RA Family, RX Family, RL78 Family, RZ Family

HS300x Sample Software Manual

Introduction

This application note describes the sample software that is for use with the HS300x humidity and temperature sensor and runs on certain MCUs of the RA family, RX family, RL78 family, and RZ family.

Target Devices

RA6M4 Group

RX65N Group

RL78/G14 Group

RL78/G23 Group

RZ/G2L Group

Trademarks

Contents

1. Overview

This sample software acquires humidity and temperature data from the HS300x humidity and temperature sensor and handles calculations on the data. In combination with the I2C driver of the FSP or FIT, the sample software controls the HS300x through the I2C in the MCU to measure humidity and temperature, acquire ADC data, and calculate the acquired data.

2. Environment for Confirming Operation

2.1 Environment for Confirming Operation on an 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 the RA Family MCU

Item	Description
Demonstration board	RTK7EKA6M4S00001BE (EK-RA6M4)
Microcontroller	RA6M4 (R7FA6M4AF3CFB: 144 pins)
Operating frequency	200 MHz
Operating voltage	5 V
Integrated development environment	e^2 Studio 2023-01
C compiler	GCC 10.3.1.20210824 IAR ANSI C/C++ Compiler V8.50.9.278/LNX for ARM ARM Compiler 6.16
FSP	V.3.7.0
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	Relative Humidity Sensor Pmod™ Board (US082-HS3001EVZ)

Table 2-2 Amount of Memory Used in the RA Family MCU

Area	Size
ROM	Non-OS version: 1,317 bytes FreeRTOS version: 1,630 bytes Azure RTOS version: 1,606 bytes
RAM	Non-OS version: 77 bytes FreeRTOS version: 257 bytes Azure RTOS version: 426 bytes

Memory size is calculated by functions and variables only related to HS300x sensor. In RTOS, memory size does not include memory size of the thread.

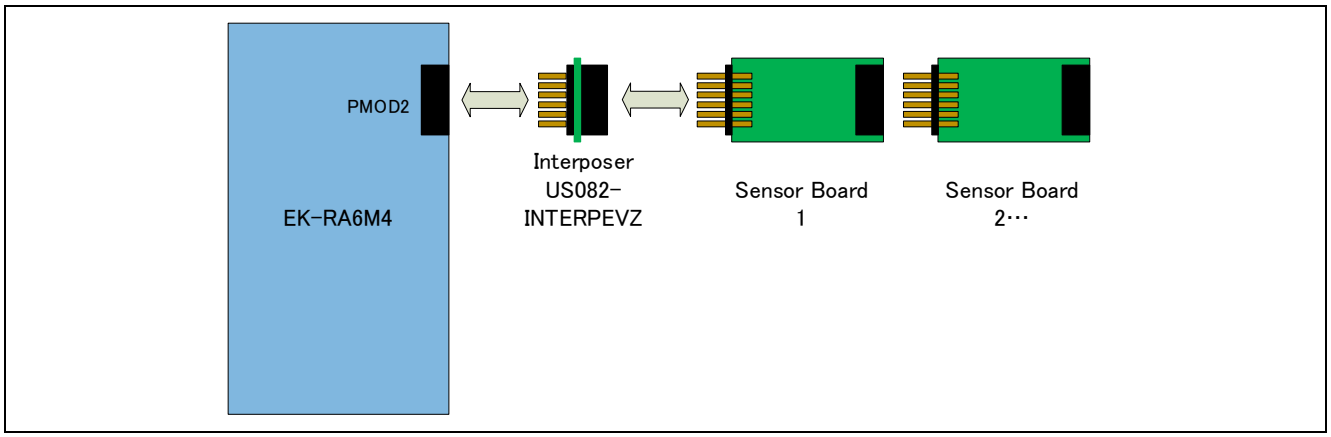


Figure 2-1 Hardware Connections for the RA Family

2.2 Environment for Confirming Operation on an 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 the RX Family MCU

Item	Description
Demonstration board	RPBRX65N (Envision Kit RX65N)
Microcontroller	RX65N (R5F565NEDDFB: 144 pins)
Operating frequency	12 MHz
Operating voltage	5 V
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	Relative Humidity Sensor Pmod™ Board (US082-HS3001EVZ)

Table 2-4 Amount of Memory Used in the RX Family MCU

Area	Size
ROM	Non-OS version: 1,652 bytes FreeRTOS version: 1,992 bytes Azure RTOS version: 2,041 bytes
RAM	Non-OS version: 133 bytes FreeRTOS version: 149 bytes Azure RTOS version: 289 bytes

Memory size is calculated by functions and variables only related to HS300x sensor. In RTOS, memory size does not include memory size of the thread.

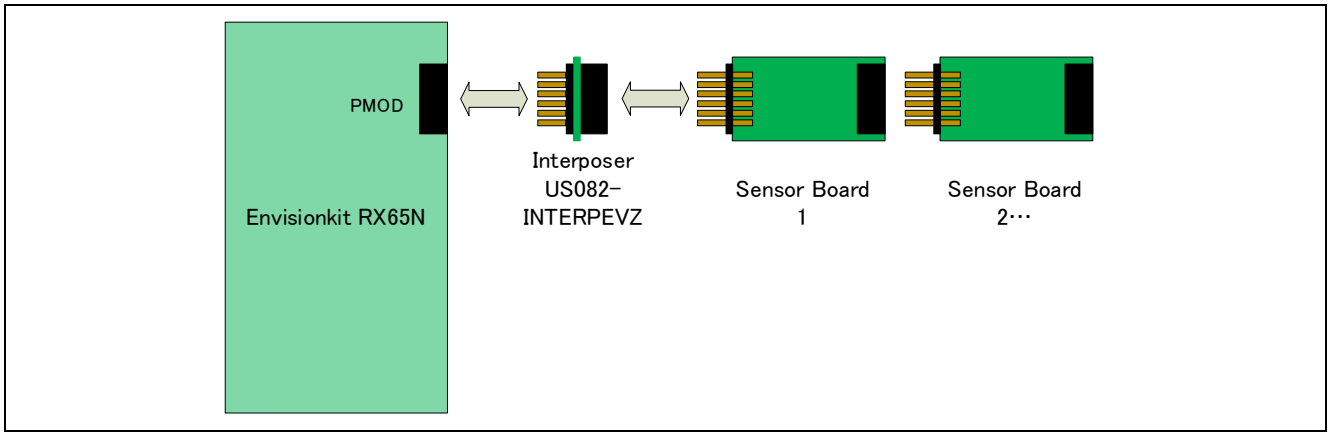


Figure 2-2 Hardware Connections for the RX Family

2.3 Environment for Confirming Operation on an 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 the RL78/G14 Group MCU

Item	Description
Demonstration board	RTK5RLG140C00000BJ (RL78/G14 Fast Prototyping Board)
Microcontroller	RL78/G14 (R5F104MLAFB :80pin)
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 GCC for Renesas RL78 4.9.2.202103 IAR Toolchain for RL78 4.21.1.2409
Emulator	On board (E2OB)(RL78/G14)
Sensor board	Relative Humidity Sensor Pmod™ Board (US082-HS3001EVZ)

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

Area	Size
ROM	1,542 bytes
RAM	83 bytes

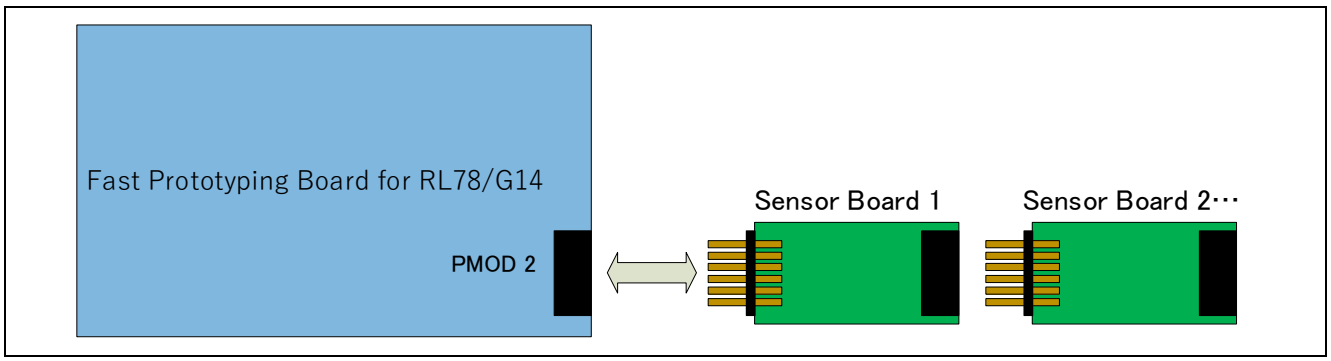


Figure 2-3 Hardware Connections for the RL78/G14 Group

2.4 Environment for Confirming Operation on an 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 the RL78/G23 Group MCU

Item	Description
Demonstration board	RTK7RLG230CSN000BJ (RL78/G23-128p Fast Prototyping Board)
Microcontroller	(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 V1.10.00 LLVM for RL78 10.0.0.202209 IAR Toolchain for RL78 4.21.1.2409
Emulator	E2 Lite
Sensor board	Relative Humidity Sensor Pmod™ Board (US082-HS3001EVZ)

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

Area	Size
ROM	1,690 bytes
RAM	80 bytes

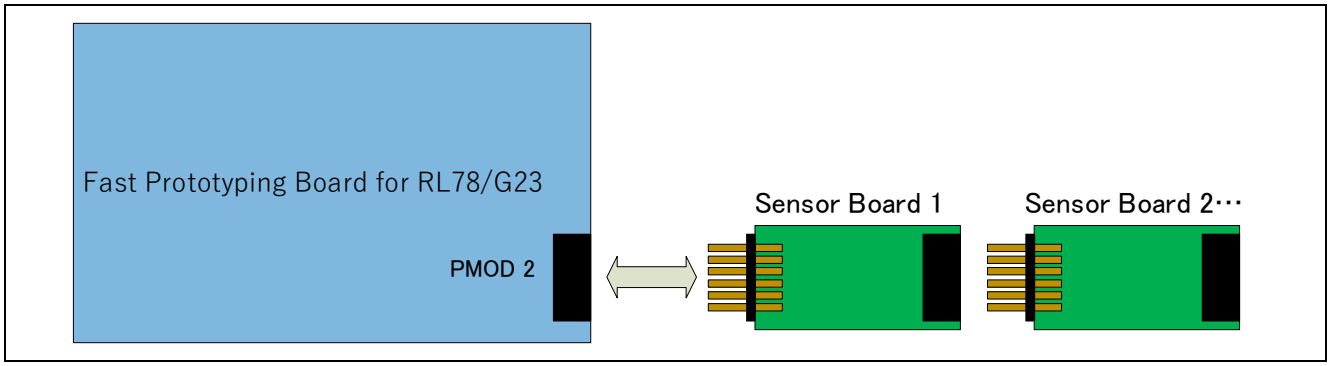


Figure 2-4 Hardware Connections for the RL78/G23 Group

2.5 Environment for Confirming Operation on an 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 the RZ Family MCU

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	5 V
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	Relative Humidity Sensor Pmod™ Board (US082-HS3001EVZ)

Table 2-10 Amount of Memory Used in the RZ Family MCU

Area	Size
ROM	Non-OS version: 1,574 bytes FreeRTOS version: 1,882 bytes
RAM	Non-OS version: 149 bytes FreeRTOS version: 429 bytes

Memory size is calculated by functions and variables only related to HS300x sensor. In RTOS, memory size does not include memory size of the thread.

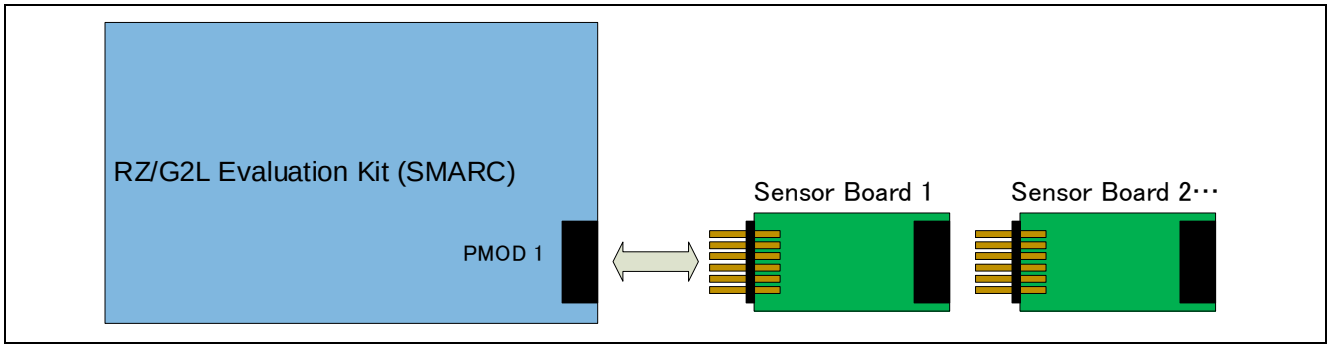


Figure 2-5 Hardware Connections for the RZ Family

3. Sensor Specifications

3.1 Overview of Sensor Specifications

Table 3-1 gives an overview of the specifications of the HS300x humidity and temperature sensor.

Table 3-1 Overview of Sensor Specifications

Item	Description
Humidity range	0%RH to 100%RH
Humidity accuracy	±1.5%RH (typ.) (HS3001: 10%RH to 90%RH, 25°C)
14-bit resolution	0.01%RH (typ.)
Independent programmable resolution settings	14 bits
Fast RH response time	1 second (typ.) (with 1 m/s air flow) 4 seconds (typ.) (in a sealed space)
Temperature sensor accuracy	±0.2°C typ. (HS3001 and HS3002: –10°C to 80°C)
Average Current	14-bit resolution at 3.3 V: 24.4 μA (max.)
Sleep Current	–40°C to 85°C: 0.6 μA (typ.) –40°C to 125°C: 1 μA (typ.)
Supply voltage	2.3 V to 5.5 V, 3.3 V (typ.)
Extended supply voltage	1.8 V (–20°C to +125°C)
Operating temperature	–40°C to +125°C

3.2 Sensor Functions

This software supports the following functions of HS300x.

Table 3-2 List of Sensor Functions

Function	Description
I2C communications	Sensor data are transferred through I2C communications.
Measurement mode	This software operates with the sensor in sleep mode most of the time. On completion of measurement triggered by a measurement request, the sensor enters sleep mode.
Measurement request	The sensor in sleep mode is placed in the measurement state upon receiving a measurement request.
Data fetch	At the end of a measurement cycle, valid data can be acquired.
Status bits	The status bits for the results of measurement indicate whether the current data are valid or old.

Note: This software does not support the following functions of HS300x.

- Access to non-volatile memory
- Setting of the measurement resolution
- Reading of the HS300x ID number

3.2.1 I2C Communication Interface

The following shows the format of measurement data transferred through I2C communications.

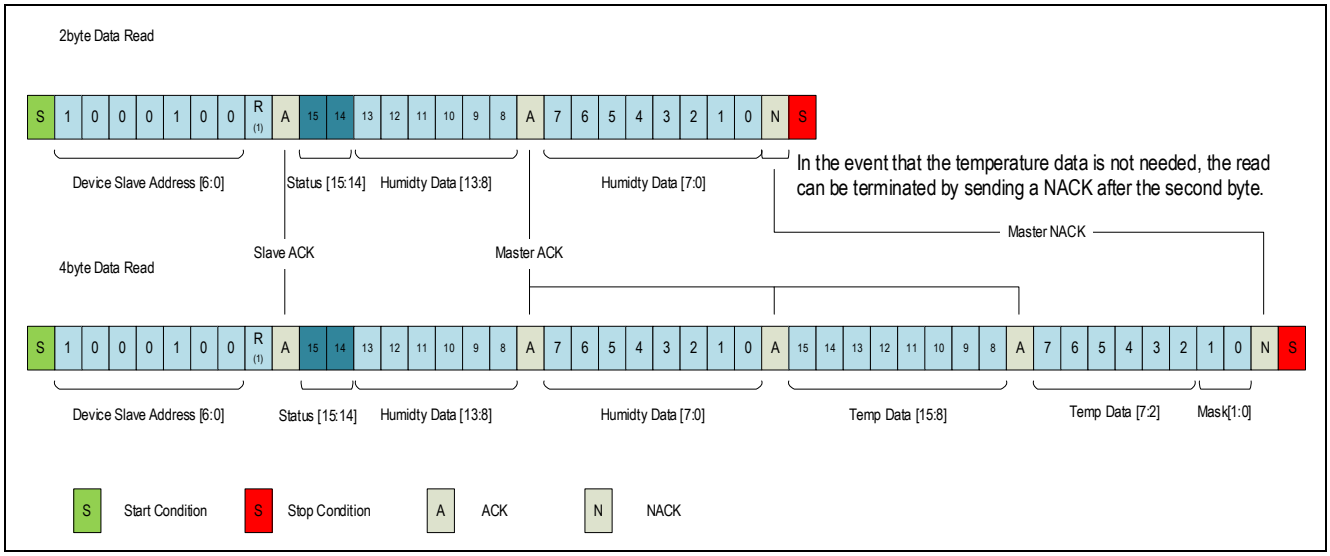


Figure 3-1 Format of I2C Communications

The status bits indicate the state of data as follows.

