v.3.4.0 or greater using the GCC ARM Embedded Toolchain 9.2.1.20191025
  - QE for BLE: Development Assistance Tool for Bluetooth® Low Energy

### 2.4.2 RL78 Microcontroller

- Firmware version **A**
  - Renesas e<sup>2</sup> studio Integrated Solution Development Environment (IDE) v 2020-10 or greater
  - Code Generator Plug-in for RL78
  - Renesas CC-RL v1.09.00 or greater
  - ZMOD4410 Indoor Air Quality eCO<sub>2</sub> Firmware ver. 2.1.1 – 20201001<sup>1</sup>

<sup>1</sup> Software License Agreement applies, see also <https://www.renesas.com/eu/en/document/oth/disclaimer002?language=en>

- ZMOD4410 Odors, Firmware for Fan Control ver. 2.1.1 – 20201001 <sup>1</sup>
- ZMOD4410 Sulfur Odors, Firmware ver. 2.1.1 – 20201001 <sup>1</sup>
- ZMOD4510 Outdoor Air Quality Firmware ver. 20191014 <sup>1</sup>
- ZMOD4450 Refrigeration Air Quality Firmware ver. 1.2.0 – 20200310 <sup>1</sup>
- Firmware version **B**
  - Renesas e2 studio Integrated Solution Development Environment (IDE) v 2021-10 or greater
  - Code Generator Plug-in for RL78
  - Renesas CC-RL v1.09.00 or greater
  - ZMOD4410 Indoor Air Quality 2<sup>nd</sup> Gen Firmware ver. 2.2.0 - 20210827 <sup>1</sup>
  - ZMOD4410 Indoor Air Quality 2<sup>nd</sup> Gen ULP Firmware ver. 1.0.0 - 20211209 <sup>1</sup>
  - ZMOD4510 Outdoor Air Quality 1<sup>st</sup> Gen Firmware ver. 3.0.0 - 20210527 <sup>1</sup>
  - ZMOD4510 Outdoor Air Quality 2<sup>nd</sup> Gen ULP Firmware ver. 3.0.0 - 20210526 <sup>1</sup>
  - ZMOD4450 Refrigeration Air Quality 1<sup>st</sup> Gen Firmware ver. 1.2.0 - 20200310 <sup>1</sup>

### 3. RL78 Firmware

#### 3.1 Firmware Layout

Figure 2 shows the RL78 Project Layout. In detail

- `<project_name>\src\` contains all the files created by the Code Generator, based on the RL78 pinout configuration on the EU045 solution kit and all the project folders that contains the application code.
- `<project_name>\src\drivers` contains the sensor drivers and the location where the Air Quality sensor's libraries have to be placed. These libraries require to sign a needs click-thru license agreement with Renesas, so they are not included into the projects. Appendix Section 6 describes these libraries and the method to include them into the project.
- `<project_name>\src\I2C` contains a Hardware Abstraction Layer between I2C API created using the Code Generator and the sensor drivers developed for this application.
- `<project_name>\src\UART` contains the Hardware Abstraction Layer between the UART API created using the Code Generator and the task that transmits the Debug Information of the UART interface [1].
- `<project_name>\src\tasks` collects all the file related to the tasks present in the application. These tasks manage the sensor communication and update the RL78 Virtual Registers available on the external I2C BUS (see section 1 "Overview").
- `<project_name>\src\common` contains the utilities functions used in the firmware.

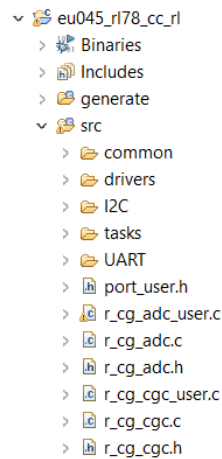


Figure 2: RL78 firmware Layout

### 3.2 Firmware Configuration

In the following file

```
<project_name>\src\common\project_conf.h,
```

all compile switches that enable/disable the firmware features are collected.

In detail:

- “HS300X\_SENSOR” (values: “SENSOR\_ABSENT”/”SENSOR\_PRESENT”, default value: “SENSOR\_PRESENT”) enables/disables the HS3001 initialization and communication procedures
- “ISL29020\_SENSOR” (values: “SENSOR\_ABSENT”/”SENSOR\_PRESENT”, default value: “SENSOR\_PRESENT”) enables/disables the ISL29020 initialization and communication procedures.
- “UART\_COMMUNICATION” (values: “ABSENT”/”PRESENT”/”TEST”, default value : “PRESENT”), enable/disables the UART communication, to transmit the Sensor Data Output on the Debug UART interface [1]. The value “TEST”: increase the number of information transmitted on the UART.
- “BATTERY\_CHARGE” (values: “ABSENT”/”PRESENT”, default value : “PRESENT”) enables/disables the task that evaluates the battery charge.

The following settings are to be used alternatively depending on Blue, Green or Yellow version. Only one sensor is present: either ZMOD4410 or ZMOD4510 or ZMOD4450

- “ZMOD4410\_SENSOR”<sup>2</sup> (values: “SENSOR\_ABSENT”/”SENSOR\_PRESENT” default value: “SENSOR\_ABSENT”) enables/disables the ZMOD4410 initialization and communication procedures; must be set to SENSOR\_PRESENT for Blue Puck version only.
- “ZMOD4410\_END\_DET”  
(values: “ZMOD4XXX\_INTERRUPT\_PIN\_ABSENT”/”ZMOD4XXX\_INTERRUPT\_PIN\_PRESENT”, default value: “ZMOD4XXX\_INTERRUPT\_PIN\_PRESENT”), that defines the detection method used by the sensor driver on the availability of new output data (see section 6.1.2 “ZMOD4410 Library” for a complete description of this compile switch).
- “ZMOD4510\_SENSOR”<sup>2</sup> (values: “SENSOR\_ABSENT”/”SENSOR\_PRESENT” default value: “SENSOR\_ABSENT”) enables/disables the ZMOD4510 initialization and communication procedures; must be set to SENSOR\_PRESENT for Green Puck version only

<sup>2</sup> The value of this compile switch has to be set according to the EU045 solution kit / Puck version, i.e. Blue, Green or Yellow. The default value of this compile switch is SENSOR\_ABSENT because the firmware is provided without the Air Quality Sensor Library since it requires to sign a click-thru license agreement with Renesas.

- “ZMOD4510\_END\_DET”  
(values: “ZMOD4XXX\_INTERRUPT\_PIN\_ABSENT”/“ZMOD4XXX\_INTERRUPT\_PIN\_PRESENT”, default value: “ZMOD4XXX\_INTERRUPT\_PIN\_PRESENT”), that defines the detection method used by sensor driver on the availability new output data (see section 6.1.4 “ZMOD4510 Library” for a complete description of this compile switch)
- “ZMOD4450\_SENSOR”<sup>2</sup> (values: “SENSOR\_ABSENT”/“SENSOR\_PRESENT” default value: “SENSOR\_ABSENT”), which enables/disables the ZMOD4450 initialization and communication procedures; must be set to SENSOR\_PRESENT for Yellow Puck version only
- “ZMOD4450\_END\_DET”  
(values: “ZMOD4XXX\_INTERRUPT\_PIN\_ABSENT”/“ZMOD4XXX\_INTERRUPT\_PIN\_PRESENT”, default value: “ZMOD4XXX\_INTERRUPT\_PIN\_PRESENT”), that defines the detection method used by the sensor driver on the availability new output data (see section 6.1.5 “ZMOD4450 Library” for a complete description of this compile switch)

### 3.3 HS3001 and ISL29020 Drivers

HS3001 and ISL29020 drivers are created based on the I2C protocol explained in their respective datasheet [3] and [4]. No external libraries are required for these sensors.

### 3.4 Integration of the external ZMOD4XXX libraries

The ZMOD4410, ZMOD4510 and ZMOD4450 air quality sensors are distributed together with a closed firmware library providing the API interface for getting the measurements from the sensors.

Although being part of this package, please note that for these libraries the click-thru license agreement with Renesas you have signed, still applies (!).

For further details on the libraries, please refer to the Appendix Section 6.

### 3.5 I2C Communication on the External Bus

#### 3.5.1 Repeat Start Signal Support

As explained in section 1 “Overview”, sensors and RL78 Microcontroller compose the sensor module, which can also be used stand-alone; it can be electrically or even mechanically separated from the rest of the solution kit. The access to the sensor data output and configuration parameters is possible through a virtual register bank, which data can be accessed via the I2C bus.

Register reading requires transmitting a message into the I2C buffer, divided into two parts (Figure 3): the first part the I2C master transmits the register address to the slave, while in the second part, the I2C master receives the register value. At the beginning of the second part, the I2C master transmits a restart bit.

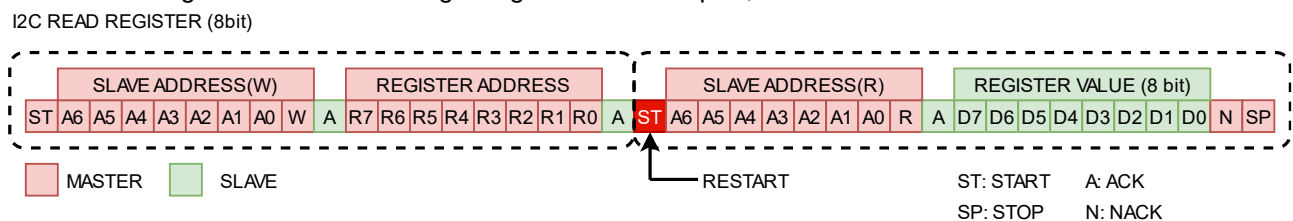


Figure 3: Read Register Message Format (8-bit data) on the I2C BUS

As it is explained in [8] (which refers for RL78/G14 family group, but it is also valid for the RL78/G13), the code generator does not support the repeat start condition for the RL78 SERIAL INTERFACE IICA peripherals, no matter whether they are configured as master or slave. To solve the problem, the document provides the firmware code snippet to be added to the I2C ISR generated by code generator.

Figure 4 shows the firmware snippet that manages the restart condition on the EU045 firmware application. This part should be included in a r\_cg\_serial\_user.c zone that is erased each time that the code is re-generated by the Code Generator.



To avoid this problem, the `r_cg_serial_user.c` file is removed by the compile toolchain (even if it is present in the software) and its data is placed in the file `serial_user.c`, that is not affected by re-generation of the code<sup>3</sup>.

```

#ifdef RESTART_SLV_SUPPORT
    if(1U == STD0)/* start or restart condition*/
    {
        /*If the i2c address has been already detected by the ISR
        * this is a Restart. The restart is used when the master
        * need to read a register value */
        if(g_iica0_slave_status_flag & _80_IICA_ADDRESS_COMPLETE)
        {
            REGMAP_I2C_Restart_Stop(I2C_SLV_CHA0, &buff_rx, &i2c_stop_out);
        }

        g_iica0_slave_status_flag = 0U;
    }
#endif

```

**Figure 4: Firmware snippet used to manage the Restart bit Reception**

### 3.5.2 Sensor Module Register Map

At power-up, the RF Module (RA4W1 MCU) starts to send messages on the I2C external bus to detect the presence of the sensor module (RL78 MCU). When the sensor module is detected, the RF module sends the sensors' configuration data and then starts requesting the sensor data output.

Table 2 shows the RL78 virtual register bank. The columns "A" and "B" refer to the firmware versions (see Table 1): the green color indicates the validity of the register while the red color means that the address is not used.

Address	Name	Type	Range	Access	Notes	A	B
0x00	ZMOD4XXX Status Register	uint8		R		Green	Green
0x01	HS3001 Status Register	uint8		R		Green	Green
0x02	ISL29020 Status Register	uint8		R		Green	Green
0x03	Input Status Register	uint8		R		Green	Green
0x04 - 0x05	ZMOD4XXX Product ID	uint16		R		Green	Green
0x06	Control Register	uint8		R/W		Green	Green
0x07 - 0x0C	ZMOD4XXX Tracking Number	6 * uint8		R		Green	Green
0x0D - 0x0E	HS3001 Measure Delay	uint16		R/W*	see "Write delay Register"	Green	Green
0x0F - 0x10	ZMOD4XXX Measure Delay	uint16		R/W*	see "Write delay Register" (this register is Used only for ZMOD4410)	Green	Green
0x11 - 0x12	ISL29020 Measure Delay	uint16		R/W*	see "Write delay Register"	Green	Green
0x20	IAQ	uint8	20, 0.1	R	ZMOD4410 IAQ Mode	Green	Green
0x21-0x22	TVOC	uint16	0-50, 0.01	R	ZMOD4410 IAQ Mode	Green	Green

<sup>3</sup> Since the file `serial_user.c` is not affected by code re-generation, in case the User changes Serial peripheral configuration (UART and I2C) and re-generates the code, the software changes on the `r_cg_serial_user.c` has to be copied in `serial_user.c`

0x23-0x24	EtOH	uint16	0-25, 0.01	R	ZMOD4410 IAQ Mode		
0x25-0x26	eCO2	uint16	400-5000	R	ZMOD4410 IAQ Mode		
0x27	Odor Control Signal State	uint8	0(OFF)/ 1(ON)	R	ZMOD4410 ODOR Mode		
0x28-0x29	Odor Concentration Ratio	uint16	0-5 , 0.01	R	ZMOD4410 ODOR Mode		
0x2A	Sulfur Odor Classification	uint8	0(Acceptable)/ 1(Sulfur)	R	ZMOD4410 SULFUR ODOR Mode		
0x2B	Sulfur Odor Intensity	uint8	0.0 - 5.0	R	ZMOD4410 SULFUR ODOR Mode		
0x2C	Refrigerator Control Signal State	uint8	0(OFF)/ 1(ON)	R	ZMOD4450		
0x2D-0x2E	Refrigerator Concentration Ratio	uint16	0-5 , 0.01	R	ZMOD4450		
0x2F-0x30	OAQ	uint16	0-500, 0.1	R	ZMOD4510		
0x31-0x32	ALS	uint16		R	ISL290xxx		
0x33-0x34	Humidity	uint16	0 - 100%, 0.01	R	HS3001		
0x35-0x36	Temperature	int16	-20 to 85, 0.01	R	HS3001		
0x37	Battery Charge	uint8	0 - 100 %	R			
0x38-0x39	FW Revision	uint16		R			
0x3A-0x3B	OAQ Fast	uint16	0-500, 0.1	R	ZMOD4510 ULP Mode		
0x3C-0x3D	O3	uint16	20-500, 0.1	R	ZMOD4510 ULP Mode		
0x3E	Control Register 2	uint8		R/W			
0x3F-0x4F	<i>Not Used</i>				Not Used		
0x50	ISL29020 Command Register	uint8		R	Debug Data		
0x51-0x52	ISL29020 DATA Raw Data	uint16		R	Debug Data		
0x53-0x54	HS3001 Temp. Raw Data	uint16		R	Debug Data		
0x55-0x56	HS3001 Humidity Raw Data	uint16		R	Debug Data		
0x57-0x5A	Log RcdA	uint32		R	Debug Data (ZMOD4410)		
0x5B-0x9A	Rmox	16 * uint32		R	Debug Data (ZMOD4XXX)		
0x9B-0x9C	Zmod Routine max exec time	uint16	0ms-6538.5ms, 0.1	R	Debug Data (ZMOD4XXX)		
0x9D-0x9E	HS300x Routine max exec time	uint16	0ms-6538.5ms, 0.1	R	Debug Data		
0x9F-0xA0	ISL29020 Routine max exec time	uint16	0ms-6538.5ms, 0.1	R	Debug Data		
0xA1-0xA2	Internal error code	uint16t		R	Debug Data		
0xA3-0xAE	Log Non Log RcdA	3* uint32		R	Debug Data (ZMOD4410 ULP Mode)		

Table 2: I2C Register Map

ZMOD4XXX Status Register (0x00)								
	7	6	5	4	3	2	1	0
	0		-		0		0	
	Ready Result		-		Error Code		Status	
<b>Status</b>	0: DISABLED, 1: INIT, 2: READY, 3: ERROR, 4: WARMUP							
<b>Error Code</b>	0: No Error, 1: I2C COMM ERROR 2: CONFIG ERROR, 3: WRONG SENSOR; 4: RUNNING ERROR							
<b>Ready Result</b>	1- Sensor Value is ready. Value is reset to 0 after the value is read via I2C							
HS3001 Status Register (0x01)								
	7	6	5	4	3	2	1	0
	0		-		0		0	
	Ready Result		-		Error Code		Status	
<b>Status</b>	0: DISABLED, 1: INIT, 2: READY, 3: ERROR							
<b>Error Code</b>	0: No Error, 1: I2C COMM ERROR;2: CONFIG ERROR							
<b>Ready Result</b>	1: Sensor Value is ready. Value is reset to 0 after the value is read via I2C							
ISL29020 Status Register (0x02)								
	7	6	5	4	3	2	1	0
	0		-		0		0	
	Ready Result		-		Error Code		Status	
<b>Status</b>	0: DISABLED, 1: INIT, 2: READY, 3: ERROR							
<b>Error Code</b>	0: No Error, 1: I2C COMM ERROR;2: CONFIG ERROR							
<b>Ready Result</b>	1: Sensor Value is ready. Value is reset to 0 after the value is read via I2C							
Input Status Register (0x03)								
	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	0
	-	-	-	-	-	-	-	ISL9301 CHG
<b>ISL9301 CHG</b>	0: Battery not charging, 1: Battery is charging							
Control Register (0x06)								
	7	6	5	4	3	2	1	0
	0	0	0		0		0	
	Soft reset	Write Delay Registers	ISL29020 Mode		HS3001 Mode		ZMOD4XXX Mode (LSB)	
<b>ZMOD4XXX Mode (LSB)</b>	0: power-down, 1: (IAQ/OAQ/RAQ), 2: odor, 3: sulfur odor							
<b>HS3001 Mode</b>	0: power-down, 1: single shot, 2 continuous mode							
<b>ISL29020 Mode</b>	0: power-down, 1: single shot, 2 continuous mode							
<b>Soft reset</b>	1: performs reset of the module and sensors							
<b>Write Delay Registers</b>	0: the sensor delay registers are not writable, 1: sensor delay registers are writable							
Control Register 2 (0x3E)								
	7	6	5	4	3	2	1	0
	-	-	-	-	-	-	-	0
								ZMOD4XXX Mode (MSB)
<b>ZMOD4XXX Mode (MSB)</b>	0: Mode configured by bits 1-0 in Control Register, 1: ULP (Ultra Low Power)							

Table 3: Configuration and status registers.

