RA Family, RX Family, RL78 Family, RZ Family

ZMOD4xxx Sample Software Manual

Introduction

This sample software acquires gas data from the ZMOD4410, ZMOD4450 and ZMOD4510 gas sensors and calculates the result. In combination with the I2C driver of the FSP, the sample software controls the ZMOD4410 and ZMOD4510 through the I2C in the MCU to measure gases, acquire ADC data, and calculate the acquired result.

Target Devices

RA6M4 Group

RX65N Group

RL78/G14 Group

RL78/G23 Group

RZ/G2L Group

Trademarks

Contents

1. Overview

This sample software acquires data from the ZMOD4xxx gas sensor and calculates the result values. In combination with the I2C driver of the FSP, the sample software controls the ZMOD4xxx through the I2C in the MCU to measure gas environment, acquire ADC data, convert and calculate the acquired results.

2. Environment for Confirming Operation

2.1 Environment for Confirming Operation on RA Family MCU

The operation of this software has been confirmed on an MCU of the RA family in the following environment.

Table 2-1 Operating Environment for RA Family

Item	Description
Demonstration board	RTK7EKA6M4S00001BE (EK-RA6M4)
Microcontroller	RA6M4 (R7FA6M4AF3CFB :144pin)
Operating frequency	200MHz
Operating voltage	5V
Integrated development environment	e^2 Studio 2023-01
C compiler	GCC 10.3.1.20210824 IAR ANSI C/C++ Compiler V9.20.2.320/LNX for ARM ARM Compiler 6.18
FSP	V.4.5.0
RTOS	FreeRTOS™ / Microsoft® Azure RTOS
Emulator	On board (J-LINK)
Interposer	Interposer Board to convert Type2/3 to Type 6A PMOD standard (US082-INTERPEVZ)
Sensor board	TVOC and Indoor Air Quality Sensor Pmod™ Board (US082-ZMOD4410EVZ) Outdoor Air Quality Sensor Pmod™ Board (US082-ZMOD4510EVZ)

Table 2-2 Amount of Memory Used in RA Family

Area	Size (Non-OS)			Size (Free RTOS)		Size (Azure RTOS)	
ZMOD sensor	**4410**	**4510**	**4450**	**4410+4510**	**4450**	**4410+4510**	**4450**
ROM [bytes]	6,856	5,588	3,556	10,472	3,892	10,600	3,840
RAM [bytes]	719	684	390	1,471	580	1,468	749

Memory size is calculated by functions and variables only related to ZMOD4xxx sensor. In RTOS, memory size does not include memory size of the thread.

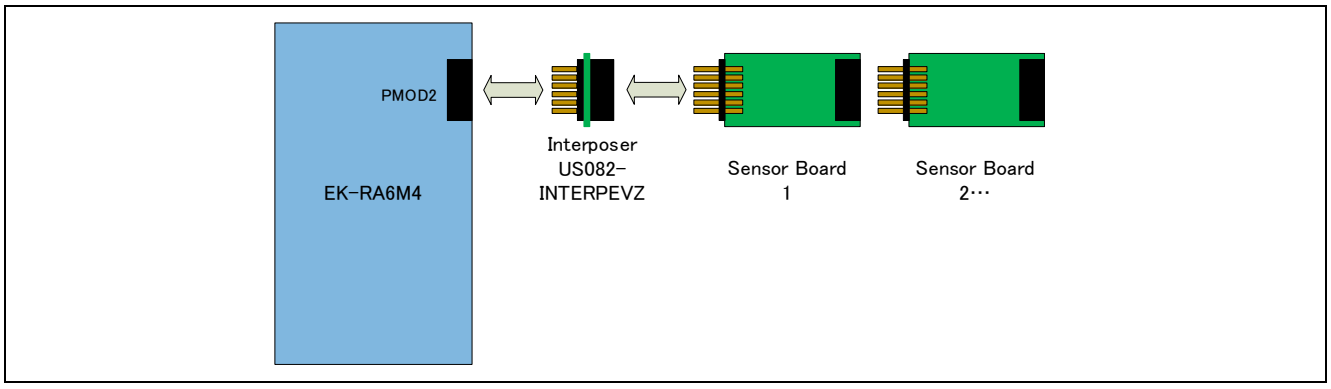


Figure 2-1 Hardware Connections for RA Family

2.2 Environment for Confirming Operation on RX Family MCU

The operation of this software has been confirmed on an MCU of the RX family in the following environment.

Table 2-3 Operating Environment for RX Family

Item	Description
Demonstration board	RPBRX65N (Envision Kit RX65N)
Microcontroller	RX65N (R5F565NEDDFB: 144pin)
Operating frequency	12MHz
Operating voltage	5V
Integrated development environment	e^2 Studio 2023-01 IAR EW for RX 4.20.1
C compiler	Renesas Electronics C/C++ compiler for RX family V.3.03.00 GCC 8.3.0.202004 IAR Toolchain for RX 8.4.10.7051
FIT	BSP V.7.20
RTOS	FreeRTOS™
Emulator	On board (E2OB)
Interposer	Interposer Board to convert Type2/3 to Type 6A PMOD standard (US082-INTERPEVZ)
Sensor board	TVOC and Indoor Air Quality Sensor Pmod™ Board (US082-ZMOD4410EVZ) Outdoor Air Quality Sensor Pmod™ Board (US082-ZMOD4510EVZ)

Table 2-4 Amount of Memory Used in RX Family

Area	Size (Non-OS)			Size (Free RTOS)		Size (Azure RTOS)	
ZMOD sensor	**4410**	**4510**	**4450**	**4410+4510**	**4450**	**4410+4510**	**4450**
ROM [bytes]	6,256	5,398	3,460	9,056	3,621	9,177	3,751
RAM [bytes]	782	651	692	1,376	752	1,513	961

Memory size is calculated by functions and variables only related to ZMOD4XXX sensor. In RTOS, memory size does not include memory size of the thread.

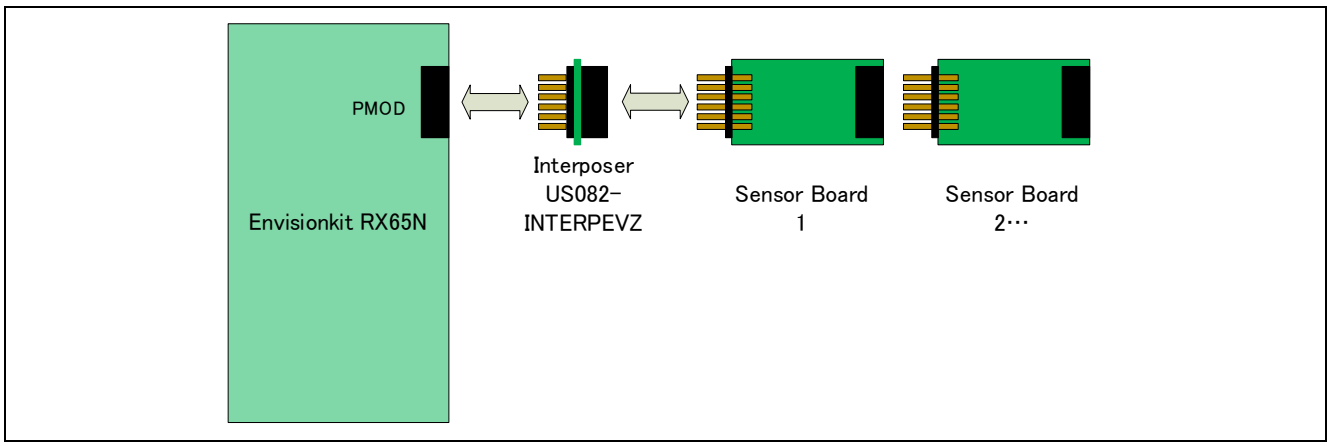


Figure 2-2 Hardware Connections for RX Family

2.3 Environment for Confirming Operation on RL78/G14 Group MCU

The operation of this software has been confirmed on an MCU of the RL78/G14 Group in the following environment.

Table 2-5 Operating Environment for RL78/G14 Group

Item	Description
Demonstration board	RTK5RLG140C00000BJ (RL78/G14 Fast Prototyping Board)
Microcontroller	RL78/G14 (R5F104MLAFB: 80pin)
Operating frequency	20MHz
Operating voltage	3.3V
Integrated development environment	e^2 studio 2023-01 IAR EW for RL78 4.21.1
C compiler	C compiler package for RL78 family V1.11.00 GCC for Renesas RL78 4.9.2.202103 IAR Toolchain for RL78 4.21.1.2409
Emulator	On board (E2OB)
Sensor board	TVOC and Indoor Air Quality Sensor Pmod™ Board (US082-ZMOD4410EVZ) Outdoor Air Quality Sensor Pmod™ Board (US082-ZMOD4510EVZ)

Table 2-6 Amount of Memory Used in RL78/G14 Group

Area	Size	Remarks
ROM	9,499 bytes	Include ZMOD4410 IAQ 2nd Gen library
RAM	551 bytes	
ROM	7,263 bytes	Include ZMOD OAQ 2nd Gen library
RAM	444 bytes	
ROM	5,235 bytes	Include ZMOD RAQ library
RAM	497 bytes	

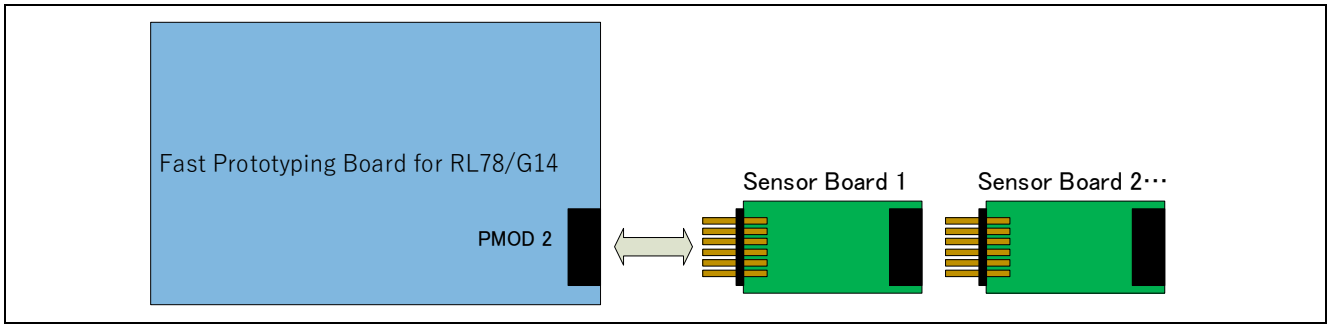


Figure 2-3 Hardware Connections for RL78/G14 Group

2.4 Environment for Confirming Operation on RL78/G23 Group MCU

The operation of this software has been confirmed on an MCU of the RL78/G23 Group in the following environment.

Table 2-7 Operating Environment for RL78/G23 Group

Item	Description
Demonstration board	RTK7RLG230CSN000BJ (RL78/G23-128p Fast Prototyping Board)
Microcontroller	RL78/G23 (R7F100GSN2DFB :128pin)
Operating frequency	32MHz
Operating voltage	3.3V
Integrated development environment	e^2 studio 2023-01 IAR EW for RL78 4.21.1
C compiler	C compiler package for RL78 family V1.11.00 LLVM for RL78 10.0.0.202209 IAR Toolchain for RL78 4.21.1.2409
Emulator	E2 Lite
Sensor board	TVOC and Indoor Air Quality Sensor Pmod™ Board (US082-ZMOD4410EVZ) Outdoor Air Quality Sensor Pmod™ Board (US082-ZMOD4510EVZ)

Table 2-8 Amount of Memory Used in RL78/G23 Group

Area	Size	Remarks
ROM	9,596 bytes	Include ZMOD4410 IAQ 2nd Gen library
RAM	479 bytes	
ROM	7,288 bytes	Include ZMOD OAQ 2nd Gen library
RAM	444 bytes	
ROM	5,500 bytes	Include ZMOD RAQ library
RAM	497 bytes	

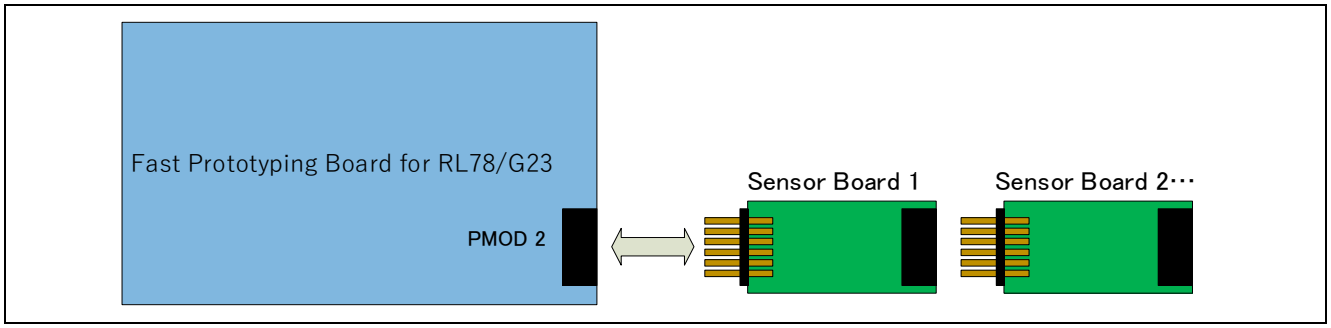


Figure 2-4 Hardware Connections for RL78/G23 Group

2.5 Environment for Confirming Operation on RZ Family MCU

The operation of this software has been confirmed on an MCU of the RZ family in the following environment.

Table 2-9 Operating Environment for RZ Family

Item	Description
Demonstration board	RTK9744L23S01000BE (RZ/G2L Evaluation Kit (SMARC))
Microcontroller	RZ/G2L (R9A07G044L23GBG:456pin)
Operating frequency	Arm® Cortex®-M33:200MHz、Arm® Cortex®-A55:1.2GHz
Operating voltage	5V
Integrated development environment	e^2 Studio 2023-01
C compiler	GCC 10.3.1.20210824
FSP	V.1.2.0
RTOS	FreeRTOS™
Emulator	SEGGER J-LINK BASE
Sensor board	TVOC and Indoor Air Quality Sensor Pmod™ Board (US082-ZMOD4410EVZ) Outdoor Air Quality Sensor Pmod™ Board (US082-ZMOD4510EVZ)

Table 2-10 Amount of Memory Used in RZ Family

Area	Size (Non-OS)		Size (FreeRTOS)
ZMOD sensor	**4410**	**4510**	**4410+4510**
ROM[bytes]	7,271	5,693	10,494
RAM[bytes]	803	636	1,713

Memory size is calculated by functions and variables only related to ZMOD4XXX sensor. In RTOS, memory size does not include memory size of the thread.

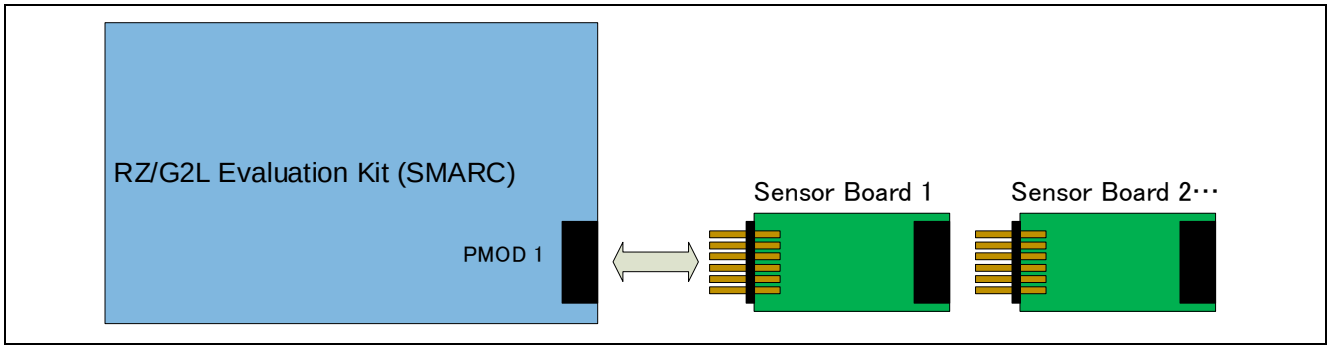


Figure 2-5 Hardware Connections for RZ Family

3. ZMOD4410 Sensor specifications

3.1 Sensor Specifications Overview

The ZMOD4410 Gas Sensor Module is designed to detect typical TVOC contaminations based on studies and international standards for indoor air quality. Characteristic module parameters are shown in Table **3-1**. The response time for a gas stimulation is always within a few seconds, depending on the TVOC and its concentration. An active or direct airflow onto the sensor module is not necessary because diffusion of ambient gas does not limit the sensor module's response time.

Important: The ZMOD4410 also can detect safety-relevant gases for indoor air, such as carbon monoxide (CO); however, the sensor module is not designed to detect these interferants reliably and therefore it is not approved for use in any safety-critical or life-protecting applications. It must not be used in such applications, and Renesas disclaims all liability for any such use.

Table 3-1- Gas Sensor Module Specifications during Operation

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit [a]
	Resistance Measurement Range	Ethanol in air	0	-	1000	ppm
			0	-	1000000	ppb [c]
	IAQ and TVOCspecified Measurement Range to meet UBA[b]	Ethanol in air for IAQ 1st Gen	160	-	30000	ppb
		Ethanol in air for IAQ 2st Gen	160	-	10000	ppb
	TVOC specified Measurement Range for PBAQ	Ethanol in air	1	-	2000	µg/m^3
	Humidity Range for Sensor Module Operation	All ZMOD operations except PBAQ operation, Non-condensing	0	-	90	% RH
		Specification to meet PBAQ, Non-condensing	-	50	-	% RH
S	Sensitivity over Lifetime	Resistance in Clean Air / Resistance at 10ppm Ethanol (R_{Air}/R_{Gas})		5		Ω/Ω
T-80	Sensor Module Response Time [d]	Samples needed to change to 80% of end value (all operation modes except ULP)	-	15	-	Samples
V_{DD}	Supply Voltage		1.7	-	3.6	V
T_{AMB}	Ambient Temperature Range for Sensor Module Operation	All operation modes except PBAQ, Non-condensing	-40	-	65	°C
		Specification according to PBAQ, Non-condensing	-	21	-	°C
	Average Power: IAQ 1st Gen	Continuous and Odor Operation Mode	-	23	-	mW
		Low Power Operation Mode	-	1.5	-	mW
	Average Power: IAQ 2s Gen, Relative IAQ and Sulfur-based Odor Discrimination		-	6	-	mW

	Average Power: IAQ 2s Gen ULP and Relative IAQ ULP	Ultra-Low Power operation	-	0.16	-	mW
	Average Power: PBAQ		-	1	-	mW
	Average Power: Odor Operation		-	23	-	mW
I_{ACTIVE}	Supply Current, Active Mode including Heater Current for IAQ 1st Gen Continuous and Odor Operation Mode	At VDD = 1.8 V	-	13	-	mA
		At VDD = 3.3V	-	7	-	mA
I_{ACTIVE}	Supply Current, Active Mode including Heater Current for IAQ 2s Gen, Relative IAQ, Ultra-low Power and Sulfur-based Odor Discrimination	At VDD = 1.8 V	-	7.4	16.2	mA
		At VDD = 3.3V	-	5.2	10.3	mA
I_{ACTIVE}	Supply Current, Active Mode including Heater Current for PBAQ	At VDD = 1.8 V	-	9.9	16.4	mA
		At VDD = 3.3V	-	6.9	10.6	mA
I_{SLEEP}	Current during measurement delays	Sleep Mode ASIC	-	450	-	nA

[a] The abbreviation ppm stands for “parts per million,” and ppb is an abbreviation for “parts per billion.” For example, 1 ppm equals 1000 ppb.

[b] Source: Umweltbundesamt, *Beurteilung von Innenraumluftkontaminationen mittels Referenz- und Richtwerten*, (Bundesgesundheitsblatt - Gesundheitsforschung - Gesundheitsschutz, 2007).

[c] Conversion from ppm to mg/m^3 for most common TVOC is by the factor approximately 2; for example, 5ppm equals approximately 10mg/m^3.

[d] Response times depend on TVOC gas and concentration.

3.2 Sensor function and methods

The ZMOD architecture leverages different “Methods of Operation” which use time, temperature, and signatures from gases that enable unique signals from a highly trained machine learning system and makes use of embedded artificial intelligence (AI) technology. This section discusses the different operation modes of the ZMOD4410. At present, five operation modes are released.

Family of IAQ software releases:

- Operation Mode 1: IAQ 1st Generation: Continuous – Measurement of UBA levels for IAQ and eCO2
- Operation Mode 2: IAQ 1st Generation: Low Power – Measurement of UBA levels for IAQ and eCO2
- Operation Mode 3: IAQ 2nd Generation: Using AI for improved ppm TVOC, IAQ and eCO2 functionality (recommended for new designs)
- Operation Mode 4: Odor – Control signal based on Air Quality Changes
- Operation Mode 5: Sulfur-based Odor Discrimination
- Operation Mode 6: Relative IAQ – Measure Relative IAQ based on Air Quality Changes
- Operation Mode 7: PBAQ – TVOC measurement to meet PBAQ standards

By default, the IAQ 2nd Generation (Mode 3) operation should be used for new designs. In case of the need for a slightly faster sample rate and a larger VOC range (up to 30ppm), it is recommended to use the IAQ 1Gen algorithms.

Additional technical information on sensitivity, selectivity, and stability for all operation modes is available in Renesas' *ZMOD4410 Application Note – TVOC Sensing.* For more information, including application notes, white papers, blog, and manuals, visit the ZMOD4410 webpage.

Table 3-2 Typical ZMOD4410 Sensor Module Accuracy Achievable with Calibration

Parameter	Conditions	Minimum	Typical	Maximum	Unit
Accuracy for TVOC	Without additional calibration		±25		%
	With additional calibration		±15		%
Accuracy for IAQ	Without additional calibration		±10		%
Consistency	Part-to-Part Variation		±25		%
Durability to Siloxanes	Change in sensitivity		±5		%

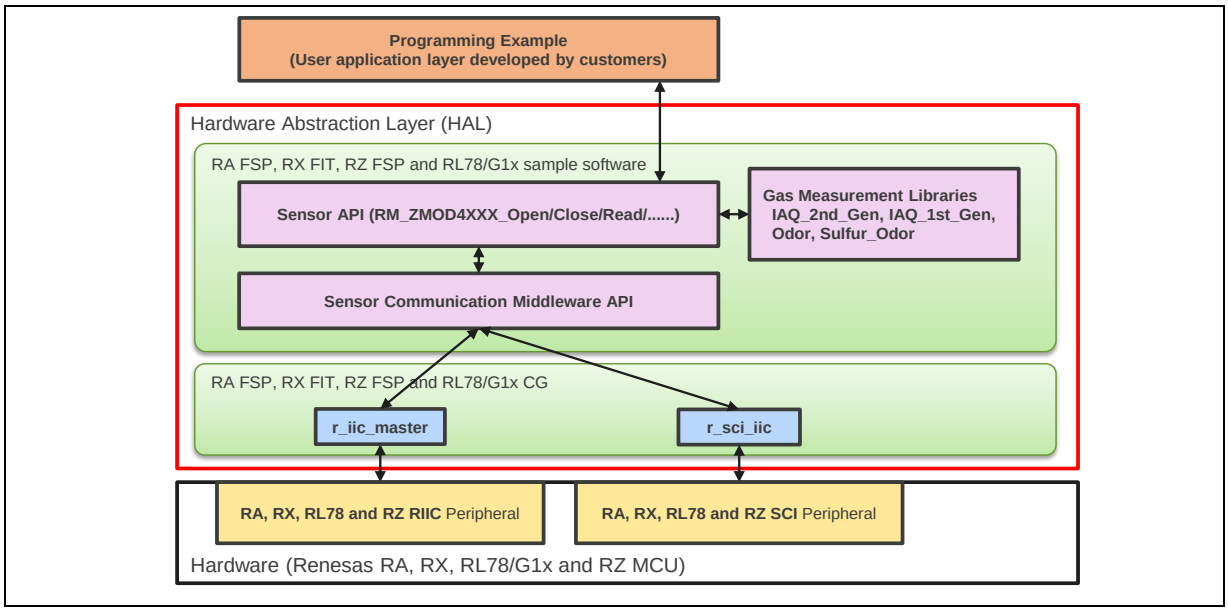
3.2.1 Conversion of output data – Firmware / API / algorithms

To operate the ZMOD4410, a firmware provided by Renesas containing an API, algorithm libraries, and an example should be used. For implementing the sensor module in a customer-specific application, detailed information on the programming is available. For downloading these documents, please visit the ZMOD4410 webpage

The structure of theZMOD4410 firmware is described and illustrated in Figure 3-1 below.

- The “Hardware Abstraction Layer (HAL)” contains hardware-specific drivers include ZMOD4410 sensor API functions with gas measurement libraries, sensor communication middleware functions and low-level I2C drivers.
- The “ZMOD4410 sensor API (Application Programming Interface)” block contains the functions needed to operate the ZMOD4410.
  The “Gas Measurement Libraries” work with sensor API, it contains the functions and data structures needed to calculate the firmware specific results for the Indoor Air Quality related parameters, such as IAQ, TVOC, eCO2 (IAQ 1st Gen and IAQ 2nd Gen) or Air Quality Change (Odor), or Sulfur Odor result. These algorithms cannot be used in parallel. This block also contains the optional cleaning procedure and the optional math library for code size reduction. The libraries are described in more detail in the documents *ZMOD4410-IAQ_xxx_Gen-lib.pdf*, *ZMOD4410-Odor-lib.pd*f, and *ZMOD4410-Sulfur_Odor-lib.pdf, ZMOD4410-PBAQ-lib.pdf, ZMOD4410-Rel_IAQ-lib.pdf, ZMOD4410-Rel_IAQ_ULP-lib.pdf*. All of these files are part of the downloadable sample software packages.
- The “Programming Example” block provides a code example that is used to initialize the ZMOD4410, perform measurements, display the data output for each specific example, and start the optional cleaning function.
- The low-level driver “r_iic_master and r_sci_iic” block is the hardware-specific implementation of the I2C interface for Renesas RA MCU. This block contains read and write functions to communicate with the ZMOD4410 sensor via I2C buses.

Figure 3-1 File Overview of ZMOD4410 Firmware



3.2.2 Typical hardware requirements

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

- 12 to 20 kB program flash for ZMOD4410-related firmware code (MCU architecture and compiler dependent), note also Table 3-3
- 1kB RAM for ZMOD4410-related operations, note also Table 3-3
- Capability to perform I2C communication, timing functions, and floating-point instructions
- The algorithm functions work with variables saved in background and need memory retention between each call

Table 3-3 Exemplary Memory Footprint of ZMOD4410 Implementation on a RL78-G13 MCU

	IAQ 2nd Gen	IAQ 2nd Gen ULP	IAQ 1st Gen	PBAQ	Relative IAQ	Relative IAQ ULP	Odor	Sulfur Odor
Program flash usage in kB	15.8	14.6	10.9	12.7	10.7	10.7	8.7	8.4
RAM usage (required variables) in bytes	496	480	284	436	400	400	168	402
RAM usage (stack size for library functions) in bytes	496	336	-	448	224	224	-	368

4. ZMOD4450 Sensor specifications

4.1 Sensor Specifications Overview

The ZMOD4450 Gas Sensor Module is designed to detect typical gases inside refrigeration applications associated with food ripening or rotting. Characteristic module parameters are shown in Table 4-1. The response time for a gas stimulation is always within a few seconds, depending on the gas and its concentration. An active or direct airflow onto the sensor module is not necessary because diffusion of ambient gas does not limit the sensor module's response time.

Important: The ZMOD4450 also can detect safety-relevant gases for indoor air, such as carbon monoxide (CO); however, the sensor module is not designed to detect these interferants reliably and therefore it is not approved for use in any safety-critical or life-protecting applications. It must not be used in such applications, and Renesas disclaims all liability for any such use.

Table 4-1- Gas Sensor Module Specifications during Operation

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit [a]
	Resistance Measurement Range	Ethylene (C_2H_4) in air	0		10	ppm
		Trimethylamine (C_3H_9N) in air	0		600	ppb
		Dimethyl sulfide (C_2H_6S) in air	0		180	ppb
RAQ	Refrigeration Air Quality Range	Change rate based on resistance	0		3	
	Temperature Range		0		25	°C
	Repeatability	Variation in sensor module signal		±10		%
T-90	Sensor Module Response Time	Time to change to 90% of end value		10		s
V_{DD}	Supply Voltage		1.7	-	3.6	V
T_{AMB}	Ambient Temperature Range for Sensor Module Operation		-40	25	65	°C
	Average Power ZMOD4450	Continuous Operation	-	23	-	mW
I_{ACTIVE}	Supply Current, Active Mode including Heater Current.	At VDD = 1.8 V	-	13	-	mA
		At VDD = 3.3V	-	7	-	mA
I_{SLEEP}	Current during measurement delays	Sleep Mode ASIC	-	450	-	nA

[a] The abbreviation ppm stands for "parts per million," and ppb is an abbreviation for "parts per billion." For example, 1 ppm equals 1000 ppb.

4.2 Sensor function and methods

The ZMOD architecture leverages different “Methods of Operation” which use time, temperature, and signatures from gases that enable unique signals from a highly trained machine learning system and makes use of embedded artificial intelligence (AI) technology. This section discusses the different operation modes of the ZMOD4450. At present, five operation modes are released.

Family of IAQ software releases:

- Operation Mode 1: RAQ – Control signal based on Refrigeration Air Quality Changes

In addition, details for sensitivity, reliability, sample rates, and sensor module influences are explained in detail in the following sections. All graphs and information show the typical responses that are to be expected from the sensor module upon exposure to a variety of test conditions. For more information, including application notes, white papers, blog, and manuals, visit the ZMOD4450 product page.

Table 4-2 Typical ZMOD4450 Sensor Module Accuracy Achievable with Calibration

Parameter	Conditions	Minimum	Typical	Maximum	Unit
Accuracy for RAQ	Without additional calibration		±10		%
Consistency	Part-to-Part Variation		±25		%
Durability to Siloxanes	Change in sensitivity		±5		%

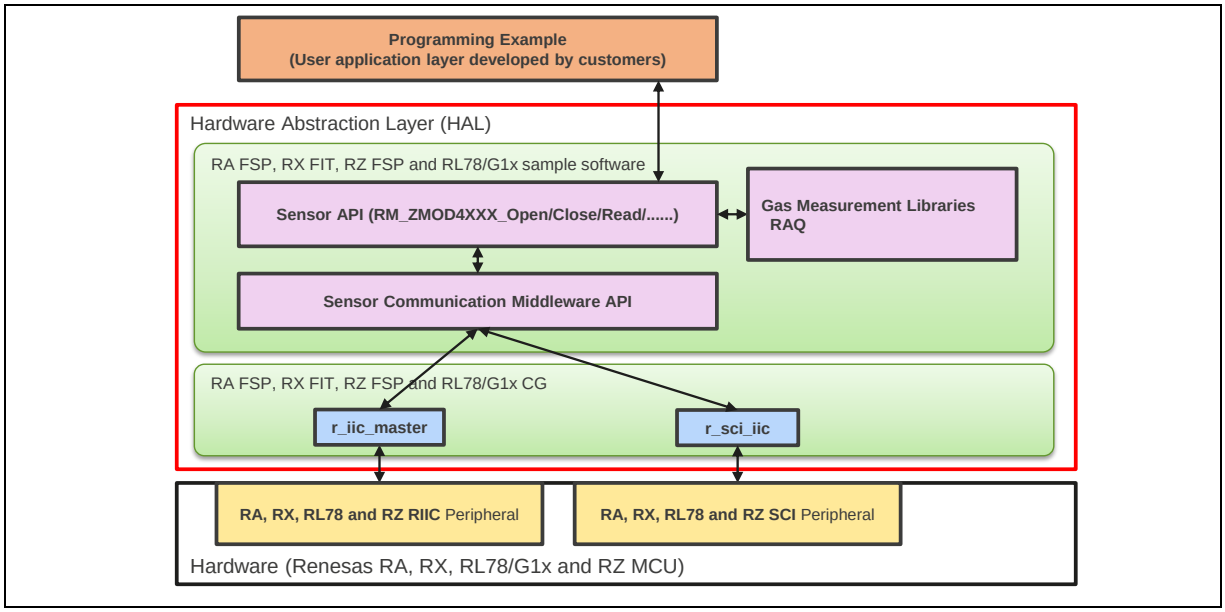
4.2.1 Conversion of output data – Firmware / API / algorithms

To operate the ZMOD4450, a firmware provided by Renesas containing an API, algorithm libraries, and an example should be used. For implementing the sensor module in a customer-specific application, detailed information on the programming is available. For downloading these documents, please visit the ZMOD4450 webpage

The structure of theZMOD4450 firmware is described and illustrated in Figure 4-1 below.