00_B: Valid data acquired at the end of a measurement cycle

01_B: Invalid data that have already been acquired

3.2.2 Expressions for Converting Output Values to Humidity and Temperature

The HS300x software uses functions for converting the acquired ADC data to values for humidity and temperature. The following shows the conversion expressions.

- Humidity conversion expression

$$Humidity\,[\%RH] = \left(\frac{Humidity\,[13:0]}{2^{14} - 1} \right) * 100$$

- Temperature conversion expression

$$Temperature\,[^{o}C] = \left(\frac{Temperature\,[15:2]}{2^{14} - 1} \right) * 165 - 40$$

4. Sample Software Specifications

This sample software package contains a total of thirteen projects: non-OS and OS versions for the RA family, non-OS and OS versions for the RX family, non-OS version for the RL78 family, and a non-OS and OS versions for the RZ family. This section describes these projects.

For the FreeRTOS settings for the RX family, refer to the FAQ.

4.1 Configuration of the Sample Software

Figure 4-1 is a block diagram of the sample software.

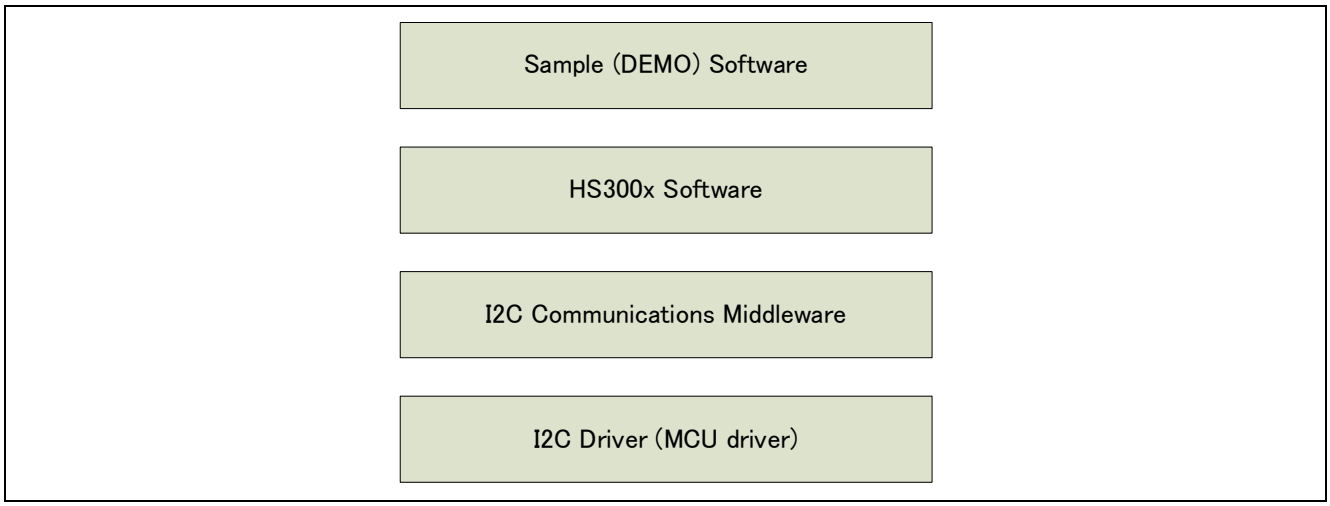


Figure 4-1 Block Diagram of the Sample Software

4.2 Specifications of Sensor API Functions

4.2.1 List of Sensor API functions

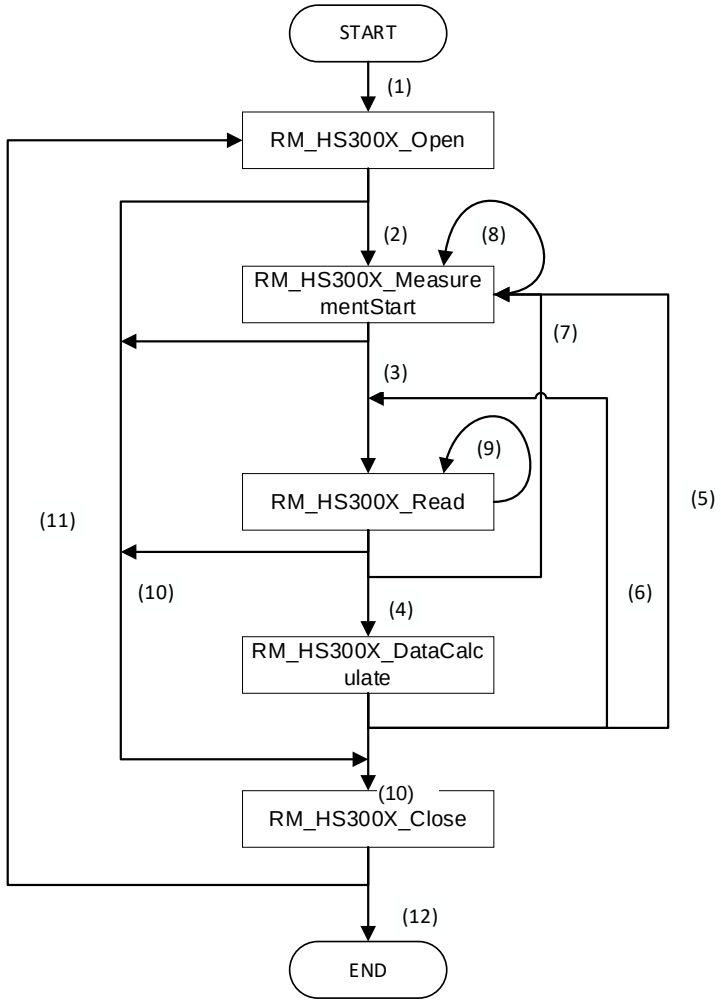
The following table lists the sensor API functions. For details of the API functions, refer to the separately provided RX Family HS300x Sensor API FIT Module application note (R01AN5893) and RL78 Family Renesas Sensor Control Modules application note (R01AN5896).

Table 4-1 List of Sensor API Functions

Function	Description
RM_HS300X_Open	Starts control of the sensor.
RM_HS300X_Close	Terminates control of the sensor.
RM_HS300X_MeasurementStart	Starts measurement by the sensor.
RM_HS300X_Read	Acquires data from the sensor.
RM_HS300X_DataCalculate	Calculates values from the data acquired from the sensor.
RM_HS300X_ProgrammingModeEnter	Starts programming mode.
RM_HS300X_SensorIdGet	Get ID of the sensor.
RM_HS300X_ResolutionChange	Change the measurement resolution.
RM_HS300X_ProgrammingModeExit	Terminates programming mode.

4.2.2 Guide to Using the API Functions

The following diagram of API function transitions shows the conditions on the usage of the individual HS300x API functions and the expected orders of function calls.



Flow No.	Function	Transition Case
(1)	Start HS300x	not open HS300x
(2)	Start Measurement HS300x	open success
(3)	ADC Read	Measurement success (callback state success)
(4)	Humidity & Temp. calculate	ADC Read success (callback state success)
(5)	next measurement start	DataCalculate return success and next ADC data Get
(6)	get retry ADC data	user get retry ADC data or DataCalculate return invalid data
(7)	next measurement (Not calculate)	The user requests only ADC data and measures the next data.
(8)	Retry Measure Start	Measurement start Waiting for response (callback none and polling)
(9)	Retry ADC Read	Read ADC Waiting for response (callback none and polling)
(10)	Close HS300x	Error state or user stop
(11)	Restart HS300x	Close success and user restart
(12)	Stop HS300x	Close success or Error

Figure 4-2 Diagram of Transitions between API Function Calls

The conditions for calling the individual functions are shown below.

- RM_HS300X_Open: (1) Activation of HS300x or (11) restart after a call of RM_HS300X_Close
- RM_HS300X_Close: (10) Successful completion or abnormal end of individual processing
- RM_HS300X_MeasurementStart: (2) Start of measurement after a call of RM_HS300X_Open, (5) (7) acquisition of the next measured data, or (8) retry after waiting for the response to the measurement start request
- RM_HS300X_Read: (3) Acquisition of measured data after a call of RM_HS300X_MeasurementStart or (9) retry after waiting for the response to the data acquisition request
- RM_HS300X_DataCalculate: (4) Calculation of humidity and temperature data after a call of RM_HS300X_Read

Notes:
Since RM_HS300X_Open checks the state of the I2C driver, the I2C driver must be opened before the RM_HS300X_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 measurement is started by RM_HS300X_MeasurementStart, the sensor stops measurement after having updated the ADC data. Therefore, be sure to call RM_HS300X_MeasurementStart at least once every time before executing processing by RM_HS300X_Read.

When using an OS and controlling the sensor with multiple threads or tasks simultaneously in use, the user will need to use a semaphore to control the bus. For the timing of the semaphore being raised and the control of blocking, refer to 4.4 Flowchart of the OS Version of the Sample Software, Flowchart of the OS Version of the Sample Software.

Shows the flow and timing of Programming mode.

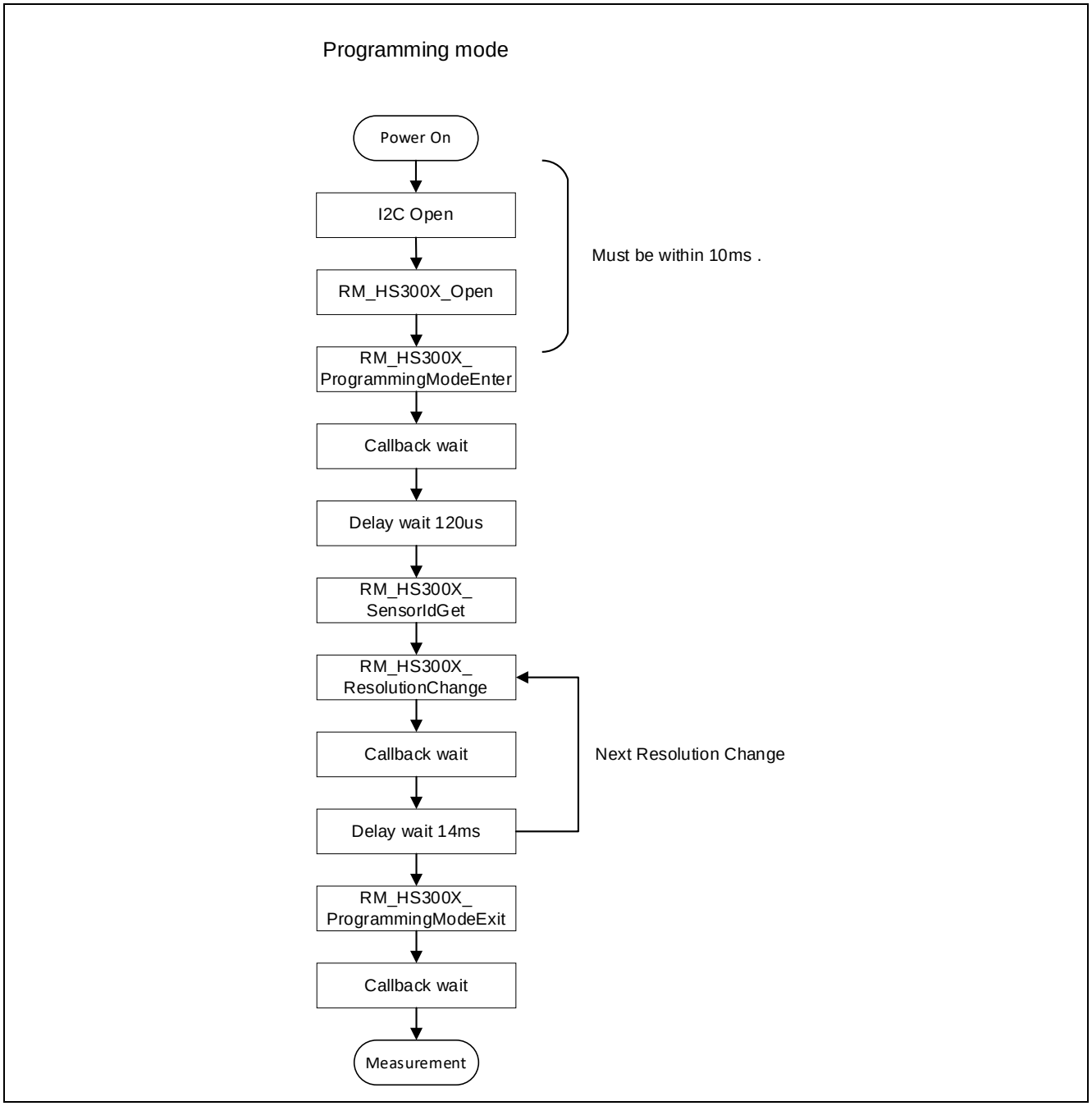


Figure 4-3 Diagram of Programming mode.

4.3 Flowchart of the Main Processing in the Non-OS Version of the Sample Software

This sample software first starts the driver and then repeats the processing for starting the measurement by the sensor, acquiring data from the sensor, and calculating values from the results of measurement.

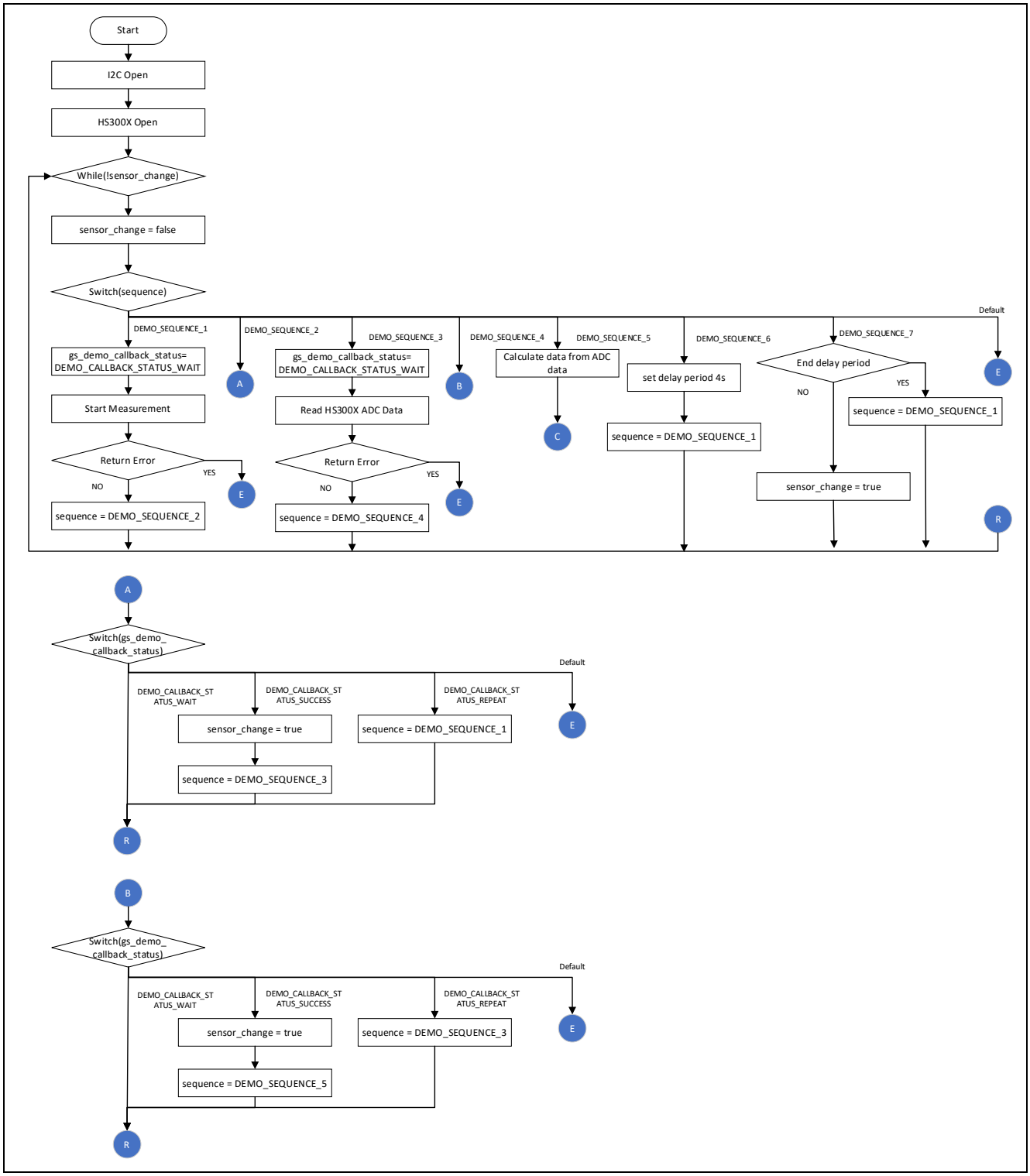


Figure 4-4 Flowchart of the Main Processing in the Non-OS Version of the Sample Software (1)