### 3.6 Debug UART Interface

The EU045 solution kit also provides a debug UART interface for the RL78 microcontroller (enabled by the compile switch `UART_COMMUNICATION`, as explained in 3.2 “Firmware Configuration”), available on the J8 connector [1]. For each sensor present on the kit, the following data are available:

- Sensor Data Output
- Sensor Measure Delay (if available)
- Sensor Status
- Sensor Error Code.

Relating to the battery, by default (i.e. when no USB/Wireless charger is connected to the solution kit) an estimate of the charge percentage is shown on the UART. However, when the USB or Qi charger inputs power to the solution kit, a message “Battery charger is connected” is shown.

Table 4 shows the UART Communication Parameters and Figure 5 shows typical debug information transmitted by the RL78 UART on a EU045 Green-Puck version as an example.

RL78 UART Parameters	
Parameter	Value
Baud Rate	38400
Data Size	8-bit
Parity Bit	None
Stop Bit	1-bit

**Table 4: RL78 Debug UART Parameters**

```

Firmware Version 0.0.9 Hardware Version EK-RL78 R101

HS300X
Temperature : 0.0C
Humidity : 0.0%
Measure Delay : 0.500s
Sensor Status : READY
Error Code : NO ERROR

ZM004410
IAQ : 0.9
TVOC : 0.1
ECO2 : 0
EtOH : 0.9
Rnox Array : 1237534 224015 100 100 100 100 100 100 100 100 100 100
Log Rcds : 0
Measure Delay : 1.990s
Sensor Status : HARMUP
Error Code : NO ERROR

ISL29020
ALS : 289
Measure Delay : 0.500s
Sensor Status : READY
Error Code : NO ERROR

BATTERY
Battery Charge : 100%

```

Figure 5: Debug information (Blue-Puck Version) transmitted by the RL78 on the UART

## 4. RA4W1 Firmware

### 4.1 Firmware Layout

Figure 6 shows the RA4W1 Project Layout. In detail

- `<project_name>\qe_gen\` contains the files generated by the QE for BLE Tool, which are related to the Server GATT Services available on the application
- `<project_name>\ra\` and `<project_name>\ra_gen\` contain the data generated by FSP, according to the RA4W1 pinout and features, for the EU045 solution kit
- `<project_name>\src\` contains the firmware task files and the project folders related to the application code.
- `<project_name>\src\common` contains the utilities functions used in the firmware
- `<project_name>\src\I2C` contains the files to interface with the RL78 Register Bank
- `<project_name>\src\mdl_param` contains the functions to store some of the application parameter in the microcontroller non-volatile memory (NVM)

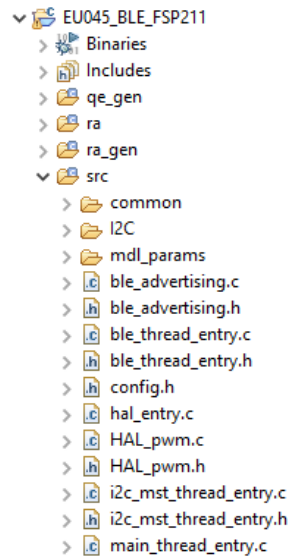


Figure 6: RA4W1 firmware Layout

## 4.2 Firmware Configuration

The RA4W1 firmware doesn't require a configuration procedure and it is the same for all the EU045 variants i.e. Blue-Puck, Green-Puck, Yellow-Puck). It auto-detects which ZMOD sensor is present on the RL78 firmware (which is different for those three variants). However, the Bluetooth® LE requires a MAC address; see the next section.

## 4.3 MAC Address

Each EU045 solution kit has a different pre-programmed MAC Address, used as default by the RA4W1 firmware. However, it is possible to configure a different MAC Address (custom MAC Address can be obtained with the procedure described in [9]) following these steps:

1. Open the FSP Configurator, double-clicking the configuration.xml file in the Project Explorer (Figure 7).
2. In the Stacks Tab, select the "BLE Abstraction Driver" present in the HAL/Common Thread.
3. In the "BLE Abstraction Driver" Property (Figure 8), the field "Debug Public Address" defines the EU045 solution kits MAC Address value. Special value "{0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF}" is used to select the kits MAC Address pre-programmed value (different for each EU045 solution kit). If a different MAC Address is required, the "Debug Public Address" array must be filled with the new value, starting from the MAC Address less significant byte<sup>4</sup>.
4. Click "Generate Project Content" to update the new configuration in the firmware.
5. Build the firmware.
6. Download / flash the new firmware to the RA4W1 microcontroller.

<sup>4</sup> If the User MAC Address is "01-02-03-04-05-06", the Debug Public Address must be set to {0x06, 0x05, 0x04, 0x03, 0x02, 0x01}

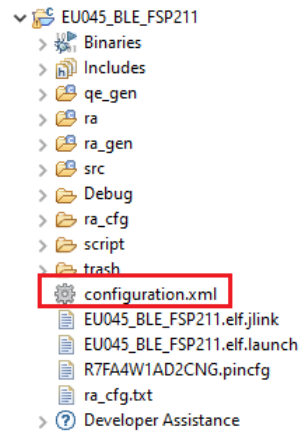


Figure 7: configuration.xml file in the e<sup>2</sup> studio Project Explorer

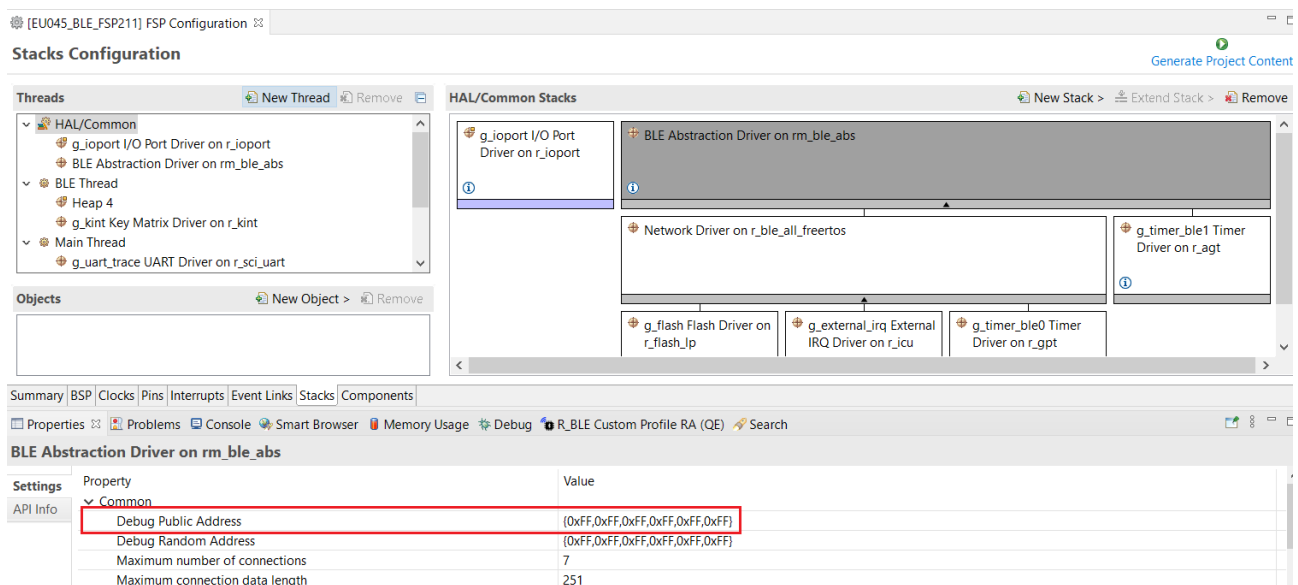


Figure 8: Debug Public Address field in the BLE Abstraction Driver Properties

#### 4.4 Debug UART Interface

The EU045 solution kit provides another debug UART interface, this time for the RA4W1, available on the J4 connector [1].

The RA4W1 transmits additional debug information referred to:

- I2C communication with the RL78, indicating when the RL78 is detected on the external bus, when the sensor configuration is going to be transmitted and when the configuration is completed
- Bluetooth® LE Communication and Advertising transmission, indicating when the Bluetooth® LE Framework starts and when an advertising message is transmitted

Table 5 shows the UART communication parameters and Figure 9 shows the typical debug information transmitted by the RA4W1 as an example.

RA4W1 UART Parameters	
Parameter	Value
Baud Rate	115200
Data Size	8-bit
Parity Bit	None
Stop Bit	1-bit

Table 5: RA4W1 Debug UART Parameters

```

Start RL78 I2C Communication
Launching BRL78 Detected
ZMOD4XXX Detection...
ZMOD4410 Detected
BRL78 Run
Starting BLE4 advertising
Updating BLE4 advertising
Updating BLE4 advertising
Updating BLE4 advertising
Updating BLE4 advertising
UPDATE OK -> Advertising Mode = 1
Updating BLE5 advertising
Updating BLE5 advertising
Updating BLE5 advertising
Updating BLE5 advertising
Updating BLE5 advertising
Updating BLE5 advertising
Updating BLE5 advertising

```

Figure 9: Debug information transmitted by the RA4W1 on the UART

## 4.5 RGB LED coding

As explained in [1], the EU045 solution kit has an RGB LED. The green LED is connected to the ISL9301 Battery charger, indicating when the battery is charging, while the blue and red LEDs are connected on the RA4W1 as GPIO (General Purpose Output). Blue and red LED are used as default, to provide information about the EU045 solution kits status, see Table 6.

RGB LED Coding <sup>5</sup>			
Priority	Information	Color Coding	Notes
1	RA4W1 is connecting to RL78	Blue	(power-on)
2	Identify (find-me)	Purple	See Section 5.1.1
3	RL78 Error	Red	
4	Low Battery Charge (<=10%)	Red (500ms Blink)	
5	Advertising Bluetooth® Mode <sup>6</sup>	Blue (1 Flash each 3s) Bluetooth® 4.2	See Section 5.1.2
		Blue (2 Flashes each 3s) Bluetooth® 5 2Mbit/s	
		Blue (3 Flashes each 3s) Bluetooth® 5 Long Range	

Table 6: Red and Blue Led Coding

<sup>5</sup> Color coding indicated in the table doesn't consider the green LED, that is ON, while the battery is charging

<sup>6</sup> Bluetooth 5 and Bluetooth 5 LR modes may not be supported by all smartphones.



## 5. Bluetooth® Communication

### 5.1 Overview

As explained in Section 1, EU045 sensor module collects the sensor data, while the RF Module makes them available via Bluetooth® LE connection using Advertising (unconnected mode) or establishing a GATT Server/Client Connection with an external device, e.g. Smartphone or Tablet.

Note that non-standard UUID are also used in this example application. If Bluetooth® Qualification is required, check if vendor specific UUID must be obtained from Bluetooth SIG.

For more information refer to Renesas Application Note on Bluetooth Qualification [10].

#### 5.1.1 GATT Server/Client Connection

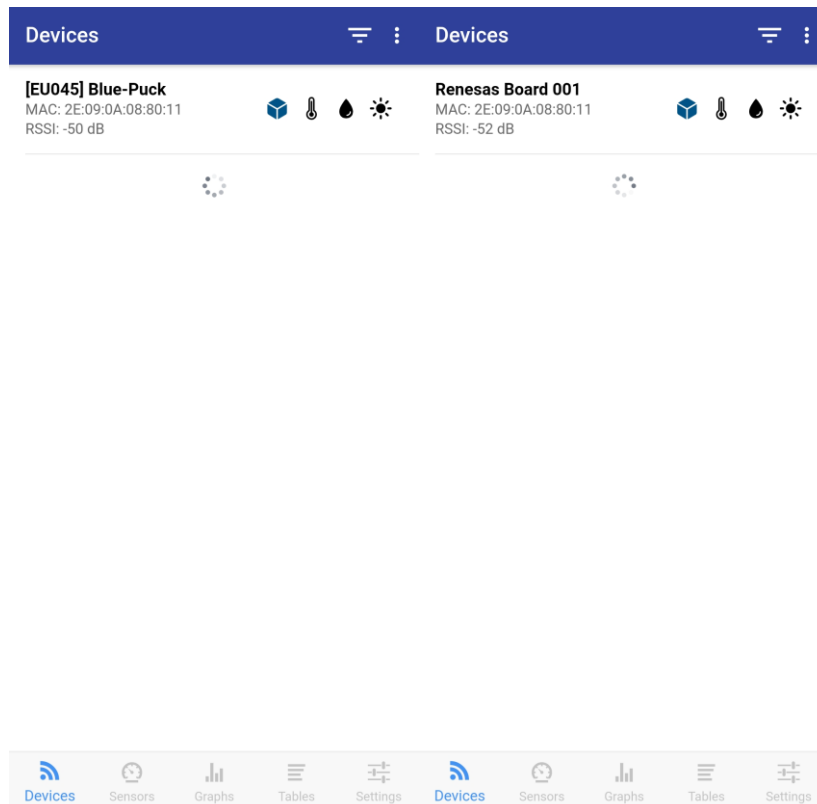
Bluetooth® GATT Server/Client Connection enables the user to access to the EU045 configuration parameters and the sensor output data using a Bluetooth® LE communication, so called Connected Mode. During the communication, the EU045 solution kit takes the role of a GATT Server while an external device takes the role of GATT Client.

Table 7, shows the GATT Server UUID list present on the application, developed using the QE for BLE Tool (Figure 11).

This list is composed of standard UUID (i.e. UUID present in the Bluetooth® Assigned Number List [11]) and non-standard UUID (general purpose UUID not related to a specific data or vendor). The Application UUID are divided in read-only data, which contains the Device Information (such as Device Name, Hardware Revision) or the sensor output data and read/write data which contains EU045 solution kit configuration parameters. For the sensor output data Bluetooth® Notification is supported.

The EU045 solution kit configuration parameters are as follows:

- **Device Name** (default value: [EU045] xxxx-Puck, where xxxx is the Puck color variant): for this application, this parameter has the same value of the Advertising Complete Local Name field present in the advertising/scan response (unconnected mode). When the Advertising Bluetooth® 4.2 Mode is selected, the Complete Local Name is transmitted via the scan response, reducing the maximum length of the Device Name to 29 bytes. Device Name can be changed by the user to easily identify a specific kit. However, intentionally the new value is reset to its default when the EU045 solution kit is reset or power down and up again.



**Figure 10: EU045 solution kit's Device name default value (left) and User Value (right)**

- **Identify (find-me)** (0: OFF, 1: ON, 2: NIGHT-MODE default value: 0: OFF): Identify is an easy way to physically detect / find the selected kit, when the user works with multiple EU045 solution kits. Setting this Service Data to 1 result in RGB LED color becoming Purple (Table 6). When this value is set to 2 the Night-Mode is enabled: the board will not indicate the Advertising Mode forcing the Blue LED always off.
- **Advertising Mode** (0: Bluetooth® 4.2 Mode, 1: Bluetooth® 5 @ 2Mbit/s Mode, 2: Bluetooth® 5 Long Range @125kb/s Mode, default value 0: Bluetooth® 4.2 Mode). This parameter defines the Advertising Mode used by the EU045 solution kit. Parameter reading and parameter Notification are also available since the Advertising Mode can also be changed by pressing SW3 (see Section 5.1.2)
- **Advertising Period[ms]** (default value: 500ms): this parameter defines the advertising transmission period. When the Bluetooth® 5 Long Range mode is selected, this parameter cannot be lower than 500ms. If this happens, the firmware automatically sets its value to 500ms.
- **HS3001 mode** (0: disabled, 1: single shot, 2: continuous, default value 2: continuous): This parameter defines the HS3001 operation mode. When the single shot mode is selected and the measurement is complete, before starting a new measurement, this parameter must be set to "0: disable" and then back to 1: single shot to obtain another shot
- **ZMDOD4XXX Mode** (default 1): This parameter defines the Air Quality Sensor mode. See Table 10 to see the Mode list for the Specific sensor present on the Puck variant
- **ISL29020 Mode**<sup>7</sup> (0: disabled, 1: single shot, 2: continuous, default value 2: continuous): This parameter defines the ISL29020 operation mode. When the single shot mode is selected and the measurement is complete, before starting a new measurement, this parameter must be set to "0: disable" and then back to 1: single shot to obtain another shot
- **HS3001 Measure Delay[ms]** (default: 500ms). This parameter defines the delay between the end of the current sensor measurement and the beginning of the next one. Therefore, when the continuous mode is selected, this parameter defines the sample period.

<sup>7</sup> Not mounted anymore in new lots.

- **ZMOD4410 Measure Delay[ms]** (default: see Table 11). This parameter sets the ZMOD4410 sampling period, defining the delay between the end of the current transmission and the beginning of the next one. Default value (Table 11) of these parameters is selected according to the example code present in the sensors' library. When the ZMDOD4XXX Mode is changed, the ZMOD4410 measure delay is automatically related to the default value
- **ISL29020 Measure Delay[ms]** (default: 500ms). This parameter defines the delay between the end of the current sensor measurement and the beginning of the next one. Therefore, when the continuous mode is selected, this parameter defines the sample period.