- The “Hardware Abstraction Layer (HAL)” contains hardware-specific drivers include ZMOD4450 sensor API functions with gas measurement libraries, sensor communication middleware functions and low-level I2C drivers.
- The “ZMOD4450 sensor API (Application Programming Interface)” block contains the functions needed to operate the ZMOD4450.
  The “Gas Measurement Libraries” work with sensor API, it contains the functions and data structures needed to calculate the firmware specific results for the Refrigeration Air Quality related parameters. These algorithms cannot be used in parallel. This block also contains the optional cleaning procedure and the optional math library for code size reduction. The library is described in more detail in the document ZMOD4450-raq-lib.pdf. All of these files are part of the downloadable sample software packages.
- The “Programming Example” block provides a code example that is used to initialize the ZMOD4450, perform measurements, display the data output for each specific example, and start the optional cleaning function.
- The low-level driver “r_iic_master and r_sci_iic” block is the hardware-specific implementation of the I2C interface for Renesas RA MCU. This block contains read and write functions to communicate with the ZMOD4450 sensor via I2C buses.

Figure 4-1 File Overview of ZMOD4450 Firmware



4.2.2 Typical hardware requirements

To operate the ZMOD4450, customer-specific hardware with a microcontroller unit (MCU) is needed. Depending on the sensor configuration and on the hardware itself, the requirements differ, and the following minimum requirements are provided as an orientation only:

- 10 to 20 kB program flash for ZMOD4450-related firmware code (MCU architecture and compiler dependent), note also Table **4-3**
- 1kB RAM for ZMOD4450-related operations, note also Table **4-3**
- Capability to perform I2C communication, timing functions, and floating-point instructions
- The algorithm functions work with variables saved in background and need memory retention between each call

Table 4-3 Exemplary Memory Footprint of ZMOD4450 Implementation on a RA6M4 MCU

	RAQ
Program flash usage in kB	1.8
RAM usage in Bytes	174

5. ZMOD4510 Sensor specifications

5.1 Sensor Specifications Overview

The ZMOD4510 Gas Sensor Module detects typical gases based on studies and international standards for outdoor air quality. Characteristic module parameters are shown in Table **5-1**. The ZMOD4510 uses a sequence of applied temperatures in order to sample the air and report an Air Quality Index (AQI) based on the EPA standard[1]. The sensor does not require an active or direct airflow onto the sensor module because diffusion of ambient gas does not limit the sensor response time.

*Important***:** The ZMOD4510 can also detect safety-relevant gases; however, the sensor is not designed to detect these interferants reliably and therefore it is not approved for use in any safety-critical or life-protecting applications. It must not be used in such applications, and Renesas disclaims all liability for any such use.

Table 5-1- Gas Sensor Module Specifications during Operation

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit [a]
AQI	Air Quality Index	OAQ 1st Gen: Rating according to EPA for ozone and nitrogen dioxide	0		500	
		OAQ 2nd Gen: Rating according to EPA for ozone	0		500	
	Measurement Range OAQ 1st Gen	Ozone (non-selective)	20		500	ppb
		Nitrogen dioxide (non-selective)	20		500	ppb
	Measurement Range OAQ 2nd Gen	Ozone (selective)	20		500	ppb
		NO_2 cross sensitivity at 200ppb		25		AQI Level
RH	Humidity Range	Non-condensing	5		90	% RH
T	Temperature Range	Typical outdoor environment	-20		50	°C
		Extended range	-40		65	°C
V_{DD}	Supply Voltage for ZMOD4510 Sensor Module		1.7	–	3.6	V
T_{AMB}	Ambient Temperature Range for Sensor Operation		-40	–	65	°C
$T_{OPERATION}$	Operation Temperature Sequence of Sense Element [a]		200		450	°C
	Average Power: OAQ 1st Gen	Outdoor Air Quality	–	21	–	mW
	Average Power: OAQ 2nd Gen	Selective ozone with ultra-low power	-	0.2	-	mW

[1] AirNow, US Environmental Protection Agency, Air Quality Index (AQI) Basics; available at: https://airnow.gov/index.cfm?action=aqibasics.aqi

I_{ACTIVE}	Supply Current, Active Mode including Heater Current for OAQ 1st Gen	At V_{DD} = 1.8 V	-	11	13	mA
		At V_{DD} = 3.3 V	-	8	10	mA
I_{ACTIVE}	Supply Current, Active Mode including Heater Current for OAQ 2nd Gen	At V_{DD} = 1.8 V	-	10	12	mA
		At V_{DD} = 3.3 V	-	6	8	mA
I_{SLEEP}	Current during measurement delays	Sleep Mode ASIC	–	450	–	nA

[b] The abbreviation ppm stands for “parts per million,” and ppb is an abbreviation for “parts per billion.” For example, 1 ppm equals 1000 ppb.

5.2 Sensor function and methods

The ZMOD architecture leverages different “Methods of Operation” which use time, temperature, and signatures from gases that enable unique signals from a highly trained machine learning system and makes use of embedded artificial intelligence (AI) technology. This section discusses the different operation modes of the ZMOD4510. Currently, two operation modes are released using the same ZMOD4510 hardware:

Family of Outdoor Air Quality (OAQ) software releases:

- Operation Mode 1 – OAQ 1st Generation: Measurement of Air Quality
- Operation Mode 2 – OAQ 2nd Generation: Selective Ozone featuring Ultra-Low Power

In addition, details for sensitivity, reliability, sample rates, and sensor module influences are explained in detail in the following sections. All graphs and information show the typical responses that are to be expected from the sensor module upon exposure to a variety of test conditions. For more information, including application notes, white papers, blog, and manuals, visit the ZMOD4510 product page.

Table 5-2 Typical ZMOD4510 Sensor Module Accuracy Achievable with Calibration

Parameter	Conditions	Minimum	Typical	Maximum	Unit
Accuracy	Without additional calibration		±50		AQI
Consistency	Part-to-Part Variation		±50		AQI
Durability to Siloxanes	Change in sensitivity during 15,000 ppm·h exposure with D4 and D5		±8		%

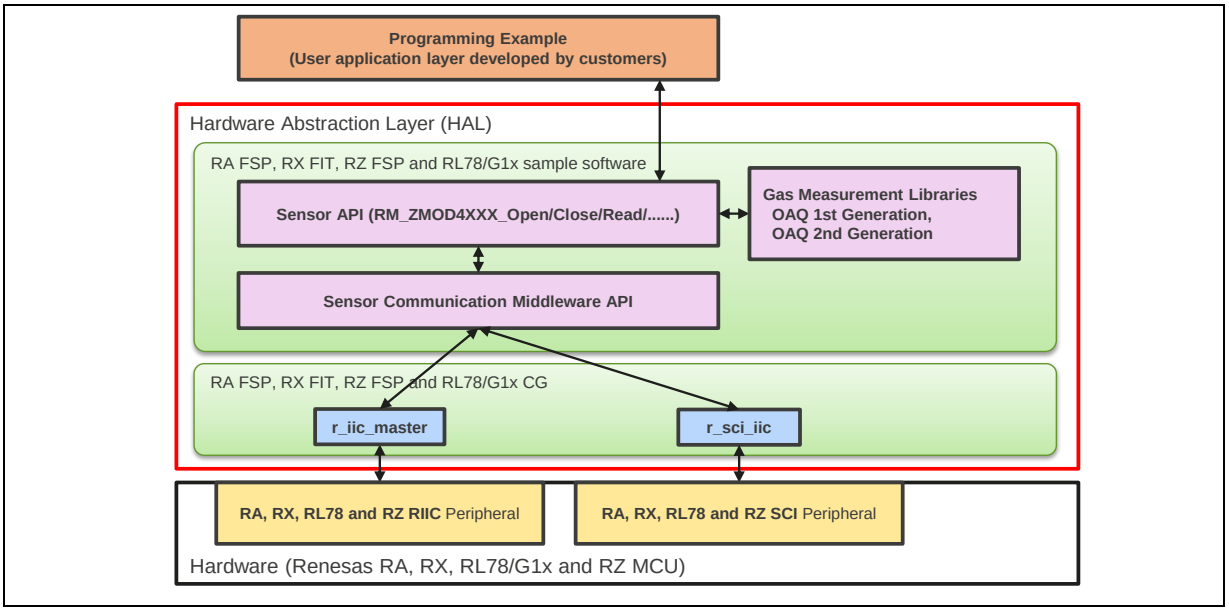
5.2.1 Conversion of output data – Firmware / API / algorithms

To operate the ZMOD4510, a firmware provided by Renesas containing an API, algorithm libraries, and an example should be used. For implementing the sensor module in a customer-specific application, detailed information on the programming is available. More information and guidance on the firmware integration, architecture, and supported platforms are available in the ZMOD4510 Programming Manual – Read Me. Code Examples in C and additional firmware descriptions for API, HAL, libraries, etc., are included at no cost in the downloadable firmware package from the ZMOD4510 product page.

The structure of theZMOD4510 firmware is described and illustrated in Figure 5-1 below.

- The “Hardware Abstraction Layer (HAL)” contains hardware-specific drivers include ZMOD4510 sensor API functions with gas measurement libraries, sensor communication middleware functions and low-level I2C drivers.
- The “ZMOD4510 sensor API (Application Programming Interface)” block contains the functions needed to operate the ZMOD4510.
  The “Gas Measurement Libraries” work with sensor API, it contains the functions and data structures needed to calculate the firmware specific results for the Air Quality Index (AQI). These algorithms cannot be used in parallel. This block also contains the optional cleaning library. The libraries are described in more detail in the documents ZMOD4510 OAQ_1st_Gen-lib.pdf and ZMOD4510 OAQ_2nd_Gen-lib.pdf.
- The “Programming Example” block provides a code example that is used to initialize the ZMOD4510, perform measurements, display the data output for each specific example, and start the optional cleaning function.
- The low-level driver “r_iic_master and r_sci_iic” block is the hardware-specific implementation of the I2C interface for Renesas RA MCU. This block contains read and write functions to communicate with the ZMOD4510 sensor via I2C buses.

Figure 5-1 File Overview of ZMOD4410 Firmware



5.2.2 Typical hardware requirements

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

- 12 to 20 kB program flash for ZMOD4510-related firmware code (MCU architecture and compiler dependent), note also Table **5-3**
- 1kB RAM for ZMOD4510-related operations, note also Table **5-3**
- Capability to perform I2C communication, timing functions, and floating-point instructions
- The algorithm functions work with variables saved in background and need memory retention between each call

Table 5-3 Exemplary Memory Footprint of ZMOD4510 Implementation on a RL78-G13 MCU

	OAQ 2nd Gen	**OAQ 1st Gen**
Program flash usage in kB	12.9	10.2
RAM usage (required variables) in bytes	250	266
RAM usage (stack size for library functions, worst case) in bytes	544	244

6. Specification of Sample Software

The sample software package contains 3 projects. Each project is described below.

6.1 Sample Software Structure

Figure 6-1 **Block diagram of Sample** Software shows structure of sample software blocks.

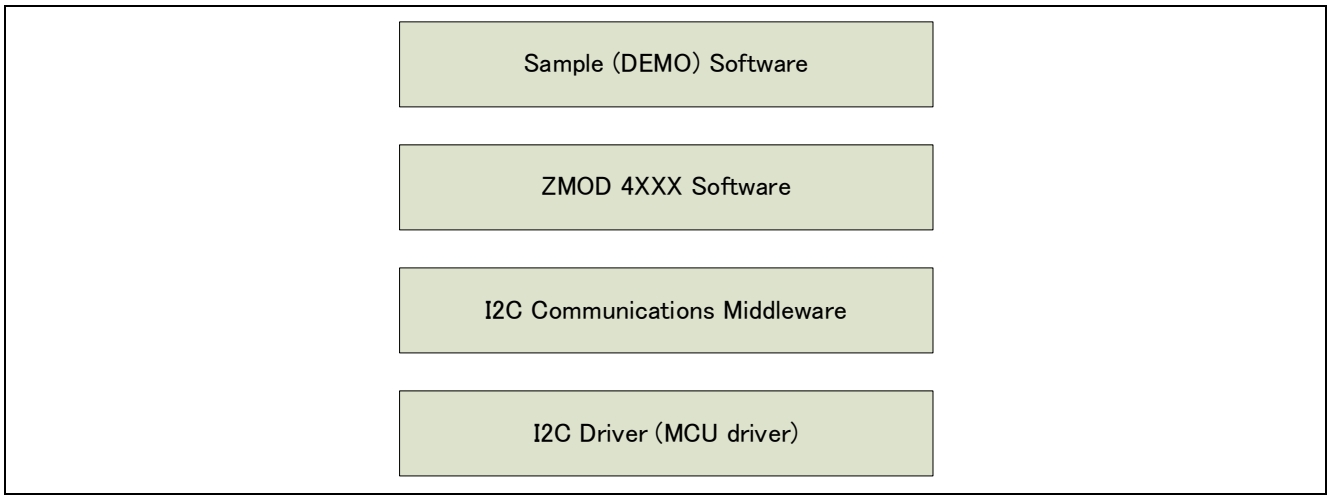


Figure 6-1 Block diagram of Sample Software

6.2 Specification of Sensor API Functions

6.2.1 List of Sensor API Functions

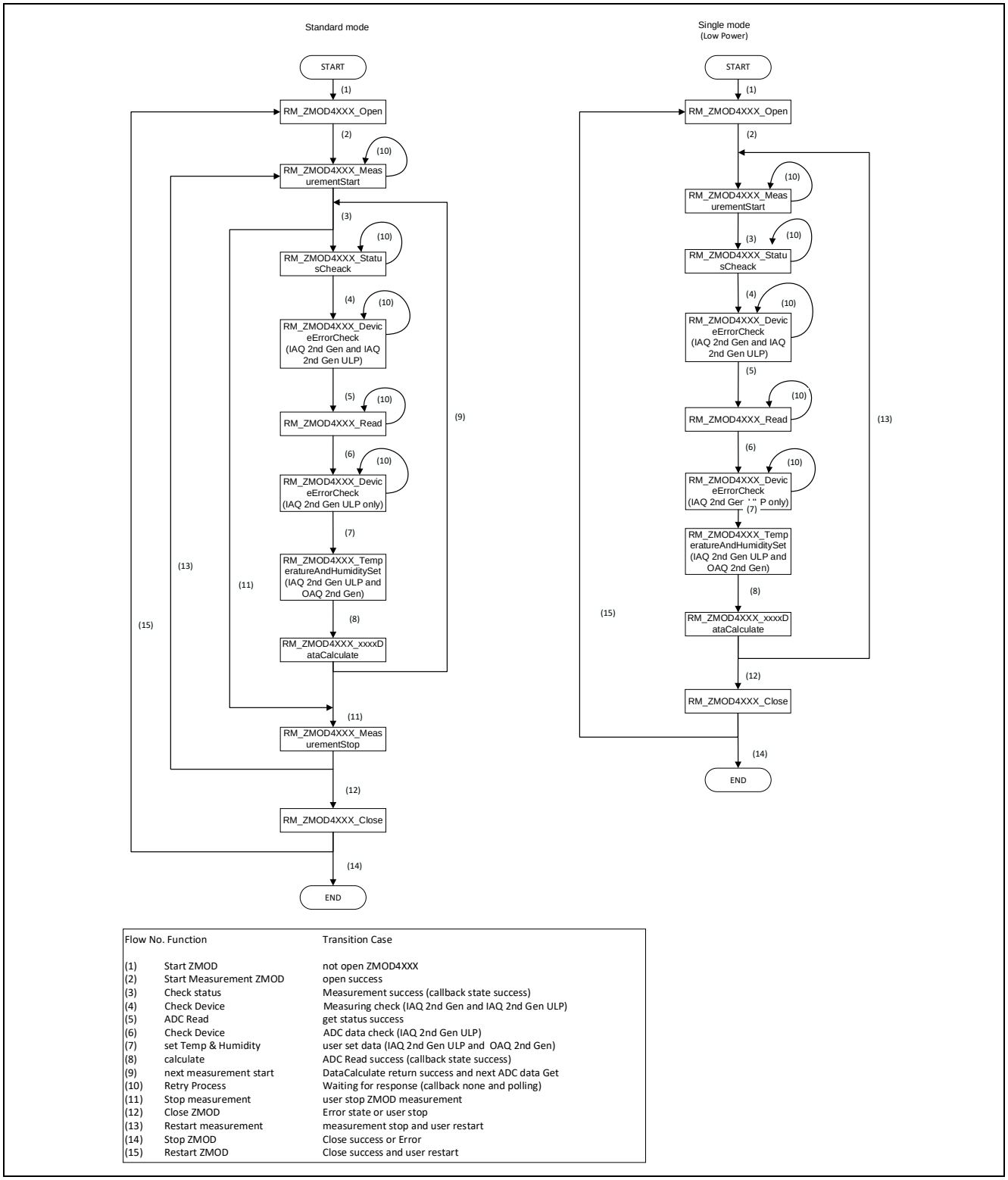
The sensor API includes following functions. Please refer to (RA Flexible Software Package Documentation) for the detail of API functions.

Table 6-1 List of Sensor API Functions

Function	Feature
RM_ZMOD4XXX_Open	Opens sensor
RM_ZMOD4XXX_Close	Closes the sensor
RM_ZMOD4XXX_MeasurementStart	Starts a measurement
RM_ZMOD4XXX_MeasurementStop	Stops a measurement
RM_ZMOD4XXX_StatusCheck	Read status of the sensor
RM_ZMOD4XXX_Read	Reads ADC data
RM_ZMOD4XXX_TemperatureAndHumiditySet	Sets temperature and humidity. (ZMOD4410 IAQ_2nd_Gen_ULP operation and ZMOD4510 OAQ_2nd_Gen operation)
RM_ZMOD4XXX_DeviceErrorCheck	ADC Data Validity Check Process
RM_ZMOD4XXX_Iaq1stGenDataCalculate	Calculate IAQ 1st Gen. values from ADC data
RM_ZMOD4XXX_Iaq2ndGenDataCalculate	Calculate IAQ 2nd Gen. values from ADC data
RM_ZMOD4XXX_OdorDataCalculate	Calculate Odor values from ADC data
RM_ZMOD4XXX_SulfurOdorDataCalculate	Calculate Sulfur Odor values from ADC data
RM_ZMOD4XXX_Oaq1stGenDataCalculate	Calculate OAQ 1st Gen. values from ADC data
RM_ZMOD4XXX_Oaq2ndGenDataCalculate	Calculate OAQ 2nd Gen. values from ADC data
RM_ZMOD4XXX_RaqDataCalculate	Calculate RAQ values from ADC data
RM_ZMOD4XXX_RelIaqDataCalculate	Calculate Rel IAQ. values from ADC data
RM_ZMOD4XXX_PbaqDataCalculate	Calculate PBAQ values from ADC data

6.2.2 API Usage Guide

Figure 6-2 Transition of API Functions shows the transition diagram of functions calling order as the usage condition of ZMOD4XXX API functions.



Flow No.	Function	Transition Case
(1)	Start ZMOD	not open ZMOD4XXX
(2)	Start Measurement ZMOD	open success
(3)	Check status	Measurement success (callback state success)
(4)	Check Device	Measuring check (IAQ 2nd Gen and IAQ 2nd Gen ULP)
(5)	ADC Read	get status success
(6)	Check Device	ADC data check (IAQ 2nd Gen ULP)
(7)	set Temp & Humidity	user set data (IAQ 2nd Gen ULP and OAQ 2nd Gen)
(8)	calculate	ADC Read success (callback state success)
(9)	next measurement start	DataCalculate return success and next ADC data Get
(10)	Retry Process	Waiting for response (callback none and polling)
(11)	Stop measurement	user stop ZMOD measurement
(12)	Close ZMOD	Error state or user stop
(13)	Restart measurement	measurement stop and user restart
(14)	Stop ZMOD	Close success or Error
(15)	Restart ZMOD	Close success and user restart

Figure 6-2 Transition of API Functions

The calling conditions for each function are as follows:

- RM_ZMOD4XXX_Open:	(1) When starting ZMOD4xxx
	(15) Re-stating after RM_ZMOD4XXX_Close
- RM_ZMOD4XXX_Close:	(12) Successful completion or abnormal end of individual processing
- RM_ZMOD4XXX_MeasurementStart：	(2) Starting measurement after RM_ZMOD4XXX_Open
	(13) When read next measurement data
	(10) Retry by waiting for measurement start response
- RM_ZMOD4XXX_MeasurementStop：	(11) When stop measurement
- RM_ZMOD4XXX_StatusCheack：	(3) Status cheack by polling
- RM_ZMOD4XXX_Read：	(5) When acquiring measurement data after RM_ZMOD4XXX_StatusCheck
	(10) Retry due to waiting for data acquisition response
・RM_ZMOD4XXX_TemperatureAndHumiditySet：	(5) RM_ZMOD4XXX_xxxxDataCalculate preprocessing (IAQ-2nd-Gen-ULP and OAQ-2nd-Gen ambient compensation)
・RM_ZMOD4XXX_DeviceErrorCheck：	(4) Get the status being measured
	(6) Checking ADC data
・RM_ZMOD4XXX_xxxxDataCalculate：	(8) Calculate data after RM_ZMOD4XXX_Read

Note:

Since RM_ZMOD4XXX_Open checks the state of the I2C driver, the I2C driver must be opened before the RM_ZMOD4XXX_Open processing.

Regarding how to open the I2C driver of the RA family and RX family, refer to the g_comms_i2c_bus0_quick_setup() function in the sample software. This is not necessary in the RL78 family devices because the I2C driver will be opened in the startup processing.

When using this API functions in a RTOS system, bus controlling by using semaphore by user is required if controlling the sensors at the same time in multiple threads/tasks.

6.3 Main Processing Flow of Sample Software

The sample software performs the driver opening process, and then repeats the process of starting sensor measurement, acquiring sensor data, and calculating measurement results.

The OS version is controlled by semaphores, and the two threads that control the sensors run in parallel.