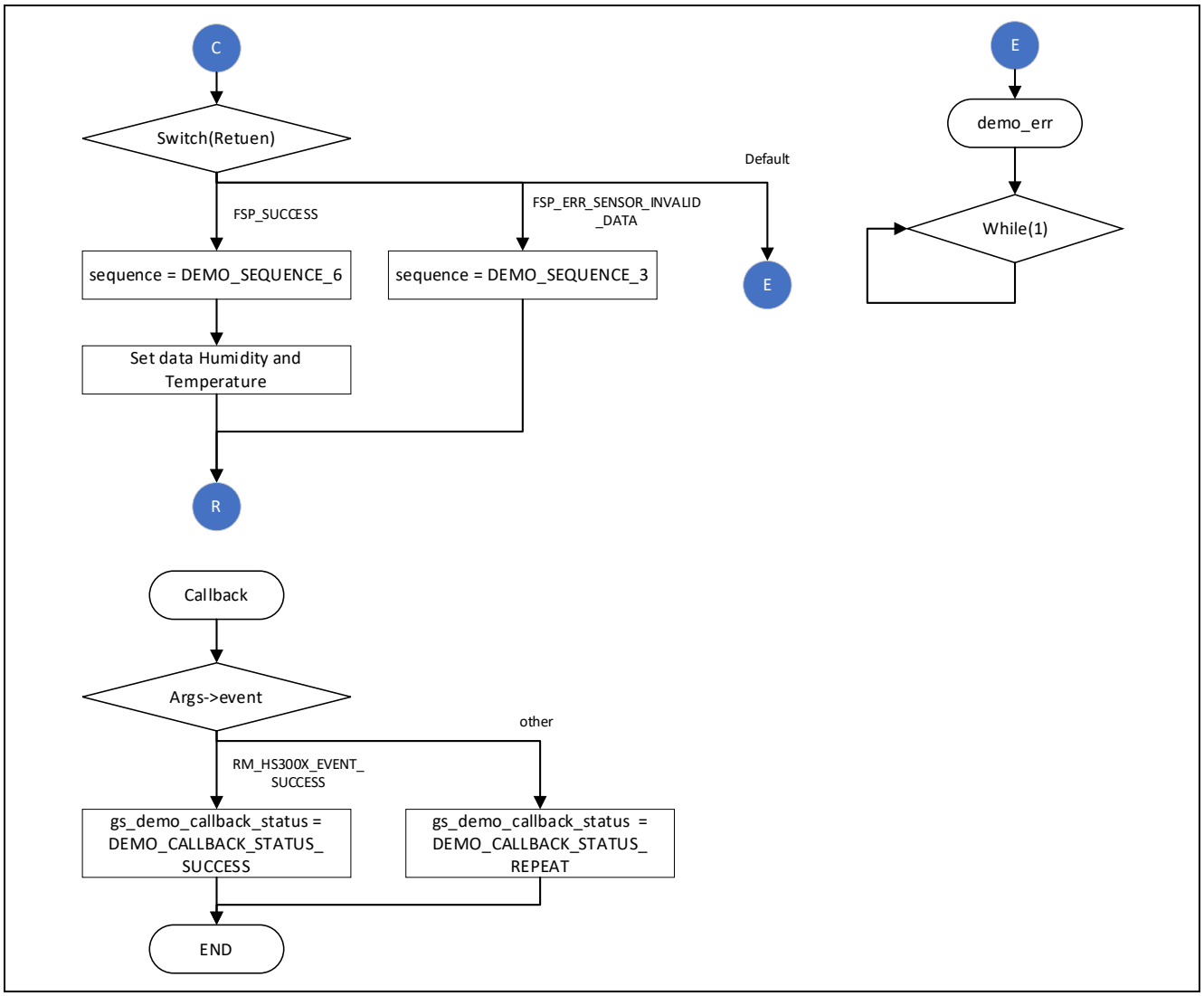


Figure 4-5 Flowchart of the Main Processing in the Non-OS Version of the Sample Software (2)

4.4 Flowchart of the OS Version of the Sample Software

The OS version uses a semaphore in control of the sensor and operates one thread for controlling the sensor in parallel.

The control of the sensor in the thread first starts the driver and then repeats the processing for starting the measurement by the sensor, acquiring data from the sensor, and calculating values from the results of measurement.

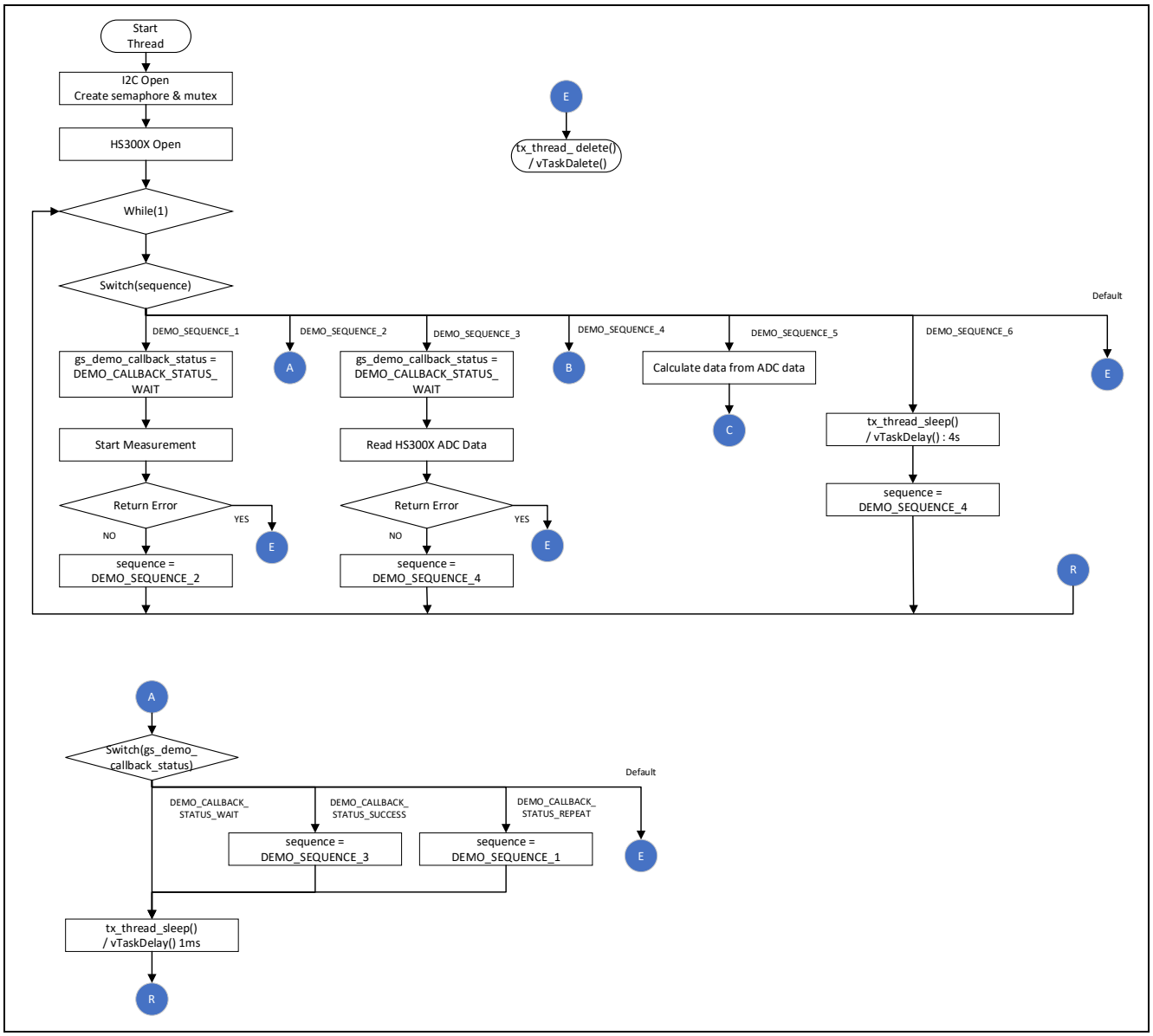


Figure 4-6 Flowchart of the Main Processing in the OS Version of the Sample Software (1)

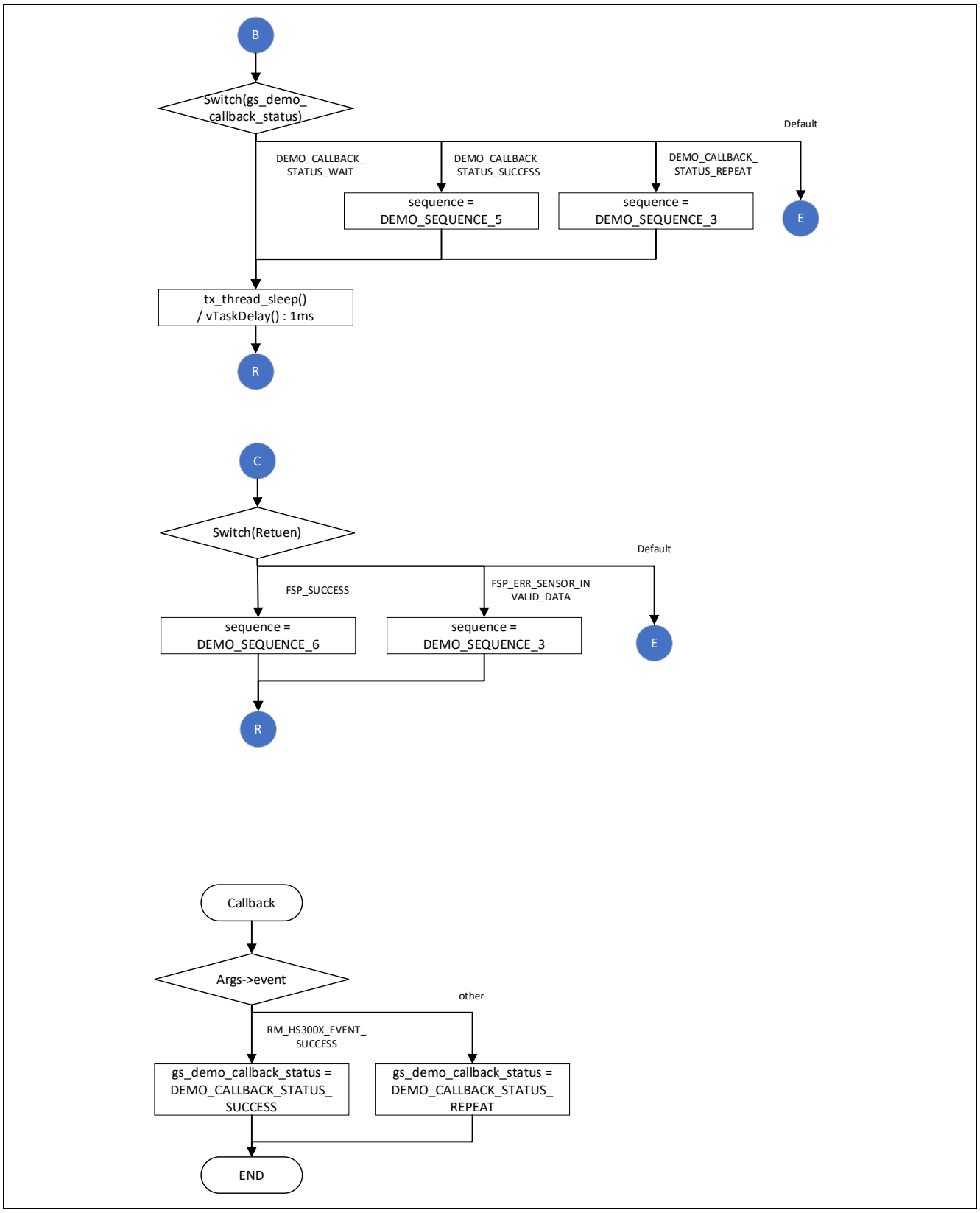


Figure 4-7 Flowchart of the Main Processing in the OS Version of the Sample Software (2)

4.5 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/hs300x_sensor_thread_entry.c

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

5. Configuration Settings

5.1 HS300x Humidity and Temperature Sensor Settings

5.1.1 RA Family

Select the rm_hs300x stack on the [Stack] tabbed page of the FSP Configurator, and the configurable items will be shown on the [Properties] tabbed page.

The following items and values can be specified.

Table 5-1 HS300x Settings for the RA Family MCU

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.
Data type	Both humidity and temperature Humidity only	Specify the type of data to be acquired from the sensor. Both the humidity and temperature or the humidity alone can be selected.
Programming Mode	OFF	Specify programming mode.
Module g_hs300x_sensor HS300X on rm_hs300x		
Name	g_hs300x_sensor0	Specify the name of the module. The module name must conform with the C language standard.
Callback	hs300x_callback	Specify the name of the user callback function. The callback function name must conform with the C language standard. When "NULL" is specified, no callback function is used.

5.1.2 RX Family

Select the r_hs300x_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 5-2 HS300x Settings for the RX Family MCU

Configurable Item	Value	Description
Configurations		
Parameter Checking	System Default	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	
Number of HS300x sensors	1	Specify the number of HS300x sensors to be connected.
	2	
Data types from HS300x sensors	Humidity and temperature	Specify the type of data to be acquired from the sensor. Both the humidity and temperature or the humidity alone can be selected.
	Humidity only	
Programming Mode	OFF	Specify programming mode.
I2C communication device No. for HS300x Sensor Device x (x = 0 or 1)	I2C Communication Device0	Specify the communications device number to be used by the sensor.
	I2C Communication Device1	
	I2C Communication Device2	
	I2C Communication Device3	
	I2C Communication Device4	
	Enabled	
Callback function for HS300x sensor device x (x = 0 or 1)	hs300x_user_callback(x) (x = 0 or 1)	Specify the name of the user callback function. The callback function name must conform with the C language standard.

5.1.3 RL78 Family

Settings can be modified by changing the values of constants defined in the ¥r_config¥r_hs300x_rl_config.h file in the project tree of the sample project.

The following items and values can be specified.

Table 5-3 HS300x Settings for the RL78 Family MCU

Constant Name	Value	Description
Configurations		
RM_HS300X_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_HS300X_CFG_DEVICE_NUM_MAX	1	Specify the number of HS300x sensors to be connected.
	2	
RM_HS300X_CFG_DATA_BOTH_HUMIDITY_TEMPERATURE	0	Specify the type of data to be acquired from the sensor. The humidity alone (0) or both the humidity and temperature (1) can be selected.
	1	
RM_HS300X_CFG_PROGRAMMING_MODE	0	Specify programming mode.
RM_HS300X_CFG_DEVICEx_COMMS_INSTANCE (x = 0 or 1)	g_comms_i2c_device(x) (x = 0 or 1)	Specify the instance name of the communications line to be used.
RM_HS300X_CFG_DEVICEx_CALLBACK (x = 0 or 1)	hs300x_user_callback1(x) (x = 0 or 1)	Specify the name of the user callback function. The callback function name must conform with the C language standard.

5.1.4 RZ Family

Select the rm_hs300x stack on the [Stack] tabbed page of the FSP Configurator, and the configurable items will be shown on the [Properties] tabbed page.

The following items and values can be specified.

Table 5-4 HS300x Settings for the RZ Family MCU

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.
Data type	Both humidity and temperature Humidity only	Specify the type of data to be acquired from the sensor. Both the humidity and temperature or the humidity alone can be selected.
Programming Mode	OFF	Specify programming mode.
Module g_hs300x_sensor HS300X on rm_hs300x		
Name	g_hs300x_sensor0	Specify the name of the module. The module name must conform with the C language standard.
Callback	hs300x_callback	Specify the name of the user callback function. The callback function name must conform with the C language standard. When "NULL" is specified, no callback function is used.

5.2 Communication Driver Middleware Settings

5.2.1 RA Family

Select the rm_comms_i2c stack on the [Stack] tabbed page of the FSP Configurator, and the configurable items will be shown on the [Properties] tabbed page.

The following items and values can be specified.