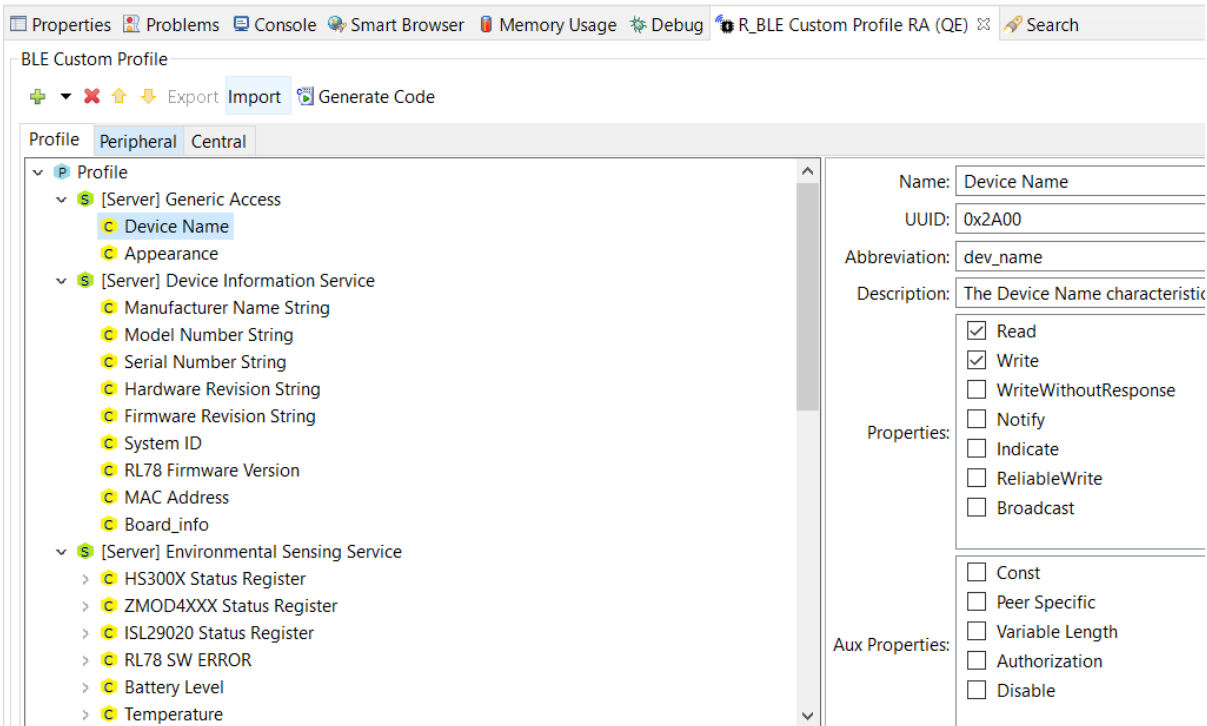


Figure 11: GATT Server UUID List in the R\_BLE Custom Profile RA

GATT Server UUID List					FW Ver.	
UUID	Name	Data Type [Format]	R/W/N	Notes	A	B
Generic Access UUID						
0x2A00	Device Name	UTF8 String	R/W	Standard UUID		
0x2A01	Appearance	UTF8 String	R	Standard UUID		
Device Information Service UUID List						
0x2A29	Manufacturer Name String	UTF8 String	R	Standard UUID		
0x2A24	Model Number String	UTF8 String	R	Standard UUID		
0x2A25	Serial Number String	UTF8 String	R	Standard UUID		
0x2A27	Hardware Revision String	UTF8 String	R	Standard UUID		

0x2A26	Firmware Revision String (RA4W1)	UTF8 String	R	Standard UUID		
0x2300	Firmware Revision String (RL78)	UTF8 String	R			
0x2A23	System ID	8 * uint8	R	Standard UUID		
0x4FF5	MAC Address	6 * uint8	R	Used by Renesas BLE-puck APP to detect Board MAC Address on IOS		
0x4FF6	Solution kit Info Flags	uint8 <sup>8</sup>	R	See Table 8		
		uint16 <sup>8</sup>	R	See Table 9		
Environmental Sensing Service List						
0x2A19	Battery Level	uint8 [0-100%]	R/N	Standard UUID		
0x2A6E	Temperature	uint16 [0.01°C]	R/N	Standard UUID		
0x2A6F	Humidity	uint16 [0.01%]	R/N	Standard UUID		
0x4910	TVOC	uint16 [0.01]	R/N	ZMOD4410 IAQ		
0x4911	eCO2	uint16 [ppm]	R/N	ZMOD4410 IAQ		
0x4912	etOH	uint16 [0.01 ppm]	R/N	ZMOD4410 IAQ		
0x4913	Odor Control Signal State	uint8_t	R/N	ZMOD4410 (ODOR Mode)		
0x4914	Odor Concentration Ratio	uint16 [0.01]	R/N	ZMOD4410 (ODOR Mode)		
0x4915	Sulfur Odor Classification	uint8	R/N	ZMOD4410 (SULFUR ODOR Mode)		
0x4916	Sulfur Odor Intensity	uint16 [0.1]	R/N	ZMOD4410 (SULFUR ODOR Mode)		
0x4920	IAQ	uint8 [0.1]	R/N	ZMOD4410		
0x4921	OAQ	uint16 [0.1]	R/N	ZMOD4510		
0x4922	RAQ	uint16 [0.01]	R/N	ZMOD4450		
0x4923	Refrigerator Control Signal State	uint8	R/N	ZMOD4450		
0x4924	ALS (ISL29020 Output)	uint16 [lux]	R/N			
0x4925	OAQ Fast	uint16 [0.1]	R/N	ZMOD4510 (ULP Mode)		
0x4926	O3	uint16 [0.1]	R/N	ZMOD4510 (ULP Mode)		
0x4FF7	Log RCDA (debug data)	uint32	R/N	ZMOD4410 Debug Data (Normal Mode)		
0x4FF8	Rmox Array Group 0	4 * uint32	R/N	ZMOD4XXX Debug data		
0x4FF9	Rmox Array Group 1	4 * uint32	R/N	ZMOD4XXX Debug data		
0x4FFA	Rmox Array Group 2	4 * uint32	R/N	ZMOD4XXX Debug data		
0x4FFB	Rmox Array Group 3	4 * uint32	R/N	ZMOD4XXX Debug data		
0x4FFC	HS3001 Status	uint8	R/N	See HS3001 Status Register Table 3		
0x4FFD	ZMOD4XXX Status	uint8	R/N	See ZMOD4XXX Status Register Table 3		

<sup>8</sup> The "Solution Kit Info Flags" field has different sizes depending on the firmware version.

0x4FFE	ISL29020 Status	uint8	R/N	See ISL29020 Status Register Table 3		
0x4FFF	RL78 Error Code	uint8	R/N			
0x5000	Identify (i.e. Find-me)	uint8	R/W	Change User Led color		
0x5001	Advertising Mode	uint8	R/W/ N			
0x5002	Advertising Period	uint16 [ms]	R/W/ N			
0x5003	HS3001 Mode	uint8	R/W	(0: disabled 1: single shot 2: continuous)		
0x5004	ZMOD4XXX Mode	uint8	R/W	See Table 10		
0x5005	ISL29020 Mode	uint8	R/W	(0: disabled 1: single shot 2: continuous)		
0x5006	HS3001 Measure Delay	uint16 [ms]	R/W			
0x5007	ZMOD4410 Measure Delay	uint16 [ms]	R/W			
0x5008	ISL29020 Measure Delay	uint16 [ms]	R/W			
0x5009	Log Non Log RCDA	3 * uint32	R/N	ZMOD4410 Debug Data (ULP Mode)		

Table 7: GATT Server UUID List

Solution kit Info Flags (Firmware version A)								
	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0
	RAQ	OAQ	Sulfur Odor	Odor	IAQ	Light Sens	Hum Sens	Temp Sens
<b>Temp Sens</b>	Temperature sensor state. 0: absent, 1: present.							
<b>Hum Sens</b>	Humidity sensor state. 0: absent, 1: present.							
<b>Light Sens</b>	Light sensor state. 0: absent, 1: present.							
<b>IAQ</b>	IAQ sensor in normal mode. 0: not active, 1: active.							
<b>Odor</b>	IAQ sensor in Odor mode state. 0: not active, 1: active.							
<b>Sulfur Odor</b>	IAQ sensor in Sulfur mode state. 0: not active, 1: active.							
<b>OAQ</b>	OAQ sensor in normal mode. 0: not active, 1: active.							
<b>RAQ</b>	RAQ sensor in normal mode. 0: not active, 1: active.							

Table 8: Solution kit Info Flags – Firmware Version “A”.

Solution kit Info Flags (Firmware version B)								
	15	14	13	12	11	10	9	8
	-	-	-	-	-	-	0	0
							OAQ ULP	IAQ ULP
	7	6	5	4	3	2	1	0
	0	0	0	0	0	0	0	0
	RAQ	OAQ	Sulfur Odor	Odor	IAQ	Light Sens	Hum Sens	Temp Sens
<b>Temp Sens</b>	Temperature sensor state. 0: absent, 1: present.							
<b>Hum Sens</b>	Humidity sensor state. 0: absent, 1: present.							
<b>Light Sens</b>	Light sensor state. 0: absent, 1: present.							
<b>IAQ</b>	IAQ sensor in normal mode. 0: not active, 1: active.							
<b>Odor</b>	IAQ sensor in Odor mode state. 0: not active, 1: active.							

<b>Sulfur Odor</b>	IAQ sensor in Sulfur mode state. 0: not active, 1: active.
<b>OAQ</b>	OAQ sensor in normal mode. 0: not active, 1: active.
<b>RAQ</b>	RAQ sensor in normal mode. 0: not active, 1: active.
<b>IAQ ULP</b>	IAQ sensor in Ultra Low Power mode: 0: not active, 1: active.
<b>OAQ ULP</b>	OAQ sensor in Ultra Low Power mode: 0: not active, 1: active.

**Table 9: Solution kit Info Flags – Firmware Version “B”.**

ZMOD4XXX Modes List					
Sensor	Value				
	0	1	2	3	4
ZMOD4410	DISABLED	IAQ	ODOR <sup>9</sup>	SULFUR ODOR <sup>9</sup>	ULP <sup>9</sup>
ZMOD4510	DISABLED	OAQ			ULP <sup>9</sup>
ZMOD4450	DISABLED	RAQ			

**Table 10: ZMOD4XXX Modes List**

ZMOD4410 Default Measure Delay Values [ms]		
IAQ	ODOR	SULFUR ODOR
1990	0	1990

**Table 11: ZMOD4410 Default Measure Delay Values based on the selected Sensor Mode**

### 5.1.2 Advertising

EU045 solution kit Advertising (unconnected mode) enables the user to read the output data, without the need to set up a GATT Server/Client connection, i.e. multiple receivers can simultaneously display the EU045 output data.

Three alternative advertising modes are used for this application

- Bluetooth® 4.2 Mode  
legacy mode with reference range and speed; always 1Mbps; works with most smartphones
- Bluetooth® 5 - 2Mbps mode  
showcasing 2x speed; works with all modern smartphones
- Bluetooth® 5 Long Range at 125 kbps mode  
showcasing 4x range; works with modern Android Smartphones, but currently not with Apple devices.

Each mode differs from the others for the maximum number of byte available for the advertising payload (31 bytes for Advertising and Scan Response in the Bluetooth® 4.2 Mode, 255 bytes for Advertising in Bluetooth® 5 Mode), the communication speed and the communication range.

At power-on the EU045 solution kit starts the communication using the Bluetooth® 4.2 legacy mode. The user can change the Advertising Mode using the related Service Data (Section 5.1.1) or pressing the SW3 (each time the switch is pressed, the Advertising Mode is changed). RGB LED can be used to check which Advertising Mode currently used by the EU045 solution kit (Table 6).

<sup>9</sup> ODOR and SULFUR ODOR modes available in firmware version “A” only;  
ULP mode available in firmware version “B” only.

Table 12 shows the complete Advertising Data List. When Bluetooth® 4.2 Mode is used, multiple Advertising Messages are required to send all the data present in the list, due to the limit number of bytes per packet in this mode.

Advertising Data List			FW Ver.	
Name	Type	Notes	A	B
Device Name	Complete Local Name (0x09)	(Transmitted on the Scan Response message for the Bluetooth® 4.2 Mode and on the Advertising message for Bluetooth® 5 and Bluetooth® 5 LR Modes)		
Temperature	Service Data (0x16)	See Table 7		
Humidity	Service Data (0x16)	See Table 7		
Battery Charge	Service Data (0x16)	See Table 7		
IAQ	Service Data (0x16)	See Table 7. Transmitted only for Blue-Puck version, when IAQ Mode is selected		
TVOC	Service Data (0x16)	See Table 7. Transmitted only for Blue-Puck version, when IAQ Mode is selected		
eCO2	Service Data (0x16)	See Table 7. Transmitted only for Blue-Puck version, when IAQ Mode is selected		
etOH	Service Data (0x16)	See Table 7. Transmitted only for Blue-Puck version, when IAQ Mode is selected		
Odor Control Signal State	Service Data (0x16)	See Table 7. Transmitted only for Blue-Puck version, when Odor Mode is selected		
Odor Concentration Ratio	Service Data (0x16)	See Table 7. Transmitted only for Blue-Puck version, when Odor Mode is selected		
Sulfur Odor Classification	Service Data (0x16)	See Table 7. Transmitted only for Blue-Puck version, when Sulfur Odor Mode is selected		
Sulfur Odor Intensity	Service Data (0x16)	See Table 7. Transmitted only for Blue-Puck version, when Sulfur Odor Mode is selected		
OAQ	Service Data (0x16)	See Table 7. Transmitted only for Green-Puck version		
OAQ Fast	Service Data (0x16)	See Table 7. Transmitted only for Green-Puck version when ULP Mode is selected		
O3	Service Data (0x16)	See Table 7. Transmitted only for Green-Puck version when ULP Mode is selected		
Refrigerator Control Signal State	Service Data (0x16)	See Table 7. Transmitted only for Yellow-Puck version		
RAQ	Service Data (0x16)	See Table 7. Transmitted only for Yellow-Puck version		
ALS	Service Data (0x16)	See Table 7.		
EU045 solution kit Info	Service Data (0x16)	Used by Renesas BLE-puck APP to detect the EU045 solution kit (see Table 8 and Table 9)		
MAC Address	Service Data (0x16)	Used by Renesas BLE-puck APP to detect solution kit MAC Address on IOS		
Manufacturer Specific Data	Manufacturer Specific Data (0xFF)	Transmitted only for Bluetooth® 5 and Bluetooth® 5 LR Modes		

**Table 12: Advertising Data List**

## 5.2 Renesas BLE-puck APP

### 5.2.1 Overview

Renesas Bluetooth LE Puck is an APP for Android and IOS, which enables the user to visualize and collect the sensor output data from one or multiple EU045 solution kits at the same time, and/or to change their configuration. The APP mainly retrieves the EU045 solution kits output data from their Advertising messages, while uses the GATT/Server Client Connection in order to retrieve additional information or change the EU045 solution kits configuration.

### 5.2.2 Install the APP

The **Renesas Bluetooth LE Puck** is available on the Google Play Store and on the Apple App Store:

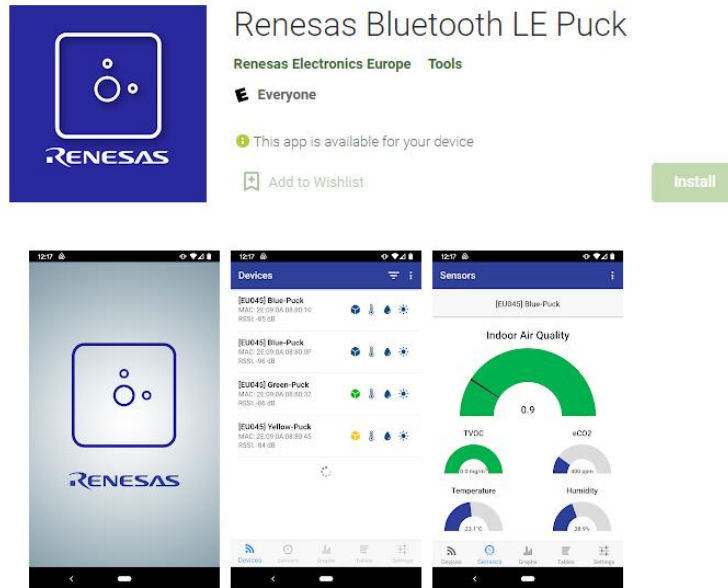
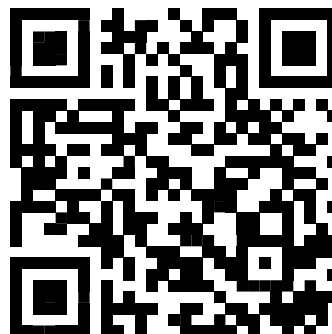


Figure 12: Renesas Bluetooth LE Puck on the Play Store.





### 5.2.3 Tabs

The APP is mainly composed of the following tabs:

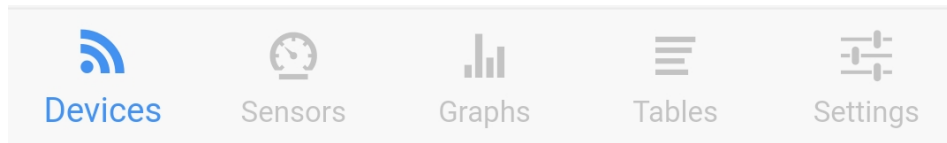


Figure 13: EU045 BLE-puck APP available Tabs.