6.3.1 ZMOD4410, ZMOD4450

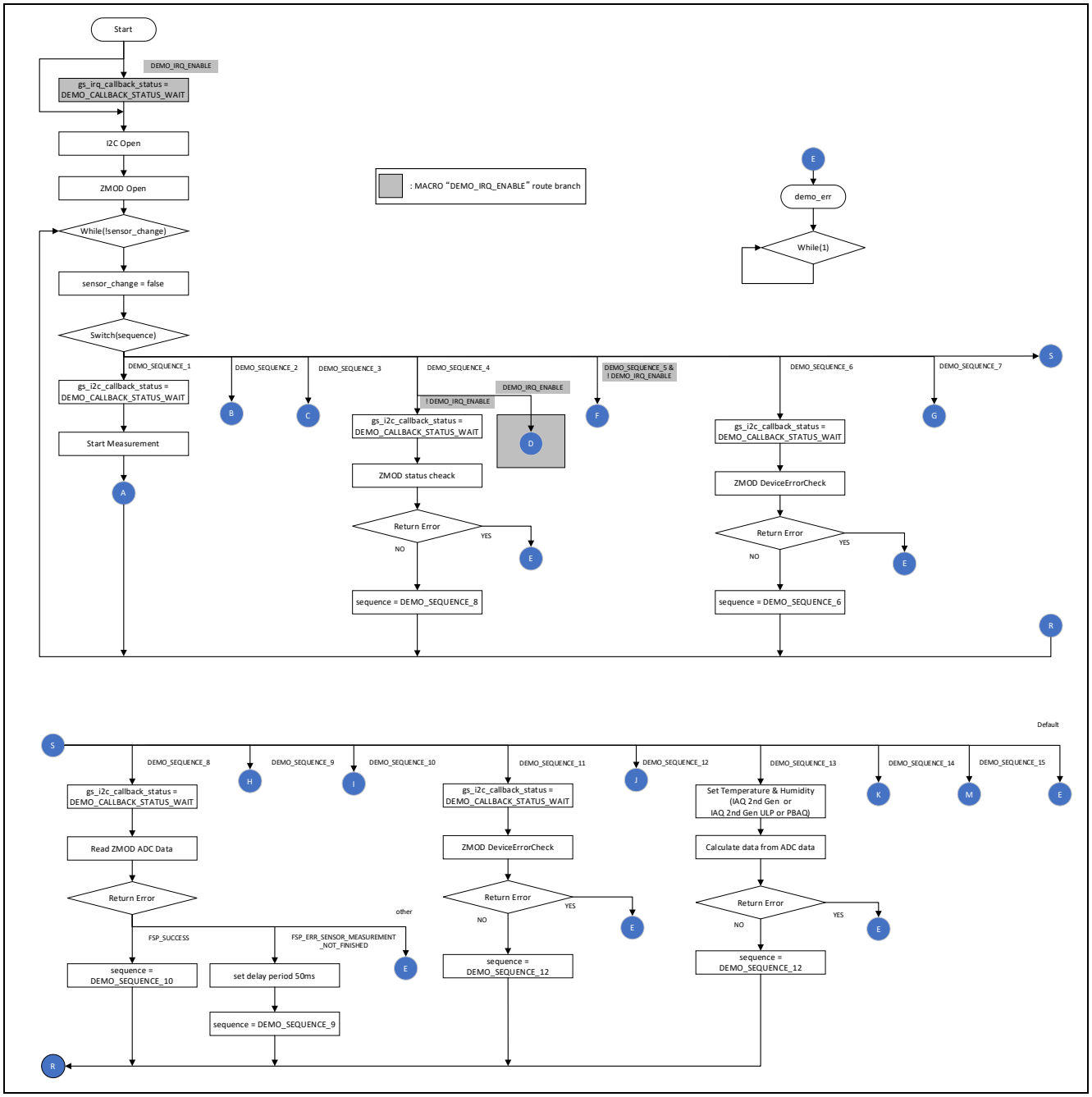


Figure 6-3 Flow 1 of ZMOD4410, ZMOD4450 Sample Software Main Processing

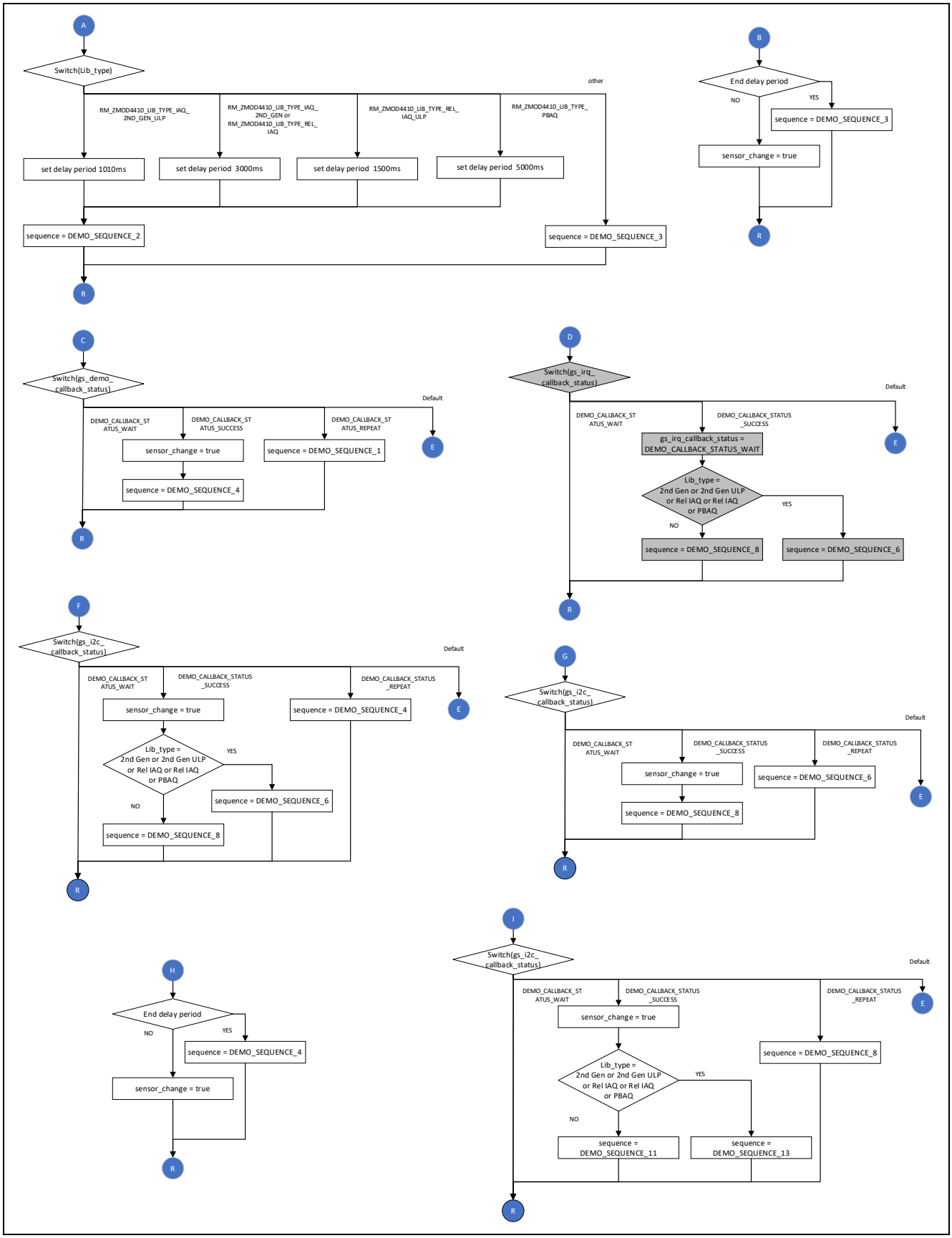


Figure 6-4 Flow 2 of ZMOD4410, ZMOD4450 Sample Software Main Processing

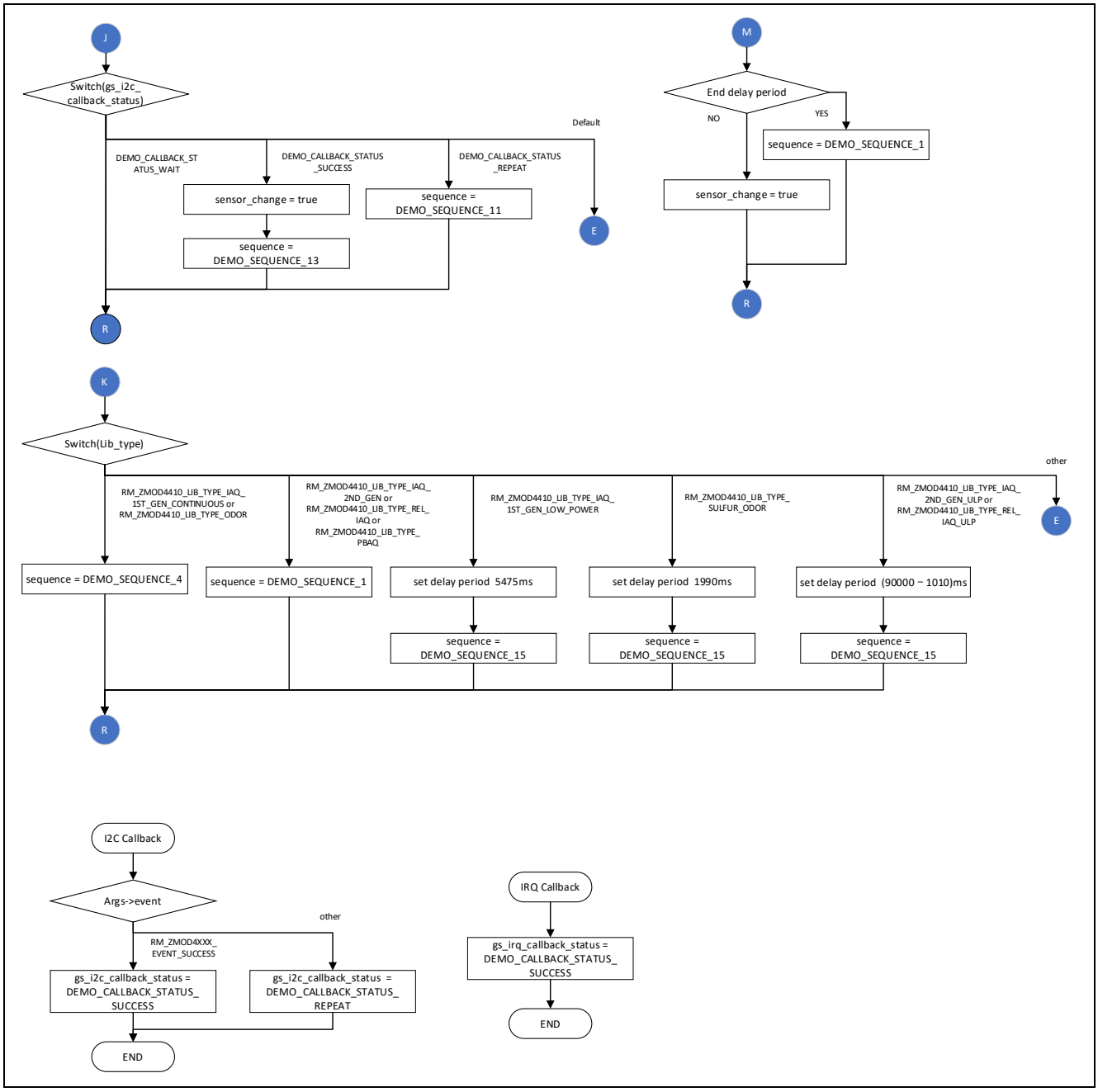


Figure 6-5 Flow 3 of ZMOD4410, ZMOD4450 Sample Software Main Processing

6.3.2 ZMOD4510

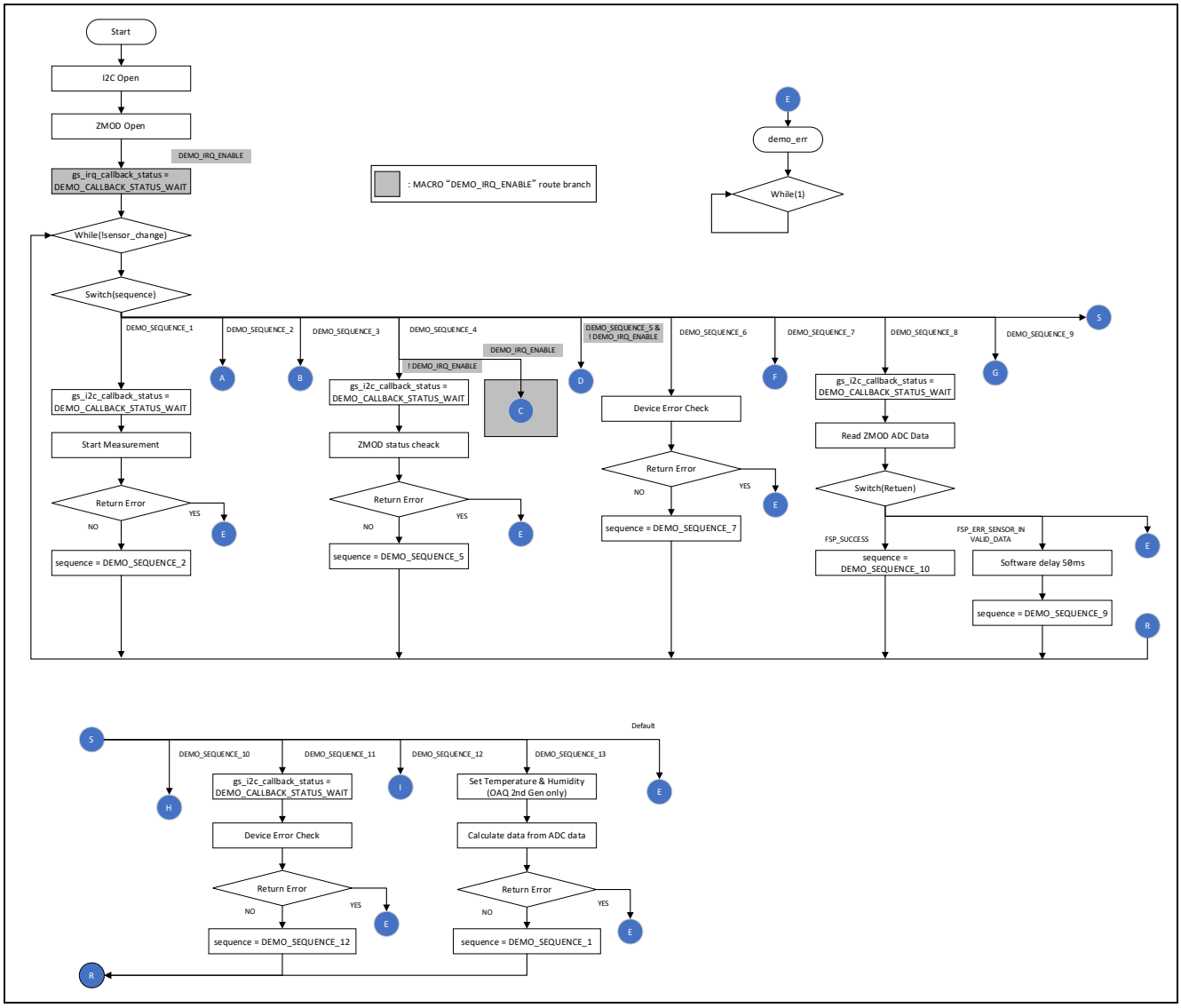


Figure 6-6 Flow 1 of ZMOD4510 Sample Software Main Processing

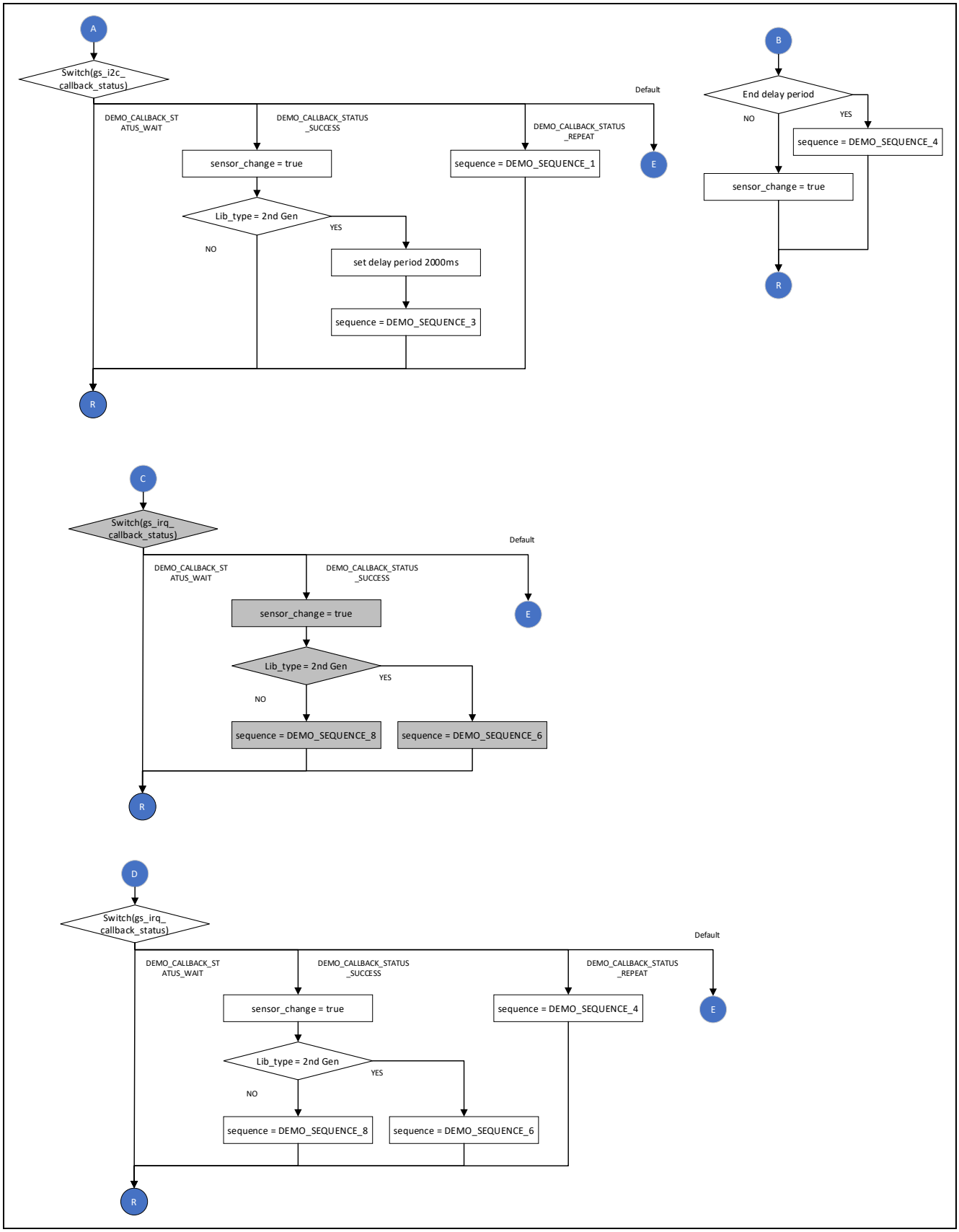


Figure 6-7 Flow 2 of ZMOD4510 Sample Software Main Processing

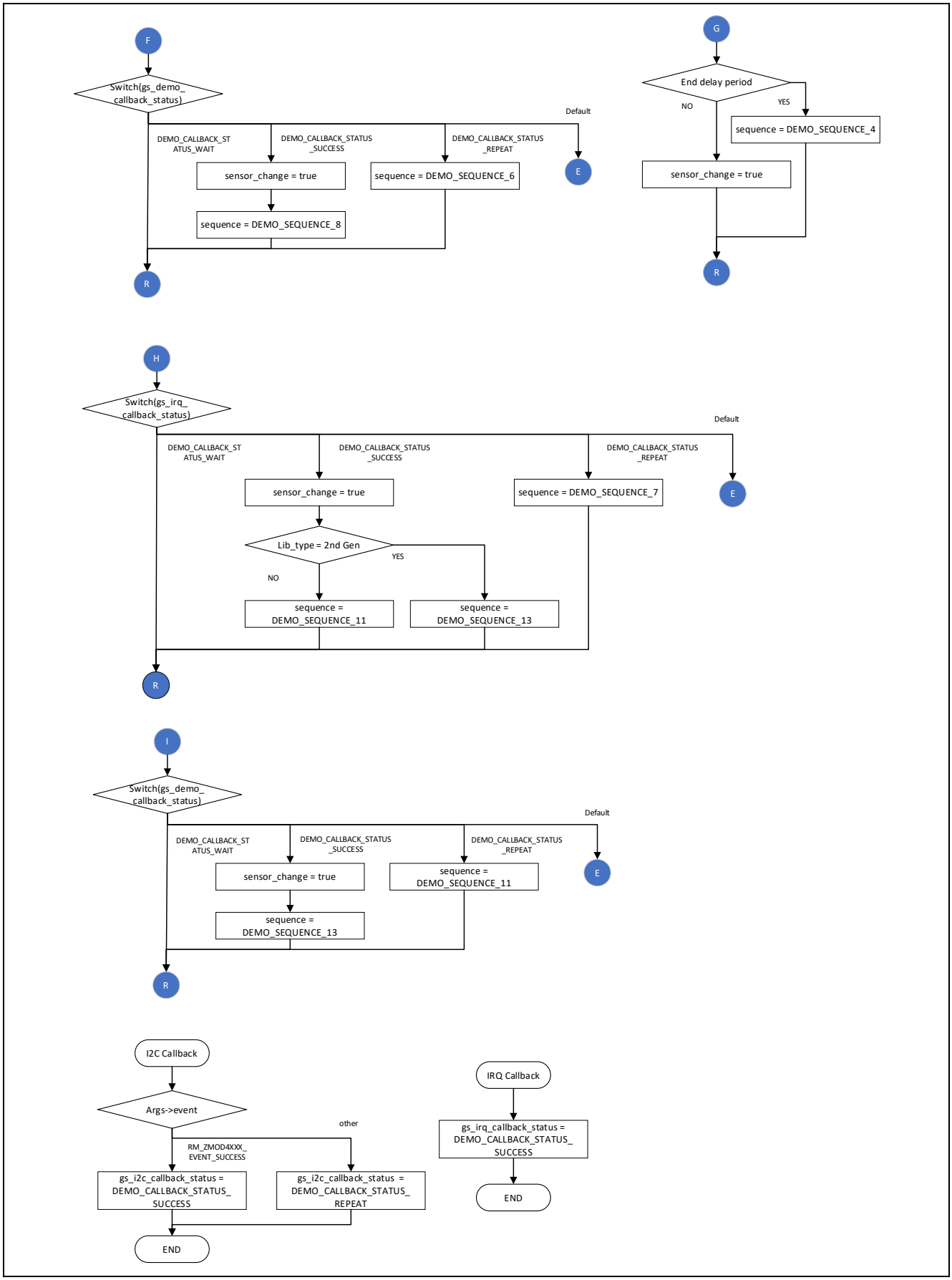


Figure 6-8 Flow 3 of ZMOD4510 Sample Software Main Processing

6.3.3 Azure RTOS Project

The RX Azure RTOS project has the following changes from the default.

1. src/hardware_setup.c

   25th line: Change from 100u to 1000u

2. src/demo_thread.c

   57th line: Add extern void tx_application_define_user (void);

   179th line : Add tx_application_define_user();

3. src/rtos_skelton/zmod4410_sensor_thread_entry.c , zmod4510_sensor_thread_entry.c

   27th line: Change from #include "azurertos_object_init.h" to #include "tx_api.h"

7. Configuration Settings

7.1 ZMOD4xxxx Gas Sensor Settings

7.1.1 RA Family

Select the rm_zmod4xxx stack in the "Stack" tabbed page of the FSP Configurator, and the configurable items are shown in the "Properties" tabbed page.

The following items and values can be specified.

Important: Only one rm_zmod4xxx stack can be registered, and multiple registrations are not allowed.

Table 7-1 ZMOD4xxx Settings for RA Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP) Enabled Disabled	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
Module g_zmod4xxx_sensor ZMOD4XXX on rm_zmod4xxx		
Name	g_zmod4xxx_sensor0	Specify the name of the module. A module name conforming to the C language standard can be specified.
Callback	zmod4xxx_comms_i2c_callback	Specify the name of the user callback function. A callback function name conforming to the C language standard can be specified. When "NULL" is specified, no callback function is used.
IRQ Callback	zmod4xxx_irq_callback	Specify the IRQ user callback function name. A callback function name conforming to the C language standard can be specified. When "NULL" is specified, no callback function is used.

7.1.2 RX Family

Select the r_zmod4xxx_rx component on the [Component] tabbed page of the Smart Configurator, and the configurable items will be shown in the [Configure] panel.

The following items and values can be specified.

Important: Only one rm_zmod4xxx stack can be registered, and multiple registrations are not

Table 7-2 ZMOD4xxx Settings for RX Family

設定項目	設定値	説明
Configurations		
Parameter Checking	Default (BSP) Enabled Disabled	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
Operation mode of ZMOD4XXX Sensors	IAQ 1st Gen. (Continuous)	Specify ZMOD4xxx sensor operation mode. 1: IAQ 1st Gen. (Continuous) 2: IAQ 1st Gen. (Low Power) 3: IAQ 2nd Gen. 4: Odor 5: Sulfur-based Odor 6: OAQ 1st Gen. 7: OAQ 2nd Gen. 8: IAQ 2nd Gen. Ultra-Low Power 9: RAQ 10:Rel IAQ. 11:Rel IAQ. Ultra-Low Power mode 12:PBAQ.
I2C communication device number	I2C Communication Device(x) (x = 0 - 15)	Specify the communications device number to be used by the sensor.
I2C Callback function name	zmod4xxx_user_i2c_callback0	Specify the name of I2C callback function. A callback function name conforming to the C language standard can be specified.
Enable IRQ	Enable or Disable	Specify IRQ enable or disable.
IRQ Callback function name	zmod4xxx_user_irq_callback0	Specify the name of IRQ callback function. A callback function name conforming to the C language standard can be specified.
IRQ number	IRQ(x) (x = 0 - 15)	Specify valid IRQ number.
IRQ Trigger	Rising	Specify IRQ trigger from Low Level, Falling, Rising, Both Edges.
IRQ Interrupt Priority	Priority(x) (x = 0 - 15)	Specify IRQ interrupt priority.

7.1.3 RL78 Family

Settings can be modified by changing the values of constants defined in the \zmod4xxx\r_config\r_zmod4xxx_rl_config.h file in the project tree of the sample project.

The following items and values can be specified.

Table 7-3 ZMOD4xxx Settings for RL78 Family

定数名	設定値	説明
Configurations		
RM_ZMOD4XXX_CFG_PARAM_CHECKING_ENABLE	0	Enable (1) or disable (0) the parameter check processing. When "1" is specified, the project is built so that the generated code includes the parameter check processing.
	1	
RM_ZMOD4XXX_CFG_DEVICE_NUM_MAX	1	Specify the number of ZMOD4xxx sensor
RM_ZMOD4XXX_CFG_DEVICE(x)_OPERATION_MODE (x = 0 – 1)	1	Specify ZMOD4xxx sensor operation mode.* 1: IAQ 1st Gen. (Continuous) 2: IAQ 1st Gen. (Low Power) 3: IAQ 2nd Gen. 8: IAQ 2nd Gen. Ultra-Low Power 4: Odor 5: Sulfur-based Odor 6: OAQ 1st Gen. 7: OAQ 2nd Gen. 9: RAQ 10:Rel IAQ. 11:Rel IAQ. Ultra-Low Power mode 12:PBAQ.
RM_ZMOD4XXX_CFG_DEVICE(x)_COMMS_INSTANCE (x = 0 – 1)	g_comms_i2c_device(x) (x = 0 - 4)	Specify the communications device number to be used by the sensor.
RM_ZMOD4XXX_CFG_DEVICE(x)_COMMS_I2C_CALLBACK (x = 0 – 1)	zmod4xxx_user_i2c_callback0	Specify the name of I2C callback function. A callback function name conforming to the C language standard can be specified.
RM_ZMOD4XXX_CFG_DEVICE(x)_IRQ_ENABLE (x = 0 – 1)	Enable or Disable	Specify IRQ enable or disable.
RM_ZMOD4XXX_CFG_DEVICE(x)_IRQ_CALLBACK (x = 0 – 1)	zmod4xxx_user_irq_callback0	Specify the name of IRQ callback function. A callback function name conforming to the C language standard can be specified.
RM_ZMOD4XXX_CFG_DEVICE(x)_IRQ_NUMBER (x = 0 – 1)	R_INTC(x) (x = 0 – 11)	Specify the number of IRQ

*When using Code Generator or using Smart Configurator of e^2 studio 2021-10 or later, library settings are not automatically set in the build settings. Please set the library settings manually after code generation.

7.1.4 RZ Family

Select the rm_zmod4xxx stack in the "Stack" tabbed page of the FSP Configurator, and the configurable items are shown in the "Properties" tabbed page.

The following items and values can be specified.

Important: Only one rm_zmod4xxx stack can be registered, and multiple registrations are not allowed.

Table 7-4 ZMOD4xxx Settings for RZ Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP)	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
	Enabled	
	Disabled	
Module g_zmod4xxx_sensor0 ZMOD4XXX Gas Sensor (rm_zmod4xxx)		
Name	g_zmod4xxx_sensor0	Specify the name of the module. A module name conforming to the C language standard can be specified.
Callback	zmod4xxx_comms_i2c_callback	Specify the name of the user callback function. A callback function name conforming to the C language standard can be specified. When "NULL" is specified, no callback function is used.
IRQ Callback	zmod4xxx_irq_callback	Specify the IRQ user callback function name. A callback function name conforming to the C language standard can be specified. When "NULL" is specified, no callback function is used.

7.2 Sensor Communication Middleware Settings

7.2.1 RA Family

Select the rm_comms_i2c stack in the "Stack" tabbed page of the FSP Configurator, and the configurable items are shown in the "Properties" tabbed page.

The following items and values can be specified.

Table 7-5 Communication Middleware Settings for RA Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP) Enabled Disabled	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
Module g_comms_i2c_device I2C Communication Device on rm_comms_i2c		
Name	g_comms_i2c_device0	Specify the name of the module. A module name conforming to the C language standard can be specified.
Semaphore Timeout	0xFFFFFFFF	For an RTOS project, specify the time of semaphore timeout.
Slave Address	0x32	Specify the slave address. When rm_zmod4xxx is used, this value is automatically specified and cannot be modified.
Address Mode	7-Bit	Specify the number of slave address bits. When rm_zmod4xxx is used, this value is automatically specified and cannot be modified.
Callback	rm_zmod4xxx_comms_i2c_callback	Specify the name of the user callback function. When rm_zmod4xxx is used, this value is automatically specified and cannot be modified.
Module g_comms_i2c_bus0 I2C Shared Bus on rm_comms_i2c		
Name	g_comms_i2c_bus0	Specify the name of the I2C module.
Bus Timeout	0xFFFFFFFF	Specify the time of I2C bus timeout.
Semaphore for blocking	Unuse Use	For an RTOS project, enable or disable the blocking processing.
Recursive Mutex for Bus	Unuse Use	For an RTOS project, enable or disable the recursive operation when blocking is enabled.

7.2.2 RX Family

Select the r_comms_i2c_rx component on the [Component] tabbed page of the Smart Configurator, and the configurable items will be shown in the [Configure] panel.

The following items and values can be specified.

Table 7-6 Communication Driver Settings for the RX Family MCU

Configurable Item	Value	Description
Configurations		
Parameter Checking	System Default Enabled Disabled	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
Number of communication lines	Unused, 1 – 16 (default: 1)	Specify the number of communications bus lines that can be connected.
Number of I2C Devices	Unused, 1 – 16 (default: 1)	Specify the number of I2C device that can be connected.
Blocking operation supporting with RTOS	Disabled Enabled	For an RTOS project, enable or disable the blocking operation.
Bus lock operation supporting with RTOS	Disabled Enabled	For an RTOS project, enable or disable the bus lock operation.
IIC Driver Type for I2C Shared bus(x) (x = 0 - 15)	RIIC SCI IIC Not selected	Specify the I2C bus type to be used for the communication bus. When using the RIIC, r_riic_rx is necessary. When using the SCI IIC, r_sci_iic_rx is necessary. If an unused FIT module is deleted, a warning message will appear but this does not affect the operation.
Channel No. for I2C Shared bus(x) (x = 0 - 15)	0	Specify the I2C channel number to be used for the communication bus.
Timeout for the bus lock of the I2C bus for I2C Shared Bus(x) (x = 0 - 15)	0xFFFFFFFF	Specify the time of I2C bus(x) timeout. (x = 0 - 15)
I2C Shared Bus No. for I2C Communication Device(x) (x = 0 - 15)	I2C Shared Bus(x) (x = 0 - 15)	Specify the configuration of used communication bus.
Slave address for communication device(x) (x = 0 - 15)	0x32	Specify the slave address of the device to be connected to the communications bus. If you are using r_zmod4xxx_rx, specify 0x32.
Slave address mode for communication device(x) (x = 0 - 15)	7 bit address mode	Specify the slave address mode. If you are using r_zmod4xxx_rx, specify the 7-bit address mode.
Callback function for Communication device(x) (x = 0 - 15)	comms_i2c_user_callback(x) (x = 0 - 15)	Specify the name of the user callback function. When using r_zmod4xxx_rx, specify rm_zmod4xxx_callback(y) (y = 0).

7.2.3 RL78 Family

Settings can be modified by changing the values of constants defined in the \zmod4xxx\r_config\r_comms_i2c_rl_config.h file in the project tree of the sample project.

The following items and values can be specified.

Table 7-7 Communication Driver Settings for the RL78 Family MCU

Constant Name	Value	Description
Configurations		
COMMS_I2C_CFG_PARAM_CHECKING_ENABLE	0	Enable (1) or disable (0) the parameter check processing. When "1" is selected, the project is built so that the generated code includes the parameter check processing.
	1	
COMMS_I2C_CFG_BUS_NUM_MAX	1	Specify the number of communication bus lines that can be connected.
	2	
	3	
	4	
	5	
COMMS_I2C_CFG_DEVICE_NUM_MAX	1	Specify the number of I2C devices can be connected.
	2	
	3	
	4	
	5	
COMMS_I2C_CFG_BUS(x)_DRIVER_TYPE (x = 0 - 4)	COMMS_DRIVER_I2C	Specify the I2C type to be used for the communication bus.
	COMMS_DRIVER_SAU_I2C	
COMMS_I2C_CFG_DEVICE(x)_BUS_CH (x = 0 - 4)	g_comms_i2c_bus(x)_extended_cfg (x = 0 - 4)	Specify the I2C bus configuration to be used for the communication bus.
COMMS_I2C_CFG_DEVICE(x)_SLAVE_ADDR (x = 0 - 4)	0x32	Specify the slave address of the device to be connected to the communications bus. If you are using rm_zmod4xxx, specify 0x32.
COMMS_I2C_CFG_BUSx_CALLBACK_ENABLE (x = 0 - 4)	0	Enable (1) or disable (0) the user callback function.
	1	
COMMS_I2C_CFG_BUSx_CALLBACK (x = 0 - 4)	comms_i2c_user_callback1(x) (x = 0 - 4)	Specify the name of the user callback function. When using rm_zmod4xxx, specify rm_zmod4xxx_callback(y) (y = 0).

7.2.4 RZ Family

Select the rm_comms_i2c stack in the "Stack" tabbed page of the FSP Configurator, and the configurable items are shown in the "Properties" tabbed page.

The following items and values can be specified.

Table 7-8 Communication Middleware Settings for RZ Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP)	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
	Enabled	
	Disabled	
Module g_comms_i2c_device0 I2C Communication Device (rm_comms_i2c)		
Name	g_comms_i2c_device0	Specify the name of the module. A module name conforming to the C language standard can be specified.
Semaphore Timeout	0xFFFFFFFF	For an RTOS project, specify the time of semaphore timeout.
Slave Address	0x32	Specify the slave address. When rm_zmod4xxx is used, this value is automatically specified and cannot be modified.
Address Mode	7-Bit	Specify the number of slave address bits. When rm_zmod4xxx is used, this value is automatically specified and cannot be modified.
Callback	rm_zmod4xxx_comms_i2c_callback	Specify the name of the user callback function. When rm_zmod4xxx is used, this value is automatically specified and cannot be modified.
Module g_comms_i2c_bus0 I2C Shared Bus (rm_comms_i2c)		
Name	g_comms_i2c_bus0	Specify the name of the I2C module.
Bus Timeout	0xFFFFFFFF	Specify the time of I2C bus timeout.
Semaphore for Blocking	Unuse	For an RTOS project, enable or disable the blocking processing.
	Use	
Recursive Mutex for Bus	Unuse	For an RTOS project, enable or disable the recursive operation when blocking is enabled.
	Use	

7.3 I2C Driver Settings

7.3.1 RA Family

Select the r_iic_master or r_sci_i2c stack in the "Stack" tabbed page of the FSP Configurator, and the configurable items are shown in the "Properties" tabbed page.

The following items and values can be specified.

Table 7-9 r_iic_master Settings for RA Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP) Enabled Disabled	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
DTC on Transmission and Reception	Enabled Disabled	Specify whether to use the DTC for transmission and reception.
10-bit slave addressing	Enabled Disabled	Specify whether to support 10-bit addressing for the slave address. When using rm_zmod4xxx, select "Disabled".
Module g_i2c_master0 I2C Master Driver on r_iic_master		
Name	g_i2c_master0	Specify the name of the module.
Channel	0	Specify the channel number to be used.
Rate	Standard Fast-mode Fast-mode plus	Specify the baud rate. When using rm_zmod4xxx, select "Standard" or "Fast-mode".
Rise Time (ns)	120	Specify the SCL rise time according to the specifications of the target board to be used.
Fall Time (ns)	120	Specify the SCL fall time according to the specifications of the target board to be used.
Duty Cycle (%)	50	Specify the SCL duty cycle.
Slave Address	0x00	This item specifies the slave address of the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
Address Mode	7-Bit 10-Bit	This item specifies the salve address mode for the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
Timeout Mode	Short Mode Long Mode	Specify the time of I2C bus timeout.
Callback	rm_comms_i2c_callback	The name of the user callback function is automatically specified by rm_comms_i2c.
Interrupt Priority Level	Priority 0 (highest) Priority 1 Priority 2 Priority 3 Priority 4 Priority 5	Specify the interrupt priority level of the I2C bus driver.

	Priority 6	
	Priority 7	
	Priority 8	
	Priority 9	
	Priority 10	
	Priority 11	
	Priority 12	
	Priority 13	
	Priority 14	
	Priority 15	
Pins		
SDA	Pxxx	The pin numbers to be used by the driver are displayed. Use the "Pins" tabbed page to modify the pin configuration.
SCL	Pxxx	

Table 7-10 r_sci_i2c Settings for RA Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP)	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
	Enabled	
	Disabled	
DTC on Transmission and Reception	Enabled	Specify whether to use the DTC for transmission and reception.
	Disabled	
10-bit slave addressing	Enabled	Specify whether to support 10-bit addressing for the slave address. When using rm_zmod4xxx, select "Disabled".
	Disabled	
Module g_i2c0 I2C Master Driver on r_sci_i2c		
Name	g_i2c0	Specify the name of the module.
Channel	0	For an RTOS project, specify the time of semaphore timeout.
Slave Address	0x00	This item specifies the slave address of the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
Address Mode	7-Bit	This item specifies the salve address mode for the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
	10-bit	
Rate	Standard	Specify the baud rate. Select "Standard" or "Fast-mode".
	Fast-mode	
	Fast-mode plus	
SDA Output Delay (nano seconds)	300	Specify the SDA output delay time.
Noise filter setting	Use clock signal divided by 1 with noise filter	Specify the noise filter to be used for input signals.
	Use clock signal divided by 2 with noise filter	

	Use clock signal divided by 4 with noise filter	
	Use clock signal divided by 8 with noise filter	
Bit Rate Modulation	Enable	Enable or disable the bit rate modulation function.
	Disable	
Callback	rm_comms_i2c_callback	The name of the user callback function is automatically specified by rm_comms_i2c.
Interrupt Priority Level	Priority 0 (highest)	Specify the interrupt priority level of the I2C bus driver.
	Priority 1	
	Priority 2	
	Priority 3	
	Priority 4	
	Priority 5	
	Priority 6	
	Priority 7	
	Priority 8	
	Priority 9	
	Priority 10	
	Priority 11	
	Priority 12	
	Priority 13	
	Priority 14	
	Priority 15	
RX Interrupt Priority Level [Only used when DTC is enabled]	Priority 0 (highest)	When using the DTC, specify the priority level of the reception interrupt.
	Priority 1	
	Priority 2	
	Priority 3	
	Priority 4	
	Priority 5	
	Priority 6	
	Priority 7	
	Priority 8	
	Priority 9	
	Priority 10	
	Priority 11	
	Priority 12	
	Priority 13	
	Priority 14	
	Priority 15	
	Disabled	
Pins		
SDA	Pxxx	The pin numbers to be used by the driver are displayed. Use the "Pins" tabbed page to modify the pin configuration.
SCL	Pxxx	

7.3.2 RX Family

Select the r_riic_rx or r_sci_iic_rx component on the [Component] tabbed page of the Smart Configurator, and the configurable items will be shown in the [Configure] panel.

The following items and values can be specified.