Table 5-5 Communication Driver Settings for the RA Family MCU

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_device I2C Communication Device on rm_comms_i2c		
Name	g_comms_i2c_device0	Specify the name of the module. The module name must conform with the C language standard.
Semaphore Timeout	0xFFFFFFFF	For an RTOS project, specify the time of semaphore timeout.
Slave Address	0x44	Specify the slave address. When rm_hs300x is used, this value is automatically specified and cannot be modified.
Address Mode	7-Bit	Specify the number of slave address bits. When rm_hs300x is used, this value is automatically specified and cannot be modified.
Callback	rm_hs300x_callback	Specify the name of the user callback function. When rm_hs300x 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	For an RTOS project, enable or disable processing for blocking.
	Use	
Recursive Mutex for Bus	Unuse	For an RTOS project, enable or disable recursive operation when blocking is enabled.
	Use	

5.2.2 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 5-6 Communication Driver Settings for the RX Family MCU

Configurable Item	Value	Description
Configurations		
Parameter Checking	System Default	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	
Number of communication lines	Unused	Specify the number of communications bus lines that can be connected.
	1	
	2	
	3	
	4	
	5	
Number of I2C Devices	Unused	Specify the number of I2C device that can be connected.
	1	
	2	
	3	
	4	
	5	
Blocking operation supporting with RTOS	Disabled	For an RTOS project, enable or disable the blocking operation.
	Enabled	
Bus lock operation supporting with RTOS	Disabled	For an RTOS project, enable or disable the bus lock operation.
	Enabled	
IIC Driver Type for I2C Shared bus(x) (x = 0 - 4)	RIIC	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.
	SCI IIC	
	Not selected	
Channel No. for I2C Shared bus(x) (x = 0 - 4)	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 - 4)	0xFFFFFFFF	Specify the time of I2C bus(x) timeout. (x = 0 - 4)
I2C Shared Bus No. for I2C Communication Device(x) (x = 0 - 4)	I2C Shared Bus(x) (x = 0 - 4)	Specify the configuration of used communication bus.
Slave address for communication device(x)	0x44	Specify the slave address of the device to be connected to the communications bus. If you are using rm_hs300x, specify 0x44.

(x = 0 - 4)		
Slave address mode for communication device(x) (x = 0 - 4)	7 bit address mode	Specify the slave address mode. If you are using rm_hs300x, specify the 7-bit address mode.
Callback function for Communication device(x) (x = 0 - 4)	comms_i2c_user_callback(x) (x = 0 - 4)	Specify the name of the user callback function. When using rm_hs300x, specify rm_hs300x_callback(y) (y = 0-1).

5.2.3 RL78 Family

Settings can be modified by changing the values of constants defined in the ¥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 5-7 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)	0x44	Specify the slave address of the device to be connected to the communications bus. If you are using rm_hs300x, specify 0x44.
COMMS_I2C_CFG_DEVICE(x)_CALLBACK (x = 0 - 4)	comms_i2c_user_callback(x) (x = 0 - 4)	Specify the name of the user callback function. When using rm_hs300x, specify rm_hs300x_callback(y) (y = 0 - 1).

5.2.4 RZ Family

Select the rm_comms_i2c stack on the [Stack] tabbed page of the FSP Configurator, and the configurable items will be shown on the [Properties] tabbed page.

The following items and values can be specified.

Table 5-8 Communication Driver Settings for the RA Family MCU

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. The module name must conform with the C language standard.
Semaphore Timeout	0xFFFFFFFF	For an RTOS project, specify the time of semaphore timeout.
Slave Address	0x44	Specify the slave address. When rm_hs300x is used, this value is automatically specified and cannot be modified.
Address Mode	7-Bit	Specify the number of slave address bits. When rm_hs300x is used, this value is automatically specified and cannot be modified.
Callback	rm_hs300x_callback	Specify the name of the user callback function. When rm_hs300x 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 processing for blocking.
Recursive Mutex for Bus	Unuse Use	For an RTOS project, enable or disable recursive operation when blocking is enabled.

5.3 I2C Driver Settings

5.3.1 RA Family

Select the r_iic_master or r_sci_i2c stack on the [Stack] tabbed page of the FSP Configurator, and the configurable items will be shown on the [Properties] tabbed page.

The following items and values can be specified.

Table 5-9 r_iic_master Settings for the RA Family MCU

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 in transmission and reception.
10-bit slave addressing	Enabled Disabled	Specify whether to support 10-bit slave addresses. If you are using rm_hs300x, 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 bit rate. If you are using rm_hs300x, select "Standard" or "Fast-mode".
Rise Time (ns)	120	Specify the SCL rise time to suit the specifications of the target board to be used.
Fall Time (ns)	120	Specify the SCL fall time to suit 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 any setting that is made here is overwritten by the setting in rm_comms_i2c.
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 any setting that is made here is overwritten by the setting in rm_comms_i2c.
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)	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	
Pins		
SDA	Pxxx	The pin numbers to be used by the driver are displayed. Use the [Pins] tabbed page to modify the pin configuration if this is required.
SCL	Pxxx	

Table 5-10 r_sci_i2c Settings for the RA Family MCU

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 in transmission and reception.
	Disabled	
10-bit slave addressing	Enabled	Specify whether to support 10-bit slave addresses. If you are using rm_hs300x, 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 any setting that is made here is overwritten by the setting in rm_comms_i2c.
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 any setting that is made here is overwritten by the setting in rm_comms_i2c.
	10-bit	
Rate	Standard	Specify the bit 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 division value of the noise filter 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 the DTC is to be used, 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 if this is required.
SCL	Pxxx	

5.3.2 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 5-11 r_riic_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 – 2)	Not supported Supported	Specify whether to support the operation of channel x.
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 One IIC phi Two IIC phi Three IIC phi Four IIC phi	Specify the digital filter for input signals.
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.
Master arbitration lost detection function for CHx (x = 0 – 2)	Unused Used	Specify whether to use the master arbitration lost detection function.
Address y format for CHx (x = 0 – 2, y = 0 – 2)	Not 7 bit address format 10 bit address format	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.
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 Used	Specify whether to use the general call function.

CHx RXI INT Priority Level (x = 0 – 2)	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 priority level of the reception interrupt.
CHx RXI INT Priority Level (x = 0 – 2)	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 priority level of the transmission interrupt.
CHx EEI INT Priority Level (x = 0 – 2)	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 priority level of the error interrupt.

CHx TEI INT Priority Level (x = 0 – 2)	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 priority level of the transmission end interrupt.
Timeout function for CHx (x = 0 – 2)	Unused Used	Specify whether to use the timeout function.
Timeout detection time for CHx (x = 0 – 2)	Long mode Short mode	Specify the time for timeout detection.
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 5-12 r_sci_iic_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 – 12)	Not supported	Specify whether to support the operation of channel x.
	Supported	
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	Specify the interrupt priority level.
	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)	
Digital noise filter (NFEN bit) for CHx (x = 0 – 12)	Disable	Specify whether to use the digital noise filter.
	Enable	
Noise Filter Setting Register (NFCS bit) for CHx (x = 0 – 12)	The clock divided by 1	Specify the function of the digital noise filter.
	The clock divided by 2	
	The clock divided by 4	
	The clock divided by 8	
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	Specify whether to include the pin function settings in the code to be generated.
	Include port setting	
Resources		
SSDAx Pins	Checked	Specify the pins to be used. Select the checkboxes for the desired pins.
SSCLx Pins	Checked	

5.3.3 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 5-13 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_hs300x, select IICxx.
	UARTxx	
	CSIxx	
	IICxx	
IICxx		
Transfer rate	1000000	Specify the bit rate. If you are using rm_hs300x, 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.

5.3.4 RZ Family

Select the r_iic_master or r_sci_i2c stack on the [Stack] tabbed page of the FSP Configurator, and the configurable items will be shown on the [Properties] tabbed page.

The following items and values can be specified.

Table 5-14 r_iic_master Settings for the RZ Family MCU

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 in transmission and reception.
	Disabled	
10-bit slave addressing	Enabled	Specify whether to support 10-bit slave addresses. If you are using rm_hs300x, select "Disabled".
	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	Specify the bit rate. If you are using rm_hs300x, select "Standard" or "Fast-mode".
	Fast-mode	
	Fast-mode plus	
Rise Time (ns)	120	Specify the SCL rise time to suit the specifications of the target board to be used.
Fall Time (ns)	120	Specify the SCL fall time to suit 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 any setting that is made here is overwritten by the setting in rm_comms_i2c.
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 any setting that is made here is overwritten by the setting in rm_comms_i2c.
	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	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	

6. Guide to 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.

6.1 RA Sample Project

Use the following procedures to modify a sample project.

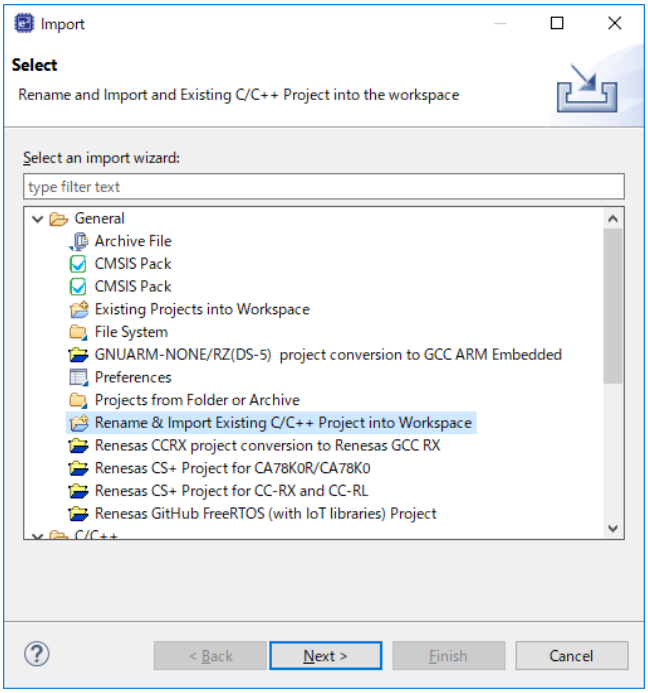
This section describes how to modify the "HS300x_RA6M4_NonOS" sample project so that it can be used on the EK-RA2E1 board as an example.

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

6.1.1 Importing the Sample Project

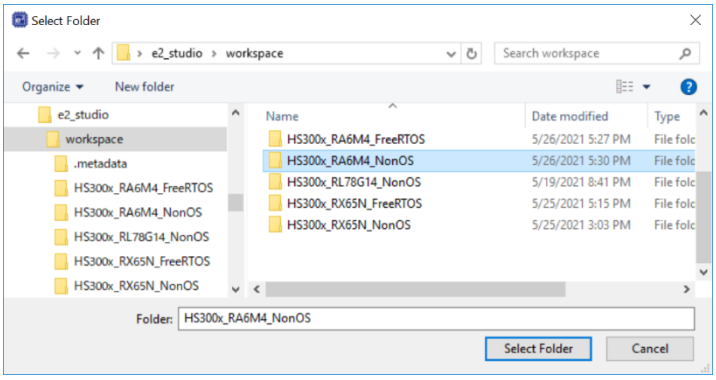
Select [Import] from the menu.

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

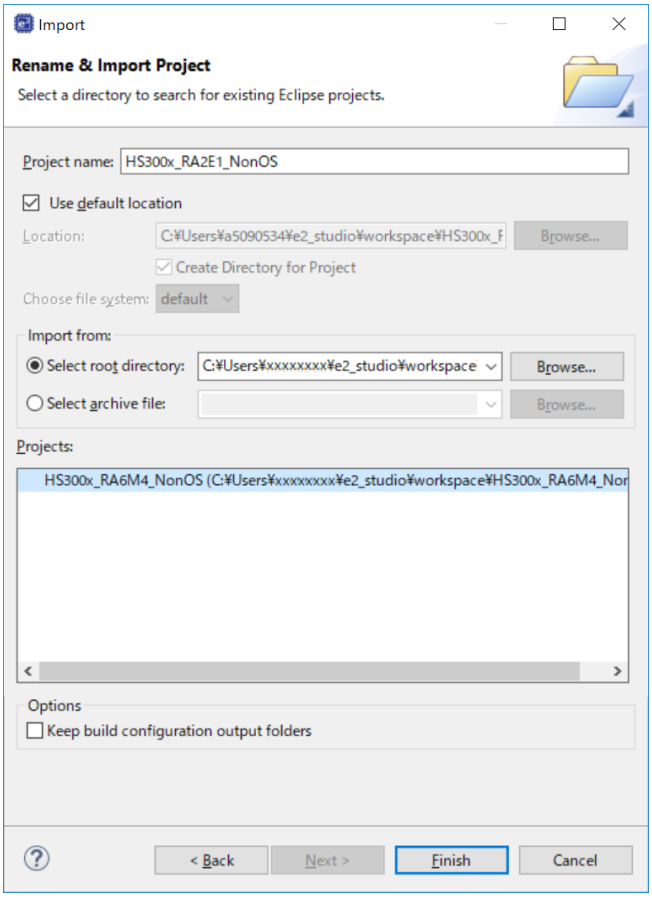


Press the [Browse] button to open the [Select Folder] window.

Select the folder of the original project for the current device from the 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.



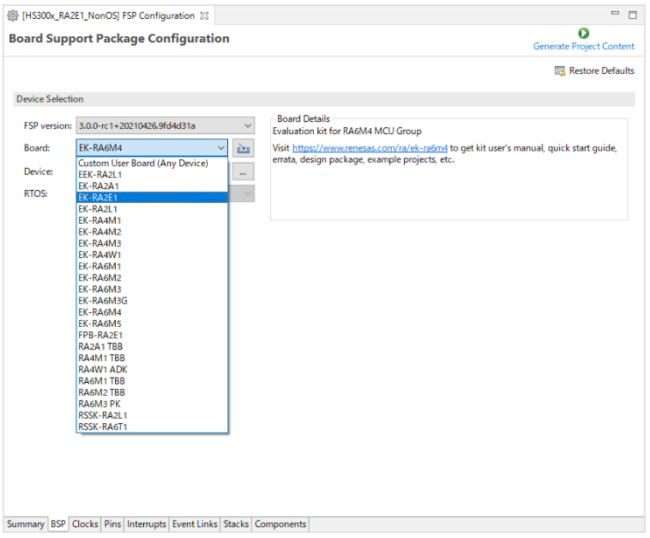
6.1.2 Modifying Settings of the FSP Configurator

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

Change the settings of "Board" and "Device" on the [BSP] tabbed page.

When selecting a Renesas board, you will only need to modify the "Board" setting.

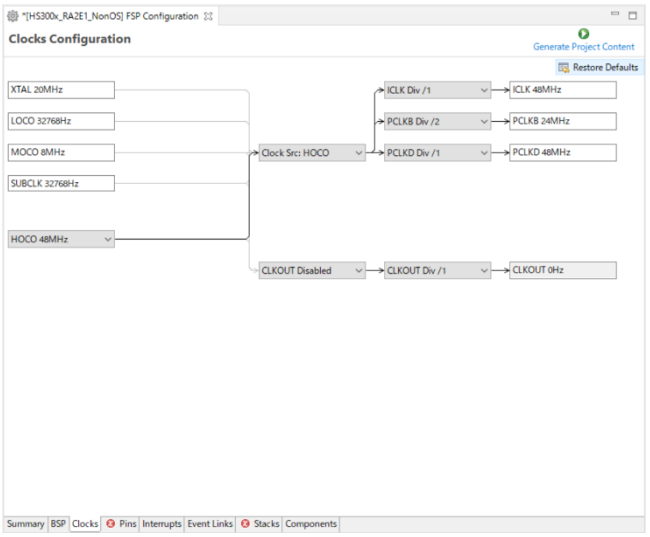
When selecting a board provided from other companies, change the "Board" setting to "Custom User Board (Any Device)" and then change the "Device" setting to the new device to be used.



Set up the clocks on the [Clocks] tabbed page.

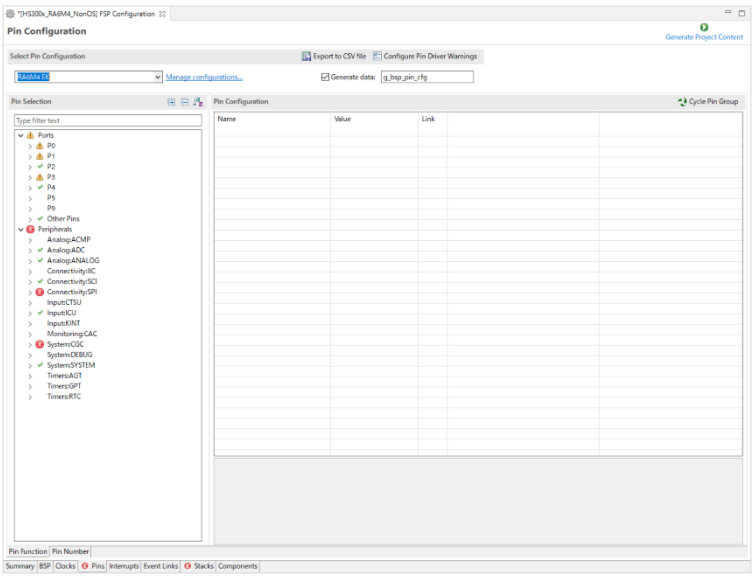
When "Custom User Board (Any Device)" is selected for "Board", set up the clocks to suit the specifications of the target board to be used.