- **Devices Tab:** This is the APP main tab showing a list of all EU045 solution kits currently transmitting the Advertisements. This list is obtained by scanning the Bluetooth® Devices, applying a filter in order to show the ones that transmits the kits Info Service Data on the Advertising message. Device filtering can be removed using the button placed at the top of the tab. Figure 14 shows an example of the Device tab with four EU045 solution kits detected by the APP. For each EU045 solution kit following information are visualized:
  - Complete Local Name (same of Device Name for this application)
  - MAC Address
  - RSSI
  - EU045 kit Puck Color/Type.
  - Sensor Status Icons (black when the related sensor is enabled, gray when it is disabled)

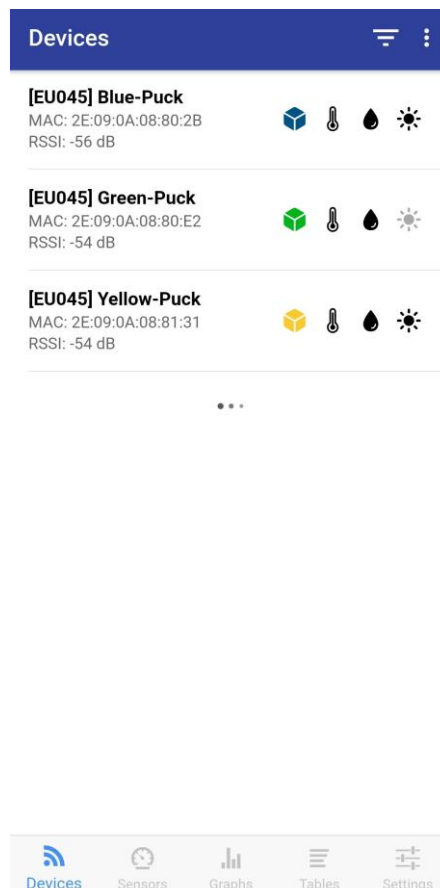
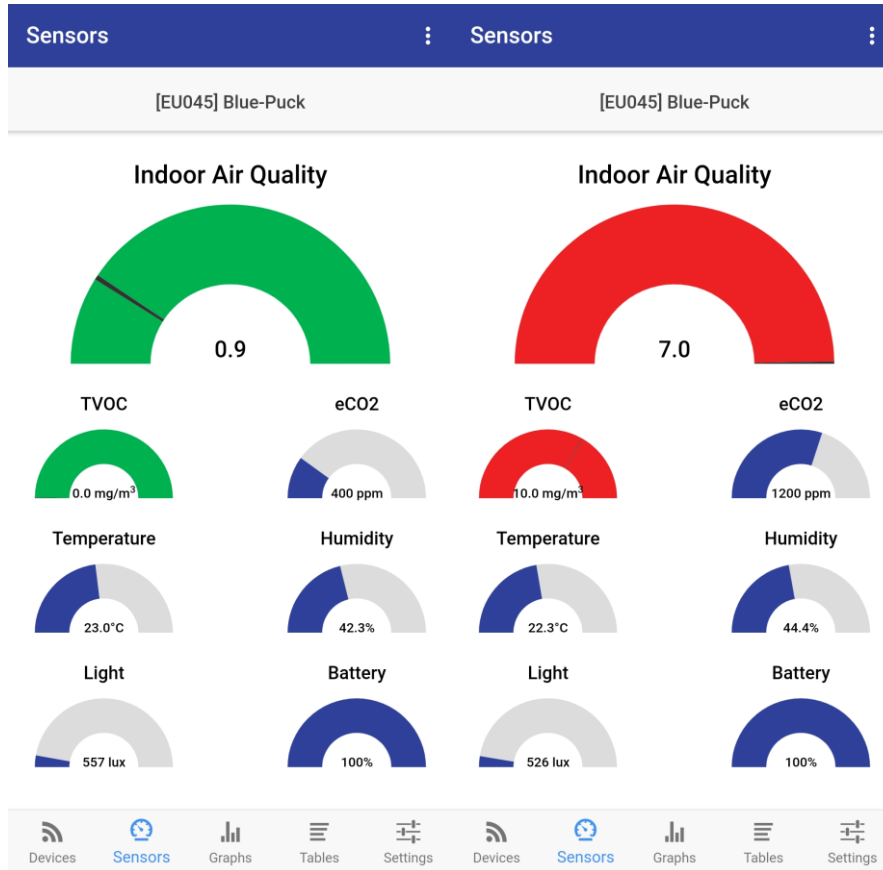


Figure 14: Device tab, with the EU045 solution kit List detected by the APP

- Sensors Tab:** This tab becomes available after the user selects (exactly) one of the EU045 solution kits shown in the Device tab List. The Tab shows the EU045 solution kit data values, through donut-style diagrams. Donut diagram colors for Indoor Air Quality and TVOC (Blue-Puck Version), Outdoor Air Quality (Green-Puck Version), and Refrigerator Air Quality (Yellow-Puck Version) parameters follow the color coding based on the available sensor datasheets (Figure 15), e.g. indicating criticality level of the air quality from green to red.



**Figure 15: Sensors Tab (Blue Puck Version), for different values of the Output Parameters**

- Graphs Tab:** This tab becomes available after the user selects one or multiple EU045 solution kits shown in the Device tab List, then showing those selected sensor data outputs as Line Graphs<sup>10</sup>. This tab enables the user to see the output data of multiple EU045 solution kits simultaneously (Figure 16). Thus, it only shows the *common* elements of the selected sensors.

<sup>10</sup> The Air Quality Sensor output graphs are visualized only when EU045 solution kits with the same Air Quality sensor are selected in the Device Tab.

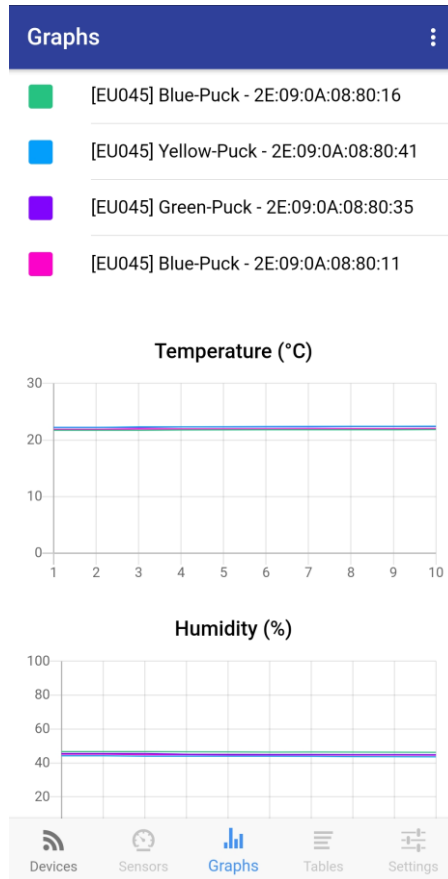


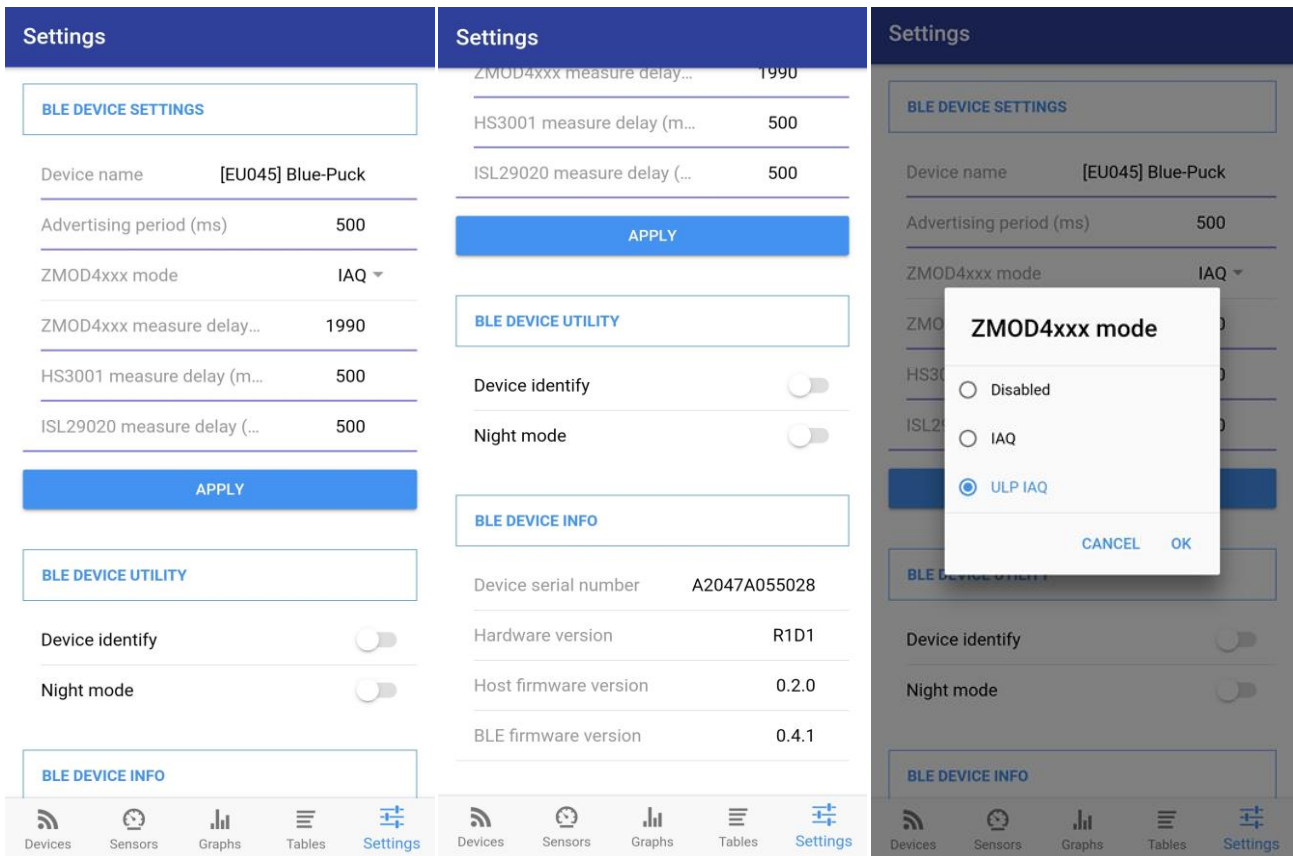
Figure 16: Output of Multiple EU045 solution kits, visualized in the Graphs Tab

- **Tables Tab:** This tab becomes available after the user selects one of the EU045 solution kits present in the Device tab List. Using this tab, the user can visualize and collect the selected EU045 solution kits output data into a table format (Figure 17) and optionally also allows export of these data into a .csv file (timestamp will be included in the csv file)

T (°C)	H (%)	(lux)	IAQ	TVOC (mg/m3)	eCO2 (ppm)
21.5	46.6	516	0.9	0.0	0
21.5	46.6	522	0.9	0.0	0
21.6	46.6	521	0.9	0.0	0
21.6	46.5	522	0.9	0.0	0
21.6	46.5	517	0.9	0.0	0
21.6	46.4	523	0.9	0.0	0
21.6	46.4	520	0.9	0.0	0
21.6	46.3	517	0.9	0.0	0
21.6	46.2	516	0.9	0.0	0
21.7	46.2	521	0.9	0.0	0
21.7	46.2	518	0.9	0.0	0
21.7	46.2	521	0.9	0.0	0
21.7	46.1	522	0.9	0.0	0

Figure 17: Tables Tab, with the EU045 solution kit’s Output Data (Blue-puck Version)

- Settings Tab:** This tab becomes available after the user selects one of the EU045 solution kits present in the Device tab List. The tab enables the user to change the selected EU045 solution kit's configuration parameters (Device Name, Sensor Parameters), identify the kit (see Identify in Section 5.1.1) and visualize additional information about the kit



**Figure 18: Setting Tab, with the EU045 solution kit’s Configuration Parameters, Serial Number, Board HW and FW versions.**

## 6. APPENDIX – Renesas ZMOD Libraries

The ZMOD libraries are already part of the solution kit.

Hence, the following steps are NOT needed in many cases.

However, to manually update the ZMOD libraries for another project or context, please see the following subsections as guidance on how to do and refer to the ZMOD product pages for more information <sup>1</sup>:

- <https://www.renesas.com/us/en/products/sensor-products/environmental-sensors/digital-gas-sensors/zmod4410-indoor-air-quality-sensor-platform>
- <https://www.renesas.com/us/en/products/sensor-products/environmental-sensors/digital-gas-sensors/zmod4510-outdoor-air-quality-sensor-platform>
- <https://www.renesas.com/us/en/products/sensor-products/environmental-sensors/digital-gas-sensors/zmod4450-refrigeration-air-quality-sensor-platform>

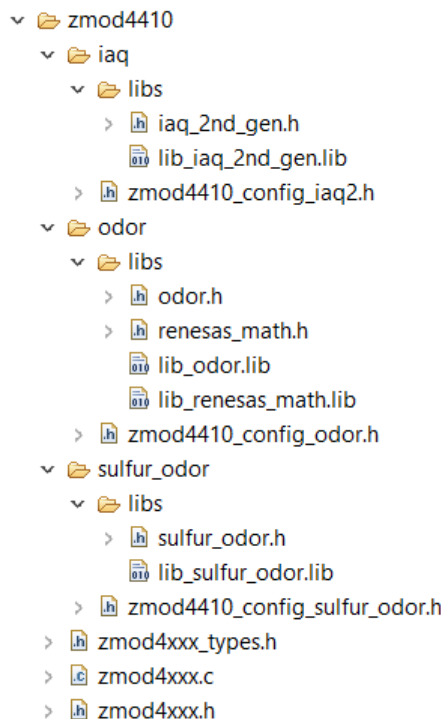
### 6.1.1 ZMOD4410 Library ver. 2.1.1 - 20201001 (used in firmware version “A”)

#### 6.1.1.1 Library Integration Procedure

Figure 25 shows all the files required for the ZMOD4410 drivers and the suggested layout inside the zmod4410 folder. The zmod4410 folder must be included into the project at:

```
<project_name>\src\drivers\
```

inserting the required files as second step.



**Figure 19: ZMOD4410's driver folder with the Library files**

The required files are part of the sensor library and must be copied from the downloaded archives

- “REN\_ZMOD4410-AirQuality-eCO2-FW-2nd-Gen-2p1p1\_SWR\_20201001.zip”.
- “REN\_ZMOD4410-Odor-Firmware-2p1p1\_SWR\_20201001.zip”.

- "REN\_ZMOD4410-Sulfur-Odors-Firmware-2p1p1\_SWR\_20201001.zip".

In detail:

1. "lib\_iaq\_2nd\_gen.lib" from the folder  
"Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.1.1\Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\ZMOD4410\_Firmware\gas-algorithm-libraries\iaq\_2nd\_gen\Renesas RL78\S2-core\ccrl"
2. "iaq\_2nd\_gen.h" from the folder  
"Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.1.1\Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\ZMOD4410\_Firmware\gas-algorithm-libraries\iaq\_2nd\_gen\Renesas RL78\S2-core\ccrl"
3. "zmod4410\_config\_iaq2.h" from the folder  
"Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.1.1\Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\ZMOD4410\_Firmware\zmod4xxx\_example\src"
4. "lib\_odor.lib" from the folder  
"Renesas\_ZMOD4410\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_odor\_firmware\ZMOD4410\_Firmware\gas-algorithm-libraries\odor\Renesas RL78\S2-core\ccrl"
5. "lib\_renesas\_math.lib" from the folder  
"Renesas\_ZMOD4410\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_odor\_firmware\ZMOD4410\_Firmware\gas-algorithm-libraries\odor\Renesas RL78\S2-core\ccrl"
6. "odor.h" from the folder  
"Renesas\_ZMOD4410\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_odor\_firmware\ZMOD4410\_Firmware\gas-algorithm-libraries\odor\Renesas RL78\S2-core\ccrl"
7. "renesas\_math.h" from the folder  
"Renesas\_ZMOD4410\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_odor\_firmware\ZMOD4410\_Firmware\gas-algorithm-libraries\odor\Renesas RL78\S2-core\ccrl"
8. "zmod4410\_config\_odor.h" from the folder  
"Renesas\_ZMOD4410\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_odor\_firmware\ZMOD4410\_Firmware\zmod4xxx\_example\src"
9. "lib\_sulfur\_odor.lib" from the folder  
"Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\ZMOD4410\_Firmware\gas-algorithm-libraries\sulfur\_odor\Renesas RL78\S2-core\ccrl"
10. "sulfur\_odor.h" from the folder  
"Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\ZMOD4410\_Firmware\gas-algorithm-libraries\sulfur\_odor\Renesas RL78\S2-core\ccrl"
11. "zmod4410\_config\_sulfur\_odor.h" from the folder  
"Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\ZMOD4410\_Firmware\zmod4xxx\_example\src"
12. "zmod4xxx.c" from one of the following folders  
"Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.1.1\Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\ZMOD4410\_Firmware\zmod4xxx\_example\src",  
"Renesas\_ZMOD4410\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_odor\_firmware\ZMOD4410\_Firmware\zmod4xxx\_example\src"  
"Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\ZMOD4410\_Firmware\zmod4xxx\_example\src"
13. "zmod4xxx.h" from one of the following folders  
"Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.1.1\Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\ZMOD4410\_Firmware\zmod4xxx\_example\src",  
"Renesas\_ZMOD4410\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_odor\_firmware\ZMOD4410\_Firmware\zmod4xxx\_example\src"

“Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\ZMOD4410\_Firmware\zmod4xxx\_example\src”

14. “zmod4xxx\_types.h” from one of the following folders

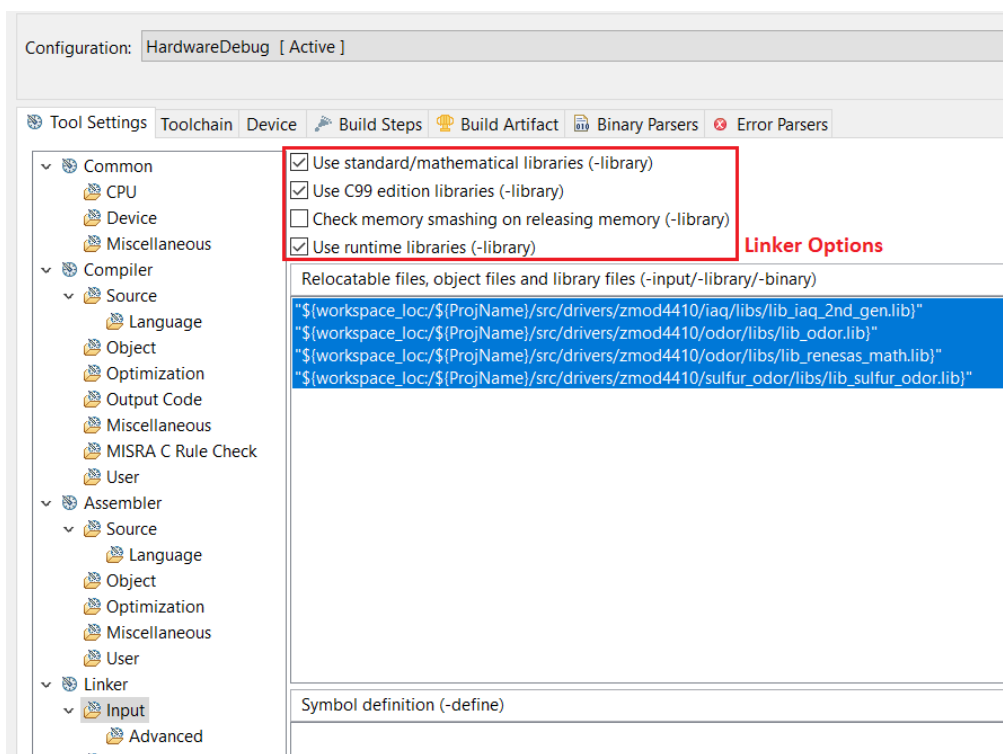
“Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.1.1\Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\ZMOD4410\_Firmware\zmod4xxx\_example\src”,

“Renesas\_ZMOD4410\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_odor\_firmware\ZMOD4410\_Firmware\zmod4xxx\_example\src”

“Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\_2.1.1\Renesas\_ZMOD4410\_Sulfur\_Odor\_Example\ZMOD4410\_Firmware\zmod4xxx\_example\src”

Once all the files are copied, the libraries must be included in the project as shown in Figure 26 and Figure 27.

Figure 26 also shows the Linker Options Values to link the required Standard Library.