Table 7-11 r_riic_rx Settings for the RX Family MCU

Configurable Item	Value	Description
Configurations		
Set parameter checking enable	System Default	Enable or disable the parameter check processing. When "Include" is selected, the project is built so that the generated code includes the parameter check processing.
	Not	
	Include	
MCU supported channels for CHx (x = 0 – 2)	Not supported	Specify whether to support the operation of channel x.
	Supported	
CHx RIIC bps(kbps) (x = 0 – 2)	400	Specify the bit rate. Set to 400 or a smaller value.
Digital filter for CHx (x = 0 – 2)	Not	Specify the digital filter for input signals.
	One IIC phi	
	Two IIC phi	
	Three IIC phi	
	Four IIC phi	
Setting port setting processing	Not include port setting	Specify whether to include the pin function settings in the code to be generated.
	Include port setting	
Master arbitration lost detection function for CHx (x = 0 – 2)	Unused	Specify whether to use the master arbitration lost detection function.
	Used	
Address y format for CHx (x = 0 – 2, y = 0 – 2)	Not	This item specifies the slave address mode for slave address y but the user does not need to make this setting because any setting that is made here is overwritten by the setting in rm_comms_i2c.
	7 bit address format	
	10 bit address format	
Slave Address y for CHx (x = 0 – 2, y = 0 – 2)	0x0025	This item specifies slave address y but the user does not need to make this setting because any setting that is made here is overwritten by the setting in rm_comms_i2c.
General call address for CHx	Unused	Specify whether to use the general call function.
	Used	
CHx RXI INT Priority Level (x = 0 – 2)	Level 1	Specify the priority level of the reception interrupt.
	Level 2	
	Level 3	
	Level 4	
	Level 5	
	Level 6	
	Level 7	
	Level 8	
	Level 9	
	Level 10	
	Level 11	
	Level 12	
	Level 13	

	Level 14	
	Level 15 (highest)	
CHx RXI INT Priority Level (x = 0 – 2)	Level 1	Specify the priority level of the transmission interrupt.
	Level 2	
	Level 3	
	Level 4	
	Level 5	
	Level 6	
	Level 7	
	Level 8	
	Level 9	
	Level 10	
	Level 11	
	Level 12	
	Level 13	
	Level 14	
	Level 15 (highest)	
CHx EEI INT Priority Level (x = 0 – 2)	Level 1	Specify the priority level of the error interrupt.
	Level 2	
	Level 3	
	Level 4	
	Level 5	
	Level 6	
	Level 7	
	Level 8	
	Level 9	
	Level 10	
	Level 11	
	Level 12	
	Level 13	
	Level 14	
	Level 15 (highest)	
CHx TEI INT Priority Level (x = 0 – 2)	Level 1	Specify the priority level of the transmission end interrupt.
	Level 2	
	Level 3	
	Level 4	
	Level 5	
	Level 6	
	Level 7	
	Level 8	
	Level 9	
	Level 10	
	Level 11	
	Level 12	
	Level 13	
	Level 14	
	Level 15 (highest)	
Timeout function for CHx (x = 0 – 2)	Unused	Specify whether to use the timeout function.
	Used	
Timeout detection time	Long mode	Specify the time for timeout detection.

for CHx (x = 0 – 2)	Short mode	
Count up during low period of timeout detection for CHx (x = 0 – 2)	Unused Used	Specify whether to increment the counter for detecting a timeout while SCL is at the low level.
Count up during high period of timeout detection for CHx (x = 0 – 2)	Unused Used	Specify whether to increment the counter for detecting a timeout while SCL is at the high level.
Set Counter of checking bus busy	1000	Specify the counter value to be judged to represent the bus busy state.
Resources		
SDAx Pins	Checked	Specify the pins to be used. Select the checkboxes for the desired pins.
SCLx Pins	Checked	

Table 7-12 r_sci_iic_rx Settings for the RX Family MCU

Configurable Item	Value	Description
Configurations		
Set parameter checking enable	System Default Not Include	Enable or disable the parameter check processing. When "Include" is selected, the project is built so that the generated code includes the parameter check processing.
MCU supported channels for CHx (x = 0 – 12)	Not supported Supported	Specify whether to support the operation of channel x.
SCI IIC bitrate (bps) for CHx (x = 0 – 12)	384000	Specify the bit rate. Set to 384000 or a smaller value.
Interrupt Priority for CHx (x = 0 – 12)	Level 1 Level 2 Level 3 Level 4 Level 5 Level 6 Level 7 Level 8 Level 9 Level 10 Level 11 Level 12 Level 13 Level 14 Level 15 (highest)	Specify the interrupt priority level.
Digital noise filter (NFEN bit) for CHx (x = 0 – 12)	Disable Enable	Specify whether to use the digital noise filter.
Noise Filter Setting Register (NFCS bit) for CHx (x = 0 – 12)	The clock divided by 1 The clock divided by 2 The clock divided by 4 The clock divided by 8	Specify the function of the digital noise filter.

I2C Mode Register 1 (IICDL bit) for CHx (x = 0 – 12)	18	Specify the number of SDA output delay cycles.
Software bus busy check counter	1000	Specify the counter value to be judged to represent the bus busy state.
Setting port setting processing	Not include port setting Include port setting	Specify whether to include the pin function settings in the code to be generated.
Resources		
SSDAx Pins	Checked	Specify the pins to be used. Select the checkboxes for the desired pins.
SSCLx Pins	Checked	

7.3.3 RL78 Family

Select "Serial" from the peripheral functions in the Code Generator, and the configurable items will be shown on the [Peripheral Functions] tabbed page.

The following items and values can be specified.

Table 7-13 Serial Settings for the RL78 Family MCU

Configurable Item	Value	Description
SAUx		
Channel		
Channel x	Unused	Specify the communication function of the channel to be used. If you are using r_zmod4xxx, select IICxx.
	UARTxx	
	CSIxx	
	IICxx	
IICxx		
Transfer rate	1000000	Specify the bit rate. If you are using r_zmod4xxx, specify 100000.
Transfer end interrupt priority (INTIICxx)	High	Specify the priority level of the transfer end interrupt.
	Level1	
	Level2	
	Low	
Master transmission end	Checked	Specify whether to use the callback function when master transmission ends.
Master reception end	Checked	Specify whether to use the callback function when master reception ends.
Master error	Checked	Specify whether to use the callback function when a communication error occurs.
IICAx		
Transfer mode		
Transfer mode	Unused	Specify the communication function of the channel to be used. Select "Single master".
	Single master	
	Slave	
Setting		
Clock mode setting	fCLK	Specify the clock to drive counting.
	fCLK/2	
Address	16	Specify the local address.
Operation mode setting	Standard	Specify the operating mode.
	Fast mode/Fast mode plus	
Transfer clock (fSCL)	100000	Specify the bit rate. Set to 400000 or a smaller value.
Communication end interrupt priority (INTIICAx)	High	Specify the priority level of the communication end interrupt.
	Level1	
	Level2	
	Low	
Master transmission end	Checked	Specify whether to use the callback function when master transmission ends.
Master reception end	Checked	Specify whether to use the callback function when master reception ends.
Master error	Checked	Specify whether to use the callback function when a communication error occurs.
Generated stop condition in master transmission/reception end callback function	Checked	Specify whether to generate a stop condition in the callback function. Deselect the checkbox.

7.3.4 RZ Family

Select the r_riic_master in the "Stack" tabbed page of the FSP Configurator, and the configurable items are shown in the "Properties" tabbed page.

The following items and values can be specified.

Table 7-14 r_riic_master Settings for RZ Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP)	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
	Enabled	
	Disabled	
10-bit slave addressing	Enabled	Specify whether to support 10-bit addressing for the slave address. When using rm_zmod4xxx, select "Disabled".
	Disabled	
Module g_i2c_master3 I2C Master Driver on r_riic_master		
Name	g_i2c_master3	Specify the name of the module.
Channel	3	Specify the channel number to be used.
Rate	Standard	Specify the baud rate. When using rm_zmod4xxx, select "Standard" or "Fast-mode".
	Fast-mode	
	Fast-mode plus	
Rise Time (ns)	120	Specify the SCL rise time according to the specifications of the target board to be used.
Fall Time (ns)	120	Specify the SCL fall time according to the specifications of the target board to be used.
Duty Cycle (%)	50	Specify the SCL duty cycle.
Noise Filter Stages	1	Removes noise below the 1 IICφ cycle.
	2	Removes noise below the 2 IICφ cycle.
	3	Removes noise below the 3 IICφ cycle.
	4	Removes noise below the 4 IICφ cycle.
Slave Address	0x00	This item specifies the slave address of the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
Address Mode	7-Bit	This item specifies the salve address mode for the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
	10-Bit	
Timeout Mode	Short Mode	Specify the time of I2C bus timeout.
	Long Mode	
Callback	rm_comms_i2c_callback	The name of the user callback function is automatically specified by rm_comms_i2c.

Interrupt Priority Level	0 (highest)	Specify the interrupt priority level of the I2C bus driver.
	1	
	2	
	3	
	4	
	5	
	6	
	7	
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	

7.4 IRQ Driver Settings

7.4.1 RZ Family

Select the r_intc_irq stack in the "Stack" tabbed page of the FSP Configurator, and the configurable items are shown in the "Properties" tabbed page.

The following items and values can be specified.

Table 7-15 r_intc_irq Settings for RZ Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP)	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
	Enabled	
	Disabled	
Module g_external_irq7 External IRQ Driver on r_intc_irq		
Name	g_external_irq7	Specify the name of the module.
Channel	7	Specify the channel number to be used.
Trigger	Falling	Specify the trigger. When using rm_zmod4xxx, select "Rising".
	Rising	
	Both Edges	
	Low Level	
Callback	rm_zmod4xxx_irq_callback	The name of the user callback function is automatically specified by r_intc_irq.
Pin Interrupt Priority	0 (highest)	Specify the interrupt priority level of the IRQ driver.
	1	
	2	
	3	
	4	
	5	
	6	
	7	
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	
Pins		
IRQ7	<unavailable>	Pin settings are made in pin_data.c in the src folder. See “8.4.3 Changing sample code” for the setting method.

8. Guide for Changing the Target Device

Use the following procedures to change the target device to a new one and run a sample project on the new device.

Before switching to a new device, import the original sample project for the current device to the workspace.

8.1 RA Sample Project

Use the following procedures to modify a sample project.

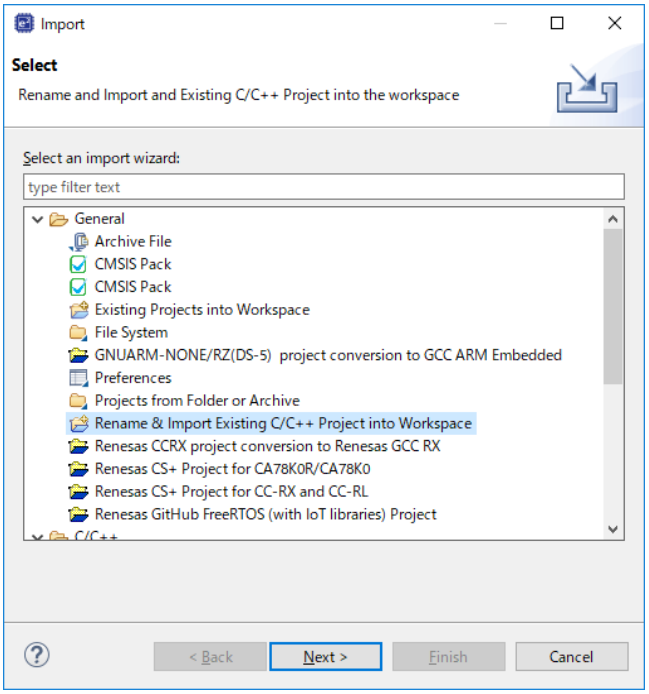
This section describes an example of modifying the sample project " ZMOD4xxx_RA6M4_NonOS" so that it can be used on the EK-RA2E1 board.

The description of PMOD1 is the procedure when using a board to which "OptionType6A" is applied.

8.1.1 Importing the Sample Project

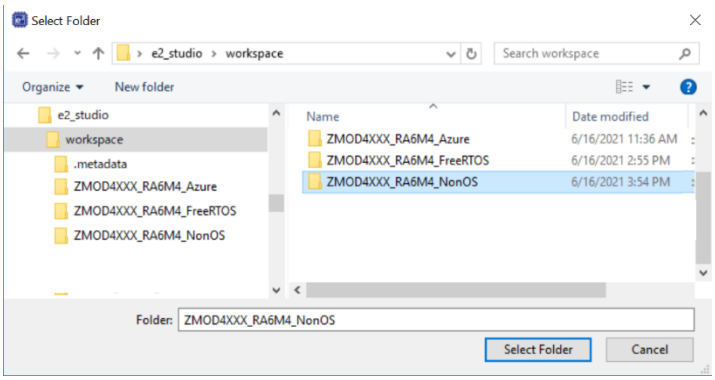
Select [Import] from the menu.

The "Import" window will appear. Select "Rename & Import Existing C/C++ Project into Workspace" in the window and press the [Next] button.

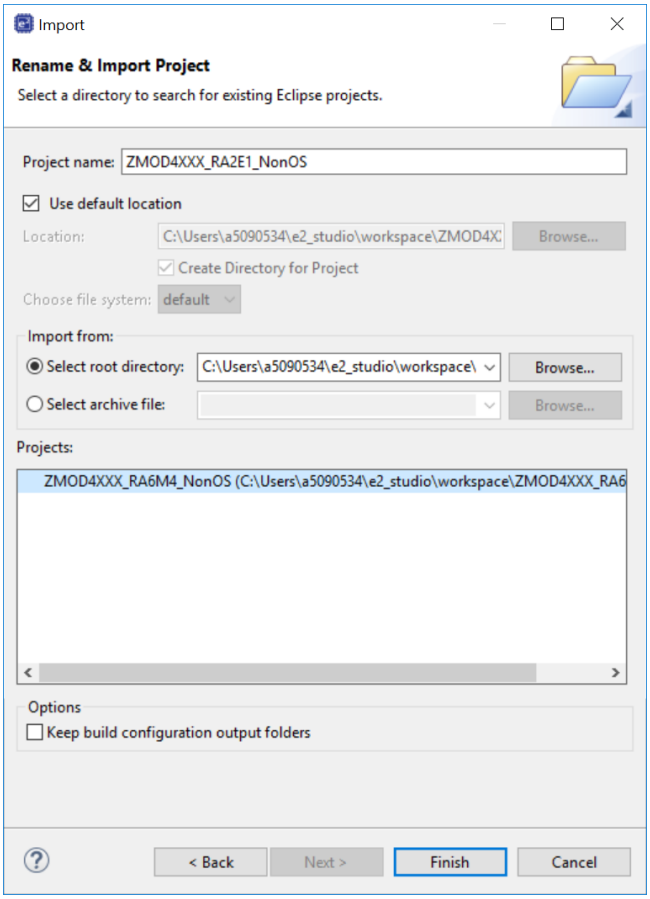


Press the [Browse] button to open the "Select Folder" window.

Select the folder of the original project for the current device from a list of imported sample projects and press the [Select Folder] button.



Enter the project name, select the original project for the current device, and press the [Finish] button.



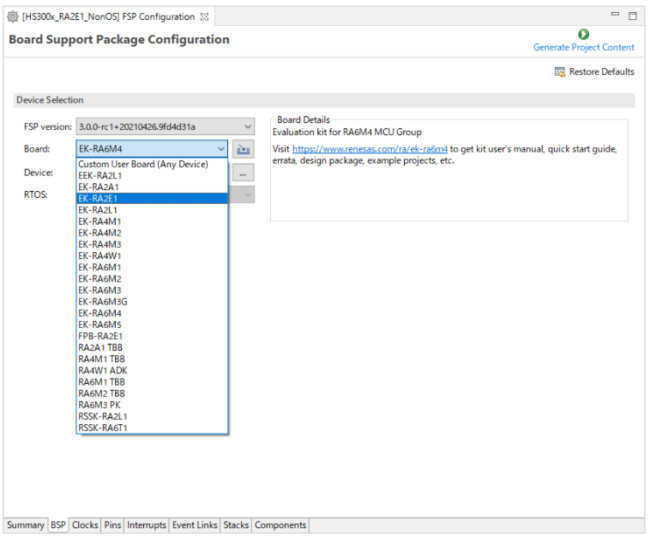
8.1.2 Modifying Settings of the FSP Configurator

Double-click on "Configuratorn.xml" in the project tree to open the FSP Configurator.

Change the settings of "Board" and "Device" in the "BSP" tabbed page.

When selecting a Renesas board, modify the "Board" setting only.

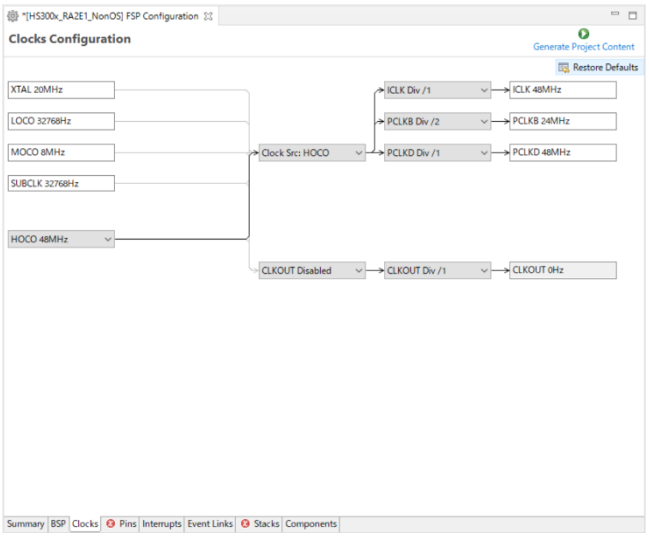
When selecting a board provided from other companies, change the "Board" setting to "Custom User Board (Any Device)" and then change the "Device" setting to the new device to be used.



Set up the clocks in the "Clocks" tabbed page.

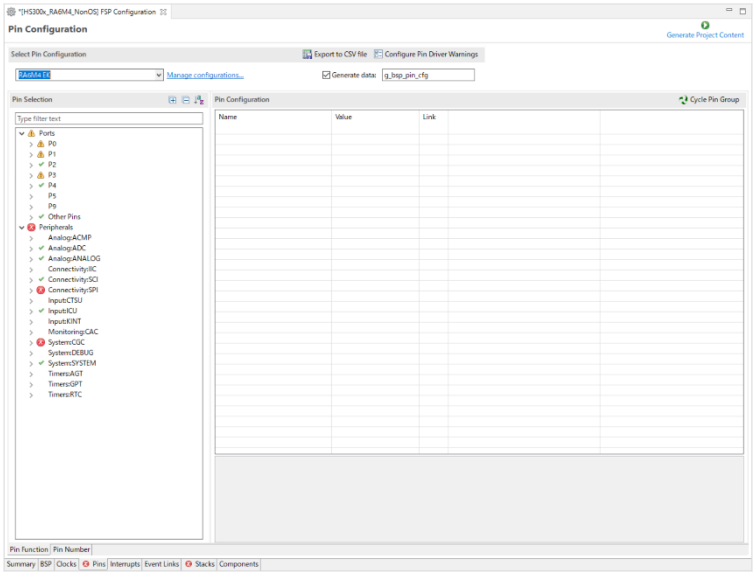
When "Custom User Board (Any Device)" is selected for "Board", set up the clocks according to the specifications of the target board to be used.

When a Renesas board is selected for "Board", the clocks are automatically set up.

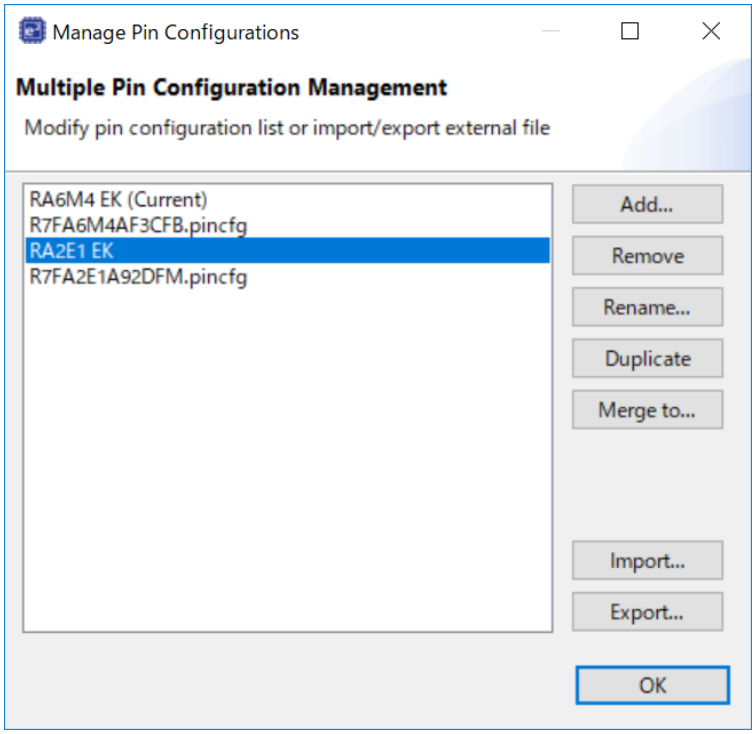


In the “Pins” tabbed page, modify the pin configuration according to the specifications of the target board to be used.

When using a Renesas board, change the selection for "Select Pin Configuration" from ”RA6M4 EK” to the target board; appropriate pins are automatically assigned.



If the desired board is not displayed in the drop-down list for "Select Pin Configuration", click on [Manage Configuration] to open the "Manage Pin Configuration" window and select the desired board in the window.



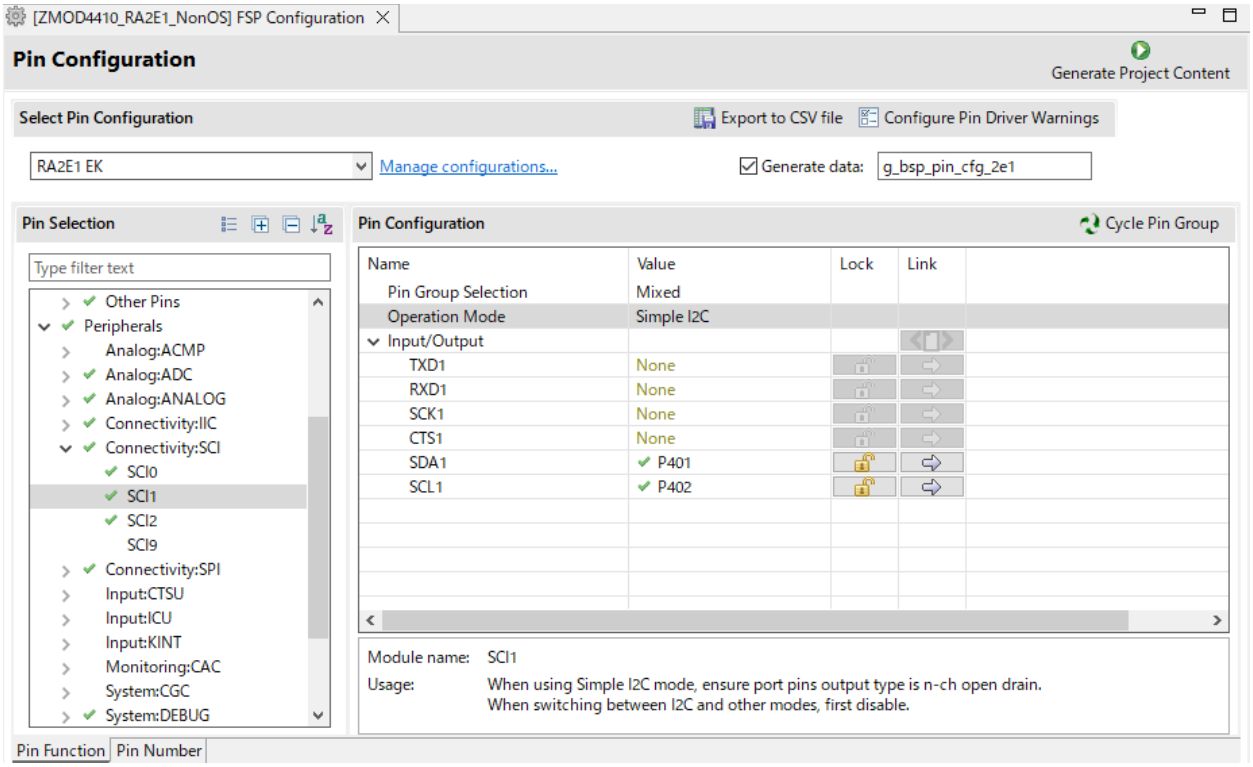
However, the "Select Pin Configuration" assignment will apply the SPI communication pin settings that support PMOD Type 2A on the EK-RA2E1 board.

This sample software uses PMOD Type 6A, therefore it is necessary to change the I2C communication pin settings that support PMOD Type 6A.

SCI2 is assigned to PMOD1 and SCI1 to PMOD2 on the EK-RA2E1 board.

I2C communication is assigned to P301 and P302 on PMOD1(OptionType6A), and it is assigned to P401 and P402 on PMOD2.

After automatic assignment of "Select Pin Configuration", reconfigure in "Pin Configuration".

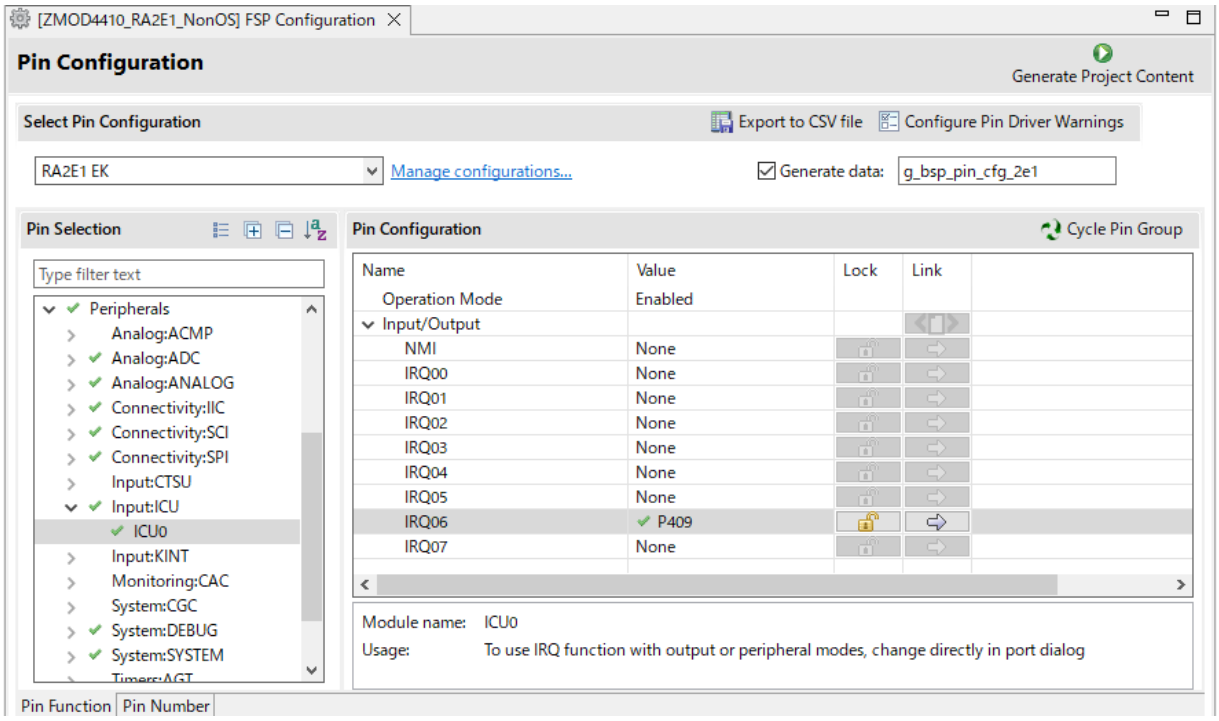


IRQ exists as an interrupt signal pin of the sensor, but pin setting is not performed by automatic assignment of “Select Pin Configuration”.

IRQ6 is assigned to PMOD2 on the EK-RA2E1 board.

To use PMOD2, P409 set to IRQ6 pin.

To use PMOD1(OptionType6A), IRQ cannot be used. Refer to section 8.1.4, Changing when not using IRQ.



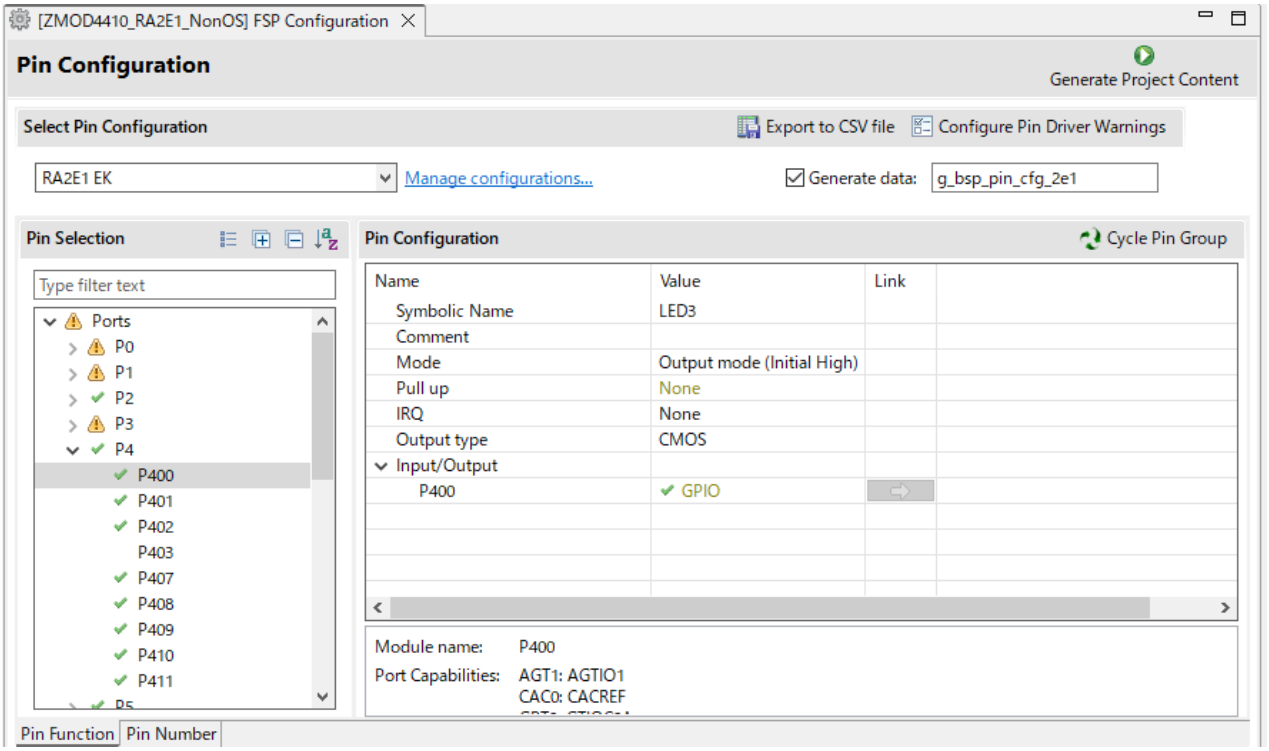
In “Select Pin Configuration”, it is automatically assigned as Type 2A.

Therefore, the pin corresponding to the RESET pin of Type 6A must be changed to I/O port pin.

To use PMOD1(OptionType6A), P101 set to I/O port pin.

To use PMOD2, P400 set to I/O port pin.