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

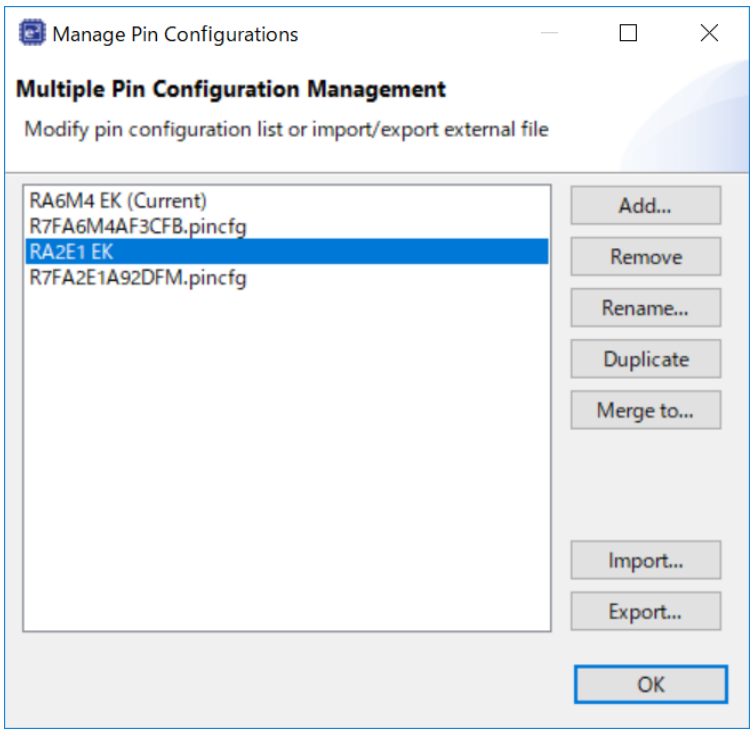


In the［Pins] tabbed page, modify the pin configuration according to the specifications of the target board to be used.

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



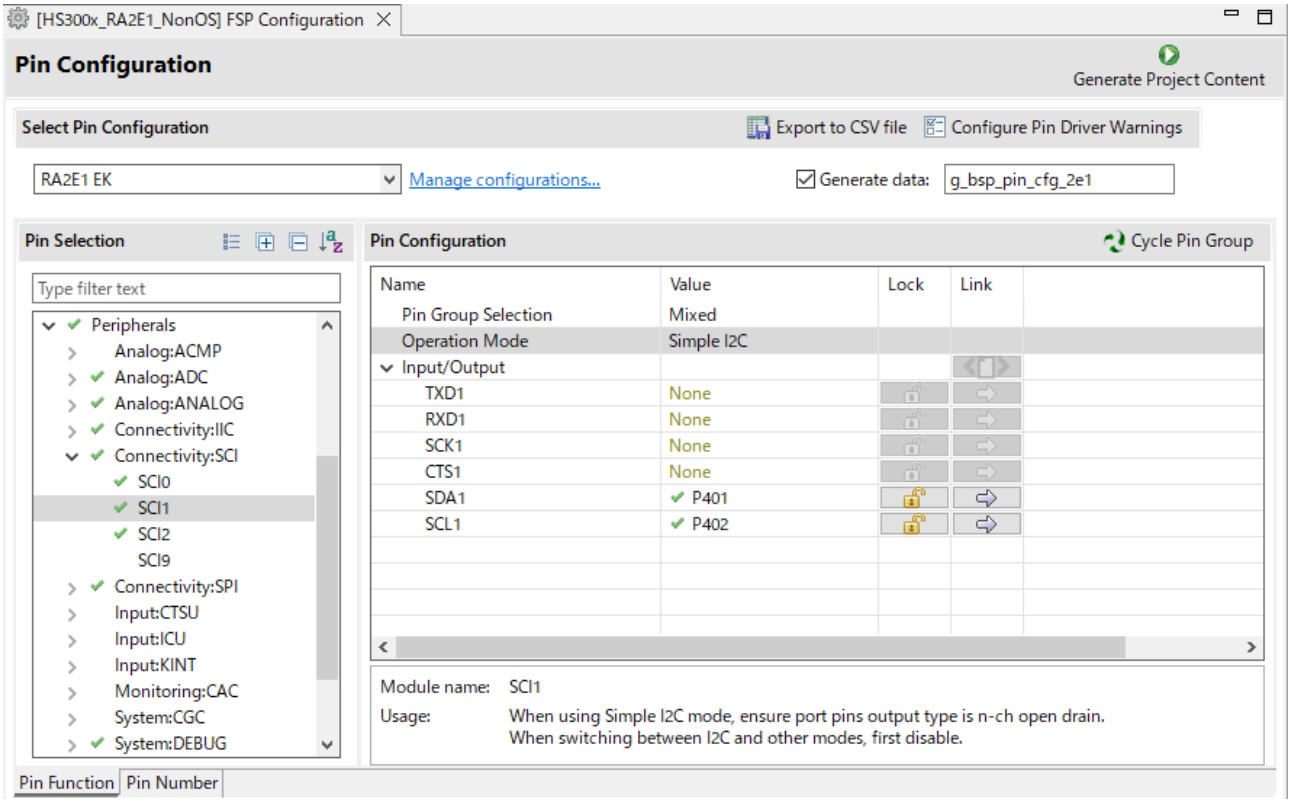
However, the "Select Pin Configuration" assignment will apply the SPI communication pin settings that support PMOD Type 2A on the EK-RA2E1 board.

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.

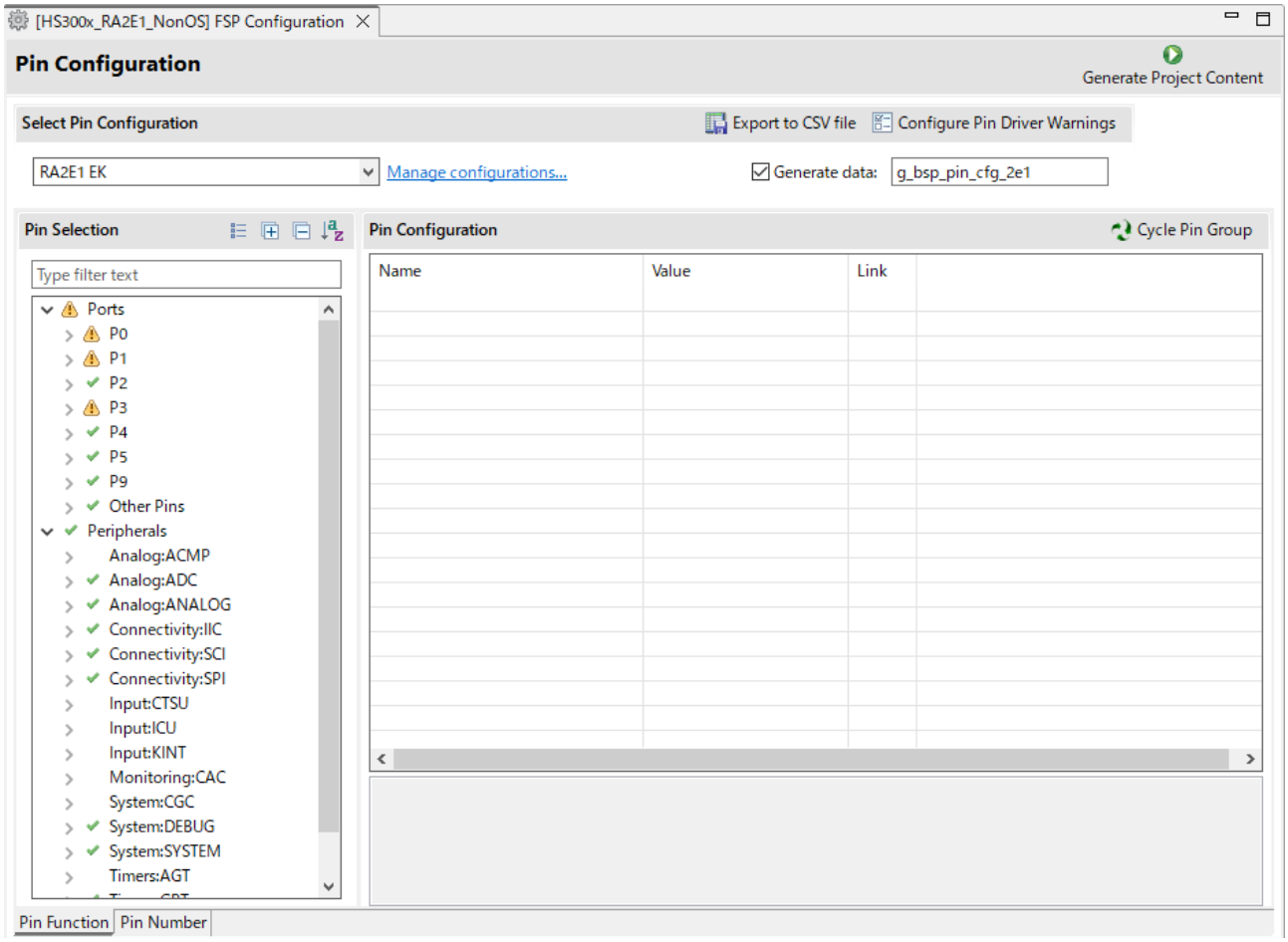
After automatic assignment of "Select Pin Configuration", reconfigure in "Pin Configuration".



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”.



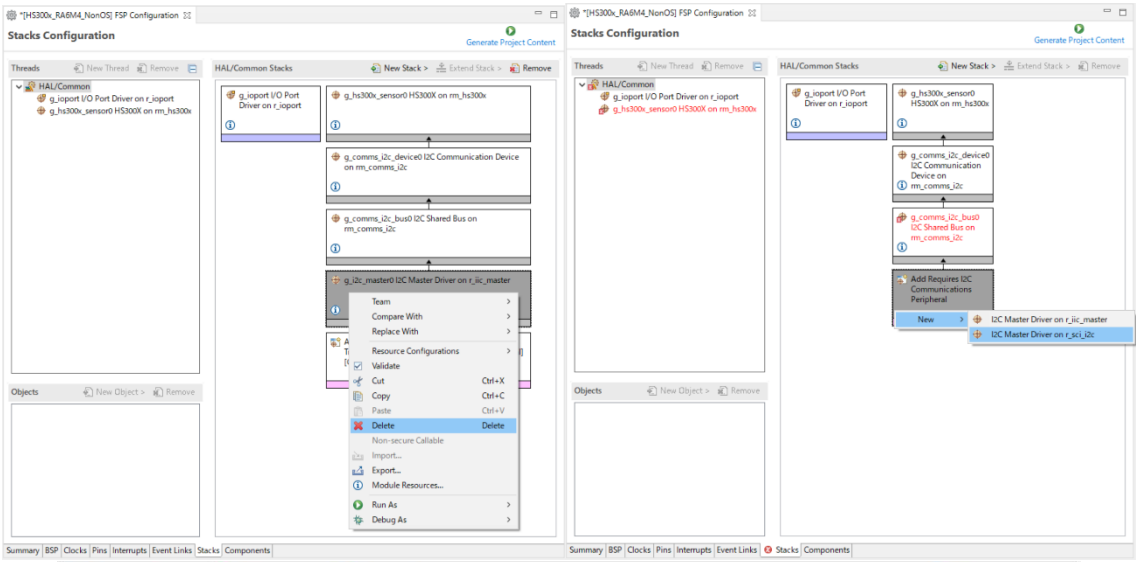
Modify the configuration of individual components on the [Stacks] tabbed page.

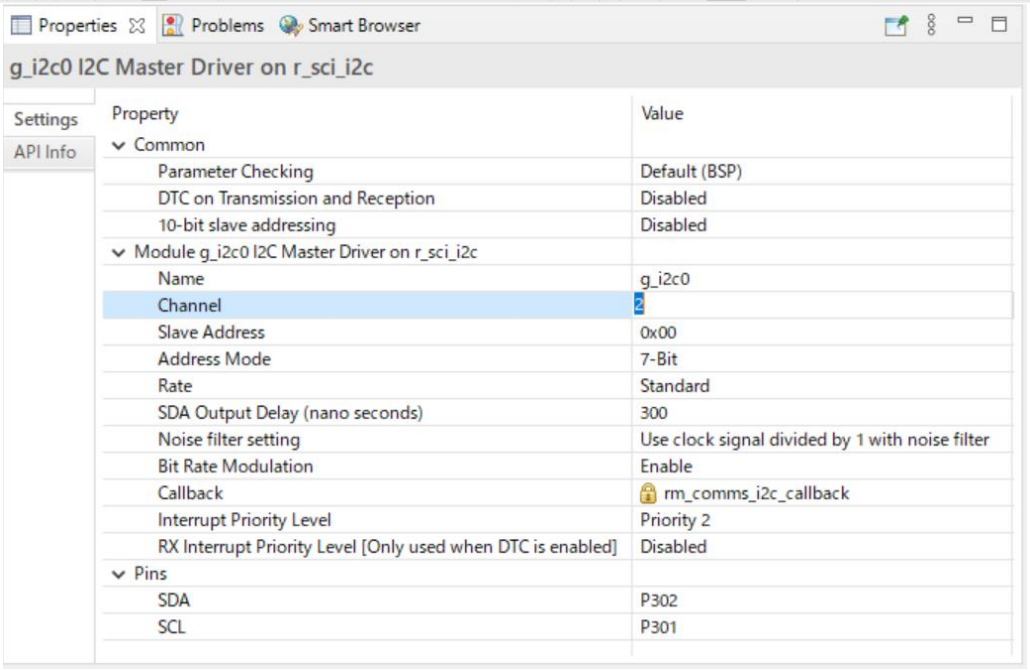
Modify the settings for r_iic_master or r_sci_i2c to suit the specifications of the target board.

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_mster" stack.

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

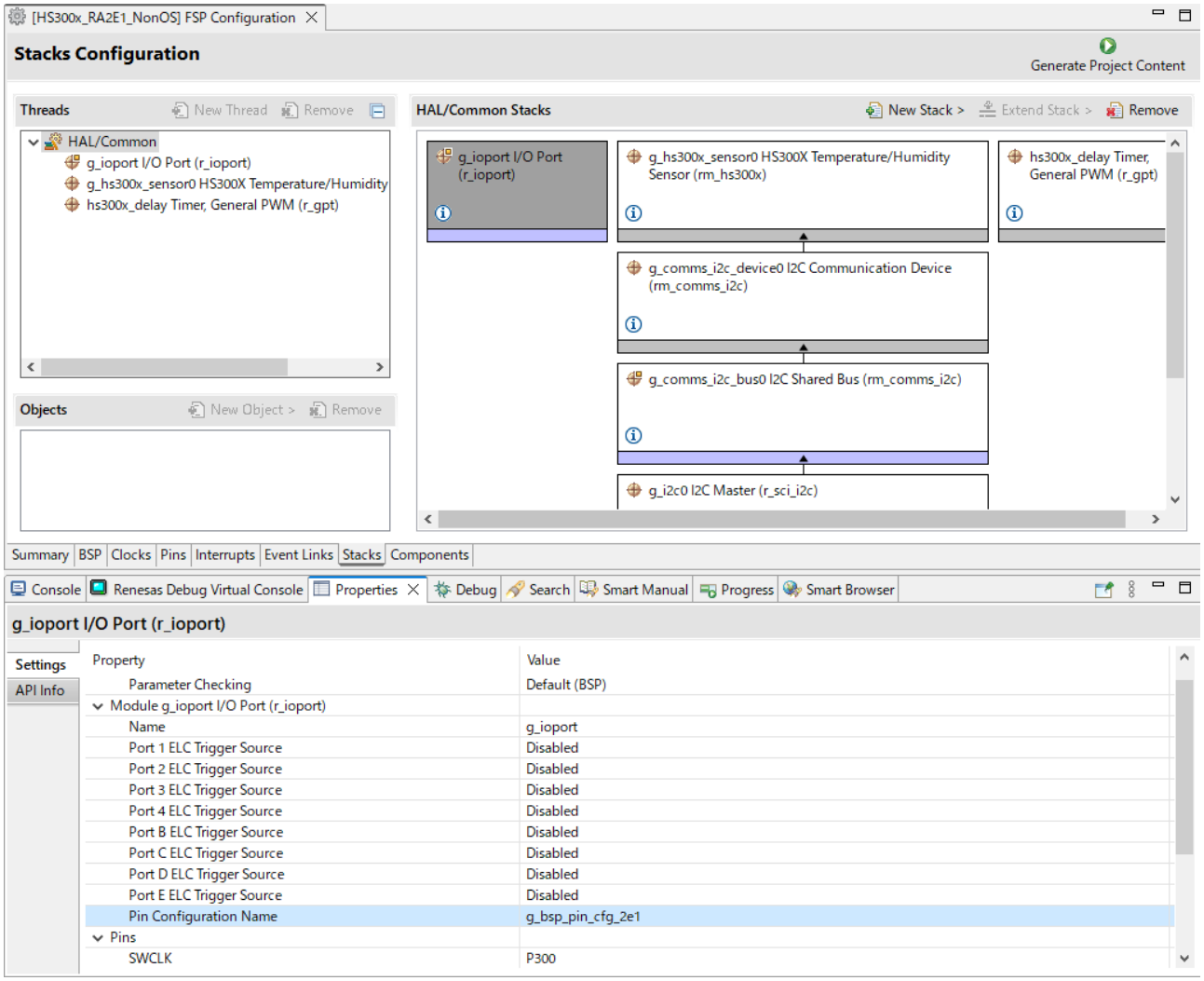
To use PMOD1, set "Channel" to "2". To use PMOD2, set it to "1".





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”.



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

Press [Generate Project Content] to generate files.