**Figure 20: ZMOD4410 compiled libraries added to the project**



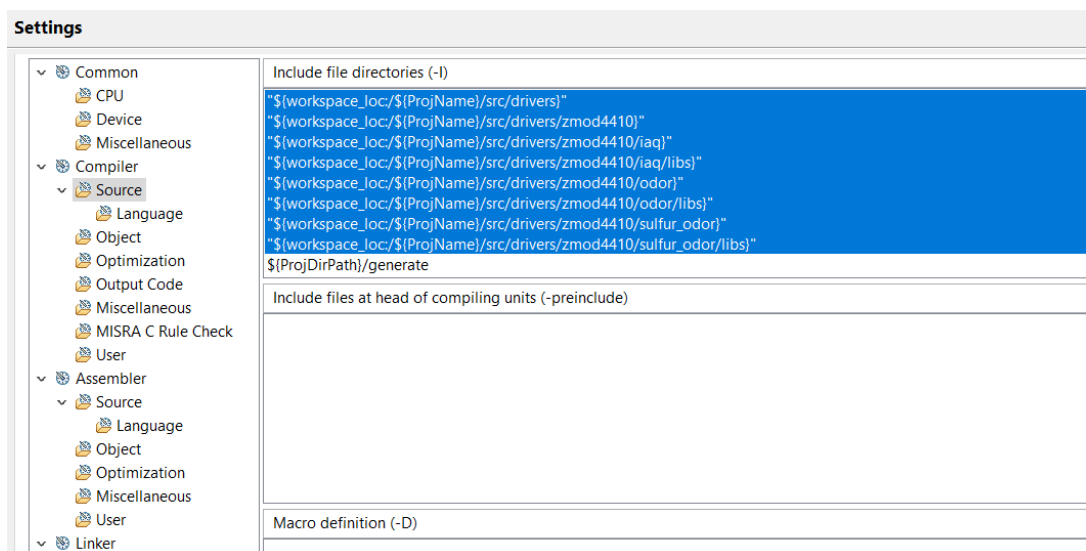


Figure 21: ZMOD4410 libraries folders added to the project

Lastly, the ZMOD4410 driver must be enabled through the file

```
<project_name>\src\common\proj_conf.h
```

defining the symbol "ZMOD4410\_SENSOR" as "SENSOR\_PRESENT" (Figure 28) .

Figure 28 also shows the symbol "ZMOD\_4410\_END\_DET" that defines the method used by the sensor driver to detect when a new Air quality Sensor Output Value is available.

The two method are:

- Polling the Sensor Status Register Value ("ZMOD4410\_END\_DET" defined as "ZMOD4XXX\_INT\_ABSENT"), (which fields indicate when the current measure is complete) on the I2C Internal Bus
- Using the ZMOD INT output pin value [5] ("ZMOD4410\_END\_DET" defined as "ZMOD4XXX\_INT\_PRESENT")

The "ZMOD4410\_END\_DET" is set "ZMOD4XXX\_INT\_PRESENT" as default in order to use the I2C peripheral to retrieve only the Sensor Output Data, optimizing the use of the I2C Internal Bus that is shared between all the sensor present on the EU045 solution kit.

```

18  *#ifndef PROJECT_CONF_H_
19  *#define PROJECT_CONF_H_
20
21
22  * Includes <System Includes> , "Project Includes"
23
24  * Macro definitions
25
26
27
28  *#include "common_macro.h"
29
30  *#define SENSOR_ABSENT ABSENT /*Sensor Absent*/
31  *#define SENSOR_PRESENT PRESENT /*Sensor Present*/
32
33  *#define ZMOD4XXX_INTERRUPT_PIN_ABSENT (0) /*ZMOD4XXX Interrupt pin not connected (polling)*/
34  *#define ZMOD4XXX_INTERRUPT_PIN_PRESENT (1) /*ZMOD4XXX Interrupt pin connected */
35
36
37  /*ZMOD4410 Configuration*/
38  *#define ZMOD4410_SENSOR SENSOR_PRESENT /*SENSOR_ABSENT/SENSOR_PRESENT*/
39  *#define ZMOD4410_END_DET ZMOD4XXX_INTERRUPT_PIN_PRESENT /*ZMOD4XXX_INTERRUPT_PIN_ABSENT/

```

Figure 22: ZMOD4410 enabled through the project\_conf.h file

### 6.1.1.2 Library Modification Procedure

To include all the ZMOD4410 Mode libraries (i.e. IAQ, Odor and Sulfur Odor), in the same firmware two modification type are required to avoid conflicts between the libraries file.

The first modification involves the files

- zmod4410\_config\_iaq2.h
- zmod4410\_config\_odor.h
- zmod4410\_config\_sulfur\_odor.h

Figure 29 shows the code change required to solve the libraries conflicts: the attribute “static” is added to all variables defined in the file. The same change is to be applied to all the files indicated above.

The second modification involves the files:

- iaq\_2nd\_gen.h
- odor.h
- sulfur\_odor.h

where the type “algorithm\_version” is defined. In these files, this type must be re-named to avoid conflicts. Figure 24 shows an example of the suggested modification where in “iaq\_2nd\_gen.h” the “algorithm\_version” is renamed into “algorithm\_version\_EU045\_iaq\_t”.

<pre> #define ZMOD4410_PROD_DATA_LEN 7  /*EU045 Added the attribute static */ static uint8_t data_set_4410i[] = {     0x00, 0x50,     0x00, 0x28, 0xC3, 0xE3,     0x00, 0x00, 0x80, 0x40};  /*EU045 Added the attribute static */ static uint8_t data_set_4410_iaq_2nd_gen[] = {     0x00, 0x50, 0xFF, 0x38,     0xFE, 0xD4, 0xFE, 0x70,     0xFE, 0x0C, 0xFD, 0xA8,     0xFD, 0x44, 0xFC, 0xE0,     0x00, 0x52, 0x02, 0x67,     0x00, 0xCD, 0x03, 0x34,     0x23, 0x03, 0xA3, 0x43,     0x00, 0x00, 0x06, 0x49,     0x06, 0x4A, 0x06, 0x4B,     0x06, 0x4C, 0x06, 0x4D,     0x06, 0x4E, 0x06, 0x97,     0x06, 0xD7, 0x06, 0x57,     0x06, 0x4E, 0x06, 0x4D,     0x06, 0x4C, 0x06, 0x4B,     0x06, 0x4A, 0x86, 0x59};  /*EU045 Added the attribute static */ static zmod4xxx_conf zmod_sensor_type[] = {     [INIT] = {         .start = 0x80,         .h = { .addr = ZMOD4410_H_ADDR, .len = 2, .data_buf = &amp;data_set_4410i[0]},         .d = { .addr = ZMOD4410_D_ADDR, .len = 2, .data_buf = &amp;data_set_4410i[2]},         .m = { .addr = ZMOD4410_M_ADDR, .len = 2, .data_buf = &amp;data_set_4410i[4]},         .s = { .addr = ZMOD4410_S_ADDR, .len = 4, .data_buf = &amp;data_set_4410i[6]},         .r = { .addr = 0x97, .len = 4},     }, }, </pre>	<pre> #define ZMOD4410_PROD_DATA_LEN 7  uint8_t data_set_4410i[] = {     0x00, 0x50,     0x00, 0x28, 0xC3, 0xE3,     0x00, 0x00, 0x80, 0x40};  uint8_t data_set_4410_iaq_2nd_gen[] = {     0x00, 0x50, 0xFF, 0x38,     0xFE, 0xD4, 0xFE, 0x70,     0xFE, 0x0C, 0xFD, 0xA8,     0xFD, 0x44, 0xFC, 0xE0,     0x00, 0x52, 0x02, 0x67,     0x00, 0xCD, 0x03, 0x34,     0x23, 0x03, 0xA3, 0x43,     0x00, 0x00, 0x06, 0x49,     0x06, 0x4A, 0x06, 0x4B,     0x06, 0x4C, 0x06, 0x4D,     0x06, 0x4E, 0x06, 0x97,     0x06, 0xD7, 0x06, 0x57,     0x06, 0x4E, 0x06, 0x4D,     0x06, 0x4C, 0x06, 0x4B,     0x06, 0x4A, 0x86, 0x59};  zmod4xxx_conf zmod_sensor_type[] = {     [INIT] = {         .start = 0x80,         .h = { .addr = ZMOD4410_H_ADDR, .len = 2, .data_buf = &amp;data_set_4410i[0]},         .d = { .addr = ZMOD4410_D_ADDR, .len = 2, .data_buf = &amp;data_set_4410i[2]},         .m = { .addr = ZMOD4410_M_ADDR, .len = 2, .data_buf = &amp;data_set_4410i[4]},         .s = { .addr = ZMOD4410_S_ADDR, .len = 4, .data_buf = &amp;data_set_4410i[6]},         .r = { .addr = 0x97, .len = 4},     }, }, </pre>
---	--

**Figure 23: zmod4410\_config\_iaq2.h after and before the code modification to solve the libraries conflicts**

```

*| * Copyright (c) 2020 Renesas Electronics Corporation
* * @file iaq_2nd_gen.h
*
*#ifndef IAQ_2ND_GEN_H
*#define IAQ_2ND_GEN_H
*
*#ifdef __cplusplus
*extern "C" {
*#endif
*
*#include <stdint.h>
*#include <math.h>
*#include "zmod4xxx_types.h"
*
* * @brief Variables that describe the library version
*#typedef struct {
*    uint8_t major;
*    uint8_t minor;
*    uint8_t patch;
*} algorithm_version_EU045_iaq_t; /*EU045 Changed the name from algorithm_version to algorithm_version_EU045_iaq_t */
*
*/** \addtogroup RetCodes Return codes of the algorithm functions.
*#define IAQ_2ND_GEN_OK (0) /**< everything okay */
*#define IAQ_2ND_GEN_STABILIZATION (1) /**< sensor in stabilization */
*/** @*/

```

Figure 24: iaq\_2nd\_gen.h after and before the code modification to solve the libraries conflicts

## 6.1.2 ZMOD4410 Library ver. 2.2.0 - 20210827 (used in firmware version “B”)

### 6.1.2.1 Library Integration Procedure

Figure 25 shows all the files required for the ZMOD4410 drivers and the suggested layout inside the drivers folder. The zmod4410 folder must be included into the project at:

```
<project_name>\src\drivers\
```

inserting the required files as second step.

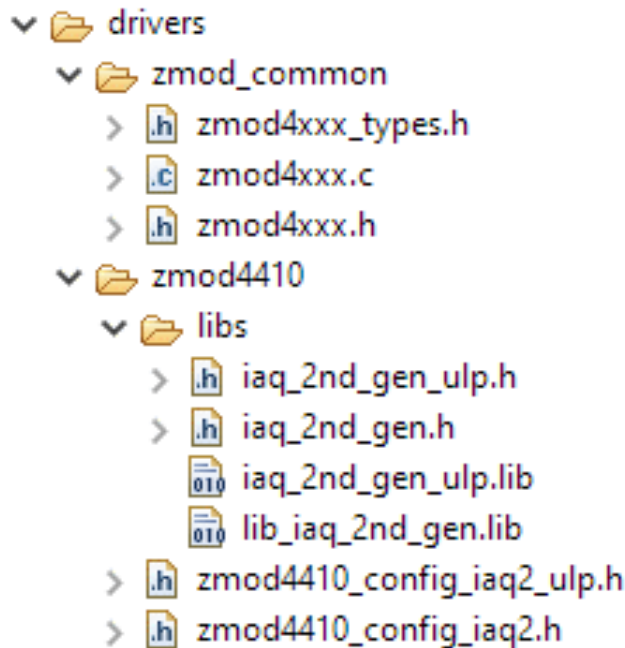


Figure 25: ZMOD4410's driver folder with the Library files

A common folder is also present. It contains the API interface of the ZMOD sensors, valid for all versions.

The required files are part of the sensor library and must be copied from the downloaded archives

- “Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.2.0.zip”.

- “zmod4xxx\_iaq\_2nd\_gen\_ulp\_example\_2021108.zip”.

In detail:

1. “lib\_iaq\_2nd\_gen.lib” from the folder  
“Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.2.0\gas-algorithm-libraries\iaq\_2nd\_gen\Renesas RL78\S2-core\ccr1”
2. “iaq\_2nd\_gen.h” from the folder  
“Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.2.0\gas-algorithm-libraries\iaq\_2nd\_gen\Renesas RL78\S2-core\ccr1”
3. “zmod4410\_config\_iaq2.h” from the folder  
“Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.2.0\zmod4xxx\_evk\_example\src”
4. “iaq\_2nd\_gen\_ulp.lib” from the folder  
“zmod4xxx\_iaq\_2nd\_gen\_ulp\_example\_2021108”
5. “iaq\_2nd\_gen\_ulp.h” from the folder  
“zmod4xxx\_iaq\_2nd\_gen\_ulp\_example\_2021108”
6. “zmod4410\_config\_iaq2\_ulp.h” from the folder  
“zmod4xxx\_iaq\_2nd\_gen\_ulp\_example\_2021108\src”
7. “zmod4xxx.c” from the folder  
“Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.2.0\zmod4xxx\_evk\_example\src”
8. “zmod4xxx.h” from the folder  
“Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.2.0\zmod4xxx\_evk\_example\src”
9. “zmod4xxx\_types.h” from the folder  
“Renesas\_ZMOD4410\_IAQ\_2nd\_Gen\_Example\_2.2.0\zmod4xxx\_evk\_example\src”

Once all the files are copied, the libraries must be included in the project as shown in Figure 26 and Figure 27.

Figure 26 also shows the Linker Options Values to link the required Standard Library.

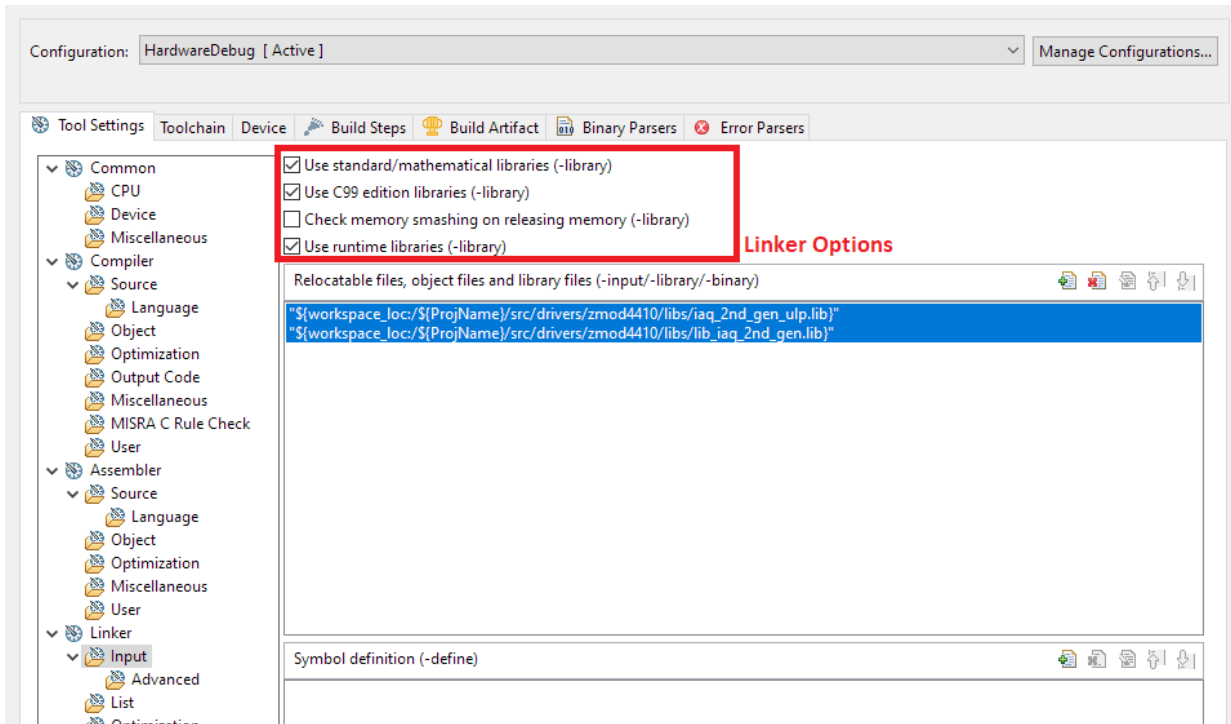


Figure 26: ZMOD4410 compiled libraries added to the project

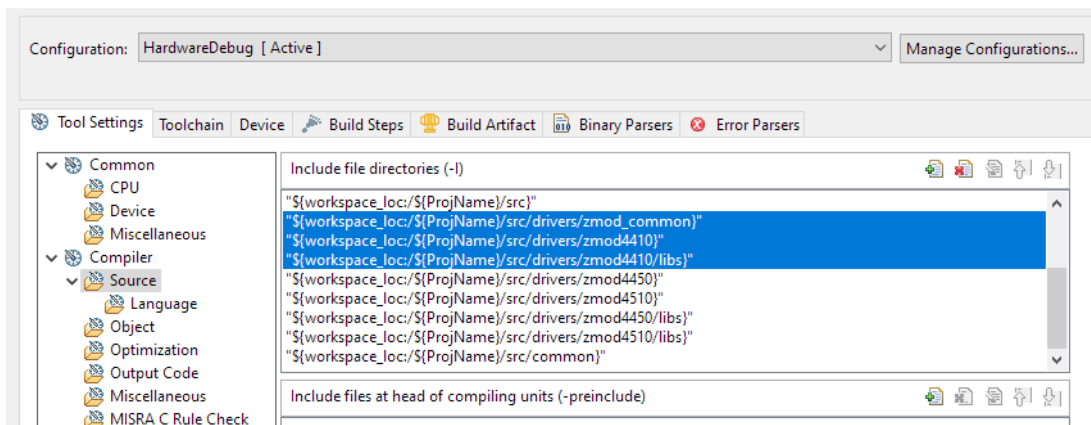


Figure 27: ZMOD4410 libraries folders added to the project

Lastly, the ZMOD4410 driver must be enabled through the file

```
<project_name>\src\common\proj_conf.h
```

defining the symbol "ZMOD4410\_SENSOR" as "SENSOR\_PRESENT" (Figure 28) .

Figure 28 also shows the symbol "ZMOD\_4410\_END\_DET" that defines the method used by the sensor driver to detect when a new Air quality Sensor Output Value is available.