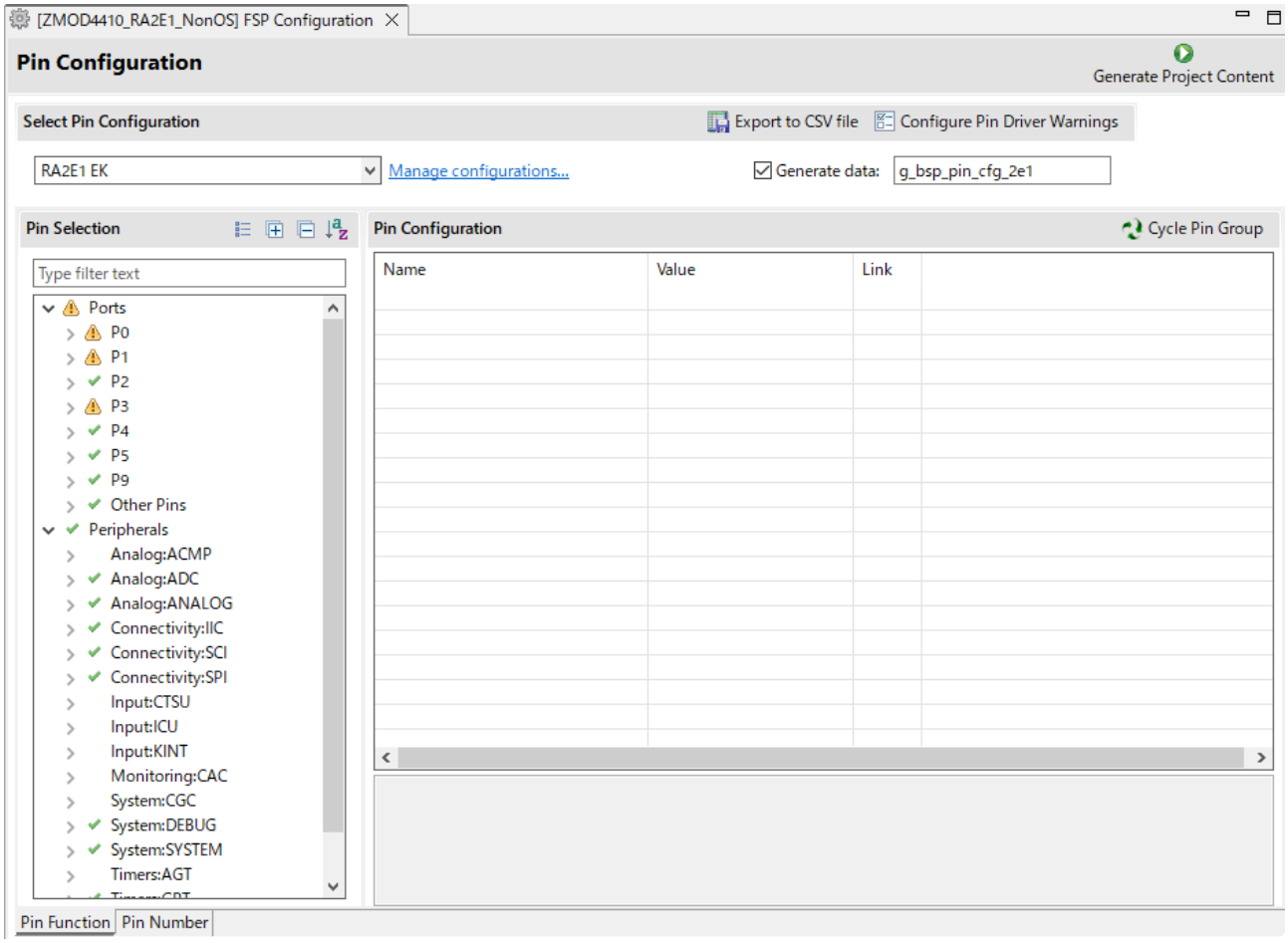
Select “Output mode (initial High)” for “Mode”.



To enable generation of pin settings, check [Generate data] check-box and enter a desired name in the text field.

The entered name is linked to the pin configuration, therefore must use a unique name that does not duplicate with other pin configurations.

In our example, it is ”g_bsp_pin_cfg_2e1”.



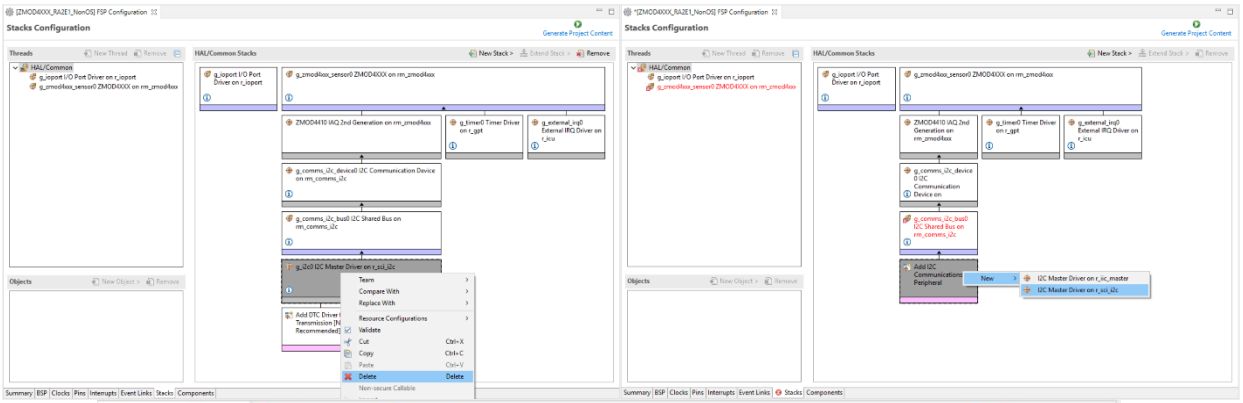
Modify the configuration of individual components in the "Stacks" tabbed page.

Modify the settings of r_iic_master or r_sci_i2c according to the specifications of the target board.

To use the pins of the IIC, delete the "I2C Master Driver on r_sci_i2c" stack and then add the "I2C Master Driver on r_iic_master" stack.

SCI2 is assigned to PMOD1 and SCI1 is assigned to PMOD2 on the EK-RA2E1 board.

To use PMOD1, set "Channel" to "2". To use PMOD2, set to "1".



Properties | Problems | Smart Browser

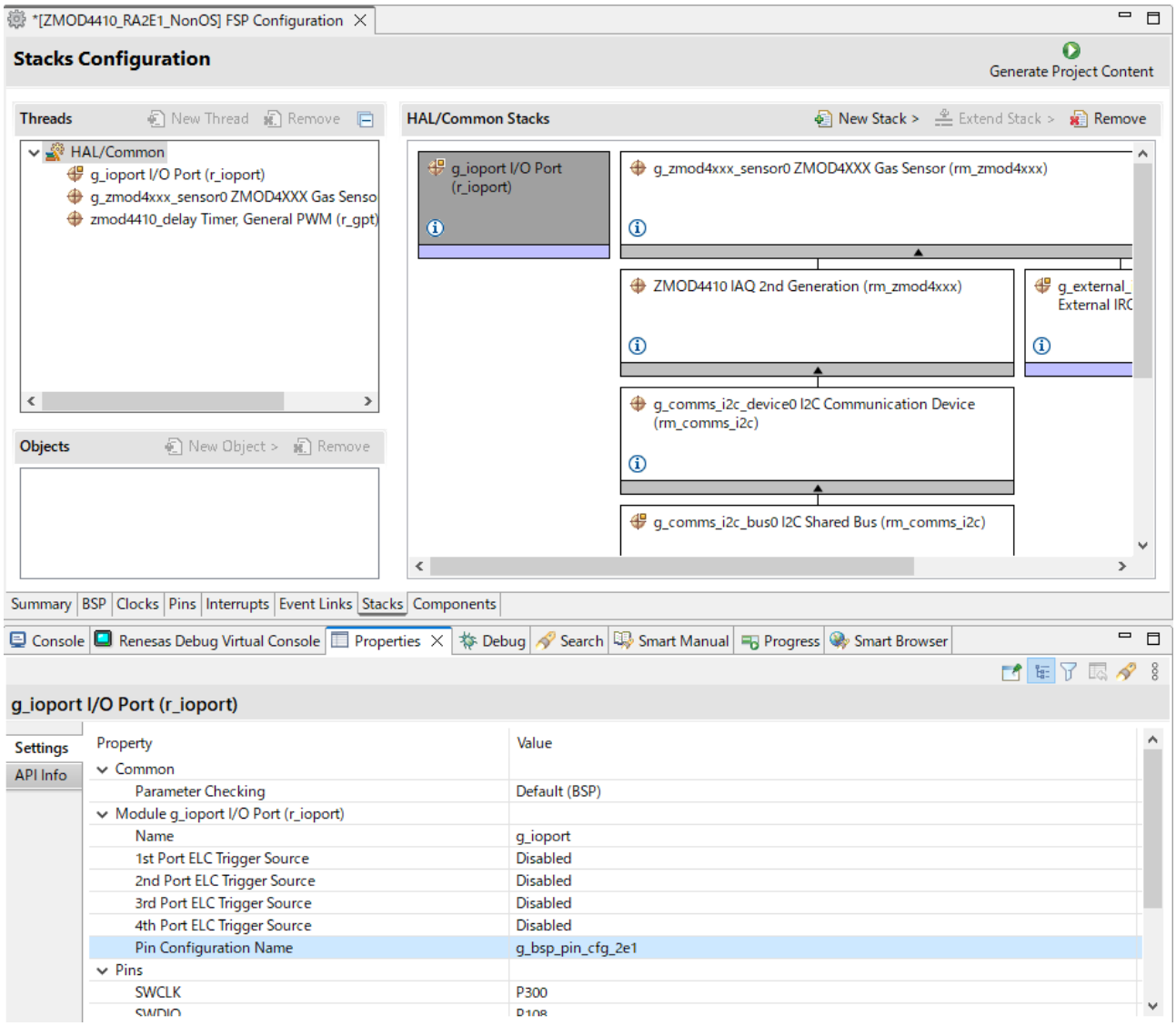
g_i2c0 I2C Master Driver on r_sci_i2c

Settings | API Info

Property	Value
Common	
Parameter Checking	Default (BSP)
DTC on Transmission and Reception	Disabled
10-bit slave addressing	Disabled
Module g_i2c0 I2C Master Driver on r_sci_i2c	
Name	g_i2c0
Channel	2
Slave Address	0x00
Address Mode	7-Bit
Rate	Standard
SDA Output Delay (nano seconds)	300
Noise filter setting	Use clock signal divided by 1 with noise filter
Bit Rate Modulation	Enable
Callback	rm_comms_i2c_callback
Interrupt Priority Level	Priority 2
RX Interrupt Priority Level [Only used when DTC is enabled]	Disabled
Pins	
SDA	P302
SCL	P301

Enter the pin configuration name to use in "Pin Configuration Name" of "g_ioport I/O Port".

In our example, it is ”g_bsp_pin_cfg_2e1”.

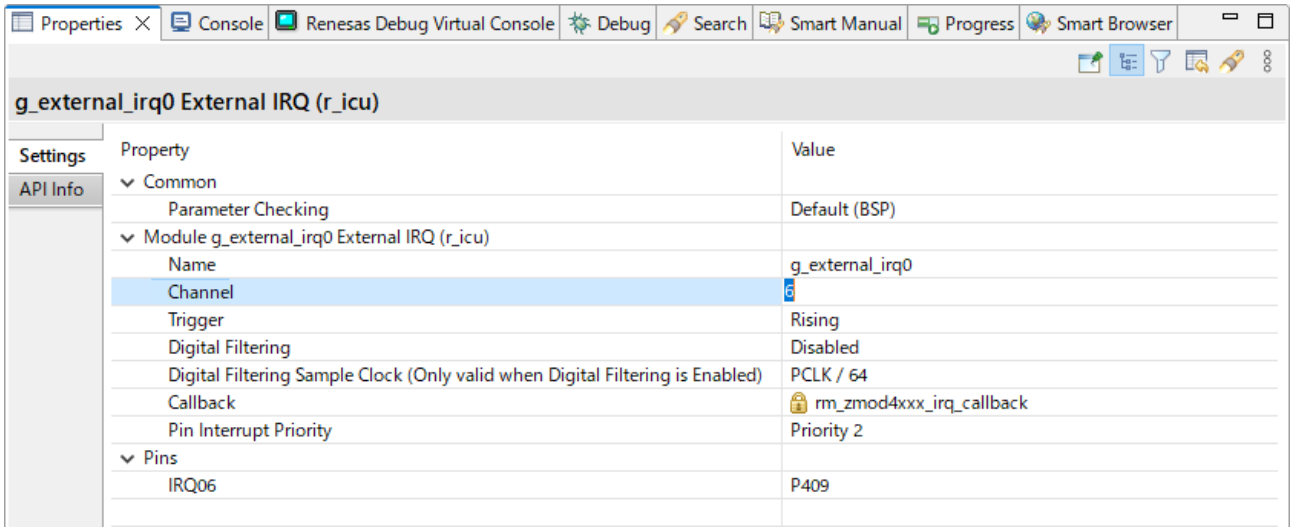


Modify the settings of r_icu according to the specifications of the target board.

IRQ6 is assigned to PMOD2 on the EK-RA2E1 board.

To use PMOD2, set "Channel" to "6".

To use PMOD1(OptionType6A), IRQ cannot be used. Refer to section 8.1.4, Changing when not using IRQ.



If an error is displayed in other stacks, modify the specified item according to the displayed error.

8.1.3 Changing sample code

Open "hal_entry.c" (Non-OS) or "(sensor_name)_sensor_thread_entry.c" (FreeRTOS, Azure) and change the write process when resetting the sensor.

Modify RESET pin designation according to the specifications of the target board.

P101 is assigned to PMOD1(OptionType6A) and P400 to PMOD2 on the EK-RA2E1 board.

To use PMOD1(OptionType6A), specify P101. To use PMOD2, specify P400.

```
/* Reset ZMOD sensor (active low). Please change to the IO port connected to the RES_N pin of the ZMOD sensor on the customer board. */
R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_04_PIN_00, BSP_IO_LEVEL_HIGH);
R_BSP_SoftwareDelay(10, BSP_DELAY_UNITS_MILLISECONDS);
R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_04_PIN_00, BSP_IO_LEVEL_LOW);
R_BSP_SoftwareDelay(10, BSP_DELAY_UNITS_MILLISECONDS);
R_IOPORT_PinWrite(&g_ioport_ctrl, BSP_IO_PORT_04_PIN_00, BSP_IO_LEVEL_HIGH);
R_BSP_SoftwareDelay(10, BSP_DELAY_UNITS_MILLISECONDS);
```

Press [Generate Project Content] to generate files.

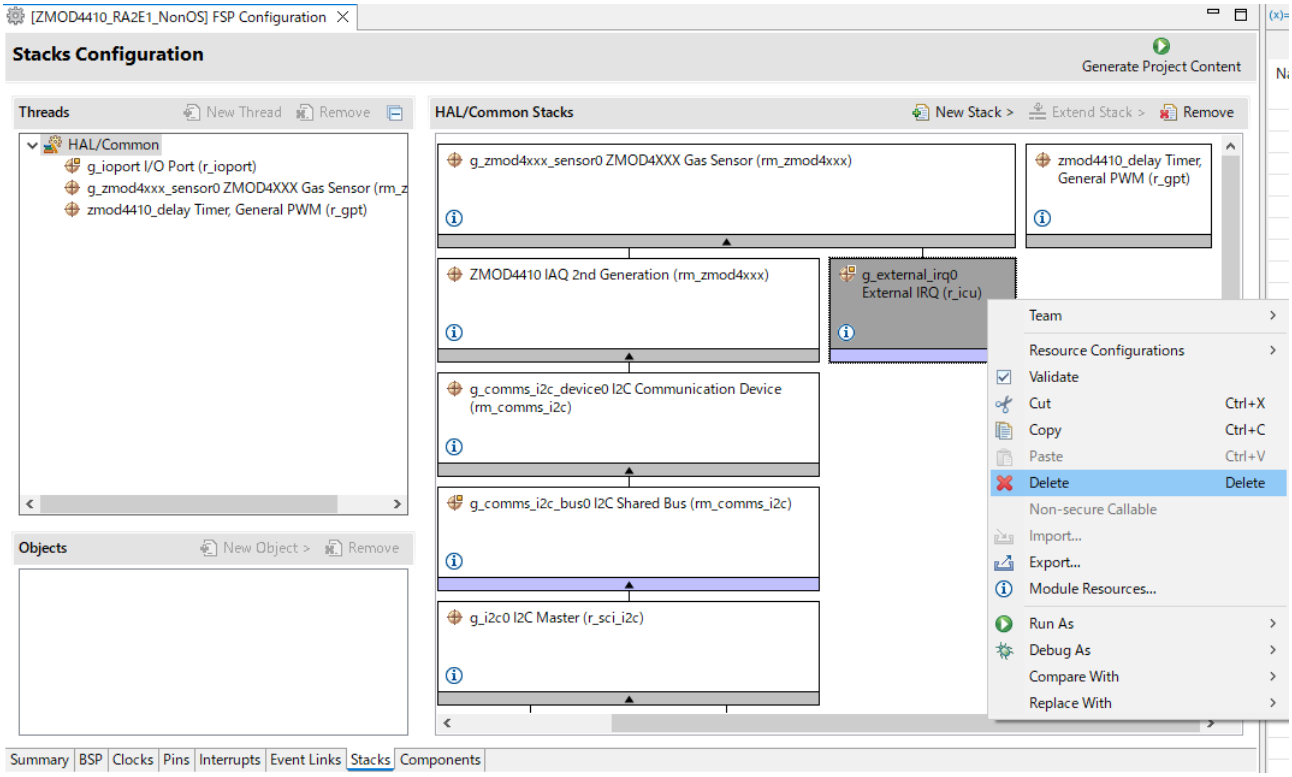
Build the project.

Select [Debug Configurations] from the menu and modify the debugger settings according to the specifications of the emulator to be connected to the target board.

8.1.4 Changing when not using IRQ

If IRQ is not used, delete stack, and change the values of constants defined.

In the "Stacks" tabbed page, Delete “g_external_irq0 External IRQ”.



Open "RA_(sensor_name).c" (Non-OS) or "(sensor_name)_sensor_thread_entry.c" (FreeRTOS, Azure) and change the value of “G_ZMOD4XXX_SENSOR0_IRQ_ENABLE” to “0”.

```
/* TODO: Enable if you want to open ZMOD4XXX */
#define G_ZMOD4XXX_SENSOR0_IRQ_ENABLE   (0)
```

8.1.5 Changing toolchain setting

If you want to use a toolchain other than the GCC ARM Embedded toolchain, copy RA_ZMOD4410.c and RA_ZMOD4510.c (Non-OS) or zmod4410_sensor_thread_entry.c, zmod4510_sensor_thread_entry.c, sensor_thread_common.c, and sensor_thread_common.c (FreeRTOS, Azure) from this project to create a new project.

8.2 RX Sample Project

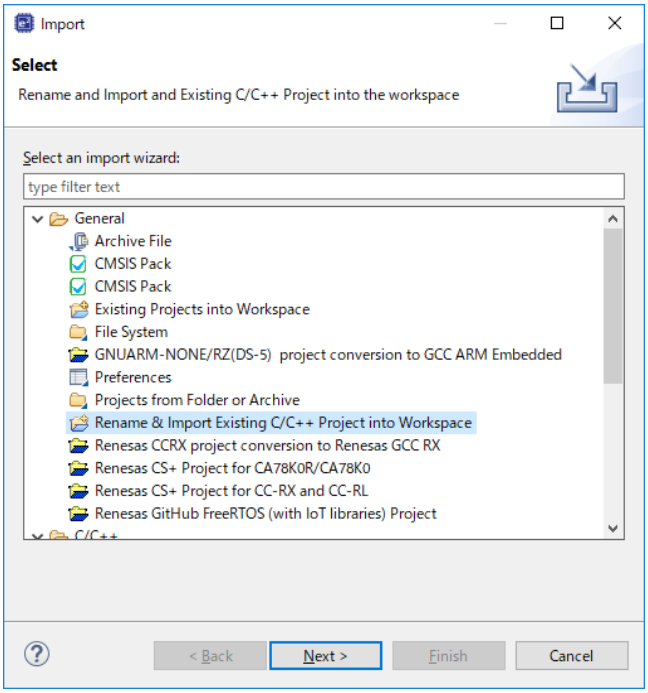
Use the following procedures to modify a sample project.

This section describes an example of modifying the sample project "ZMOD4410_RX65N_NonOS" so that it can be used on the RSKRX231 board.

8.2.1 Importing the Sample Project

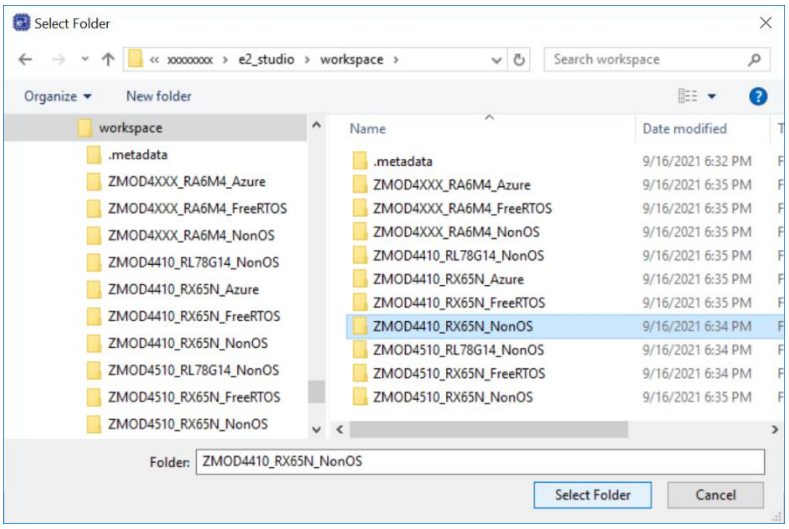
Select [Import] from the menu.

The "Import" window will appear. Select "Rename & Import Existing C/C++ Project into Workspace" in the window and press the [Next] button.

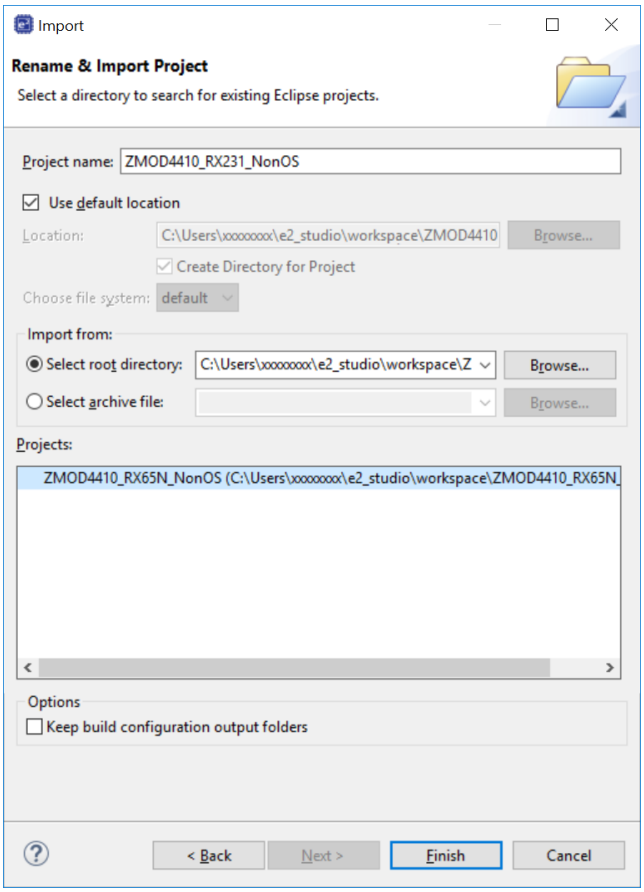


Press the [Browse] button to open the "Select Folder" window.

Select the folder of the original project for the current device from a list of imported sample projects and press the [Select Folder] button.

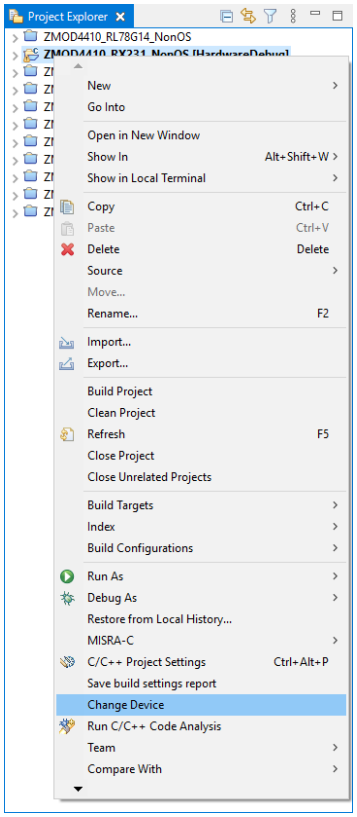


Enter the project name, select the original project for the current device, and press the [Finish] button.

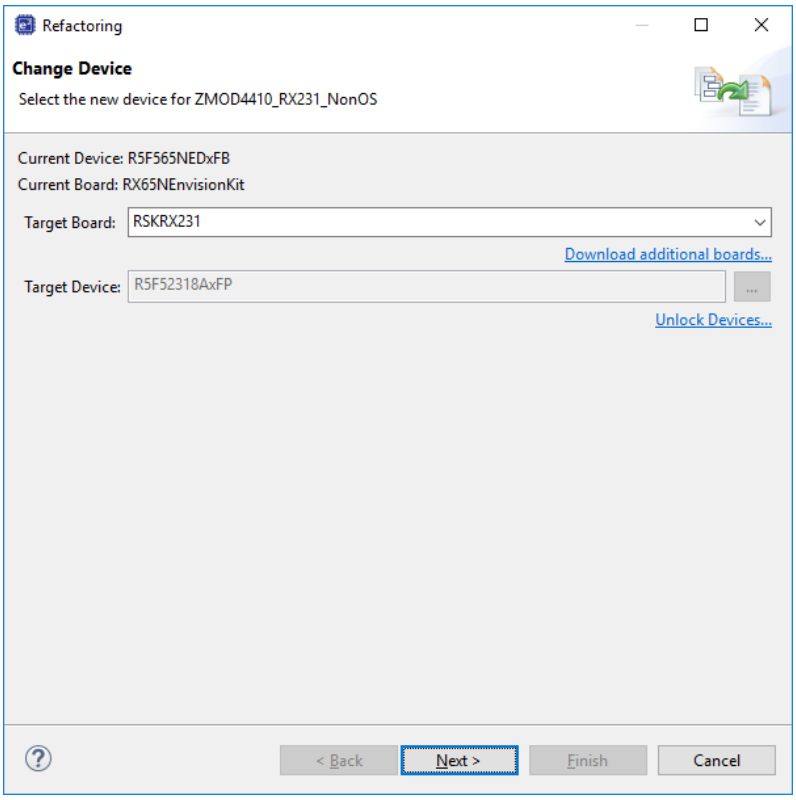


8.2.2 Changing the Device

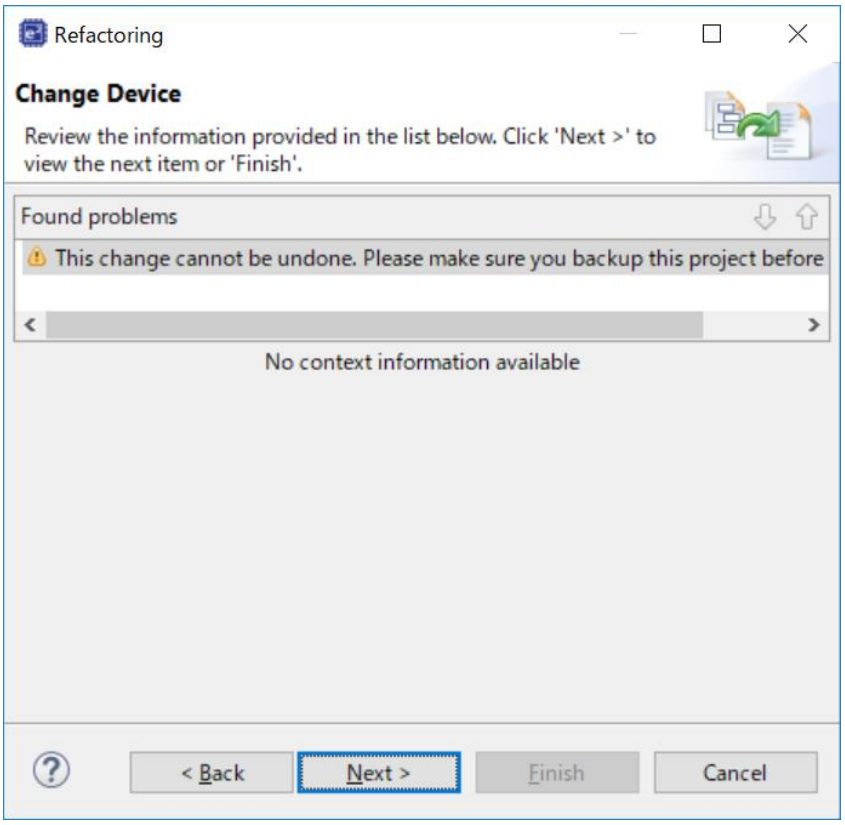
Select the imported project from the project tree and right-click on int to open the context menu. Select "Change Device" from the menu.



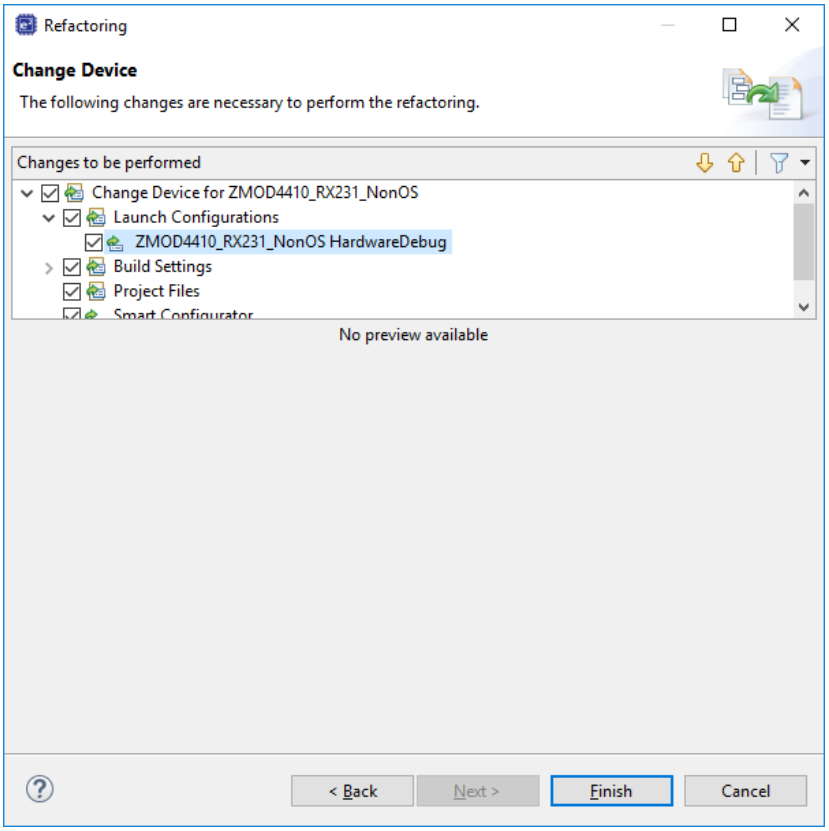
Select a desired board or device in the "Change Device" window and press the [Next] button.



If a warning message appears, read it and check if there is a problem in proceeding with the procedure. Press [Next] to move to the next step.

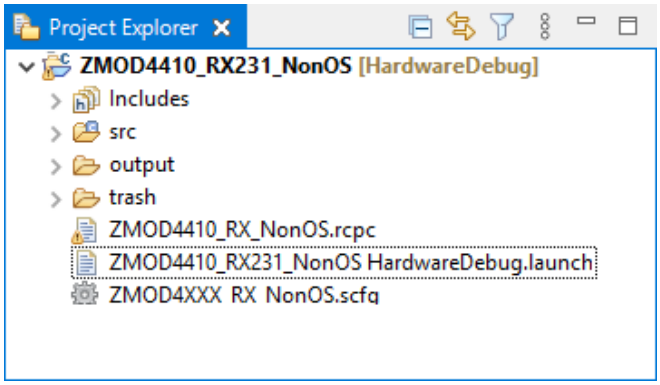


The changes you have made in the settings will be displayed. Press the [Finish] button to apply the changes to the project.