Build the project.

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

6.1.3 Changing toolchain setting

If you want to use a toolchain other than the GCC ARM Embedded toolchain, copy RA_HS300X.c (Non-OS) or hs300x_sensor_thread_entry.c, sensor_thread_common.c, and sensor_thread_common.c (FreeRTOS, Azure) from this project to create a new project.

6.2 RX Sample Project

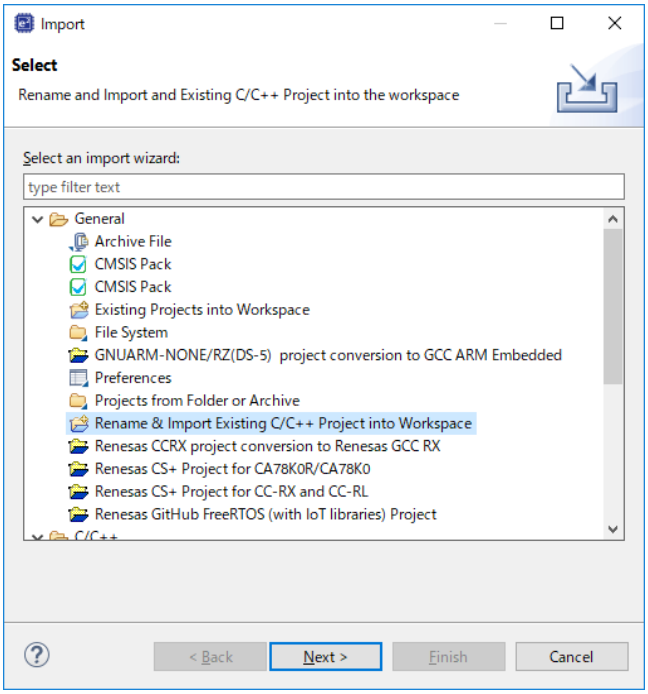
Use the following procedures to modify a sample project.

This section describes how to modify the "HS300x_RX65N_NonOS" sample project so that it can be used on the RSKRX231 board as an example.

6.2.1 Importing the Sample Project

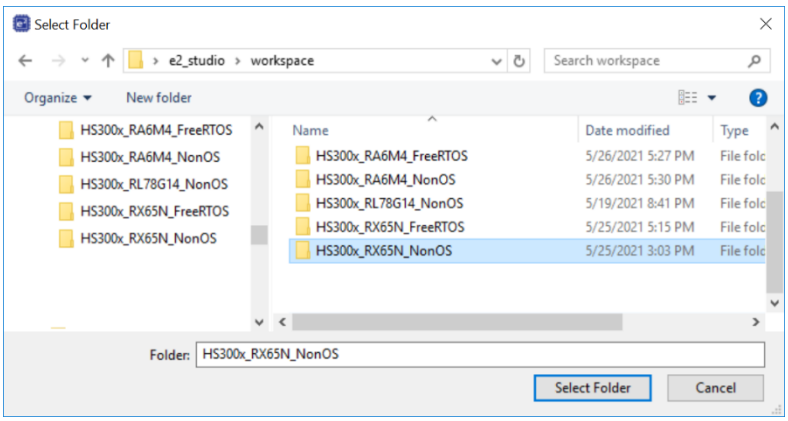
Select [Import] from the menu.

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

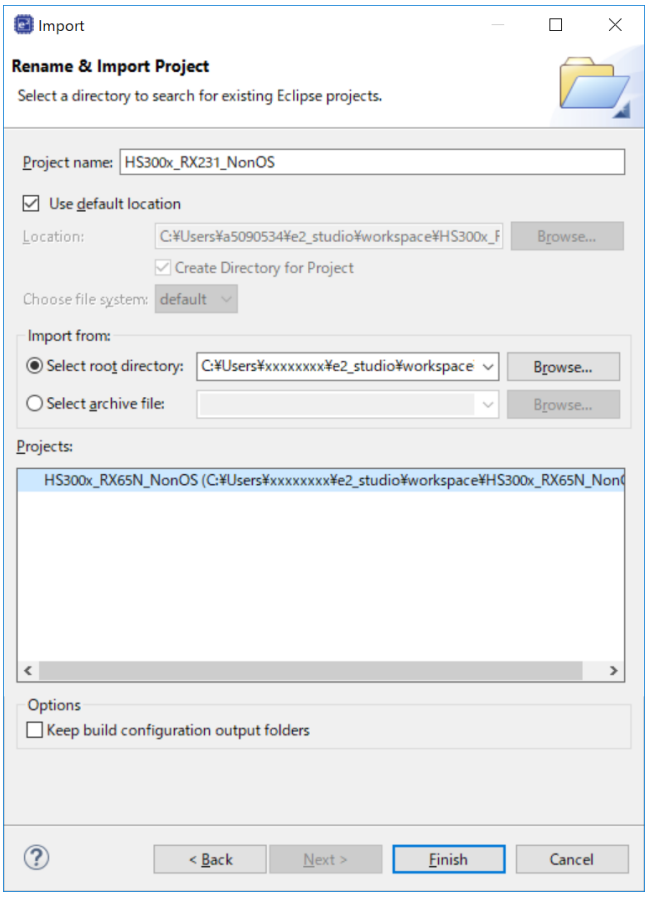


Press the [Browse] button to open the [Select Folder] window.

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

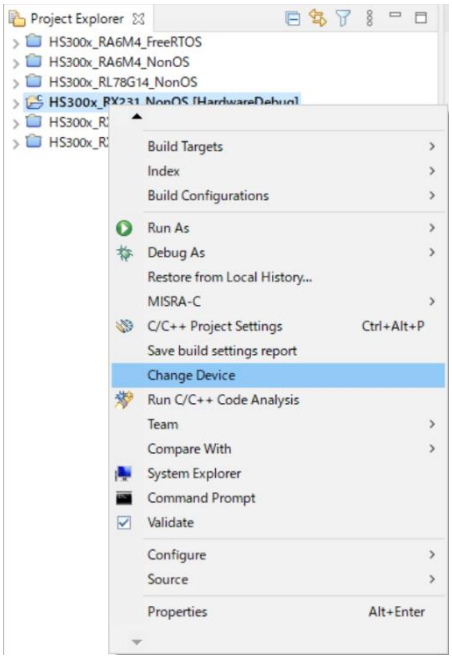


Enter the project name, select the original project for the current device, and press the [Finish] button.

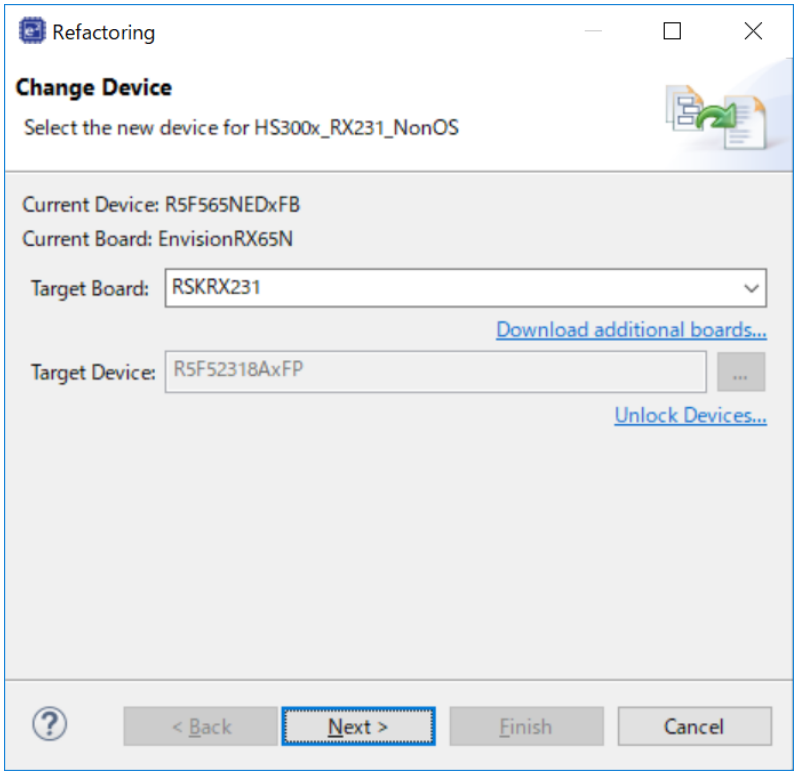


6.2.2 Changing the Device

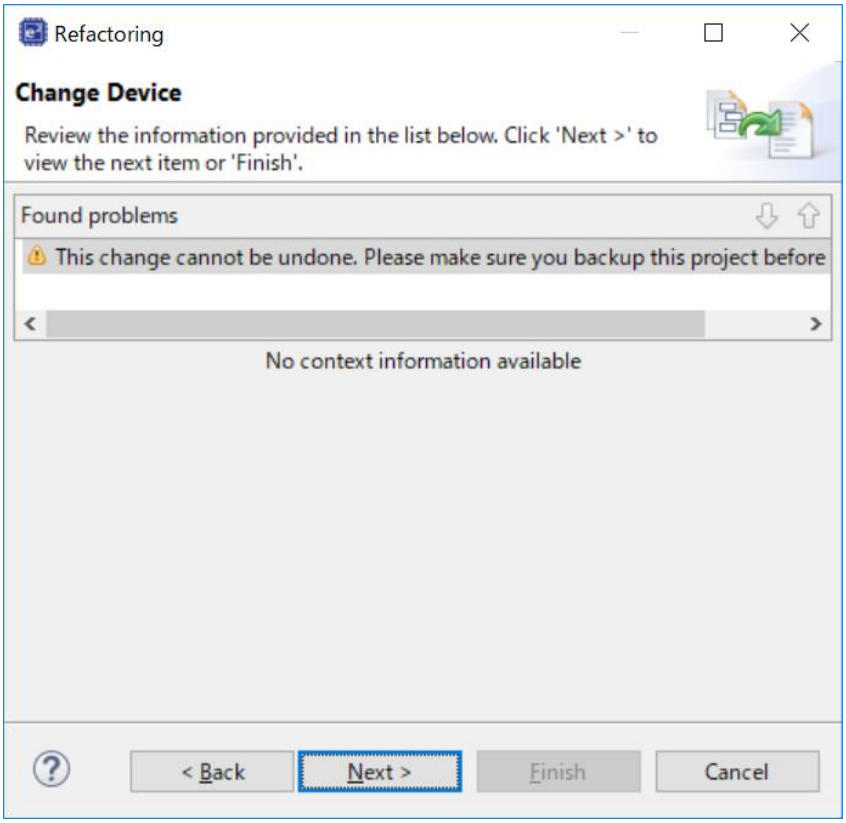
Select the imported project from the project tree and right-click on the name to open the context menu. Select [Change Device] from the menu.



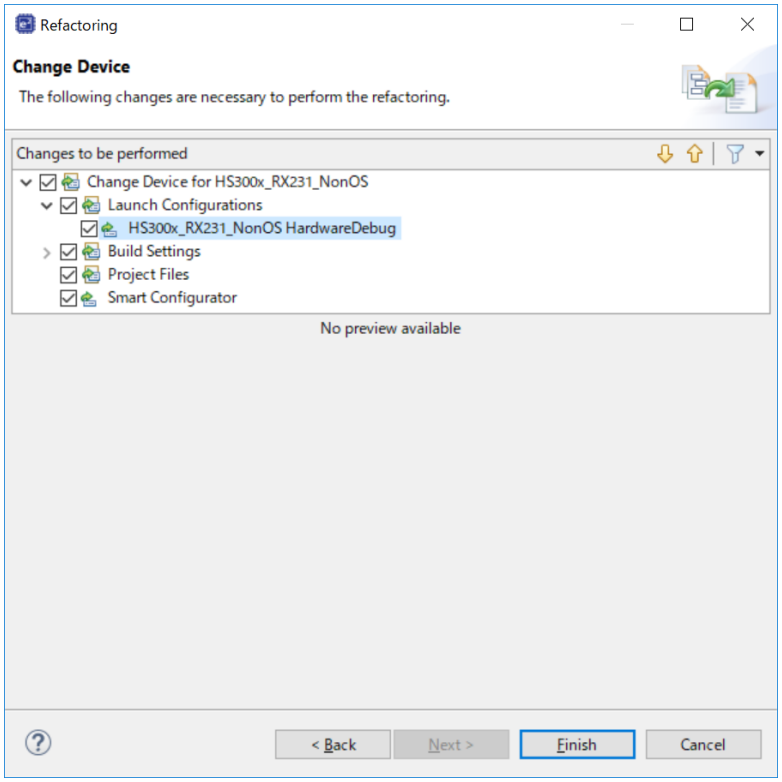
Select the desired board or device in the [Change Device] window and press the [Next] button.



If a warning message appears, read it and check if it will create a problem in continuing with the procedure. Press [Next] to go to the next step.

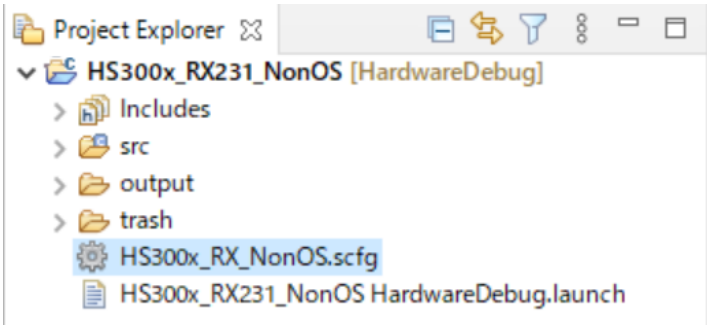


The changes you have made to the settings will be displayed. Press the [Finish] button to apply the changes to the project.

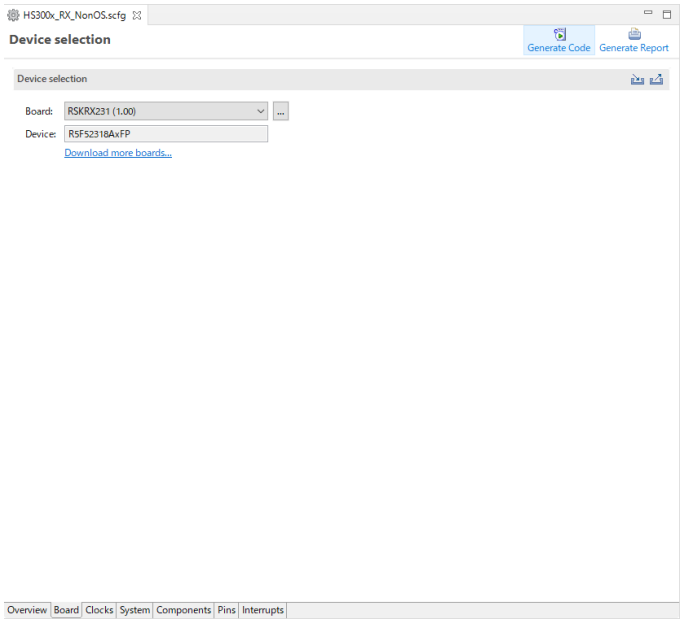


6.2.3 Modifying Settings of the Smart Configurator

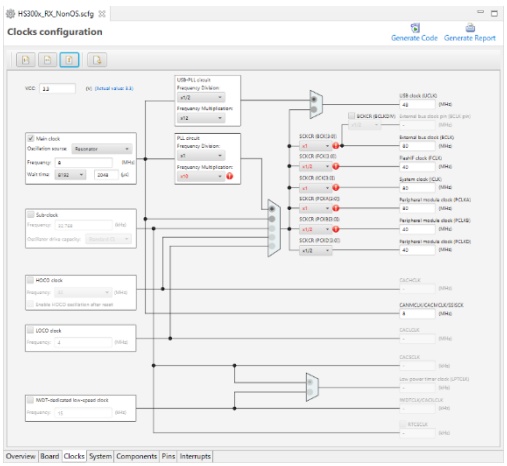
On the project tree, double-click on the .scfg file of the imported project in which the target device has been changed; this will open the Smart Configurator window.

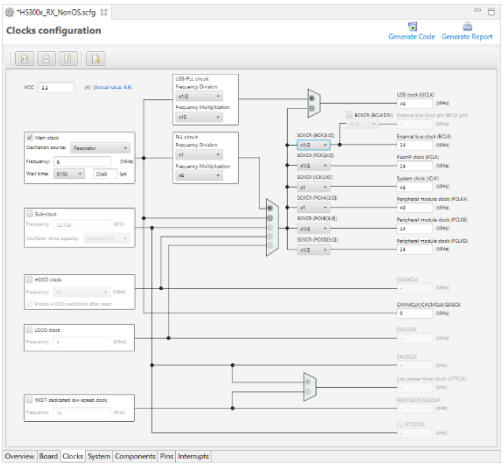


Select the [Board] tabbed page to check that the board and device have been changed correctly.



Set up the clocks on the [Clocks] tabbed page to suite the specifications of the target board to be used.