The two method are:

- Polling the Sensor Status Register Value ("ZMOD4410\_END\_DET" defined as "ZMOD4XXX\_INT\_ABSENT"), (which fields indicate when the current measure is complete) on the I2C Internal Bus
- Using the ZMOD INT output pin value [5] ("ZMOD4410\_END\_DET" defined as "ZMOD4XXX\_INT\_PRESENT")

The "ZMOD4410\_END\_DET" is set "ZMOD4XXX\_INT\_PRESENT" as default in order to use the I2C peripheral to retrieve only the Sensor Output Data, optimizing the use of the I2C Internal Bus that is shared between all the sensor present on the EU045 solution kit.

```

18  *#ifndef PROJECT_CONF_H_
19  #define PROJECT_CONF_H_
20
22  * Includes <System Includes> , "Project Includes"
25  * Macro definitions
27
28  #include "common_macro.h"
29
30  #define SENSOR_ABSENT ABSENT /*Sensor Absent*/
31  #define SENSOR_PRESENT PRESENT /*Sensor Present*/
32
33  #define ZMOD4XXX_INTERRUPT_PIN_ABSENT (0) /*ZMOD4XXX Interrupt pin not connected (polling)*/
34  #define ZMOD4XXX_INTERRUPT_PIN_PRESENT (1) /*ZMOD4XXX Interrupt pin connected */
35
36
37  /*ZMOD4410 Configuration*/
38  #define ZMOD4410_SENSOR SENSOR_PRESENT /*SENSOR_ABSENT/SENSOR_PRESENT*/
39  #define ZMOD4410_END_DET ZMOD4XXX_INTERRUPT_PIN_PRESENT /*ZMOD4XXX_INTERRUPT_PIN_ABSENT/

```

Figure 28: ZMOD4410 enabled through the project\_conf.h file

### 6.1.2.2 Library Modification Procedure

To include all the ZMOD4410 Mode libraries (i.e. IAQ and IAQ ULP), in the same firmware two modification type are required to avoid conflicts between the libraries file.

The modifications involve the files

- iaq\_2nd\_gen\_ulp.h
- zmod4410\_config\_iaq2\_ulp.h

Figure 29 and Figure 30 show the code change required to solve the library conflicts.

<pre> ...@file...iaq_2nd_gen_ulp.h ...@file...iaq_2nd_gen_ulp.h  #ifndef IAQ_2ND_GEN_ULP_H #define IAQ_2ND_GEN_ULP_H  #ifdef __cplusplus extern "C" { #endif  #include &lt;stdint.h&gt; #include &lt;math.h&gt; #include "zmod4xxx_types.h"  /**@brief Variables that describe the library version  *#if 0  *typedef struct {  *...uint8_t major;  *...uint8_t minor;  *...uint8_t patch;  *}algorithm_version;  *#endif  /**@addtogroup RetCodes Return codes of the algorithm functions.  *#define IAQ_2ND_GEN_ULP_OK (0) /**&lt; everything okay */  *#define IAQ_2ND_GEN_ULP_STABILIZATION (1) /**&lt; sensor in stabilization */  *#endif  /**@brief Variables that describe the sensor or the algorithm state.  *typedef struct {  *...float log_nonlog_rcda[3]; /**&lt; various baselines. */  *...uint8_t  *...stabilization_sample; /**&lt; Number of samples still needed for stabilization. */  *...uint8_t need_filter_init;  *...float tvoc_smooth;  *...float tvoc_deltafilter;  *...float acchw;  *...float acchw; </pre>	<pre> ...@file...iaq_2nd_gen_ulp.h ...@file...iaq_2nd_gen_ulp.h  #ifndef IAQ_2ND_GEN_ULP_H #define IAQ_2ND_GEN_ULP_H  #ifdef __cplusplus extern "C" { #endif  #include &lt;stdint.h&gt; #include &lt;math.h&gt; #include "zmod4xxx_types.h"  /**@brief Variables that describe the library version  *typedef struct {  *...uint8_t major;  *...uint8_t minor;  *...uint8_t patch;  *}algorithm_version;  *#endif  /**@addtogroup RetCodes Return codes of the algorithm functions.  *#define IAQ_2ND_GEN_ULP_OK (0) /**&lt; everything okay */  *#define IAQ_2ND_GEN_ULP_STABILIZATION (1) /**&lt; sensor in stabilization */  *#endif  /**@brief Variables that describe the sensor or the algorithm state.  *typedef struct {  *...float log_nonlog_rcda[3]; /**&lt; various baselines. */  *...uint8_t  *...stabilization_sample; /**&lt; Number of samples still needed for stabilization. */  *...uint8_t need_filter_init;  *...float tvoc_smooth;  *...float tvoc_deltafilter;  *...float acchw;  *...float acchw;  *...float etoh; </pre>
--	---

Figure 29: iaq\_2nd\_gen\_ulp.h after and before the code modification to solve the libraries conflicts

```

#define ZMOD4410_IAQ2_ULP_SEQ_RUN_TIME_WITH_MARGIN (1500U)
#define ZMOD4410_IAQ2_ULP_SAMPLE_TIME (90000U)

// clang-format off
uint8_t data_set_4410_ulp[] = {
// REMOVE heater
..... 0x00, 0x50,
// REMOVE delay, measurement
..... 0x00, 0x28, 0xC3, 0xE3,
// REMOVE sequencer
..... 0x00, 0x00, 0x80, 0x40};

uint8_t data_set_4410_iaq_2nd_gen_ulp[] = {
// REMOVE heater
..... 0x00, 0x50, 0xFF, 0x38,
..... 0xFE, 0xD4, 0xFE, 0x70,
..... 0xFE, 0x0C, 0xFD, 0xA8,
..... 0xFD, 0x44, 0xFC, 0xE0,
// REMOVE delay
..... 0x00, 0x52, 0x02, 0x67,
..... 0x00, 0xCD, 0x03, 0x34,
// REMOVE measurement
..... 0x23, 0x03, 0xA3, 0x43,
// REMOVE sequencer
..... 0x00, 0x00, 0x06, 0x49,
..... 0x06, 0x4A, 0x06, 0x4B,
..... 0x06, 0x4C, 0x06, 0x4D,
..... 0x06, 0x4E, 0x06, 0x97,
..... 0x06, 0xD7, 0x06, 0x57,
..... 0x06, 0x4E, 0x06, 0x4D,
..... 0x06, 0x4C, 0x06, 0x4B,
..... 0x06, 0x4A, 0x06, 0x59};

// clang-format on

zmod4xxx_conf zmod_sensor_type_ulp[] = {
..... [INIT] = {
..... .start = 0x80,
..... .h = { .addr = ZMOD4410_H_ADDR, .len = 2, .data_buf = &data_set_4410_ulp[0]},
..... .d = { .addr = ZMOD4410_D_ADDR, .len = 2, .data_buf = &data_set_4410_ulp[2]},
..... .m = { .addr = ZMOD4410_M_ADDR, .len = 2, .data_buf = &data_set_4410_ulp[4]},
..... .s = { .addr = ZMOD4410_S_ADDR, .len = 4, .data_buf = &data_set_4410_ulp[6]},
..... .r = { .addr = 0x97, .len = 4},
..... },
};

```

Figure 30: zmod4410\_config\_iaq2\_ulp.h after and before the code modification to solve the library conflicts

### 6.1.3 ZMOD4510 Library ver. 20191014 (used in firmware version “A”)

#### 6.1.3.1 Library Integration Procedure

Similarly, to the ZMOD4410, in Figure 36 shows all the files required for the ZMOD4510 drivers and the suggested layout inside the zmod4510 folder. The zmod4510 folder must be included into the project at:

```
<project_name>\src\drivers\
```

inserting the required files as second step.

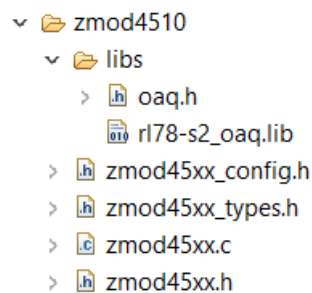


Figure 31: ZMOD4510's driver folder with the Library files

The required files are part of the sensor library and must be copied from the downloaded archive:

```
IDT_ZMOD4510-OAQ-Firmware_SWR_20191014.zip
```

In details:

1. “oaq.h” from the folder  
“IDT\_ZMOD4510-Outdoor-Air-Quality-Firmware\_SWR\_20191014\ZMOD4510-oaq-firmware\oaq-libraries\_4.0.0.zip\RL78\oaq\_rl78”

2. "rl78\_s2\_oaq.lib" from the folder  
"IDT\_ZMOD4510-Outdoor-Air-Quality-Firmware\_SWR\_20191014\ZMOD4510-oaq-firmware\oaq-libraries\_4.0.0.zip\RL78\oaq\_rl78\lib"
3. "zmod45xx\_config.h" from the folder  
"IDT\_ZMOD4510-Outdoor-Air-Quality-Firmware\_SWR\_20191014\ZMOD4510-oaq-firmware\ZMOD45xx\_example\_2.0.0.zip\src"
4. "zmod45xx\_types.h" from the folder  
"IDT\_ZMOD4510-Outdoor-Air-Quality-Firmware\_SWR\_20191014\ZMOD4510-oaq-firmware\ZMOD45xx\_example\_2.0.0.zip\src"
5. "zmod45xx.h" from the folder  
"IDT\_ZMOD4510-Outdoor-Air-Quality-Firmware\_SWR\_20191014\ZMOD4510-oaq-firmware\ZMOD45xx\_example\_2.0.0.zip\src"
6. "zmod45xx.c" from the folder  
"IDT\_ZMOD4510-Outdoor-Air-Quality-Firmware\_SWR\_20191014\ZMOD4510-oaq-firmware\ZMOD45xx\_example\_2.0.0.zip\src"

Once copied all the files, the libraries must be included in the project as shown in Figure 37 and Figure 38. Figure 37 also shows the Linker Options Values to link the required Standard Library.

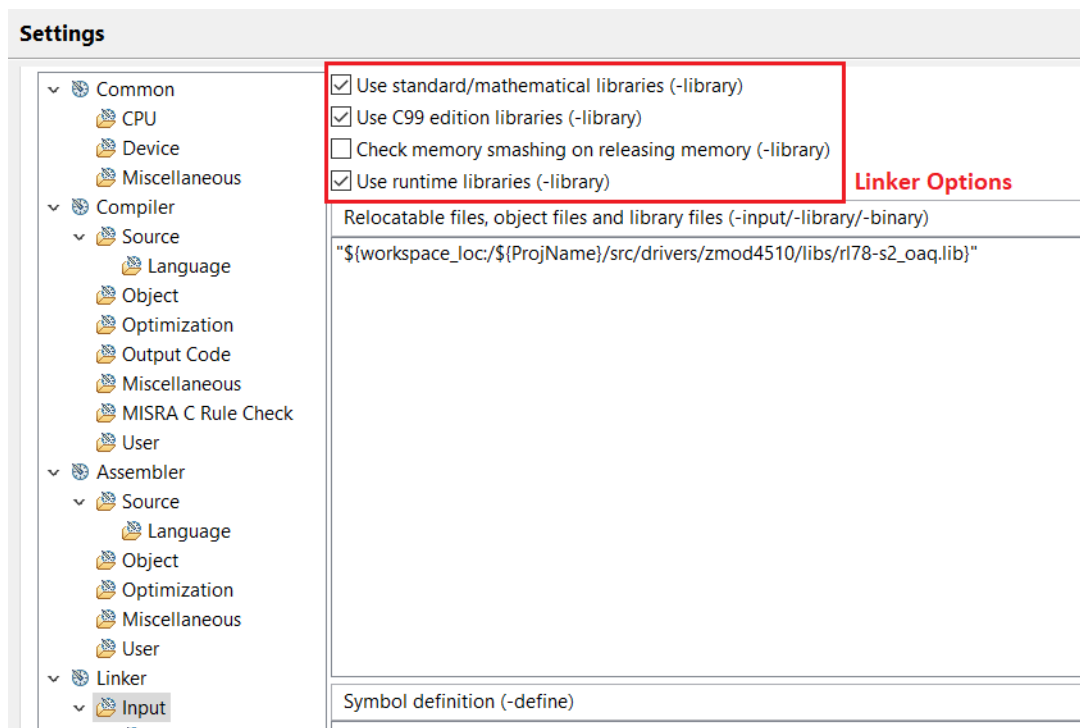


Figure 32: ZMOD4510 compiled libraries added to the project.



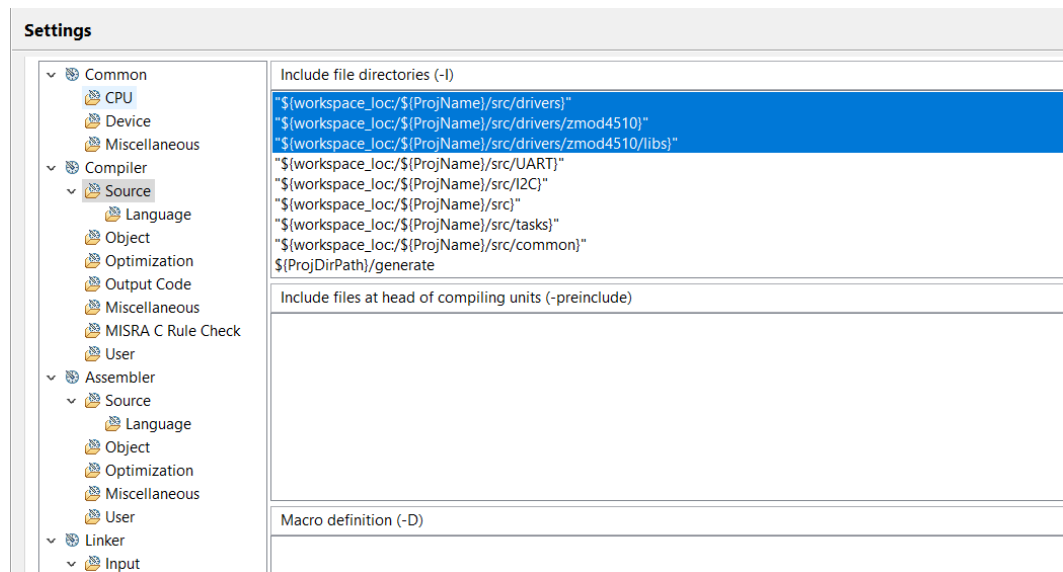
Lastly, the ZMOD4510 driver must be enabled setting the symbol “ZMOD4510\_SENSOR” as “SENSOR\_PRESENT” in the file

```
<project_name>\src\common\proj_conf.h
```

as indicated in Figure 39.

Figure 39 also shows the symbol “ZMOD\_4510\_END\_DET” that defines the method used by the sensor driver to detect when a new Air quality Sensor Output Value is available.

- Polling the Sensor Status Register Value (“ZMOD4510\_END\_DET” defined as “ZMOD4XXX\_INT\_ABSENT”), (which fields indicate when the current measure is complete) on the I2C Internal Bus



**Figure 33: ZMOD4510 libraries folders added to the project.**

- Using the ZMOD INT output pin value [6] (“ZMOD4510\_END\_DET” defined as “ZMOD4XXX\_INT\_PRESENT”)

The “ZMOD4510\_END\_DET” is set “ZMOD4XXX\_INT\_PRESENT” as default in order to use the I2C peripheral to retrieve only the Sensor Output Data, optimizing the use of the I2C Internal Bus that is shared between all the sensor present on the EU045 solution kit.

```

22  * Includes <System Includes> , "Project Includes"
25  * Macro definitions
27
28  #include "common_macro.h"
29
30  #define SENSOR_ABSENT ABSENT /*Sensor Absent*/
31  #define SENSOR_PRESENT PRESENT /*Sensor Present*/
32
33  #define ZMOD4XXX_INTERRUPT_PIN_ABSENT (0) /*ZMOD4XXX Interrupt pin not connected (polling)*/
34  #define ZMOD4XXX_INTERRUPT_PIN_PRESENT (1) /*ZMOD4XXX Interrupt pin connected */
35
36
37  /*ZMOD4410 Configuration*/
38  #define ZMOD4410_SENSOR SENSOR_ABSENT /*SENSOR_ABSENT/SENSOR_PRESENT*/
39  #define ZMOD4410_END_DET ZMOD4XXX_INTERRUPT_PIN_PRESENT /*ZMOD4XXX_INTERRUPT_PIN_ABSENT/
40
41  /*ZMOD4510 Configuration*/
42  #define ZMOD4510_SENSOR SENSOR_PRESENT /*SENSOR_ABSENT/SENSOR_PRESENT*/
43  #define ZMOD4510_END_DET ZMOD4XXX_INTERRUPT_PIN_PRESENT /*ZMOD4XXX_INTERRUPT_PIN_ABSENT/
44

```

Figure 34: ZMOD4510 enabled through the project\_conf.h file

### 6.1.3.2 Library Modification Procedure

Figure 40 shows the modification to apply to the ZMOD4510 Sensor API file at the path:

```
<project_name>\src\drivers\zmod4510\zmod4510.c @line 152
```

The file contains a “printf” function that has to be removed.