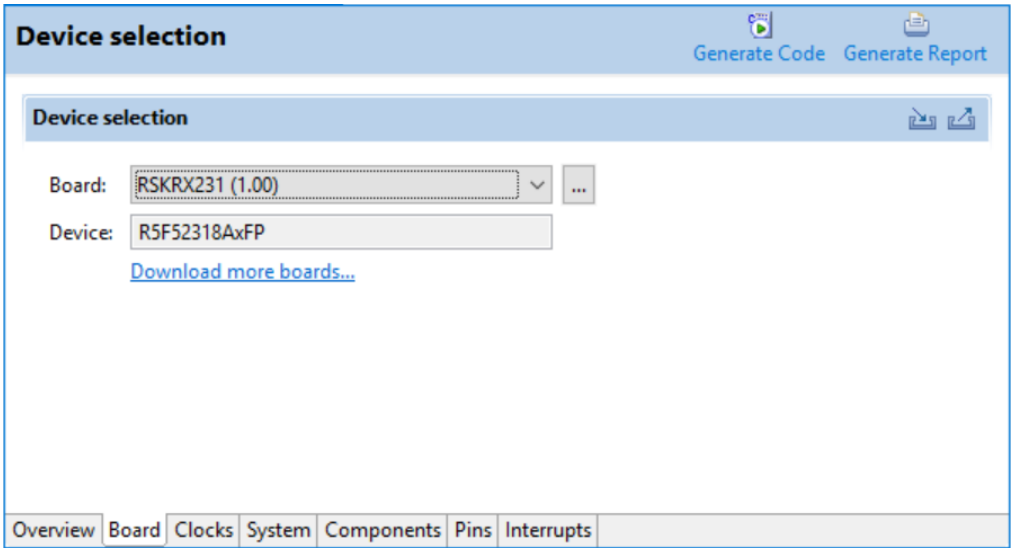


8.2.3 Modifying Settings of the Smart Configurator

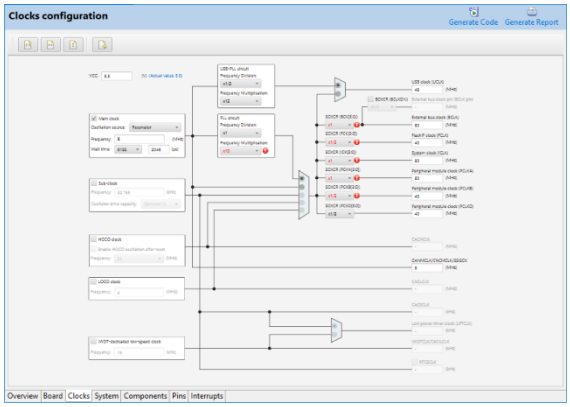
On the project tree, double-click on the .scfg file of the imported project in which the target device has been changed; the Smart Configurator window will open.

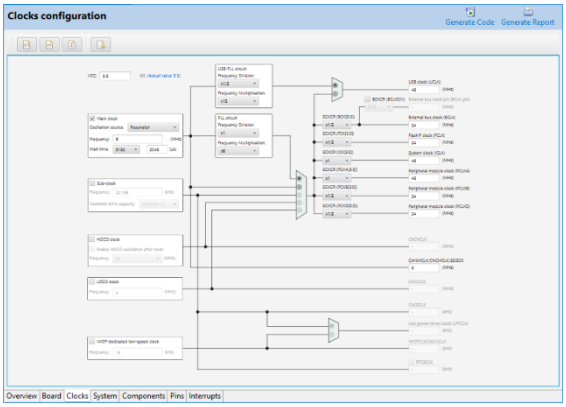


Select the "Board" tabbed page to check that the board and device have been changed correctly.



Set up the clocks in the "Clocks" tabbed page according to the specifications of the target board to be used.

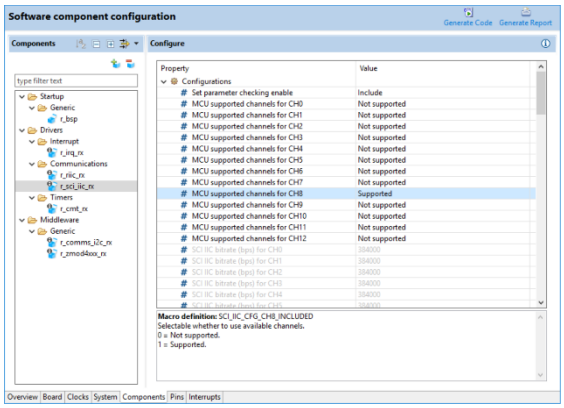


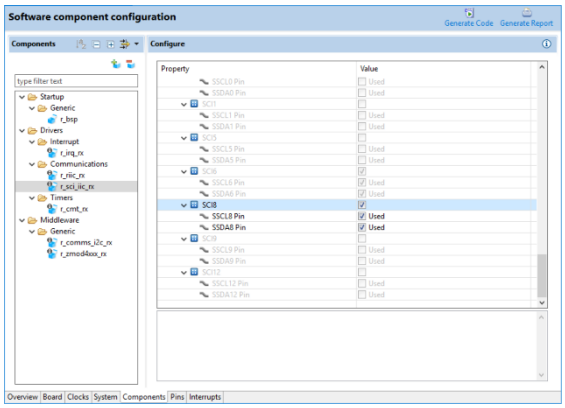


Modify the settings of individual components in the "Components" tabbed page according to the specifications of the target board.

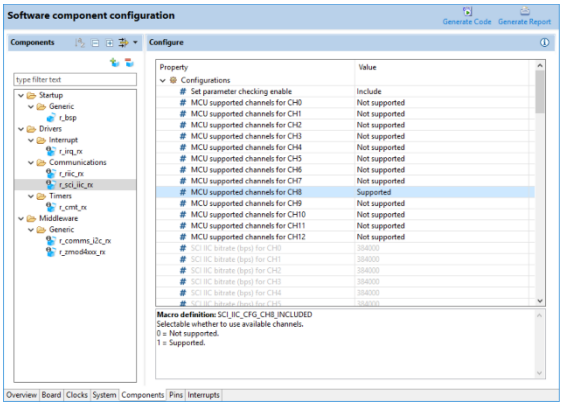
As SCI8 is assigned to PMOD on the RSK RX231 board, change the setting of "MCU supported channels for CH2" to "Not supported" and "MCU supported channels for CH8" to "Supported" in r_sci_iic_rx.

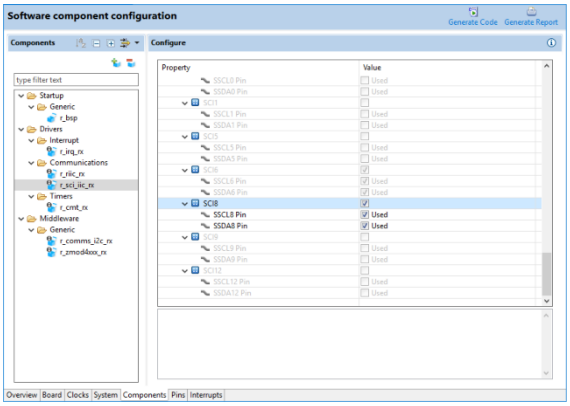
Check the settings of "SSCL8 Pin" and "SSDA8 Pin" for "SCI8" under "Resources".



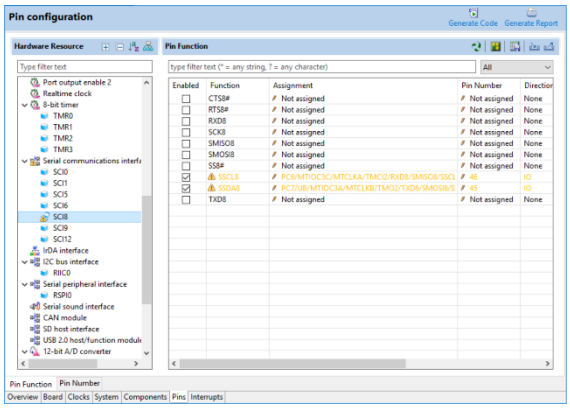


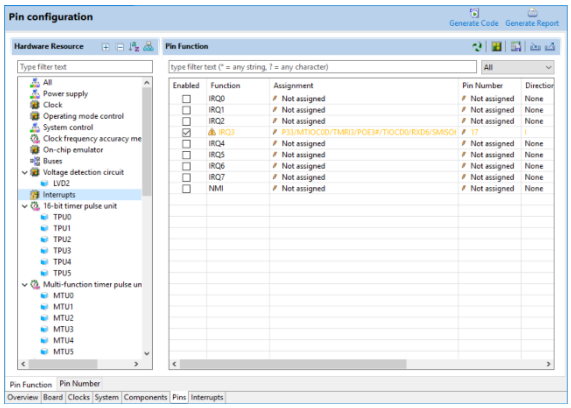
And IRQ3 is assigned to PMOD on the RSK RX231 board, check the settings of “IRQ3 Pin” for “Interrupts” under “Resources” in r_irq_rx and change the settings of “IRQ number for ZMOD4XXX sensor device0” to “IRQ3” in r_zmod4xxx_rx.





Open the "Pins" tabbed page and check that functions are assigned to the SCI8 pins and IRQ3 pins in the "Pin function" panel.

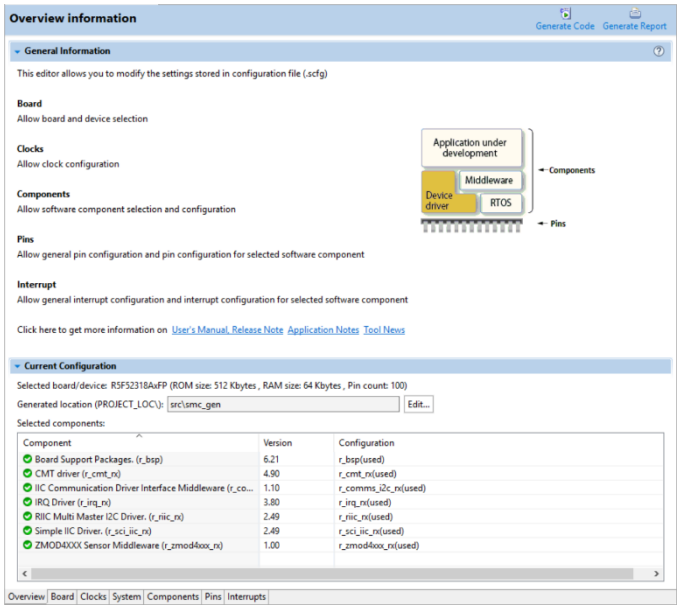




As the use of PMOD Type 2A (extended SPI) is specified in the RSK RX231 board information, a warning message will appear when I2C is used, but this does not produce any problems.

To connect a sensor board, a board for converting PMOD Type 2A to PMOD Type 6A is necessary.

Press the [Generate Code] icon to generate code.



Build the project.

Select [Debug Configurations] from the menu and modify the debugger settings according to the specifications of the emulator to be connected to the target board.

8.2.4 Changing toolchain setting

If you want to use a toolchain other than the CC-RX toolchain, copy RX_ZMOD4410.c and RX_ZMOD4510 (Non-OS), or main.c and zmod4410_sensor_thread_entry.c and zmod4510_sensor_thread_entry.c (FreeRTOS), or zmod4410_sensor_thread_entry.c, zmod4510_sensor_thread_entry.c, sensor_thread_common.c, and sensor_thread_common.c (Azure) from this project to create a new project.

8.3 RL78 Sample Project

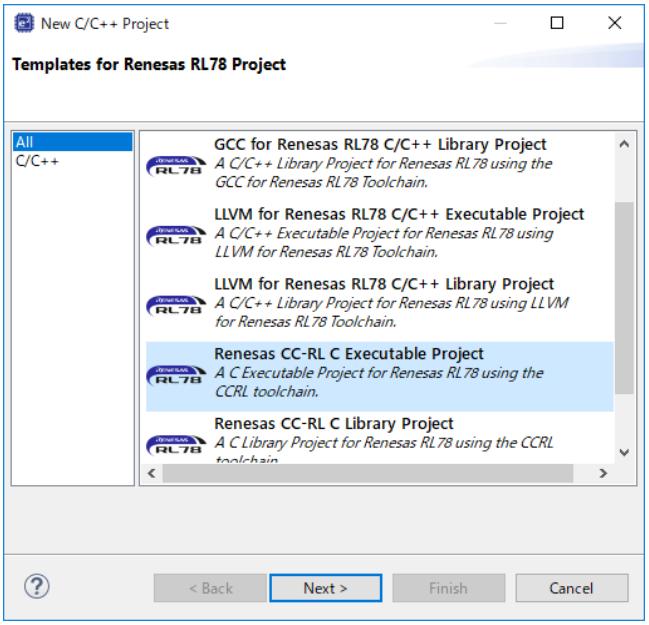
Changing the target device within the RL78 family requires creating a new project.

This section describes an example of creating a new project that can be used on a RL78/G1A custom board.

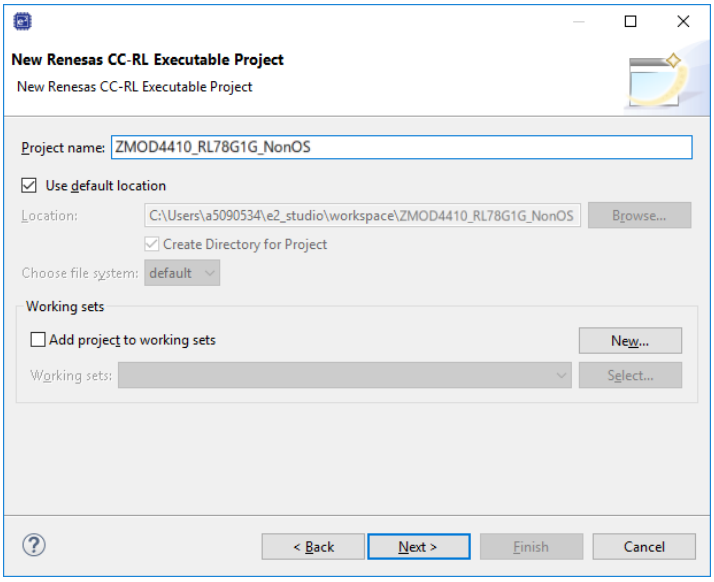
8.3.1 Creating a New Project

Select [File] → [New] → [Renesas C/C++ project] → [Renesas RL78] from the menu.

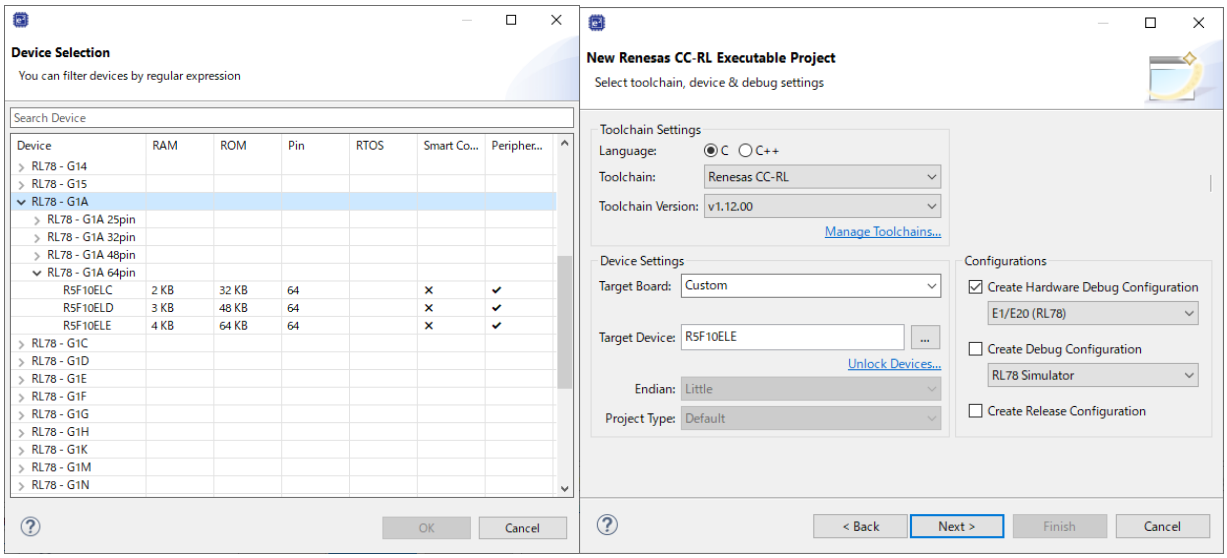
Select the template "Renesas CC-RL C Executable Project" and press the [Next] button.



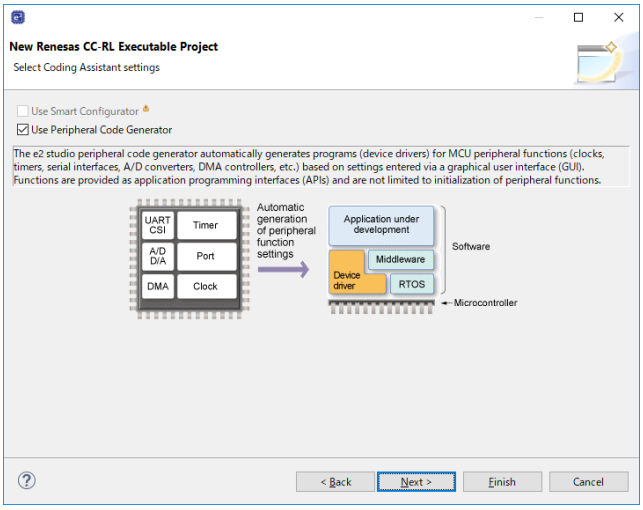
Enter the project name (example: "ZMOD4410_RL78G1A_NonOS") and press the [Next] button.



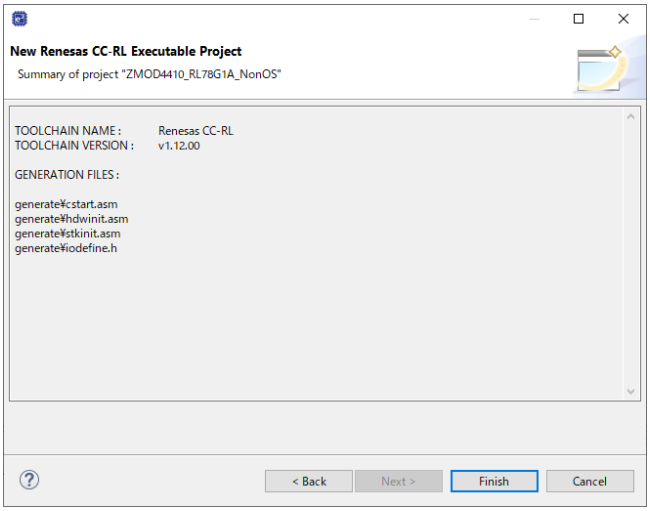
Change "Target Device" to a desired device and press the [Next] button.



Select the checkbox for "Use Peripheral Code Generator" and press the [Next] button.

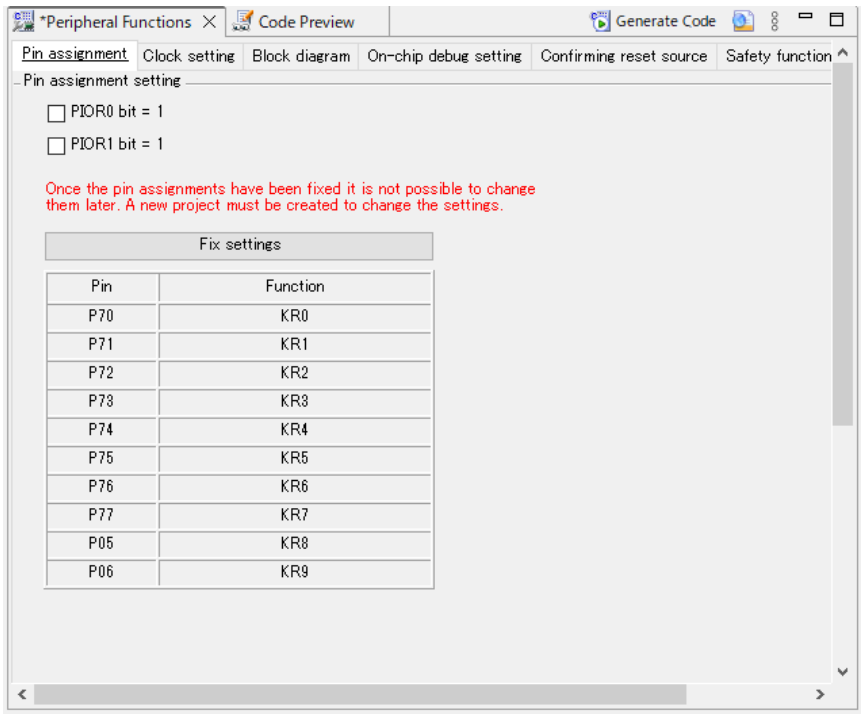


Press the [Finish] button.

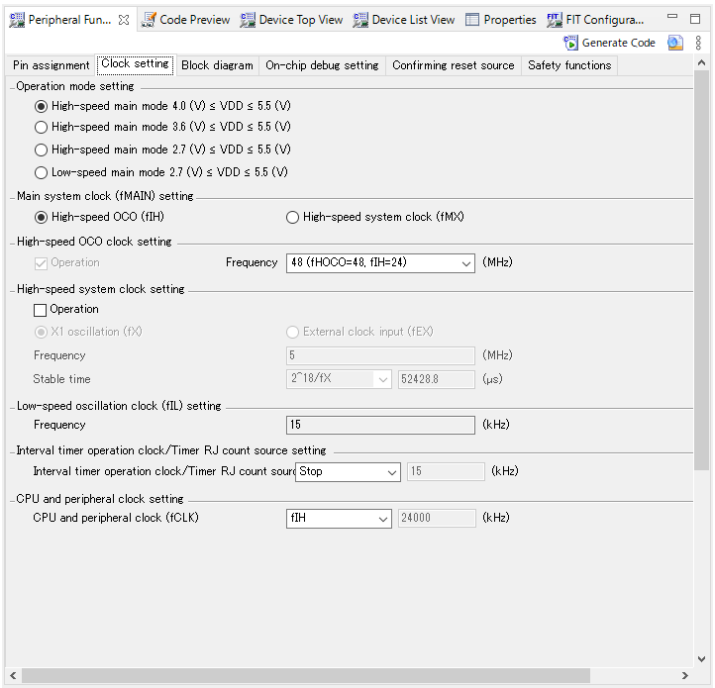


8.3.2 Settings of the Code Generator

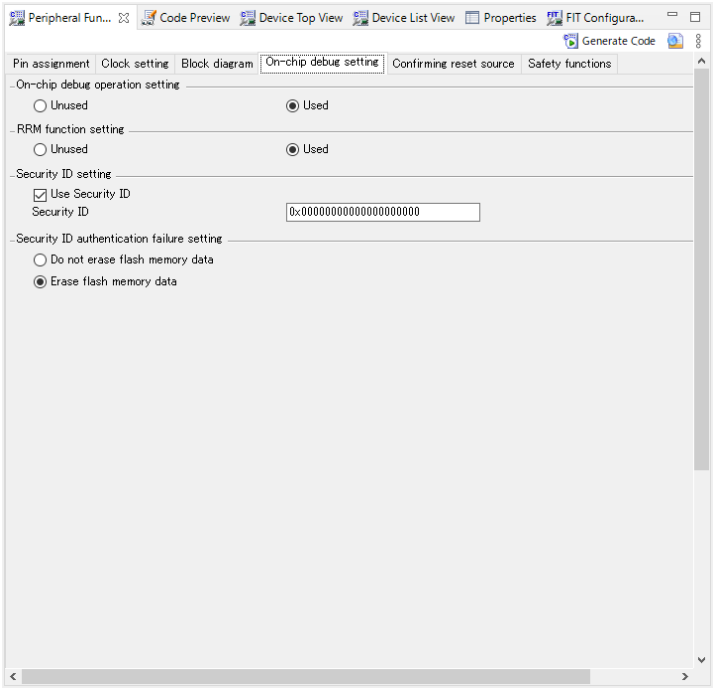
Modify the pin assignment in the "Pin assignment" tabbed page for "Common/Clock Generator" according to the specifications of the target board to be used.



Modify the clock settings in the "Clock setting" tabbed page for "Common/Clock Generator" according to the specifications of the target board.

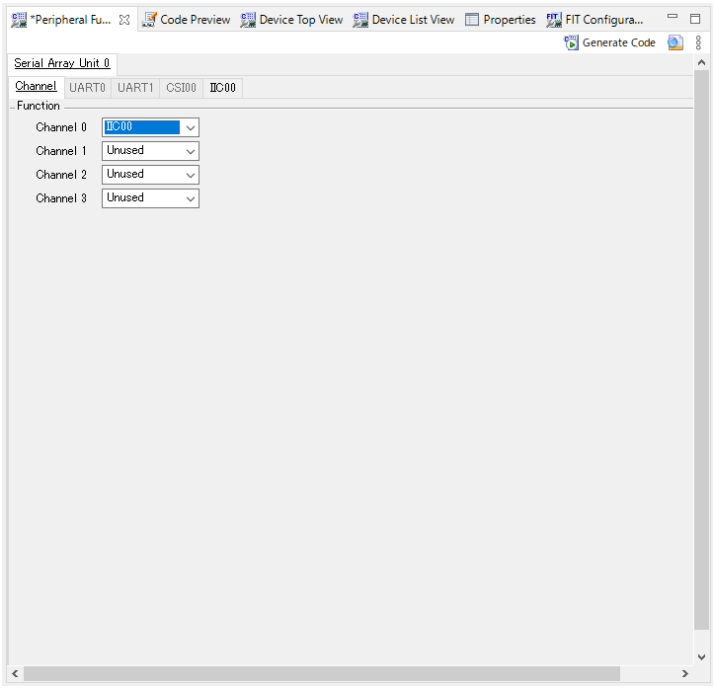


Select "Used" for "On-chip debug operation setting" in the "On-chip debug setting" tabbed page for "Common/Clock Generator".

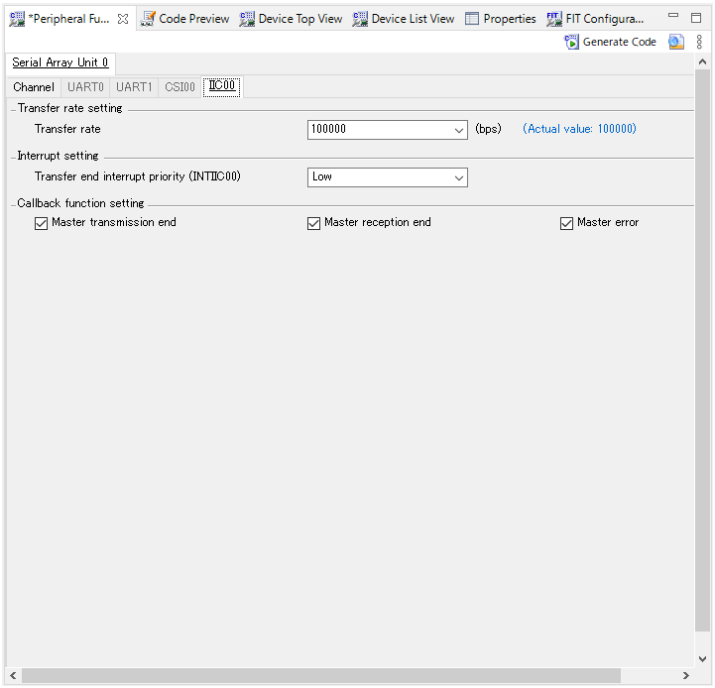


To use the serial array unit, set the channel assigned to PMOD on the target board to "IICxx" in the "Serial Array Unit" or "Serial" tabbed page.

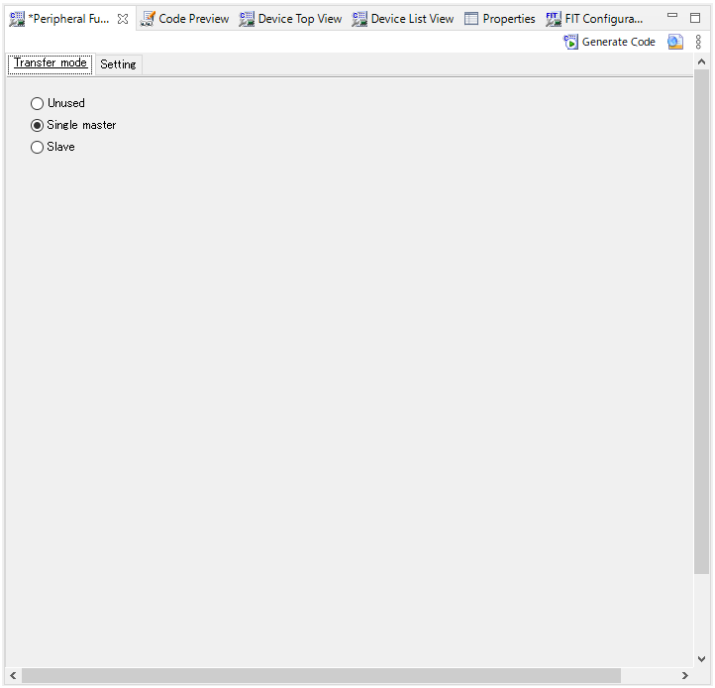
Note: The corresponding pin must be selected as N-ch by "Port".



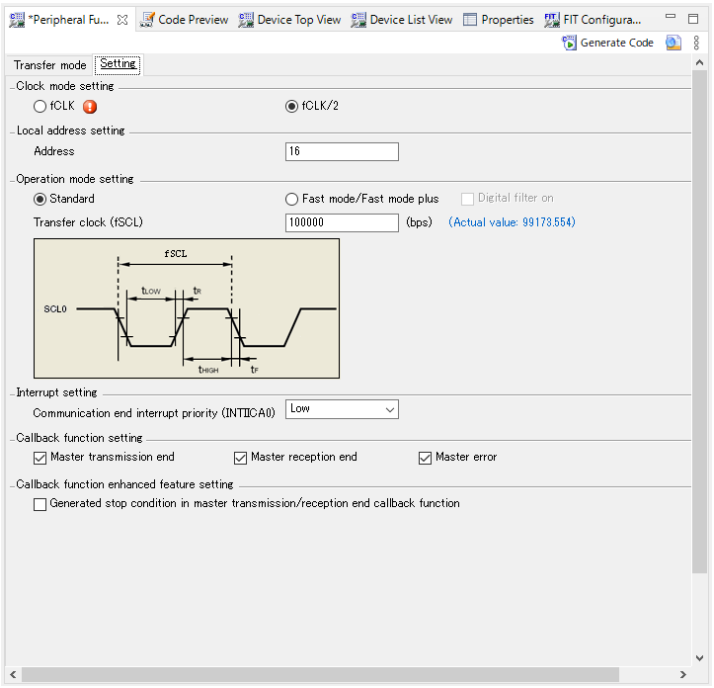
In the tabbed page for IICxx enabled in the serial array unit, set "Transfer rate" to 400000 or 100000, set "Transfer end interrupt priority" to a desired value, and enable all functions under "Callback function setting".
Note : When using a serial array unit, the Nch open drain of the pin to be used is set automatically. If an error icon on the port was displayed, open the Ports tab and check the port settings.



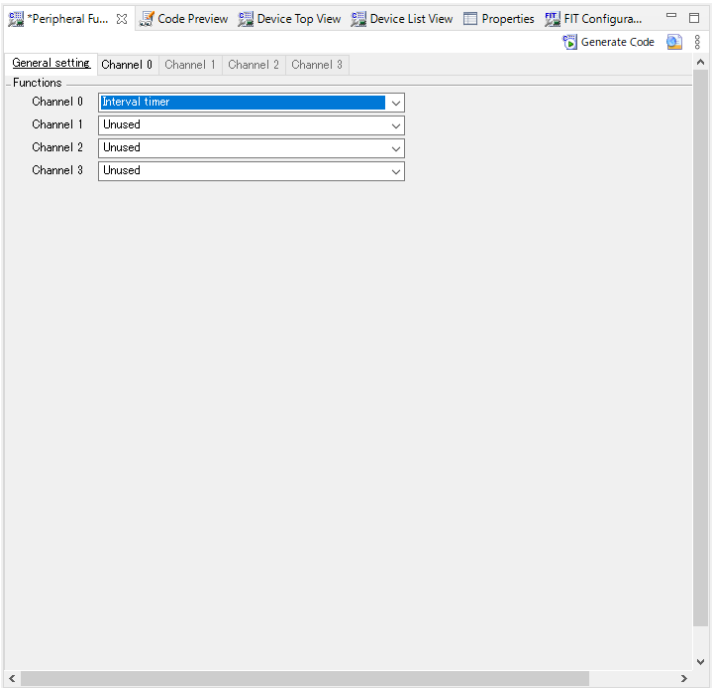
To use the serial interface IICA, select "Single master" in the "Transfer mode" tabbed page for the channel assigned to PMOD on the target board in the "Serial Interface IICA" or "Serial" setting window.