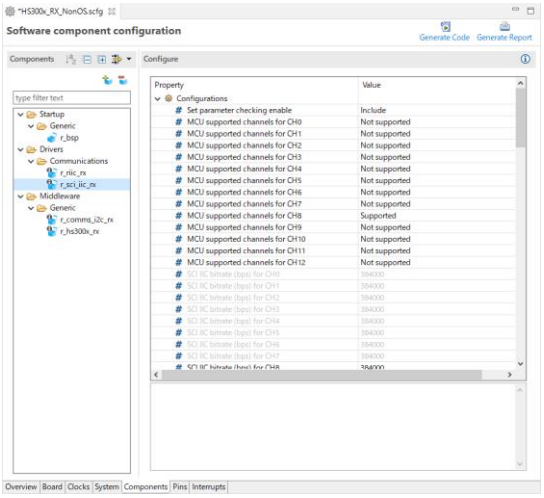


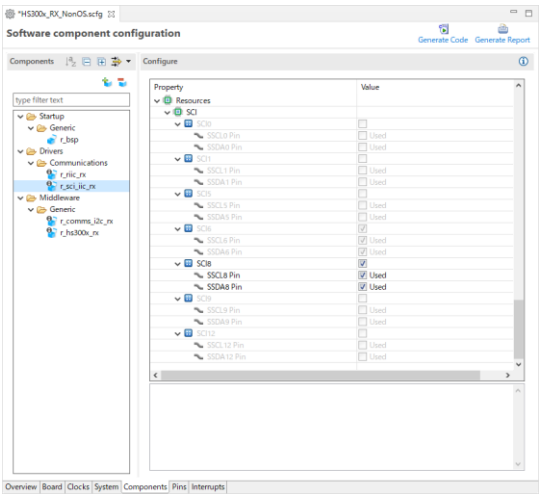


Modify the settings for individual components on the [Components] tabbed page to suit the specifications of the target board.

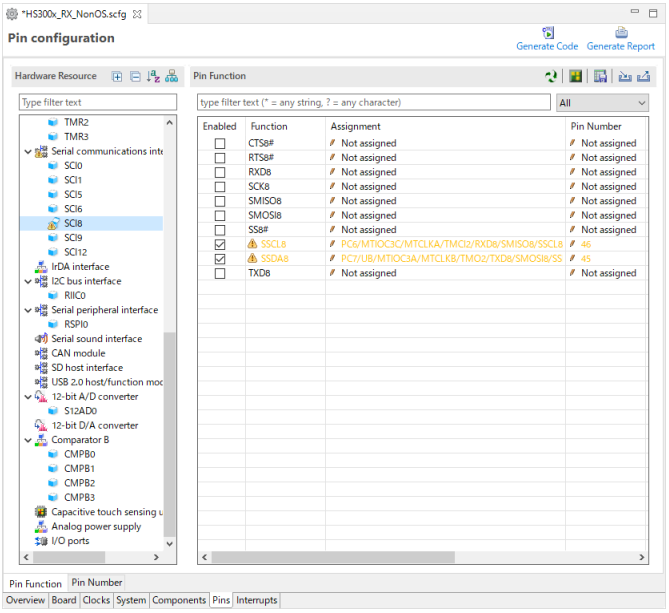
As SCI8 is assigned to PMOD on the RSK RX231 board, change the setting of "MCU supported channels for CH2" to "Not supported" and "MCU supported channels for CH8" to "Supported" in r_sci_iic_rx.

Select "SSCL8 Pin" and "SSDA8 Pin" for "SCI8" under "Resources".





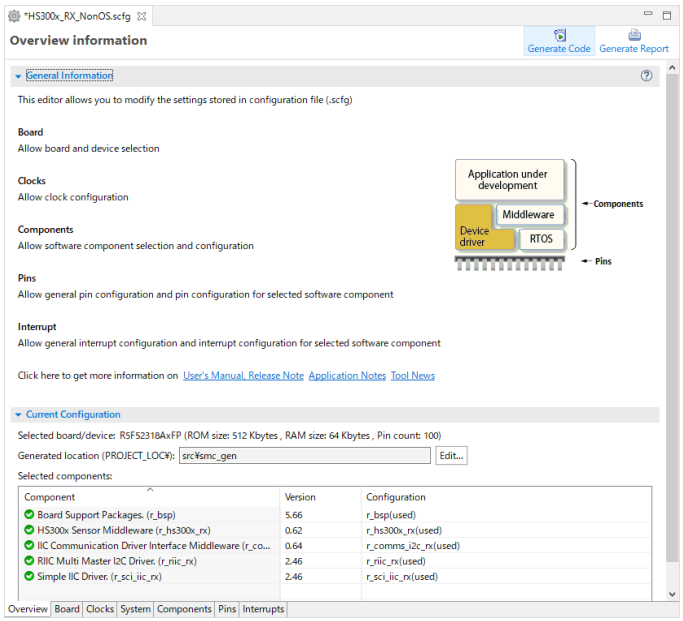
Open the [Pins] tabbed page and check that SCI8 pin functions are assigned to the desired pins in the [Pin function] panel.



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

Connecting a sensor board requires a board for converting the PMOD Type 2A interface to PMOD Type 6A.

Press the [Generate Code] icon to generate the code.



Build the project.

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

6.2.4 Changing toolchain setting

If you want to use a toolchain other than the CC-RX toolchain, copy RA_HS300X.c (Non-OS), or main.c and hs300x_sensor_thread_entry.c (FreeRTOS), or hs300x_sensor_thread_entry.c, sensor_thread_common.c, and sensor_thread_common.c (Azure) from this project to create a new project.

6.3 RL78 Sample Project

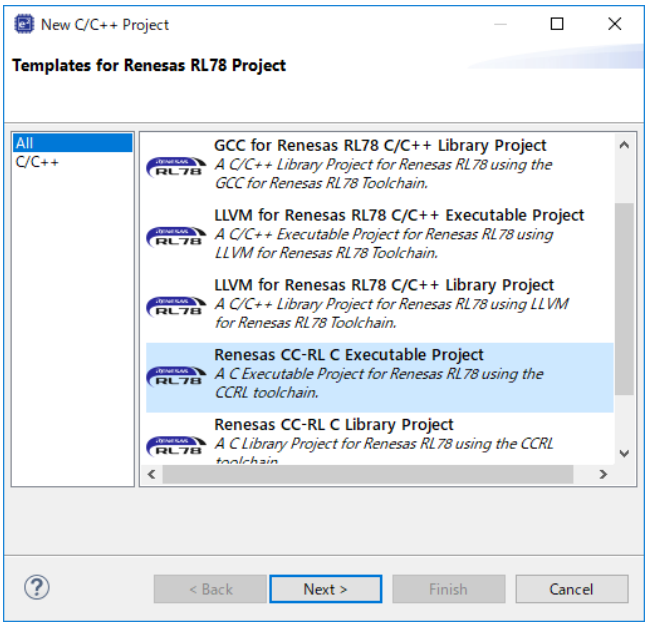
Changing the target device within the RL78 family requires creating a new project.

This section describes an example of creating a new project that can be used on the RSK RL78/G1G board.

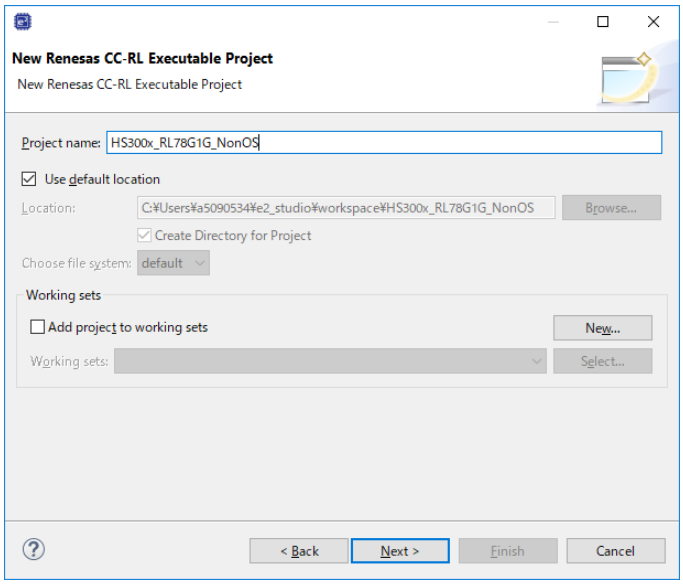
6.3.1 Creating a New Project

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

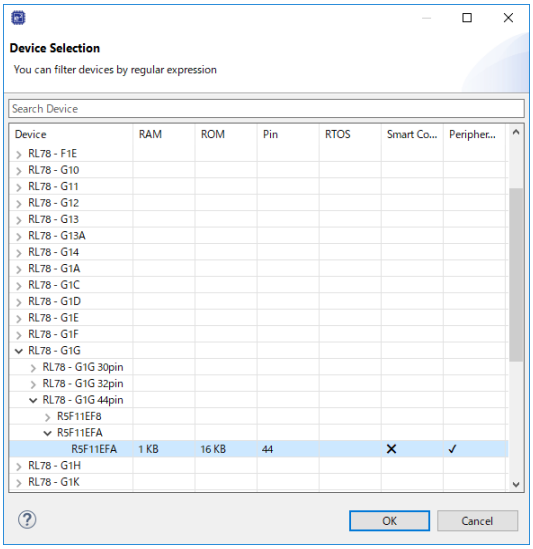
Select the template "Renesas CC-RL C Executable Project" and press the [Next] button.

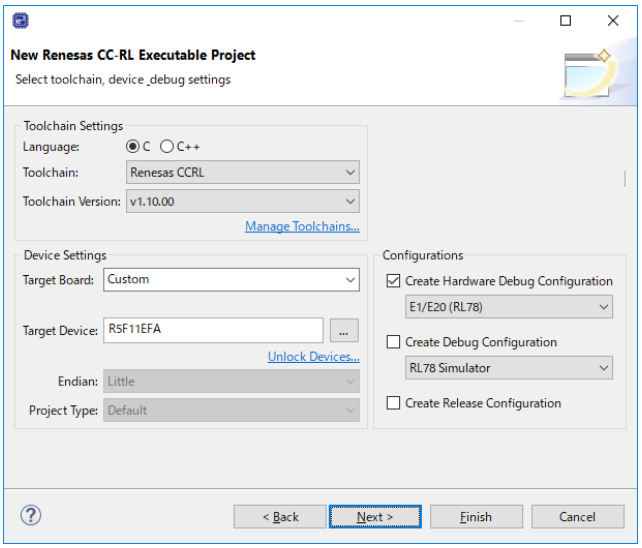


Enter the project name ("HS300x_RL78G1G_NonOS" in this example) and press the [Next] button.

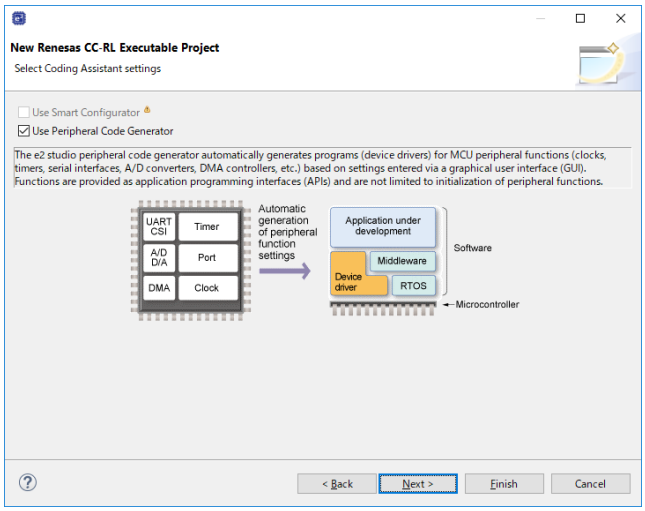


Change "Target Device" to the desired device (R5F11EFA in this example) and press the [Next] button.

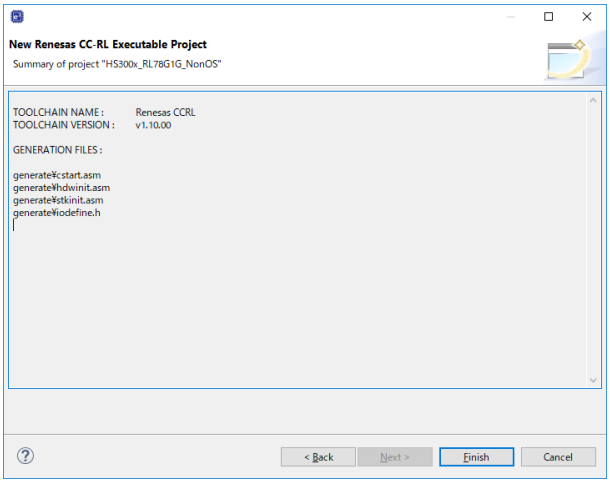




Select the checkbox for "Use Peripheral Code Generator" and press the [Next] button.

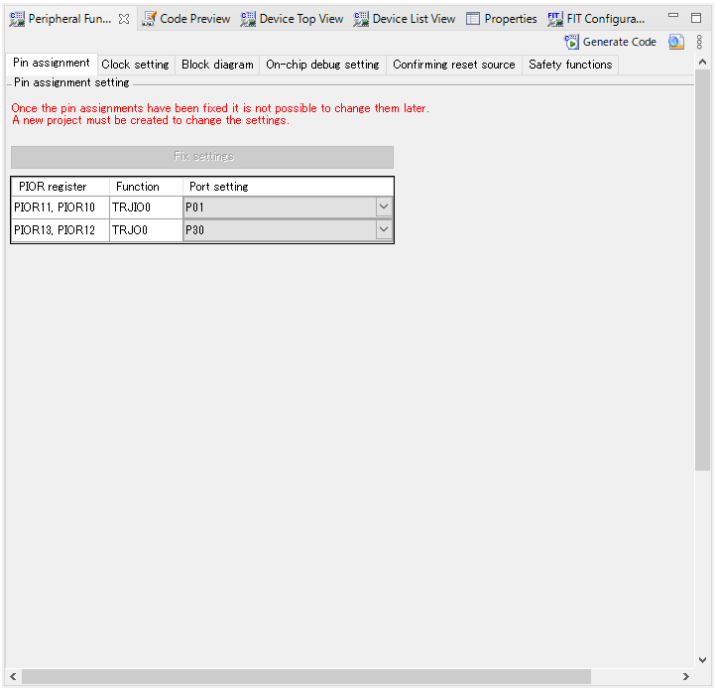


Press the [Finish] button.

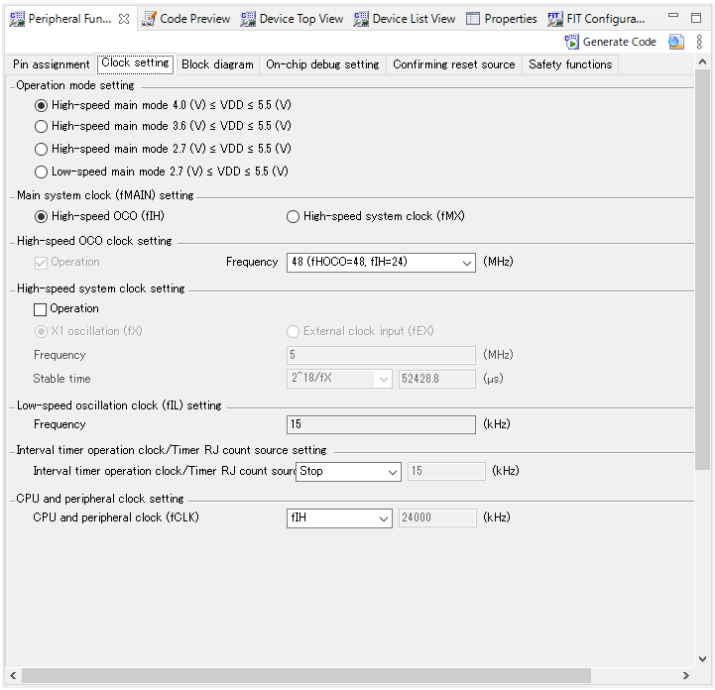


6.3.2 Settings of the Code Generator

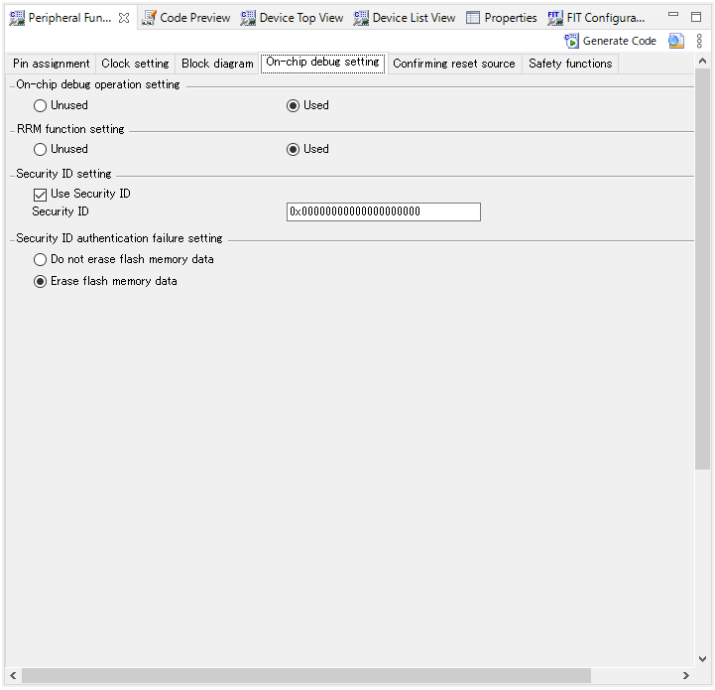
Modify the pin assignment on the [Pin assignment] tabbed page for "Peripheral Functions" to suit the specifications of the target board to be used.