<pre> int8_t zmod45xx_init_sensor(zmod45xx_dev_t *dev) {     int8_t ret = 0;     uint8_t data[4] = { 0 };     uint8_t zmod44xx_status;      if (!dev-&gt;init_conf) {}      ret = dev-&gt;read(dev-&gt;i2c_addr, 0x87, data, 1);     if (ret) {}      ret = zmod45xx_calc_factor(dev, 80, data);     if (ret) {}      ret = dev-&gt;write(dev-&gt;i2c_addr, dev-&gt;init_conf-&gt;h.addr, data,         dev-&gt;init_conf-&gt;h.len);     if (ret) {}      ret = dev-&gt;write(dev-&gt;i2c_addr, dev-&gt;init_conf-&gt;d.addr,         dev-&gt;init_conf-&gt;d.data, dev-&gt;init_conf-&gt;d.len);     if (ret) {}      ret = dev-&gt;write(dev-&gt;i2c_addr, dev-&gt;init_conf-&gt;m.addr,         dev-&gt;init_conf-&gt;m.data, dev-&gt;init_conf-&gt;m.len);     if (ret) {}      ret = dev-&gt;write(dev-&gt;i2c_addr, dev-&gt;init_conf-&gt;s.addr,         dev-&gt;init_conf-&gt;s.data, dev-&gt;init_conf-&gt;s.len);     if (ret) {}      ret =     dev-&gt;write(dev-&gt;i2c_addr, ZMOD45XX_ADDR_CMD, &amp;dev-&gt;init_conf-&gt;start, 1);     if (ret) {         return ERROR_I2C;     }      do {         ret = zmod45xx_read_status(dev, &amp;zmod44xx_status);         if (ret) {             //printf("Error %d, exiting program!\n", ret);             return ret;         }     } </pre>	<pre> int8_t zmod45xx_init_sensor(zmod45xx_dev_t *dev) {     int8_t ret = 0;     uint8_t data[4] = { 0 };     uint8_t zmod44xx_status;      if (!dev-&gt;init_conf) {}      ret = dev-&gt;read(dev-&gt;i2c_addr, 0x87, data, 1);     if (ret) {}      ret = zmod45xx_calc_factor(dev, 80, data);     if (ret) {}      ret = dev-&gt;write(dev-&gt;i2c_addr, dev-&gt;init_conf-&gt;h.addr, data,         dev-&gt;init_conf-&gt;h.len);     if (ret) {}      ret = dev-&gt;write(dev-&gt;i2c_addr, dev-&gt;init_conf-&gt;d.addr,         dev-&gt;init_conf-&gt;d.data, dev-&gt;init_conf-&gt;d.len);     if (ret) {}      ret = dev-&gt;write(dev-&gt;i2c_addr, dev-&gt;init_conf-&gt;m.addr,         dev-&gt;init_conf-&gt;m.data, dev-&gt;init_conf-&gt;m.len);     if (ret) {}      ret = dev-&gt;write(dev-&gt;i2c_addr, dev-&gt;init_conf-&gt;s.addr,         dev-&gt;init_conf-&gt;s.data, dev-&gt;init_conf-&gt;s.len);     if (ret) {}      ret =     dev-&gt;write(dev-&gt;i2c_addr, ZMOD45XX_ADDR_CMD, &amp;dev-&gt;init_conf-&gt;start, 1);     if (ret) {         return ERROR_I2C;     }      do {         ret = zmod45xx_read_status(dev, &amp;zmod44xx_status);         if (ret) {             printf("Error %d, exiting program!\n", ret);             return ret;         }     } </pre>
---	---

Figure 35: zmod45xx.c (ZMOD4510 API File) after and before the code modification

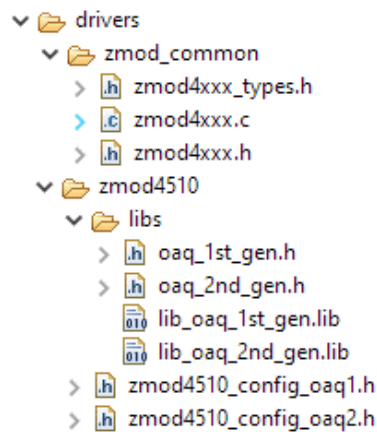
## 6.1.4 ZMOD4510 Library ver. 3.0.0 - 20210527 (used in firmware version “B”)

### 6.1.4.1 Library Integration Procedure

Similarly, to the ZMOD4410, in Figure 36 shows all the files required for the ZMOD4510 drivers and the suggested layout inside the zmod4510 folder. The zmod4510 folder must be included into the project in:

```
<project_name>\src\drivers\
```

placing then the required files as second step.



**Figure 36: ZMOD4510's driver folder with the Library files**

A common folder is also present. It contains the API interface of the ZMOD sensors, valid for all the versions.

The required files are part of the sensor library and must be copied from the downloaded archive:

Renesas\_ZMOD4510\_OAQ\_1st\_Gen\_Example\_3.0.0.zip

Renesas\_ZMOD4510\_OAQ\_2nd\_Gen\_Example\_3.0.0.zip

In details:

1. "oaq\_1st\_gen.h" from the folder  
"Renesas\_ZMOD4510\_OAQ\_1st\_Gen\_Example\_3.0.0\gas-algorithm-libraries\oaq\_1st\_gen\Renesas RL78\S2-core\ccr1"
2. "lib\_oaq\_1st\_gen.lib" from the folder  
"Renesas\_ZMOD4510\_OAQ\_1st\_Gen\_Example\_3.0.0\gas-algorithm-libraries\oaq\_1st\_gen\Renesas RL78\S2-core\ccr1"
3. "zmod4510\_config\_oaq1.h" from the folder  
"Renesas\_ZMOD4510\_OAQ\_1st\_Gen\_Example\_3.0.0\zmod4xxx\_evk\_example\src"
4. "oaq\_2st\_gen.h" from the folder  
"Renesas\_ZMOD4510\_OAQ\_2nd\_Gen\_Example\_3.0.0\gas-algorithm-libraries\oaq\_2nd\_gen\Renesas RL78\S2-core\ccr1"
5. "lib\_oaq\_2st\_gen.lib" from the folder  
"Renesas\_ZMOD4510\_OAQ\_2nd\_Gen\_Example\_3.0.0\gas-algorithm-libraries\oaq\_2nd\_gen\Renesas RL78\S2-core\ccr1"
6. "zmod4510\_config\_oaq2.h" from the folder  
"Renesas\_ZMOD4510\_OAQ\_2st\_Gen\_Example\_3.0.0\zmod4xxx\_evk\_example\src"
7. "zmod4xxx.c" from the folder  
"Renesas\_ZMOD4510\_OAQ\_2nd\_Gen\_Example\_3.0.0\zmod4xxx\_evk\_example\src"
8. "zmod4xxx.h" from the folder  
"Renesas\_ZMOD4510\_OAQ\_2nd\_Gen\_Example\_3.0.0\zmod4xxx\_evk\_example\src"
9. "zmod4xxx\_types.h" from the folder  
"Renesas\_ZMOD4510\_OAQ\_2nd\_Gen\_Example\_3.0.0\zmod4xxx\_evk\_example\src"

Once copied all the files, the libraries must be included in the project as shown in Figure 37 and Figure 38. Figure 37 also shows the Linker Options Values to link the required Standard Library.

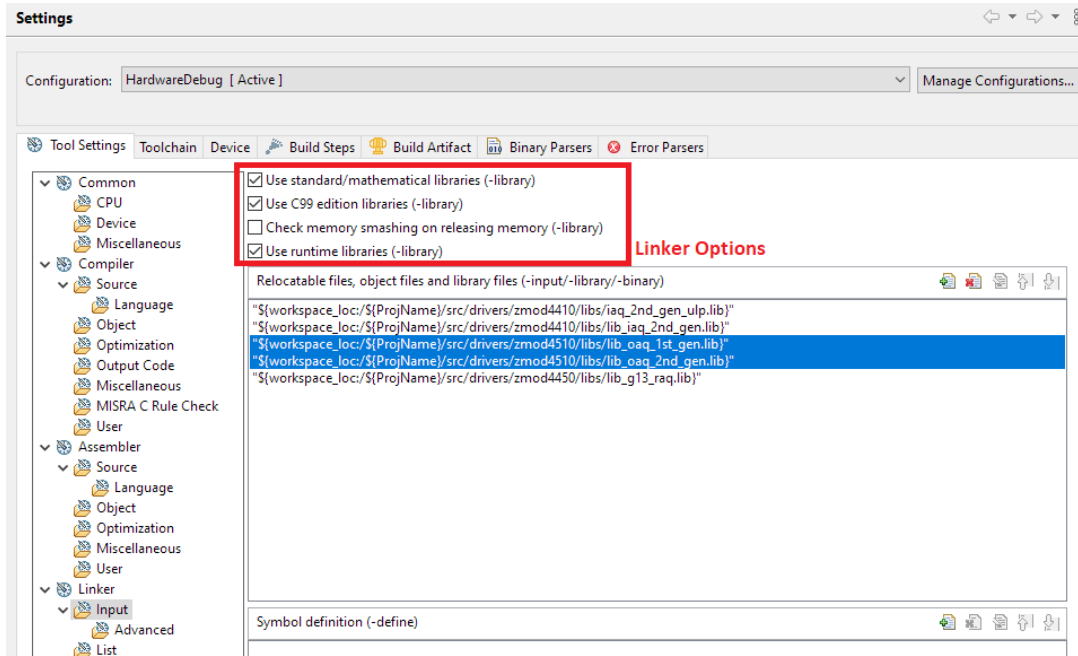


Figure 37: ZMOD4510 compiled libraries added to the project.

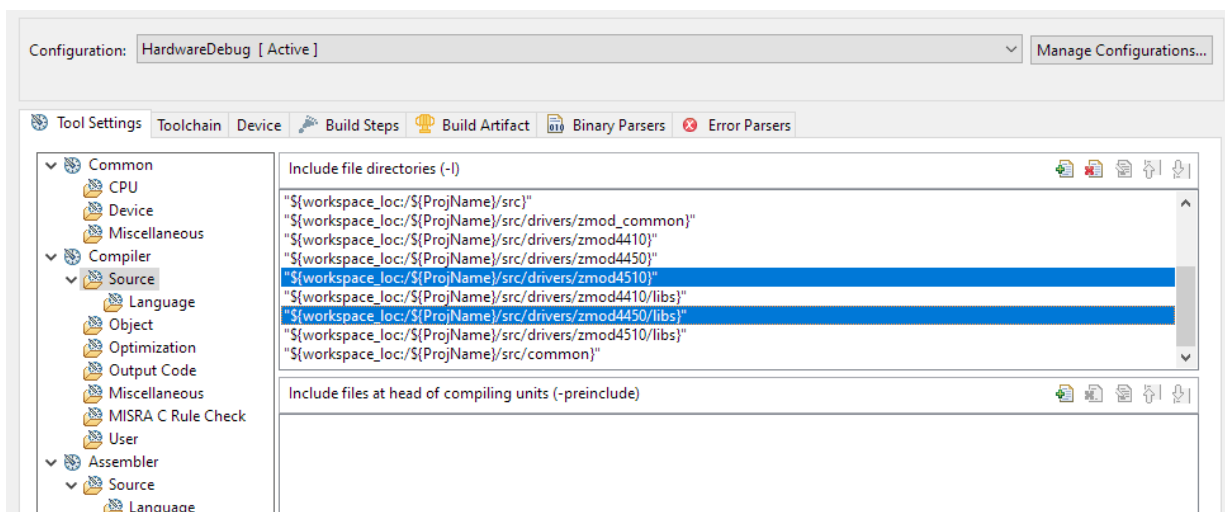


Figure 38: ZMOD4510 libraries folders added to the project.

Lastly, the ZMOD4510 driver must be enabled setting the symbol “ZMOD4510\_SENSOR” as “SENSOR\_PRESENT” in the file

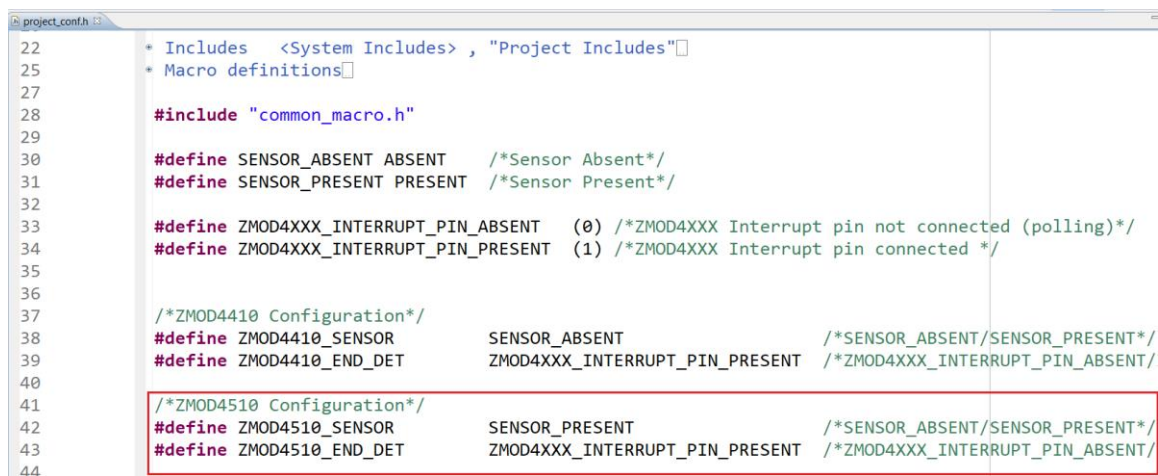
```
<project_name>\src\common\proj_conf.h
```

as indicated in Figure 39.

Figure 39 also shows the symbol “ZMOD\_4510\_END\_DET” that defines the method used by the sensor driver to detect when a new Air quality Sensor Output Value is available.

- Polling the Sensor Status Register Value (“ZMOD4510\_END\_DET” defined as “ZMOD4XXX\_INT\_ABSENT”), (which fields indicate when the current measure is complete) on the I2C Internal Bus
- Using the ZMOD INT output pin value [6] (“ZMOD4510\_END\_DET” defined as “ZMOD4XXX\_INT\_PRESENT”)

The “ZMOD4510\_END\_DET” is set “ZMOD4XXX\_INT\_PRESENT” as default in order to use the I2C peripheral to retrieve only the Sensor Output Data, optimizing the use of the I2C Internal Bus that is shared between all the sensor present on the EU045 solution kit.



```

22 * Includes <System Includes> , "Project Includes"
25 * Macro definitions
27
28 #include "common_macro.h"
29
30 #define SENSOR_ABSENT ABSENT /*Sensor Absent*/
31 #define SENSOR_PRESENT PRESENT /*Sensor Present*/
32
33 #define ZMOD4XXX_INTERRUPT_PIN_ABSENT (0) /*ZMOD4XXX Interrupt pin not connected (polling)*/
34 #define ZMOD4XXX_INTERRUPT_PIN_PRESENT (1) /*ZMOD4XXX Interrupt pin connected */
35
36
37 /*ZMOD4410 Configuration*/
38 #define ZMOD4410_SENSOR SENSOR_ABSENT /*SENSOR_ABSENT/SENSOR_PRESENT*/
39 #define ZMOD4410_END_DET ZMOD4XXX_INTERRUPT_PIN_PRESENT /*ZMOD4XXX_INTERRUPT_PIN_ABSENT/
40
41 /*ZMOD4510 Configuration*/
42 #define ZMOD4510_SENSOR SENSOR_PRESENT /*SENSOR_ABSENT/SENSOR_PRESENT*/
43 #define ZMOD4510_END_DET ZMOD4XXX_INTERRUPT_PIN_PRESENT /*ZMOD4XXX_INTERRUPT_PIN_ABSENT/
44

```

Figure 39: ZMOD4510 enabled through the project\_conf.h file

#### 6.1.4.2 Library Modification Procedure

To include all the ZMOD4510 Mode libraries (i.e. OAQ and OAQ ULP), in the same firmware two modification type are required to avoid conflicts between the libraries file.

The modifications involve the files

- oaq\_2nd\_gen.h

Figure 40 show the code change required to solve the libraries conflicts.

```

...@file... oaq_2nd_gen.h
#ifdef OAQ_2ND_GEN_H
#define OAQ_2ND_GEN_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <math.h>
#include "zmod4xx_types.h"

* @brief Variables that describe the library version[]
#if 0
typedef struct {
...uint8_t major;
...uint8_t minor;
...uint8_t patch;
...algorithm_version;
} oaq_2nd_gen_handle_t;
#endif

/** \addtogroup RetCodes Return codes of the algorithm functions.[]
#define OAQ_2ND_GEN_OK ..... (0) /**< everything okay */
#define OAQ_2ND_GEN_STABILIZATION (1) /**< sensor in stabilization */
/** @} */

* @brief Variables that describe the sensor or the algorithm state.[]
typedef struct {
...uint16_t
...stabilization_sample; /**< Number of samples still needed for stabilization. */
...float gcda[8]; /**< baseline conductances. */
...float log_ra;
...float log_b;
...float beta2;
...float O3_conc_ppb;
...float o3_1h_ppb;
...float o3_8h_ppb;
} oaq_2nd_gen_handle_t;

...@file... oaq_2nd_gen.h
#ifdef OAQ_2ND_GEN_H
#define OAQ_2ND_GEN_H

#ifdef __cplusplus
extern "C" {
#endif

#include <stdint.h>
#include <math.h>
#include "zmod4xxx_types.h"

* @brief Variables that describe the library version[]
typedef struct {
...uint8_t major;
...uint8_t minor;
...uint8_t patch;
} algorithm_version;
/** \addtogroup RetCodes Return codes of the algorithm functions.[]
#define OAQ_2ND_GEN_OK ..... (0) /**< everything okay */
#define OAQ_2ND_GEN_STABILIZATION (1) /**< sensor in stabilization */
/** @} */

* @brief Variables that describe the sensor or the algorithm state.[]
typedef struct {
...uint16_t
...stabilization_sample; /**< Number of samples still needed for stabilization. */
...float gcda[8]; /**< baseline conductances. */
...float log_ra;
...float log_b;
...float beta2;
...float O3_conc_ppb;
...float o3_1h_ppb;
...float o3_8h_ppb;
} oaq_2nd_gen_handle_t;

```

Figure 40: oaq\_2nd\_gen.h (ZMOD4510 API File) after and before the code modification

## 6.1.5 ZMOD4450 Library ver. 1.2.0 - 20200310 (used in firmware version “A” and “B”)

### 6.1.5.1 Library Integration Procedure

Similarly, to the ZMOD4410 and ZMOD4510, Figure 41 shows all the files required for the ZMOD4450 drivers and the suggested layout inside the zmod4450 folder. The zmod4450 folder must be included into the project at:

```
<project_name>\src\drivers\
```

inserting the required files as second step.

```

v zmod4450
v libs
  > idt_math.h
  > raq.h
  > lib_g13_idt_math.lib
  > lib_g13_raq.lib
  > zmod44xx_config.h
  > zmod44xx_types.h
  > zmod44xx.c
  > zmod44xx.h

```

Figure 41: ZMOD4450's driver folder with the Library files

The required files are part of the sensor library and must be copied from the downloaded archive

“IDT\_ZMOD4450-Firmware\_SWR\_20200310.zip”

In detail:

1. “lib\_g13\_idt\_math.lib” from the folder  
“IDT\_ZMOD4450-Firmware\_SWR\_20200306\ZMOD4450-raq-firmware\raq-libraries\ccr\lg13”
2. “lib\_g13\_raq.lib” from the folder  
“IDT\_ZMOD4450-Firmware\_SWR\_20200306\ZMOD4450-raq-firmware\raq-libraries\ccr\lg13”
3. “idt\_math.h” from the folder  
“IDT\_ZMOD4450-Firmware\_SWR\_20200306\ZMOD4450-raq-firmware\raq-libraries\ccr\lg13”

4. "raq.h" from the folder  
"IDT\_ZMOD4450-Firmware\_SWR\_20200306\ZMOD4450-raq-firmware\raq-libraries\ccrl\g13"
5. "zmod44xx.h" from the folder  
"IDT\_ZMOD4450-Firmware\_SWR\_20200306\ZMOD4450-raq-firmware\ZMOD4450-api-example\_1.2.0\src"
6. "zmod44xx.c" from the folder  
"IDT\_ZMOD4450-Firmware\_SWR\_20200306\ZMOD4450-raq-firmware\ZMOD4450-api-example\_1.2.0\src"
7. "zmod44xx\_types.h" from the folder  
"IDT\_ZMOD4450-Firmware\_SWR\_20200306\ZMOD4450-raq-firmware\ZMOD4450-api-example\_1.2.0\src"
8. "zmod44xx\_config.h" from the folder  
"IDT\_ZMOD4450-Firmware\_SWR\_20200306\ZMOD4450-raq-firmware\ZMOD4450-api-example\_1.2.0\src"

Once all the files are copied, the libraries must be included in the project as shown in Figure 42 and Figure 43.