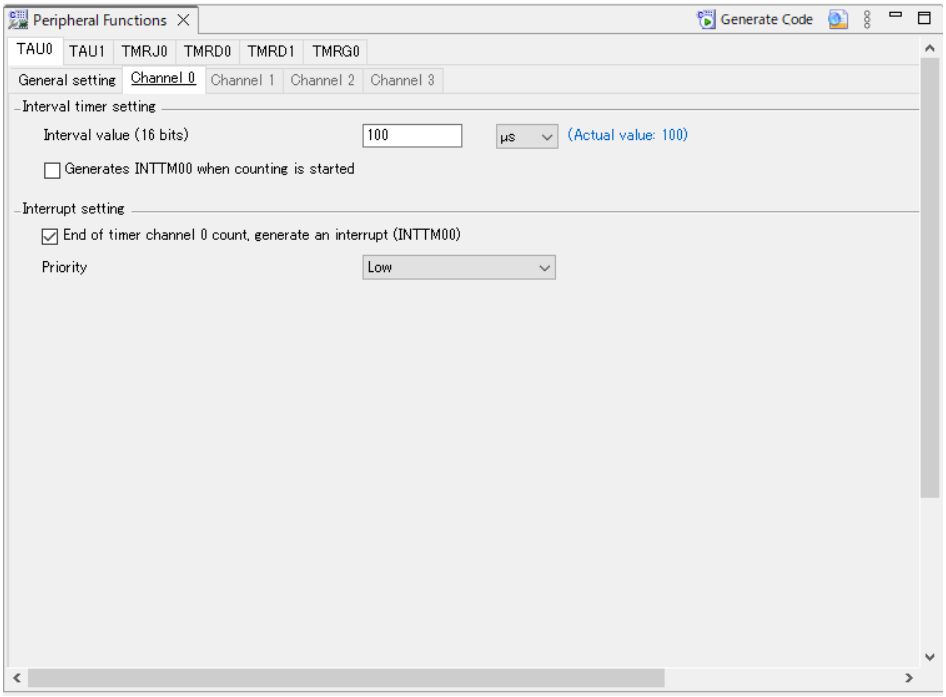
In the "Setting" tabbed page for the channel set to the single master, set "Operation mode setting" to either a combination of "Fast mode" and "400000" or a combination of "Standard" and 100000, set the interrupt priority to a desired level, enable all functions under "Callback function setting", and disable "Callback function enhanced feature setting".



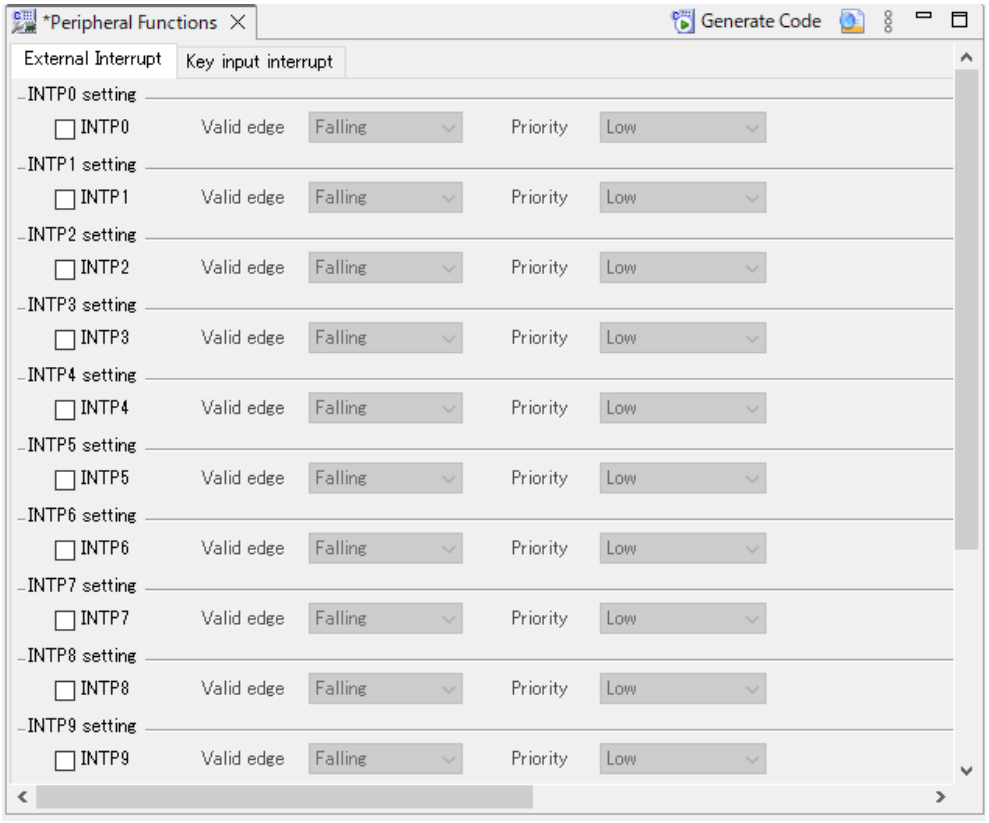
In the "General setting" tabbed page for a desired channel of the timer array unit or a desired TAU of the timer, select "Interval timer" under "Functions".



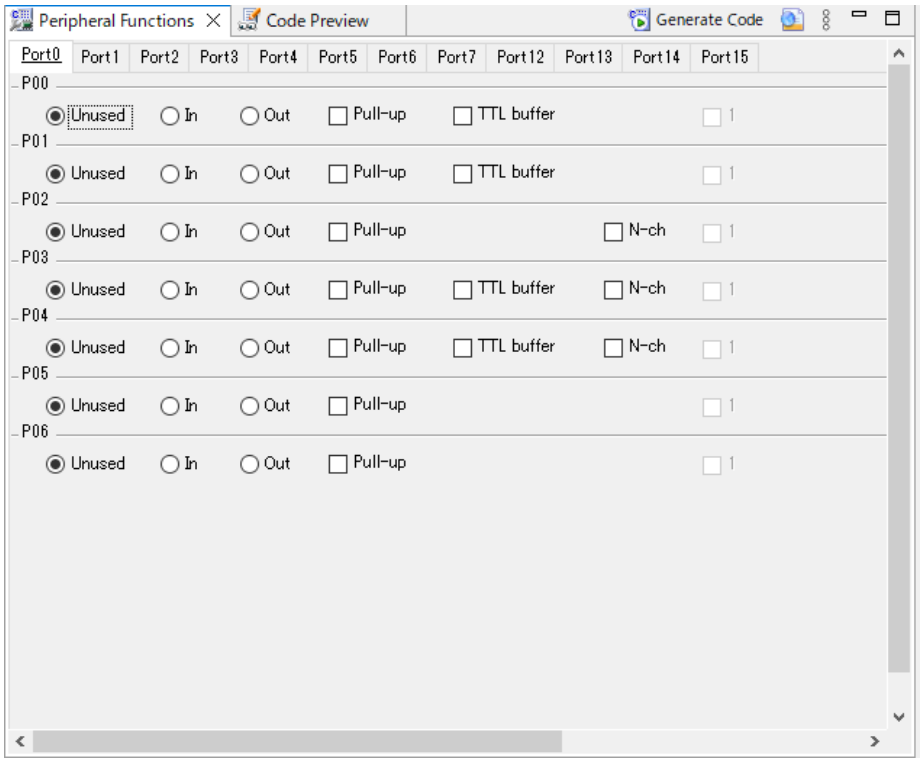
In the page for the channel set to the interval timer, set "Interval value" to "100 µs", enable timer interrupts, and set the interrupt priority to a desired level.



Set IRQ pin on PMOD in the "External Interrupt" tabbed page for "Interrupt" according to the specifications of the target board.



Set RESET pin on PMOD in the "Port (x)" tabbed page for "Port" according to the specifications of the target board.



Press the [Code Generate] button to generate code.

8.3.3 Modifying the Generated Code

Perhaps Code Generator output destination different from this sample software, because Code Generator version differs depending on the MCU used.

Open r_cg_sau_user.c, r_cg_iica_user.c, or r_cg_serial_user.c and add the following code.

Definition for including r_comms_i2c_if.h:

```
/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_sau.h"
/* Start user code for include. Do not edit comment generated here */
#include "r_comms_i2c_if.h"
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"
```

Addition of the rm_comms_i2c_bus0_callback() function to the callback function:

Specify the "false" parameter for the transmission and reception end callback functions and the "true" parameter for the error callback function.

```
/***********************************************************************
* Function Name: r_iic00_callback_master_error
* Description  : This function is a callback function when IIC00 master
err
* Arguments    : flag -
*                    status flag
* Return Value : None
***********************************************************************/
static void r_iic00_callback_master_error(MD_STATUS flag)
{
    /* Start user code. Do not edit comment generated here */
    rm_comms_i2c_bus0_callback(true);
    /* End user code. Do not edit comment generated here */
}
/***********************************************************************
* Function Name: r_iic00_callback_master_receiveend
* Description  : This function is a callback function when IIC00 finishes
* Arguments    : None
* Return Value : None
***********************************************************************/
static void r_iic00_callback_master_receiveend(void)
{
    /* Start user code. Do not edit comment generated here */
    rm_comms_i2c_bus0_callback(false);
    /* End user code. Do not edit comment generated here */
}
/***********************************************************************
* Function Name: r_iic00_callback_master_sendend
* Description  : This function is a callback function when IIC00 finishes
* Arguments    : None
* Return Value : None
***********************************************************************/
static void r_iic00_callback_master_sendend(void)
{
    /* Start user code. Do not edit comment generated here */
    rm_comms_i2c_bus0_callback(false);
    /* End user code. Do not edit comment generated here */
}
```

Open t_cg_tau_user.c or r_cg_timer_user.c and add the following code.

Declaration of external for the (sensor_name)_delay_callback() function:

```
/***********************************************************************
Global variables and functions
***********************************************************************/
/* Start user code for global. Do not edit comment generated here */
extern void zmod4410_delay_callback(void);
/* End user code. Do not edit comment generated here */
```

Addition of the call of the (sensor_name)_delay_callback() function to the timer interrupt callback function:

```
/***********************************************************************
* Function Name: r_tau0_channel0_interrupt
* Description  : This function INTTM00 interrupt service routine.
* Arguments    : None
* Return Value : None
***********************************************************************/
static void __near r_tau0_channel0_interrupt(void)
{
    /* Start user code. Do not edit comment generated here */
    zmod4410_delay_callback();
    /* End user code. Do not edit comment generated here */
}
```

Open t_cg_tau.c or r_cg_timer.c and add the following code.

Define the R_TAU0_Channel0_Reset() function in the user code description part:

```
void R_TAU0_Channel0_Reset(void)
{
      /* function not supported by this module */
}
```

Open t_cg_tau.h or r_cg_timer.h and add the following code.

Declaration of prototype for the R_TAU0_Channel0_Reset() function:

```
/***********************************************************************
Global functions
***********************************************************************/
void R_TAU0_Create(void);
void R_TAU0_Channel0_Start(void);
void R_TAU0_Channel0_Stop(void);
/* Start user code for function. Do not edit comment generated here */
void R_TAU0_Channel0_Reset(void);
/* End user code. Do not edit comment generated here */
```

Open r_cg_main.c or r_main.c and add the following code.

Definition for including r_bsp_common.h:

```
/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_cgc.h"
#include "r_cg_port.h"
#include "r_cg_tau.h"
#include "r_cg_sau.h"
/* Start user code for include. Do not edit comment generated here */
#include "r_bsp_common.h"
/* End user code. Do not edit comment generated here */
#include "r_cg_userdefine.h"
```

Declaration of prototype for each function:

```
/***********************************************************************
Global variables and functions
***********************************************************************/
/* Start user code for global. Do not edit comment generated here */
void g_comms_i2c_bus0_quick_setup(void);
void demo_err(void);

void g_zmod4xxx_sensor0_quick_setup(void);
void start_zmod4410_demo(void);
/* End user code. Do not edit comment generated here */
```

Addition of the following code to the main() function:

Modify RESET pin designation according to the target board to be used.

```
    /* Open the Bus */
    g_comms_i2c_bus0_quick_setup();

    /* Reset ZMOD sensor (active low). Please change to the IO port
    connected
     to the RES_N pin of the ZMOD sensor on the customer board. */
    P5 = _00_Pn7_OUTPUT_0 | _00_Pn6_OUTPUT_0 | _00_Pn5_OUTPUT_0 |
         _00_Pn4_OUTPUT_0 | _00_Pn3_OUTPUT_0 | _00_Pn2_OUTPUT_0 |
         _02_Pn1_OUTPUT_1 | _00_Pn0_OUTPUT_0;
    R_BSP_SoftwareDelay(10, BSP_DELAY_MILLISECS);
    P5 = _00_Pn7_OUTPUT_0 | _00_Pn6_OUTPUT_0 | _00_Pn5_OUTPUT_0 |
         _00_Pn4_OUTPUT_0 | _00_Pn3_OUTPUT_0 | _00_Pn2_OUTPUT_0 |
         _00_Pn1_OUTPUT_0 | _00_Pn0_OUTPUT_0;
    R_BSP_SoftwareDelay(10, BSP_DELAY_MILLISECS);
    P5 = _00_Pn7_OUTPUT_0 | _00_Pn6_OUTPUT_0 | _00_Pn5_OUTPUT_0 |
         _00_Pn4_OUTPUT_0 | _00_Pn3_OUTPUT_0 | _00_Pn2_OUTPUT_0 |
         _02_Pn1_OUTPUT_1 | _00_Pn0_OUTPUT_0;
    R_BSP_SoftwareDelay(10, BSP_DELAY_MILLISECS);

    /* Open ZMOD4XXX */
    g_zmod4xxx_sensor0_quick_setup();

    while(1U)
    {
        start_zmod4410_demo();
    }
```

Define of the g_comms_i2c_bus0_quick_setup() function and the demo_err() function:

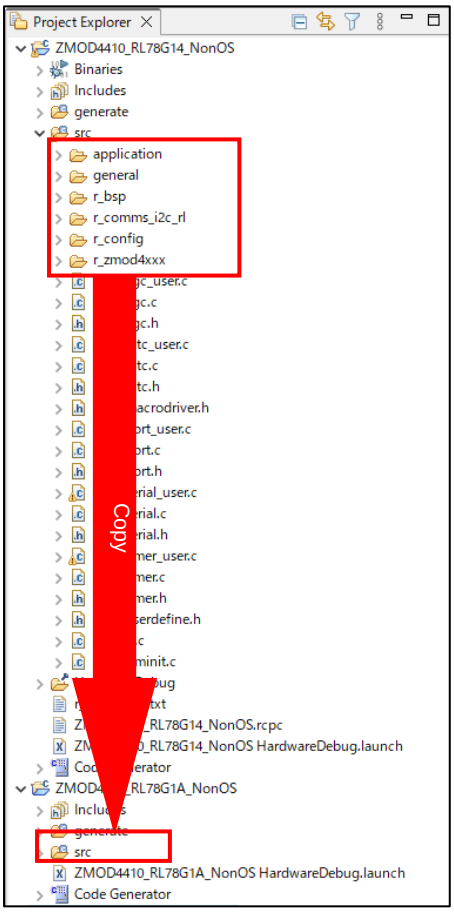
```
void g_comms_i2c_bus0_quick_setup(void)
{
    /* bus has been opened by startup process */
}

void demo_err(void)
{
    while(1)
    {
        // nothing
    }
}
```

8.3.4 Modifying Sample Source Files

Right-click on the "application" "general" "r_bsp" "r_comms_i2c_rl" "r_config" "r_zmod4xxx"folder in the project tree of the sample project "ZMOD4410_RL78G14_NonOS", "ZMOD4450_RL78G14_NonOS" or "ZMOD4510_RL78G14_NonOS" and select "Copy" from the context menu.

Then, right-click on the "src" folder in the newly created project and select "Paste" from the context menu to paste the copied files to the folder.



Open r_comms_i2c_rl_config.h in the “r_config” folder and modify the values of the following definitions.

- COMMS_I2C_CFG_BUSx_DRIVER_TYPE
- COMMS_I2C_CFG_BUSx_DRIVER_CH

When channel 0 of the serial array unit is used:

```
/* SPECIFY DRIVER TYPE, CHANNEL NO. */
/* For Bus No.0 */
#define COMMS_I2C_CFG_BUS0_DRIVER_TYPE          (COMMS_DRIVER_SAU_I2C) /*
Driver type of I2C Bus */
#define COMMS_I2C_CFG_BUS0_DRIVER_CH            (0) /* Channel No. */
```

When channel 0 of the serial interface IICA is used:

```
/* SPECIFY DRIVER TYPE, CHANNEL NO. */
/* For Bus No.0 */
#define COMMS_I2C_CFG_BUS0_DRIVER_TYPE          (COMMS_DRIVER_I2C) /*
Driver type of I2C Bus */
#define COMMS_I2C_CFG_BUS0_DRIVER_CH            (0) /* Channel No. */
```

For the other definitions, refer to section 7, Configuration Settings.

When "serial array unit", "serial interface IICA", or "timer array unit" is used as a peripheral function name in the code generator, modify the sample source code as follows.

src/general/r_smc_entry.h

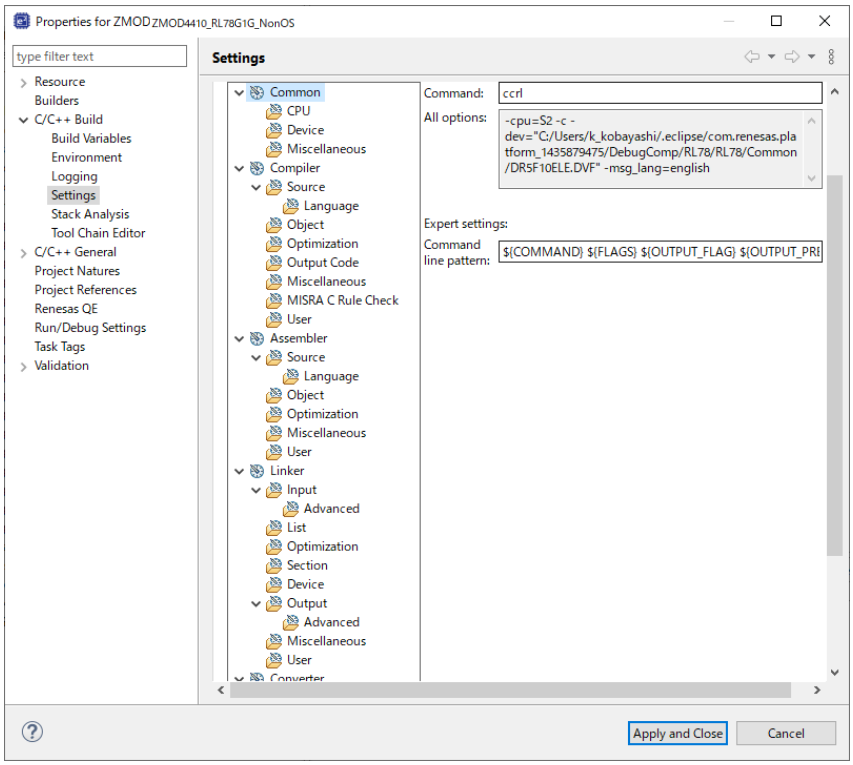
Modify "r_cg_serial.h" to "r_cg_sau.h" or "r_cg_iica.h".

Modify "r_cg_timer.h" to "r_cg_tau.h".

```
/***********************************************************************
Includes
***********************************************************************/
#include "r_cg_macrodriver.h"
#include "r_cg_sau.h"
#include "r_cg_tau.h"
#include "r_cg_port.h"
#include "r_cg_cgc.h"
#include "r_cg_userdefine.h"
```

Open the "Properties" window for the project.

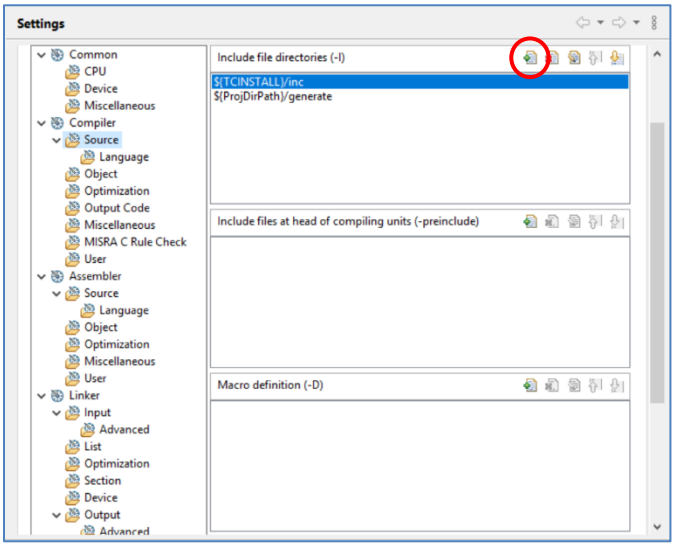
Select [C/C++ Build] → [Settings] in the "Properties" window to open the "Settings" panel.

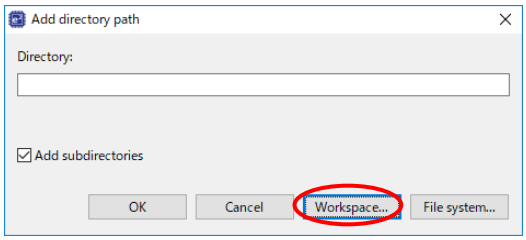


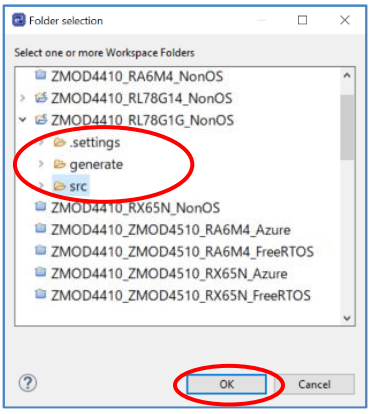
Select [Compiler] → [Source] in the "Tool Settings" tabbed page and press the [Add] icon.

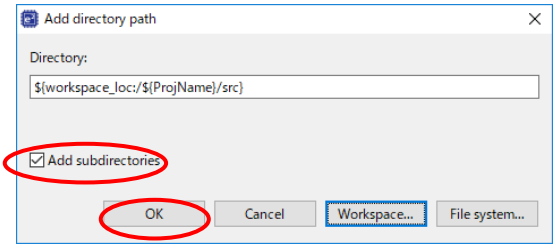
Press the [Workspace] button in the [Add directory path] dialog box and a list of projects will appear. Select the "src" folder for the newly created project in the list and press the [OK] button.

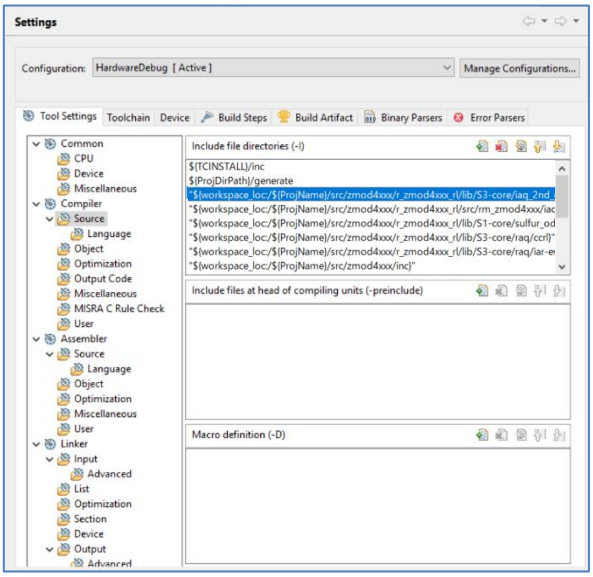
Select the checkbox for "Add subdirectories" and press the [OK] button.





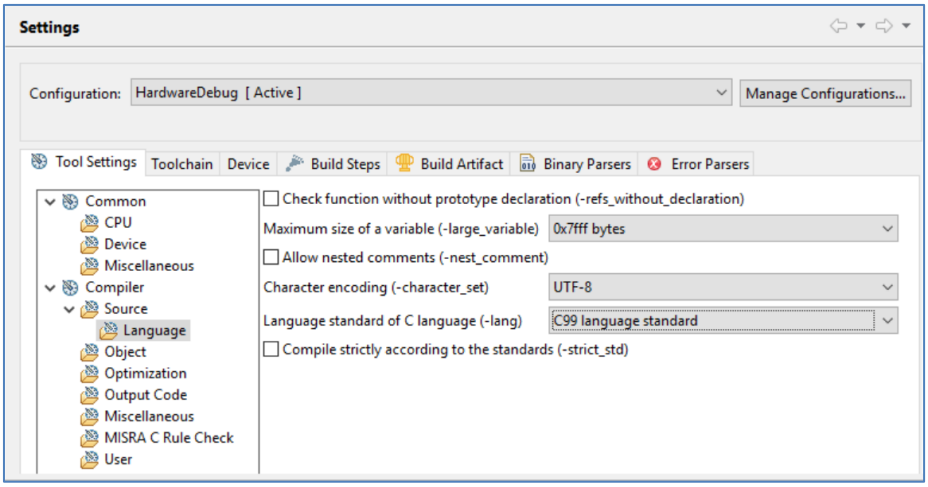






Select [Compiler] → [Source] → [Language] in the "Tool Settings" tabbed page and change the setting of "Language standard of C language" to "C99 language standard".

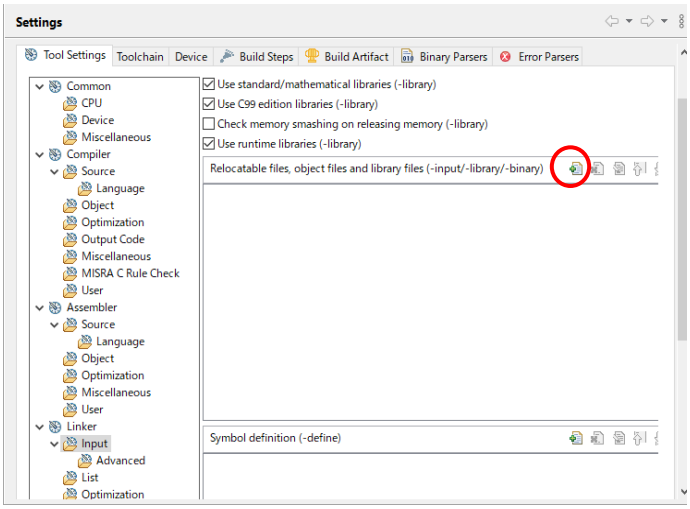
Press the [Apply and Close] button to close the "Properties" window.

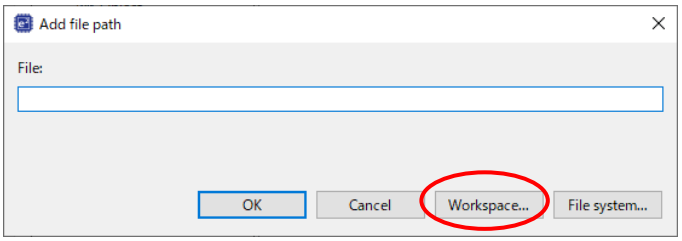


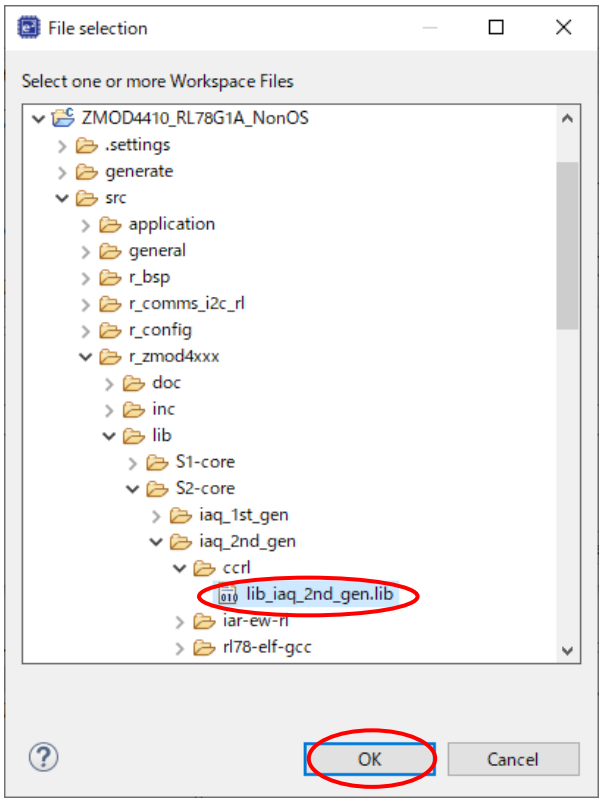
Select [Linker] → [Input] in the "Tool Settings" tabbed page and press the [Add] icon.

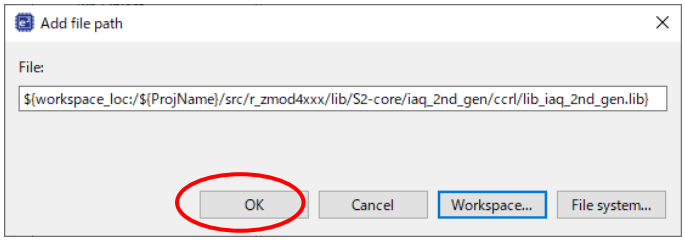
Press the [Workspace] button in the [Add directory path] dialog box and a list of projects will appear.

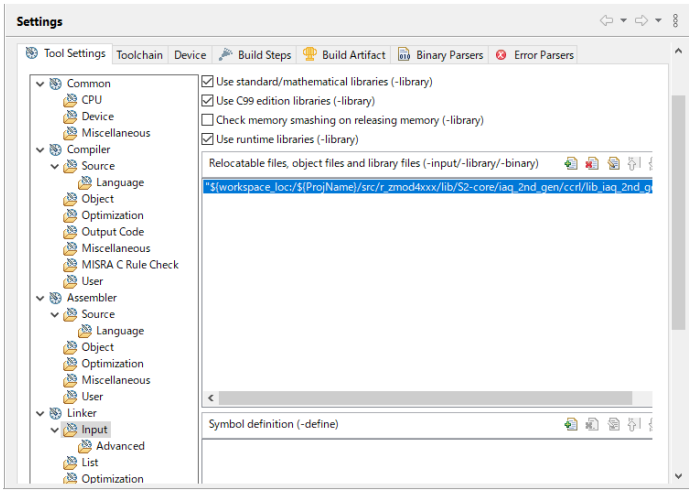
Select the "src" folder for the newly created project in the list and select the lib file that according to MCU architecture, measurement mode, and compiler to be used from the "r_zmod4xxx/lib" folder.











Build the project.

Select [Debug Configurations] from the menu and modify the debugger settings according to the specifications of the emulator to be connected to the target board.

8.4 RZ Sample Project

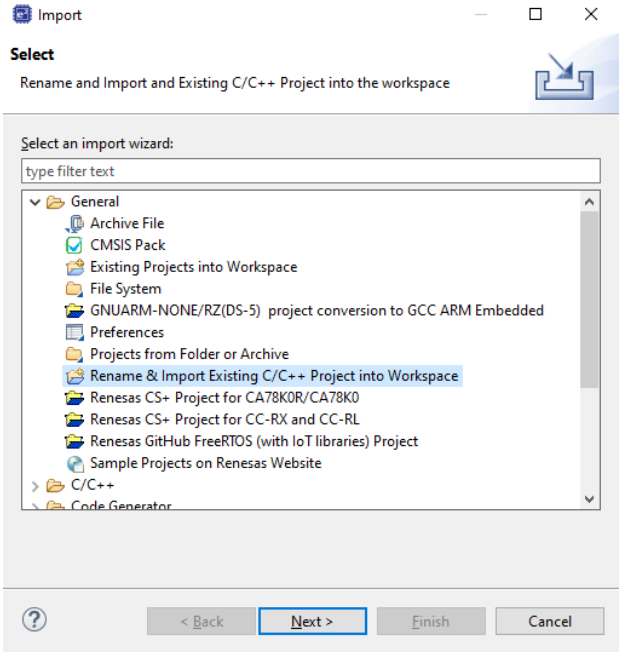
Use the following procedures to modify a sample project.

This section describes an example of modifying the sample project "ZMOD4410_RZG2L_NonOS" so that it can be used on the RZ/G2L Evaluation Kit (SMARC) board.