Figure 42 also shows the Linker Options Values to link the required Standard Library.

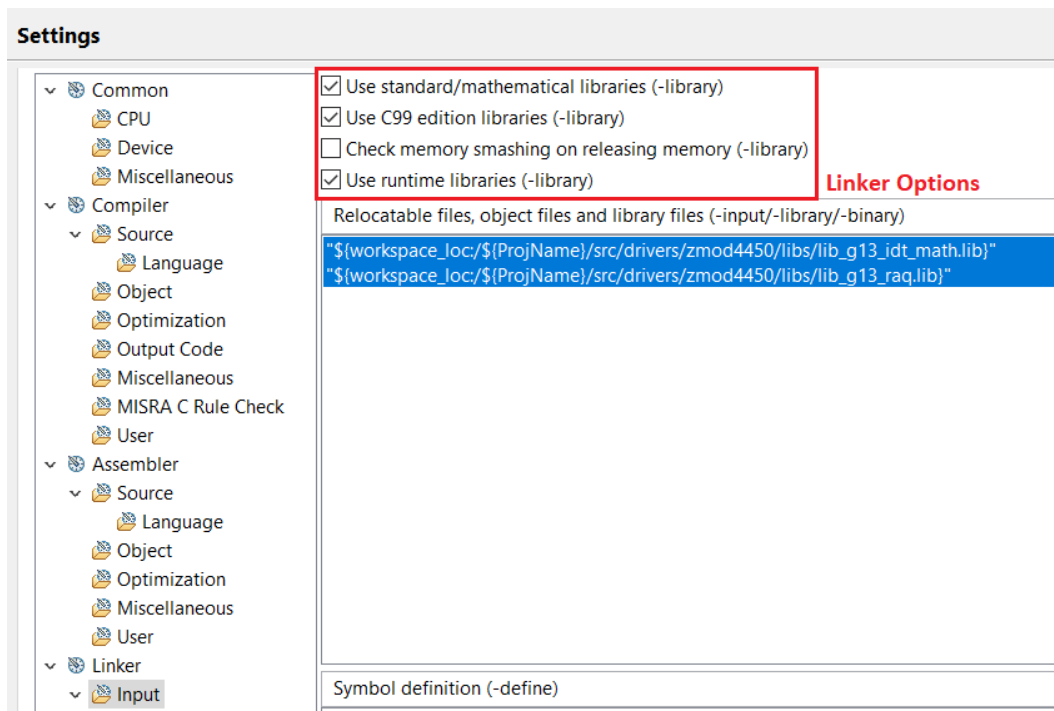
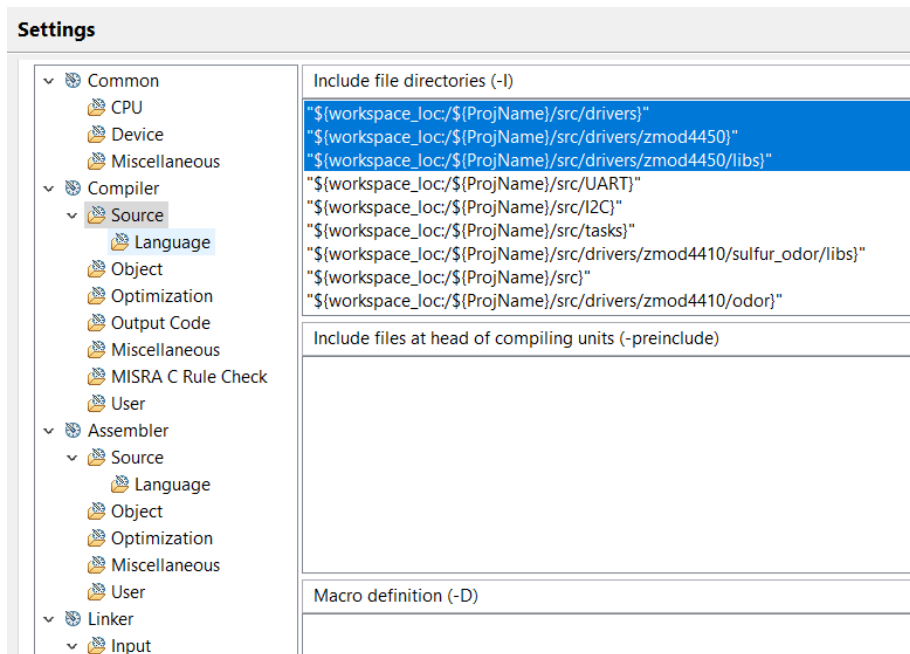


Figure 42: ZMOD4450 compiled libraries added to the project



**Figure 43: ZMOD4450 libraries folders added to the project**

Lastly, the ZMOD4450 driver must be enabled setting the symbol "ZMOD4450\_SENSOR" as "SENSOR\_PRESENT" in the file

```
<project_name>\src\common\proj_conf.h
```

as indicated in Figure 44.

Figure 44 also shows the symbol "ZMOD\_4450\_END\_DET" that defines the method used by the sensor driver to detect when a new Air quality Sensor Output Value is available.

The two method are:

- Polling the Sensor Status Register Value ("ZMOD4450\_END\_DET" defined as "ZMOD4XXX\_INT\_ABSENT"), (which fields indicate when the current measure is complete) on the I2C Internal Bus
- Using the ZMOD INT output pin value [7] ("ZMOD4450\_END\_DET" defined as "ZMOD4XXX\_INT\_PRESENT")

The "ZMOD4450\_END\_DET" is set "ZMOD4XXX\_INT\_PRESENT" as default in order to use the I2C peripheral to retrieve only the Sensor Output Data, optimizing the use of the I2C Internal Bus that is shared between all the sensor present on the EU045 solution kit.



```

28 #include "common_macro.h"
29
30 #define SENSOR_ABSENT ABSENT /*Sensor Absent*/
31 #define SENSOR_PRESENT PRESENT /*Sensor Present*/
32
33 #define ZMOD4XXX_INTERRUPT_PIN_ABSENT (0) /*ZMOD4XXX Interrupt pin not connected
34 #define ZMOD4XXX_INTERRUPT_PIN_PRESENT (1) /*ZMOD4XXX Interrupt pin connected */
35
36
37 /*ZMOD4410 Configuration*/
38 #define ZMOD4410_SENSOR SENSOR_PRESENT /*SENSOR_ABSENT/SE
39 #define ZMOD4410_END_DET ZMOD4XXX_INTERRUPT_PIN_PRESENT /*ZMOD4XXX_INTERRU
40
41 /*ZMOD4510 Configuration*/
42 #define ZMOD4510_SENSOR SENSOR_ABSENT /*SENSOR_ABSENT/SE
43 #define ZMOD4510_END_DET ZMOD4XXX_INTERRUPT_PIN_PRESENT /*ZMOD4XXX_INTERRU
44
45 /*ZMOD4450 Configuration*/
46 #define ZMOD4450_SENSOR SENSOR_ABSENT /*SENSOR_ABSENT/SE
47 #define ZMOD4450_END_DET ZMOD4XXX_INTERRUPT_PIN_PRESENT /*ZMOD4XXX_INTERRU

```

Figure 44: ZMOD4450 enabled through the project\_conf.h file

### 6.1.5.2 Library Modification Procedure

Figure 45 shows the modification to apply to the ZMOD4450 Sensor API file at the path:

```
<project_name>\src\drivers\zmod4450\zmod44xx.c
```

The symbol "CONTINUOUS\_MODE" must be defined at the beginning of this file, to start the communication with the ZMOD4450 sensor.

<pre> * Copyright (c) 2018 Integrated Device Technology, Inc. * @file zmod44xx.c #include "zmod44xx.h" #include "zmod44xx_config.h" #define CONTINUOUS_MODE int8_t zmod44xx_read_sensor_info(zmod44xx_dev_t* dev) {     int8_t ret = 0;     uint8_t data[ZMOD44XX_LEN_PID];     uint8_t status = 0;     uint8_t cmd = 0;     uint16_t i = 0;      /* wait for sensor ready */     do {          ret = dev-&gt;write(dev-&gt;i2c_addr, ZMOD44XX_ADDR_CMD, &amp;cmd, 1);         if(ret) {             return ERROR_I2C;         }     } </pre>	<pre> * Copyright (c) 2018 Integrated Device Technology, Inc. * @file zmod44xx.c #include "zmod44xx.h" #include "zmod44xx_config.h" int8_t zmod44xx_read_sensor_info(zmod44xx_dev_t* dev) {     int8_t ret = 0;     uint8_t data[ZMOD44XX_LEN_PID];     uint8_t status = 0;     uint8_t cmd = 0;     uint16_t i = 0;      /* wait for sensor ready */     do {          ret = dev-&gt;write(dev-&gt;i2c_addr, ZMOD44XX_ADDR_CMD, &amp;cmd, 1);         if(ret) {             return ERROR_I2C;         }     } </pre>
---	---

Figure 45: zmod44xx.c (ZMOD4450 API File) after and before the code modification

## 7. References

- [1] Renesas Electronics, "EU045 - Hardware User's Guide" – R30AN0366ED0100.
- [2] Renesas Electronics, "EU045 – Quick Start Guide" – R30QS0003ED0100.
- [3] Renesas Electronics (IDT), "HS300x Datasheet - High Performance Relative Humidity and Temperature Sensor", Apr. 22, 2020.
- [4] Renesas Electronics (IDT), "ISL29020 Datasheet - A Low Power, High Sensitivity, Light-to Digital Sensor With I2C Interface", Rev 1.00, Aug. 20 2009.
- [5] Renesas Electronics (IDT), "ZMOD4410 Datasheet - Gas Sensor Module for TVOC and Indoor Air Quality", June 3, 2020.
- [6] Renesas Electronics (IDT), "ZMOD4510 Datasheet - Gas Sensor Module for Outdoor Air Quality", Feb. 7, 2020.
- [7] Renesas Electronics (IDT), "ZMOD4450 Datasheet - Gas Sensor Module for Refrigeration Air Quality", Oct. 30, 2019.
- [8] Renesas Electronics, "I2C Repeat Start Signal, Application Note" - R01AN1947.
- [9] IEEE Registration Authority - [Link](#).
- [10] Renesas Electronics, Application Note "BLE microcomputer/module Bluetooth qualification acquisition" - R01AN3177.
- [11] Bluetooth® Assigned Numbers - [Link](#).

**Revision History**

Rev.	Date	Description	
		Page	Summary
0.01	24 Sep 2020		Initial version.
0.02	02 Dec 2020		First Revision after Renesas comments Added MAC Address Section Added RGB Led Coding Section Added Bluetooth® Communication (GATT Server/Client Connection, Overview and Renesas BLE-puck APP)
1.00	18 Jan 2021		Second Revision after Renesas comments.
1.10	04 Nov 2021		Added new "Firmware Upgrade" information.
1.11	12 Nov 2021		Minor revision after Renesas comments.
2.00	22 Dec 2021		Editorial changes

# General Precautions in the Handling of Microprocessing Unit and Microcontroller Unit Products

The following usage notes are applicable to all Microprocessing unit and Microcontroller unit products from Renesas. For detailed usage notes on the products covered by this document, refer to the relevant sections of the document as well as any technical updates that have been issued for the products.

## 1. Precaution against Electrostatic Discharge (ESD)

A strong electrical field, when exposed to a CMOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop the generation of static electricity as much as possible, and quickly dissipate it when it occurs. Environmental control must be adequate. When it is dry, a humidifier should be used. This is recommended to avoid using insulators that can easily build up static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work benches and floors must be grounded. The operator must also be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions must be taken for printed circuit boards with mounted semiconductor devices.

## 2. Processing at power-on

The state of the product is undefined at the time when power is supplied. The states of internal circuits in the LSI are indeterminate and the states of register settings and pins are undefined at the time when power is supplied. In a finished product where the reset signal is applied to the external reset pin, the states of pins are not guaranteed from the time when power is supplied until the reset process is completed. In a similar way, the states of pins in a product that is reset by an on-chip power-on reset function are not guaranteed from the time when power is supplied until the power reaches the level at which resetting is specified.

## 3. Input of signal during power-off state

Do not input signals or an I/O pull-up power supply while the device is powered off. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements. Follow the guideline for input signal during power-off state as described in your product documentation.

## 4. Handling of unused pins

Handle unused pins in accordance with the directions given under handling of unused pins in the manual. The input pins of CMOS products are generally in the high-impedance state. In operation with an unused pin in the open-circuit state, extra electromagnetic noise is induced in the vicinity of the LSI, an associated shoot-through current flows internally, and malfunctions occur due to the false recognition of the pin state as an input signal become possible.

## 5. Clock signals

After applying a reset, only release the reset line after the operating clock signal becomes stable. When switching the clock signal during program execution, wait until the target clock signal is stabilized. When the clock signal is generated with an external resonator or from an external oscillator during a reset, ensure that the reset line is only released after full stabilization of the clock signal. Additionally, when switching to a clock signal produced with an external resonator or by an external oscillator while program execution is in progress, wait until the target clock signal is stable.

## 6. Voltage application waveform at input pin

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between  $V_{IL}$  (Max.) and  $V_{IH}$  (Min.) due to noise, for example, the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between  $V_{IL}$  (Max.) and  $V_{IH}$  (Min.).

## 7. Prohibition of access to reserved addresses

Access to reserved addresses is prohibited. The reserved addresses are provided for possible future expansion of functions. Do not access these addresses as the correct operation of the LSI is not guaranteed.

## 8. Differences between products

Before changing from one product to another, for example to a product with a different part number, confirm that the change will not lead to problems. The characteristics of a microprocessing unit or microcontroller unit products in the same group but having a different part number might differ in terms of internal memory capacity, layout pattern, and other factors, which can affect the ranges of electrical characteristics, such as characteristic values, operating margins, immunity to noise, and amount of radiated noise. When changing to a product with a different part number, implement a system-evaluation test for the given product.

## Notice

1. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation or any other use of the circuits, software, and information in the design of your product or system. Renesas Electronics disclaims any and all liability for any losses and damages incurred by you or third parties arising from the use of these circuits, software, or information.
2. Renesas Electronics hereby expressly disclaims any warranties against and liability for infringement or any other claims involving patents, copyrights, or other intellectual property rights of third parties, by or arising from the use of Renesas Electronics products or technical information described in this document, including but not limited to, the product data, drawings, charts, programs, algorithms, and application examples.
3. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
4. You shall not alter, modify, copy, or reverse engineer any Renesas Electronics product, whether in whole or in part. Renesas Electronics disclaims any and all liability for any losses or damages incurred by you or third parties arising from such alteration, modification, copying or reverse engineering.
5. Renesas Electronics products are classified according to the following two quality grades: "Standard" and "High Quality". The intended applications for each Renesas Electronics product depends on the product's quality grade, as indicated below.
  - "Standard": Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; industrial robots; etc.
  - "High Quality": Transportation equipment (automobiles, trains, ships, etc.); traffic control (traffic lights); large-scale communication equipment; key financial terminal systems; safety control equipment; etc.Unless expressly designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not intended or authorized for use in products or systems that may pose a direct threat to human life or bodily injury (artificial life support devices or systems; surgical implantations; etc.), or may cause serious property damage (space system; undersea repeaters; nuclear power control systems; aircraft control systems; key plant systems; military equipment; etc.). Renesas Electronics disclaims any and all liability for any damages or losses incurred by you or any third parties arising from the use of any Renesas Electronics product that is inconsistent with any Renesas Electronics data sheet, user's manual or other Renesas Electronics document.
6. When using Renesas Electronics products, refer to the latest product information (data sheets, user's manuals, application notes, "General Notes for Handling and Using Semiconductor Devices" in the reliability handbook, etc.), and ensure that usage conditions are within the ranges specified by Renesas Electronics with respect to maximum ratings, operating power supply voltage range, heat dissipation characteristics, installation, etc. Renesas Electronics disclaims any and all liability for any malfunctions, failure or accident arising out of the use of Renesas Electronics products outside of such specified ranges.
7. Although Renesas Electronics endeavors to improve the quality and reliability of Renesas Electronics products, semiconductor products have specific characteristics, such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Unless designated as a high reliability product or a product for harsh environments in a Renesas Electronics data sheet or other Renesas Electronics document, Renesas Electronics products are not subject to radiation resistance design. You are responsible for implementing safety measures to guard against the possibility of bodily injury, injury or damage caused by fire, and/or danger to the public in the event of a failure or malfunction of Renesas Electronics products, such as safety design for hardware and software, including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult and impractical, you are responsible for evaluating the safety of the final products or systems manufactured by you.
8. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. You are responsible for carefully and sufficiently investigating applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive, and using Renesas Electronics products in compliance with all these applicable laws and regulations. Renesas Electronics disclaims any and all liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
9. Renesas Electronics products and technologies shall not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations. You shall comply with any applicable export control laws and regulations promulgated and administered by the governments of any countries asserting jurisdiction over the parties or transactions.
10. It is the responsibility of the buyer or distributor of Renesas Electronics products, or any other party who distributes, disposes of, or otherwise sells or transfers the product to a third party, to notify such third party in advance of the contents and conditions set forth in this document.
11. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products.

(Note1) "Renesas Electronics" as used in this document means Renesas Electronics Corporation and also includes its directly or indirectly controlled subsidiaries.

(Note2) "Renesas Electronics product(s)" means any product developed or manufactured by or for Renesas Electronics.

(Rev.4.0-1 November 2017)

## Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,  
Koto-ku, Tokyo 135-0061, Japan  
[www.renesas.com](http://www.renesas.com)

## Trademarks

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

## 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/](http://www.renesas.com/contact/).