8.4.1 Importing the Sample Project

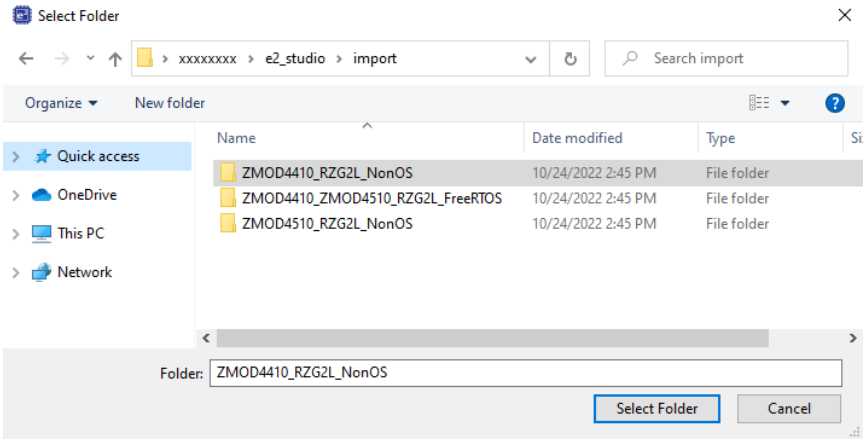
Select [Import] from the menu.

The "Import" window will appear. Select "Rename & Import Existing C/C++ Project into Workspace" in the window and press the [Next] button.

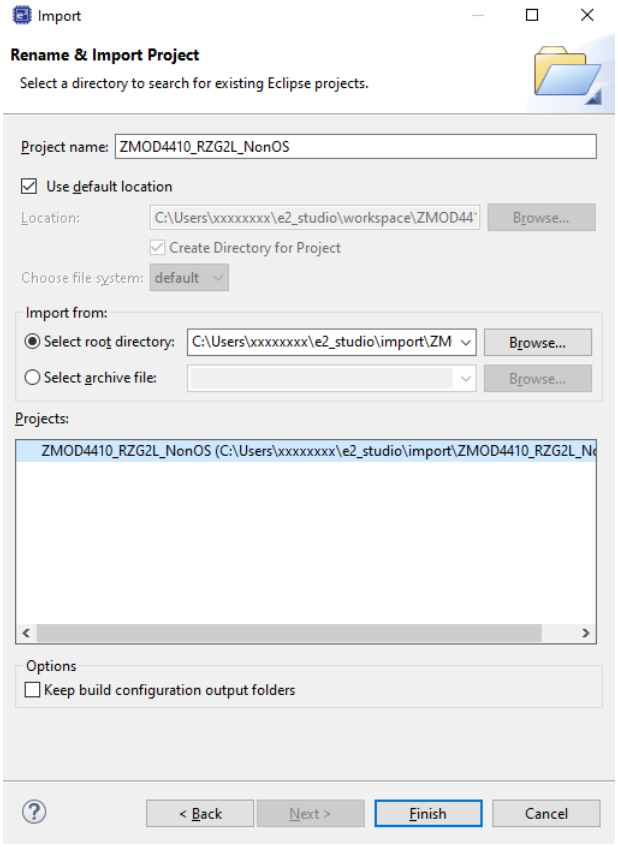


Press the [Browse] button to open the "Select Folder" window.

Select the folder of the original project for the current device from a list of imported sample projects and press the [Select Folder] button.



Enter the project name, select the original project for the current device, and press the [Finish] button.



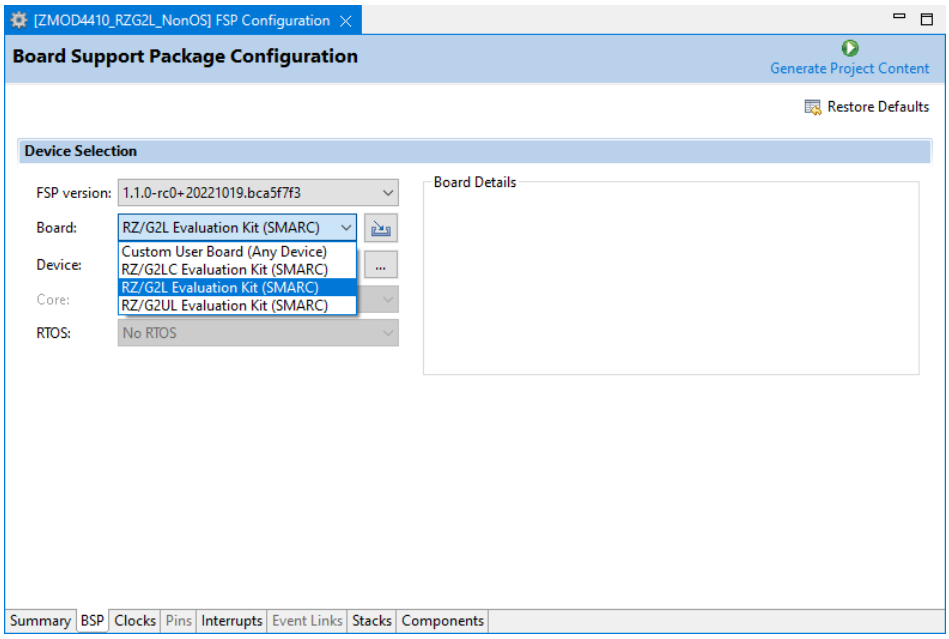
8.4.2 Modifying Settings of the FSP Configurator

Double-click on "Configuratorn.xml" in the project tree to open the FSP Configurator.

Change the settings of "Board" and "Device" in the "BSP" tabbed page.

When selecting a Renesas board, modify the "Board" setting only.

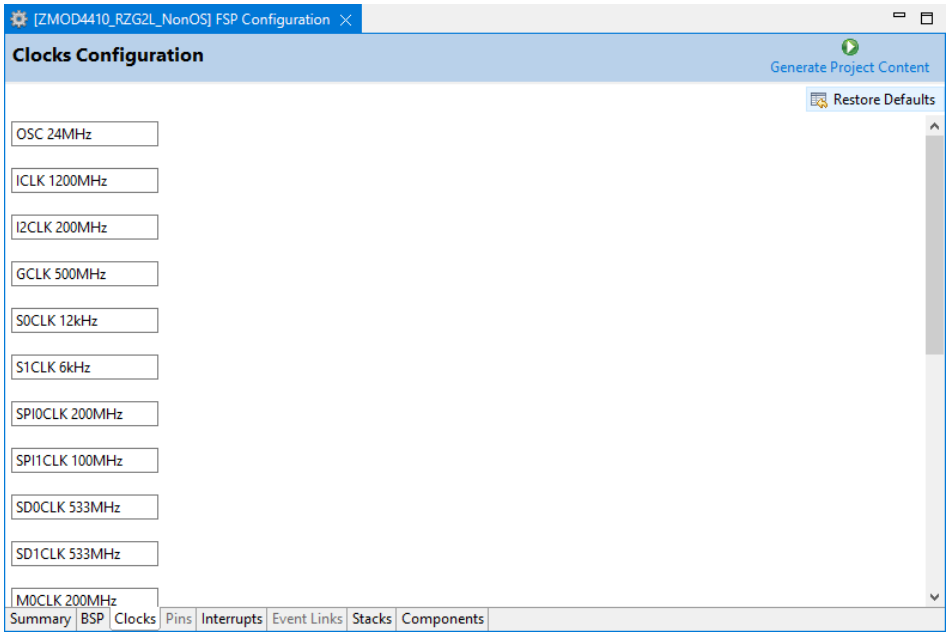
When selecting a board provided from other companies, change the "Board" setting to "Custom User Board (Any Device)" and then change the "Device" setting to the new device to be used.



Set up the clocks in the "Clocks" tabbed page.

When "Custom User Board (Any Device)" is selected for "Board", set up the clocks according to the specifications of the target board to be used.

When a Renesas board is selected for "Board", the clocks are automatically set up.

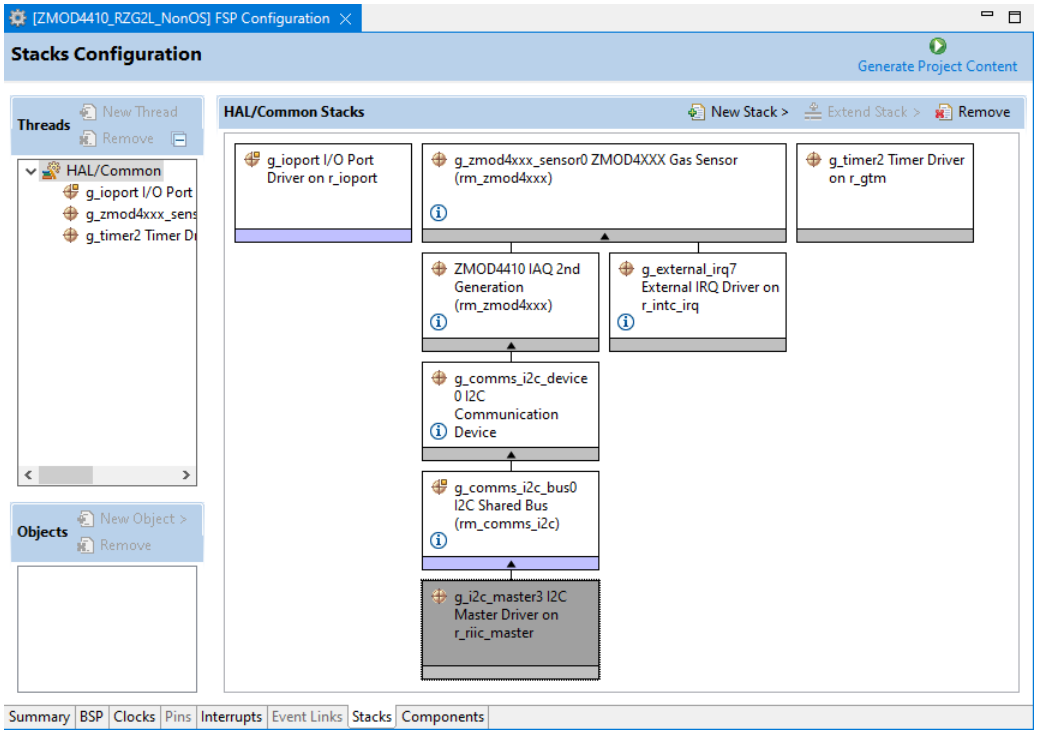


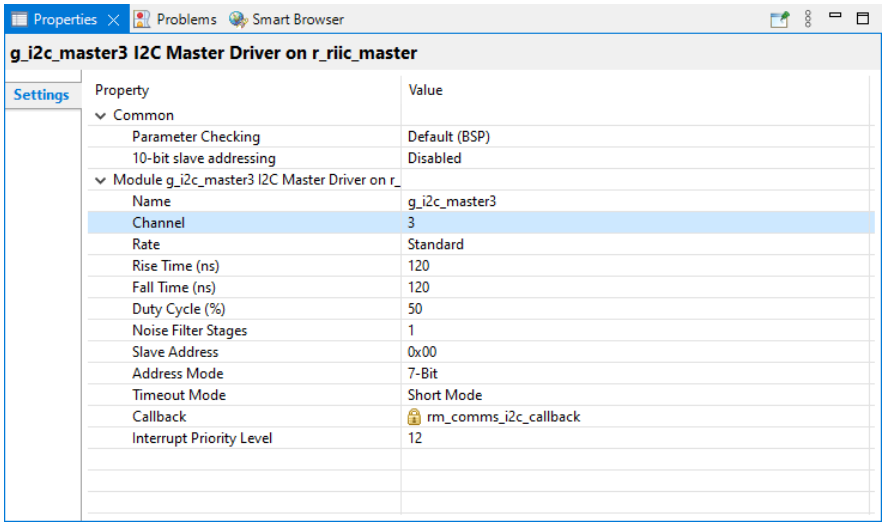
Modify the configuration of individual components in the "Stacks" tabbed page.

Modify the settings of r_riic_master and r_intc_irq according to the specifications of the target board.

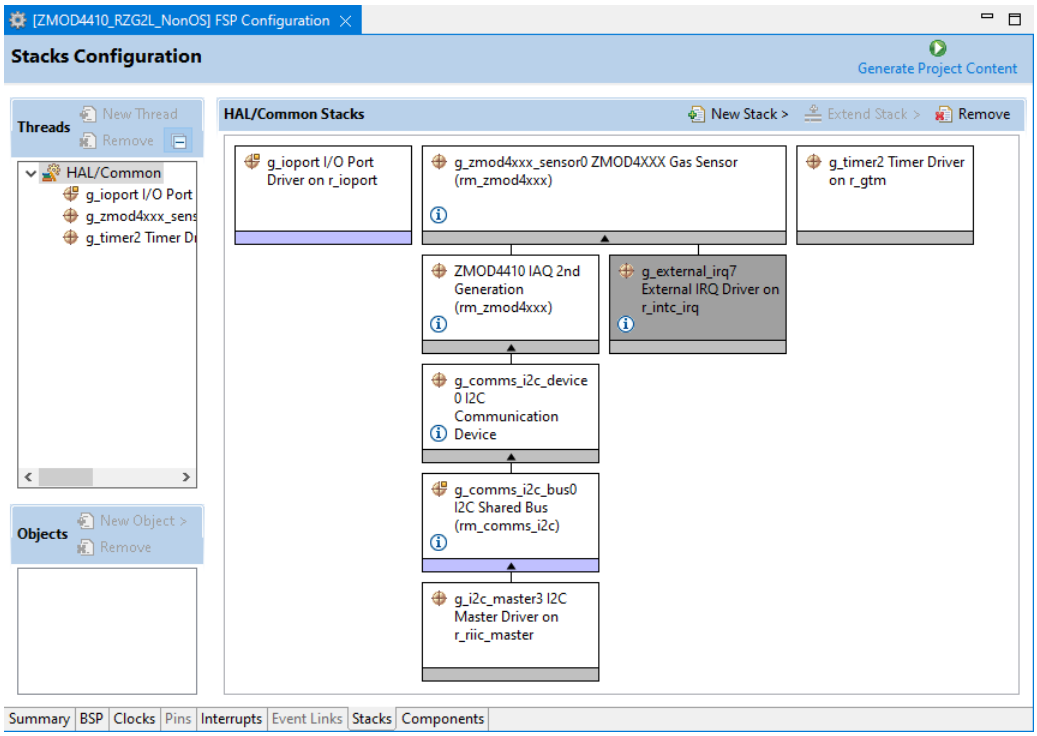
RIIC3 and IRQ7 are assigned to PMOD1 on the RZ/G2L Evaluation Kit (SMARC) board.

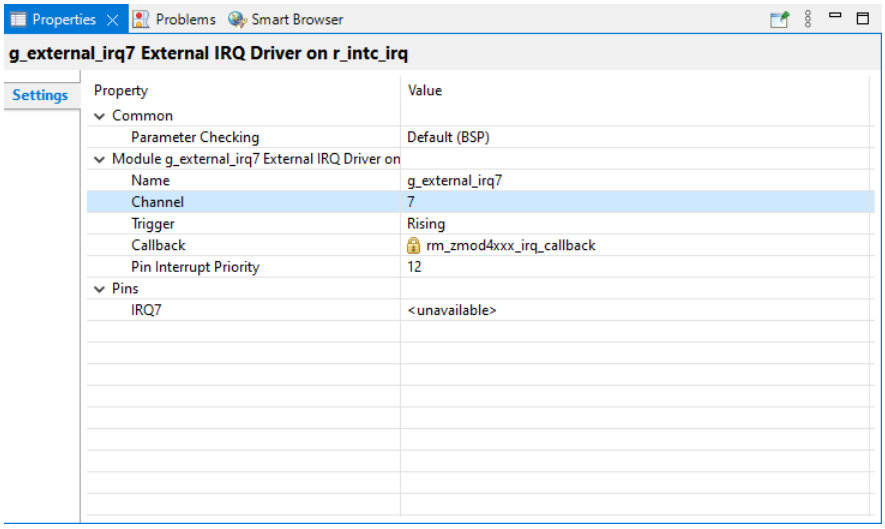
To use PMOD1, set "Channel" to "3" on r_riic_master.





To use PMOD1, set "Channel" to "7" on r_intc_irq.





Press [Generate Project Content] to generate files.

Build the project.

Select [Debug Configurations] from the menu and modify the debugger settings according to the specifications of the emulator to be connected to the target board.

8.4.3 Changing sample code

Open pin_data.c in the src folder and change the g_bsp_pin_cfg_data settings to match the board you are using.

On the RZ/G2L Evaluation Kit (SMARC) board, PMOD1 is assigned RIIC3 and IRQ7.

If you are using PMOD1, set the P18_0 to RIIC3_SDA (Function 3), the P18_1 to RIIC3_SCL (Function 3), and the P3_1 to IRQ7 (Function 1).

```
const ioport_pin_cfg_t g_bsp_pin_cfg_data[] =
{
    {.pin     = BSP_IO_PORT_18_PIN_00,
     .pin_cfg = ((uint32_t) IOPORT_CFG_PERIPHERAL_PIN |
                 (uint32_t) IOPORT_PERIPHERAL_MODE3)},
    {.pin     = BSP_IO_PORT_18_PIN_01,
     .pin_cfg = ((uint32_t) IOPORT_CFG_PERIPHERAL_PIN |
                 (uint32_t) IOPORT_PERIPHERAL_MODE3)},
    {.pin     = BSP_IO_PORT_03_PIN_01,
     .pin_cfg = ((uint32_t) IOPORT_CFG_PERIPHERAL_PIN |
                 (uint32_t) IOPORT_PERIPHERAL_MODE1)},
};
```

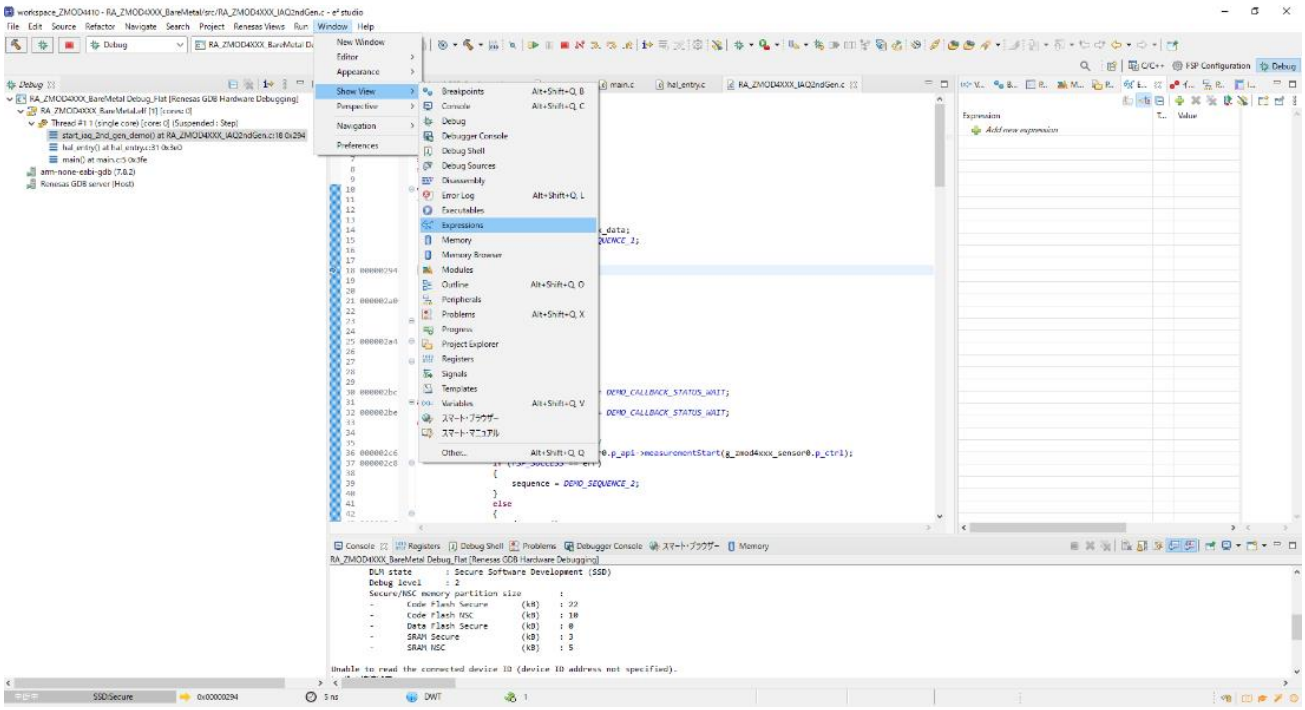
9. Viewing gas data

To check the real-time gas data, follow the steps below.

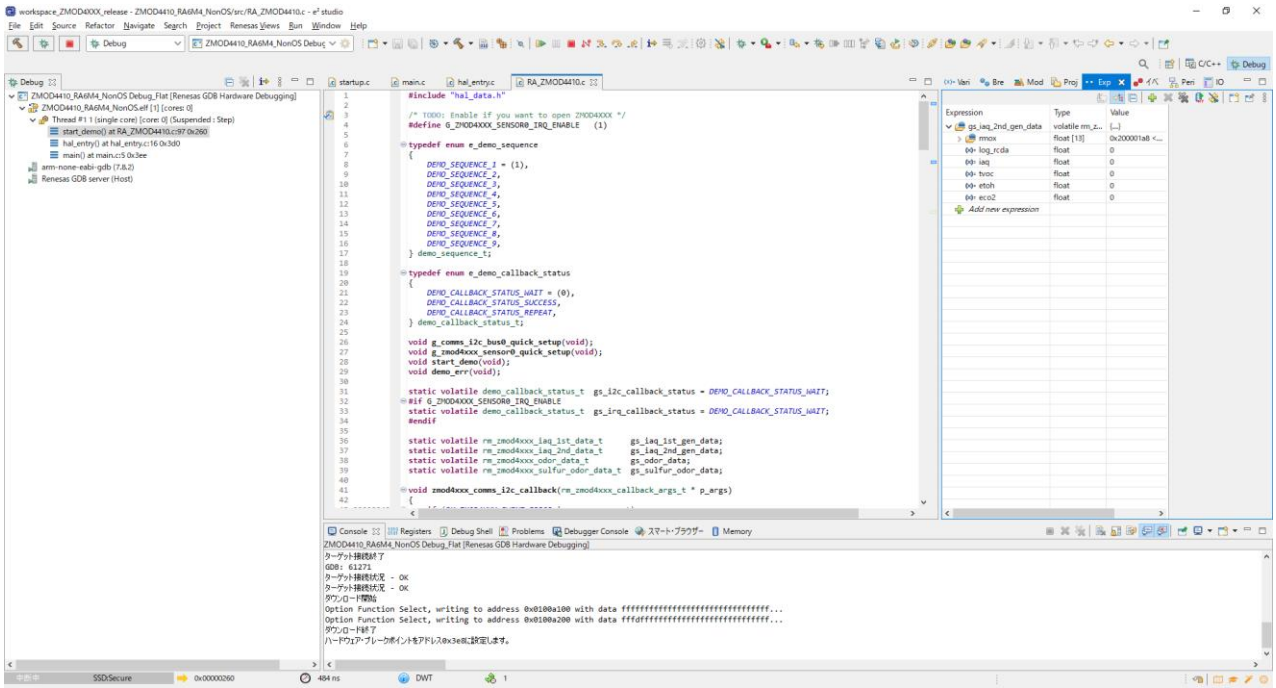
Take the “IAQ 2nd Generation” as an example.

After running the Debug, open the “Expressions” window.

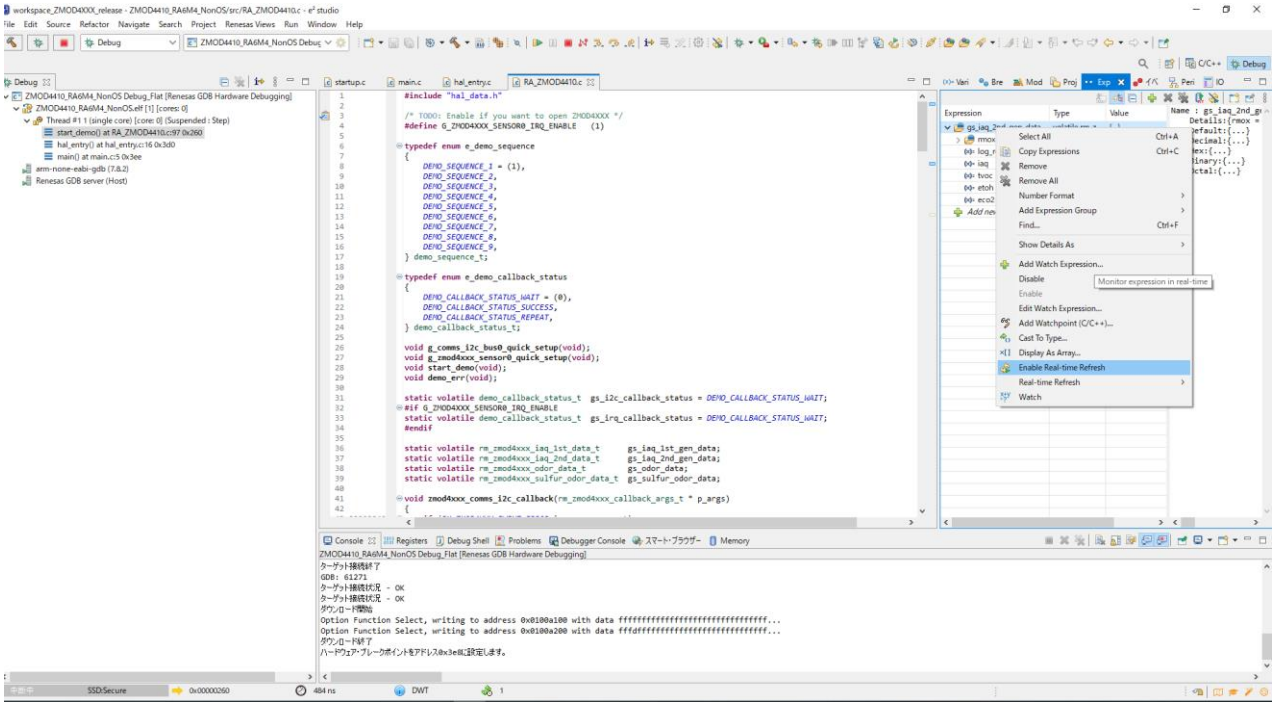
“Expressions” window is available from [Window]->[Show View]->[Expressions].



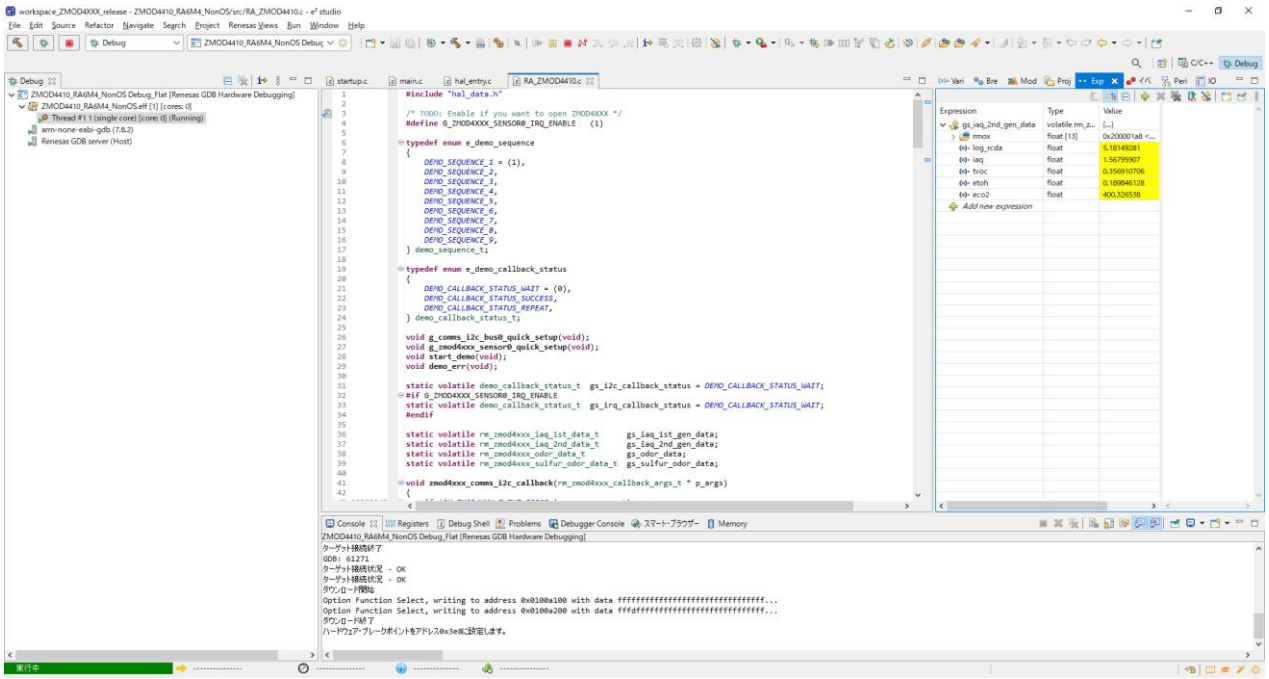
Click “Add new expression” in the “Expressions” and add “gs_iaq_2nd_gen_data”.



Right-click on the added variable and select the “Enable Real-time Refresh”.



Start the Debug and check the real-time values.



Revision History

Rev.	Date	Description	
		Page	Summary
Rev 1.00	June 30, 2021	-	First Release
Rev 1.10	September 30, 2021	-	Add RX Family, RL78 Family and RE01 256KB Group supporting
Rev.1.20	December 20, 2021	-	Add: Support multiple ZMOD sensors usage Add: Support RX Azure Other minor changes
Rev.1.30	March 1, 2022	-	Update chapters 2, 5 and 6 to add IAQ 2nd Gen ULP mode
Rev.1.31	March 11, 2022	-	Fix RA sample projects
		P9	Changed Figure 2-6 Hardware Connections for RE01 1500KB
Rev.1.40	June 30, 2022	P8, P9	Changed Environment for Confirming Operation on an RE01 Fixed: RE01 sample project codes
	August 31, 2022	-	Added: ZMOD4450 new. Fixed: Sample project codes
Rev.1.50	November 30, 2022	-	Added: RZ items Other minor changes
Rev.1.51	February 20, 2023	-	Bug fixes Updated: Environments for RL78
Rev.1.52	March 29, 2023	-	Updated: Environments for RA, RX, RL78, RZ Updated: Main Processing Flow of Sample Software Updated: Guide for Changing the Target Device
Rev.1.53	June 28, 2023	-	Updated: Environments for RA Updated: ZMOD4410 Sensor specifications Updated: Specification of Sample Software Updated: MOD4xxx Settings for RX, RL78
Rev.1.54	September 7, 2023	-	Updated: Guide for Changing the Target Device Deleted: RE01 items

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 be responsible for determining what licenses are required from any third parties, and obtaining such licenses for the lawful import, export, manufacture, sales, utilization, distribution or other disposal of any products incorporating Renesas Electronics products, if required.
5. 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.
6. 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.
7. No semiconductor product is absolutely secure. Notwithstanding any security measures or features that may be implemented in Renesas Electronics hardware or software products, Renesas Electronics shall have absolutely no liability arising out of any vulnerability or security breach, including but not limited to any unauthorized access to or use of a Renesas Electronics product or a system that uses a Renesas Electronics product. RENESAS ELECTRONICS DOES NOT WARRANT OR GUARANTEE THAT RENESAS ELECTRONICS PRODUCTS, OR ANY SYSTEMS CREATED USING RENESAS ELECTRONICS PRODUCTS WILL BE INVULNERABLE OR FREE FROM CORRUPTION, ATTACK, VIRUSES, INTERFERENCE, HACKING, DATA LOSS OR THEFT, OR OTHER SECURITY INTRUSION (“Vulnerability Issues”). RENESAS ELECTRONICS DISCLAIMS ANY AND ALL RESPONSIBILITY OR LIABILITY ARISING FROM OR RELATED TO ANY VULNERABILITY ISSUES. FURTHERMORE, TO THE EXTENT PERMITTED BY APPLICABLE LAW, RENESAS ELECTRONICS DISCLAIMS ANY AND ALL WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT AND ANY RELATED OR ACCOMPANYING SOFTWARE OR HARDWARE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. This document shall not be reprinted, reproduced or duplicated in any form, in whole or in part, without prior written consent of Renesas Electronics.
14. 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.5.0-1 October 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
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/.