Modify the clock settings on the [Clock setting] tabbed page for "Peripheral Functions" to suit the specifications of the target board.

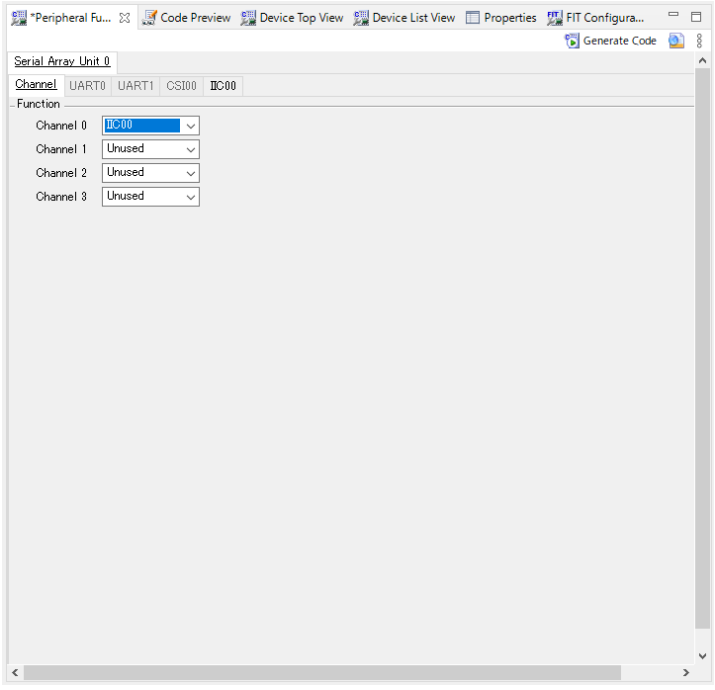


Select "Used" for "On-chip debug operation setting" on the [On-chip debug setting] tabbed page for "Common/Clock Generator".

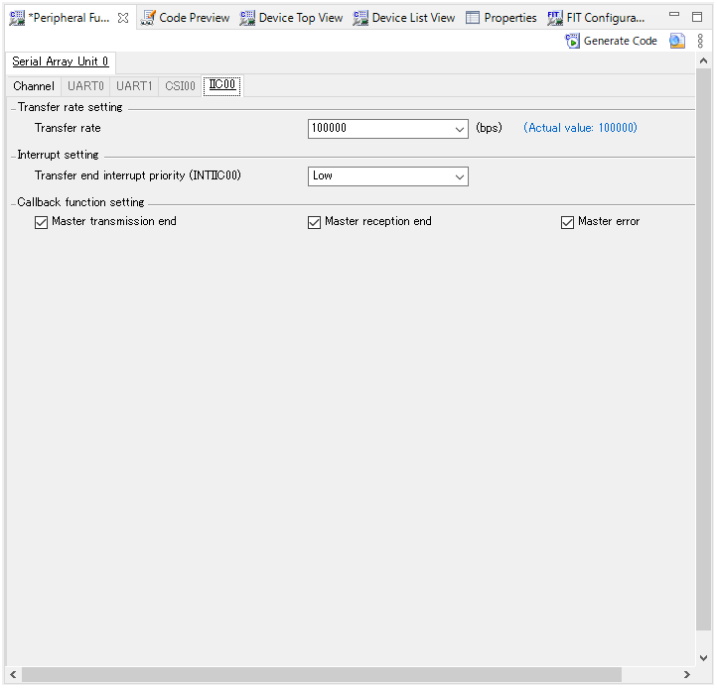


To use the serial array unit, set the channel assigned to PMOD on the target board to "IICxx" on the [Serial Array Unit] or [Serial] tabbed page.

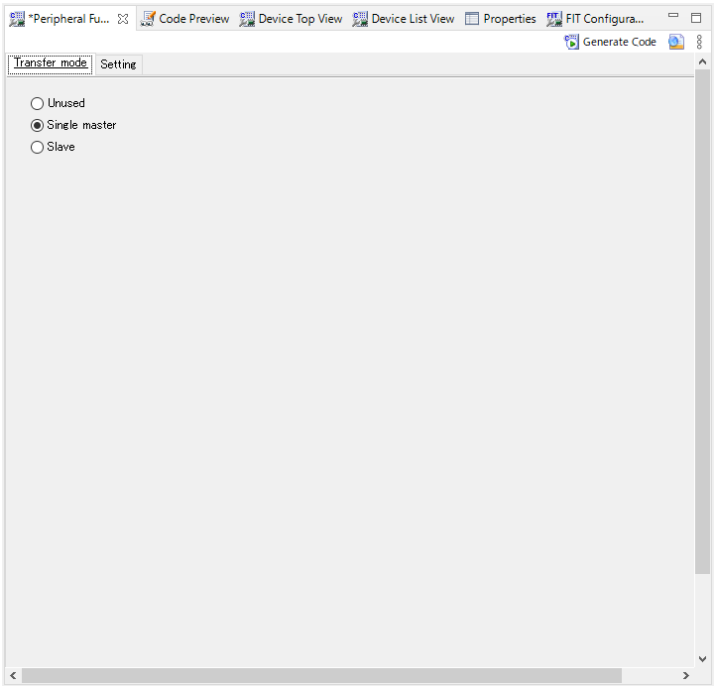
Note: The corresponding pin must be selected as N-ch by [Port].



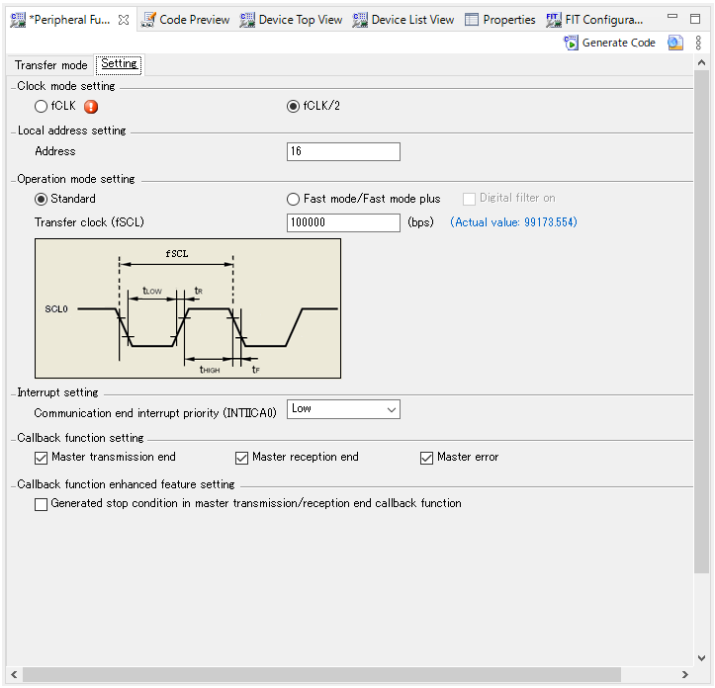
On the tabbed page for IICxx enabled in the serial array unit, set "Transfer rate" to 400000 or 100000, set "Transfer end interrupt priority" to a desired value, and enable all functions under "Callback function setting".



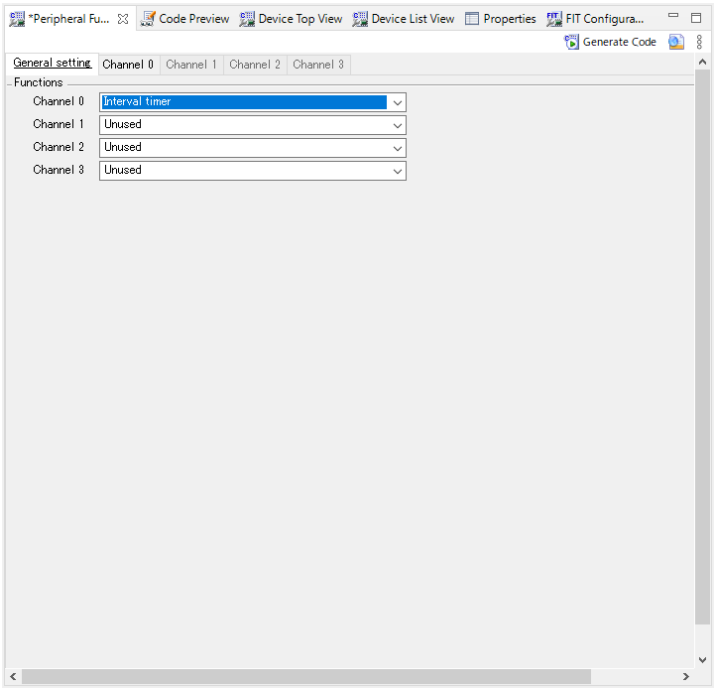
To use the serial interface IICA, select "Single master" on the [Transfer mode] tabbed page for the channel assigned to the PMOD interface on the target board in the [Serial Interface IICA] or [Serial] setting window.



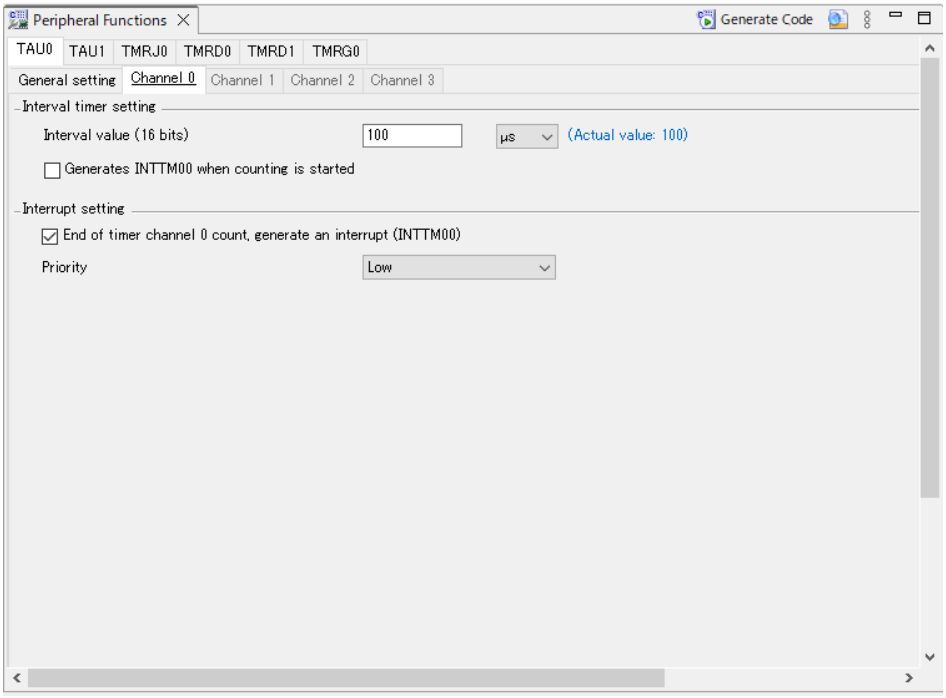
On the [Setting] tabbed page for the channel set as a single master in the previous step, set "Operation mode setting" to either the combination of "Fast mode" and "400000" or the combination of "Standard" and "100000", set the interrupt priority to the desired level, enable all functions under "Callback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



On 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.



Press the [Code Generate] button to generate the code.

6.3.3 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.

Directive 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 as 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 hs300x_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 */
    hs300x_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.

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_hs300x_sensor0_quick_setup(void);
void start_hs300x_demo(void);
/* End user code. Do not edit comment generated here */
```

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

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

    /* Open HS300x */
    g_hs300x_sensor0_quick_setup();

    while (1U)
    {
        start_hs300x_demo();
    }
```

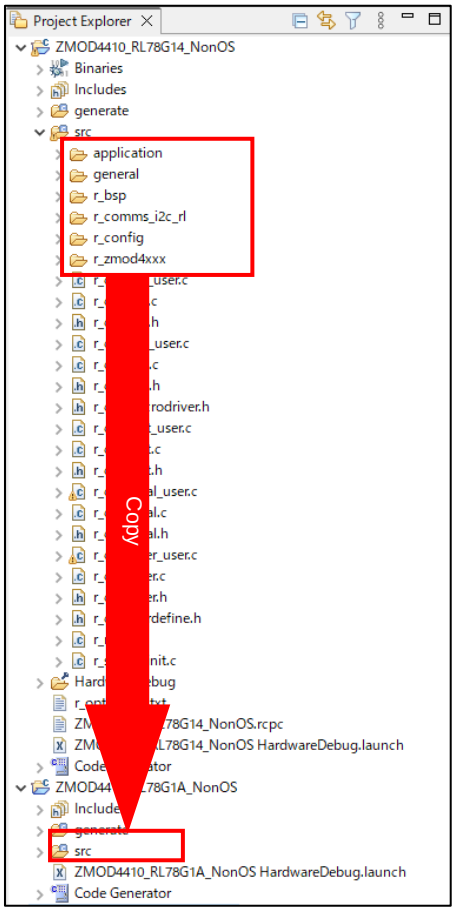
Define of the g_comms_i2c_bus0_quick_setup() function and the demo_err() function:

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

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

    }
}
```

6.3.4 Modifying Sample Source Files

Right-click on the "application" "general" "r_bsp" "r_comms_i2c_rl" "r_config" "r_hs300x" folder in the project tree of the "HS300x_RL78G14_NonOS" sample project and select [Copy] from the context menu.

After that, 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 the "r_comms_i2c_rl_config.h" file 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 5, 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

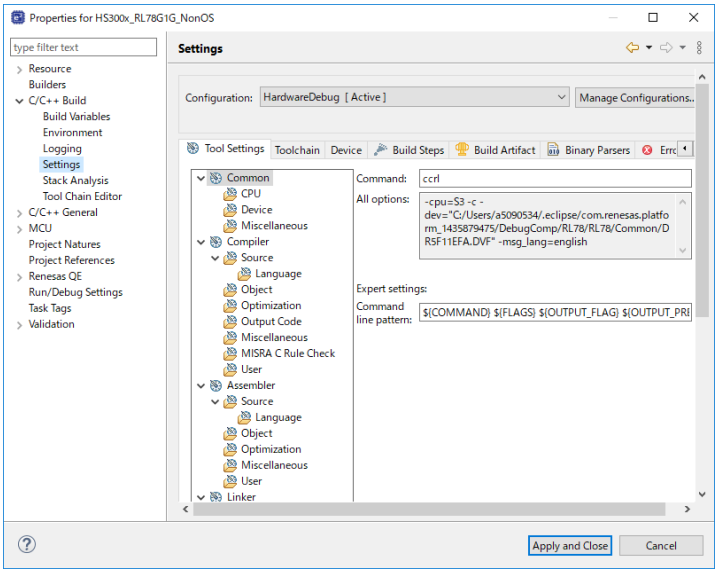
Modify "r_cg_serial.h" to "r_cg_sau.h" or "r_cg_iica.h":

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.

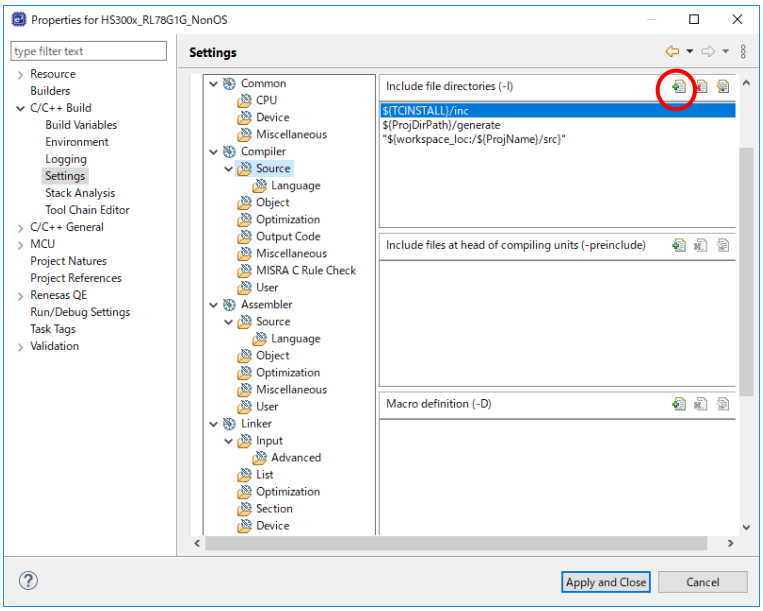
Select [C/C++ Build] → [Settings] in the [Properties] window to open the [Settings] panel.

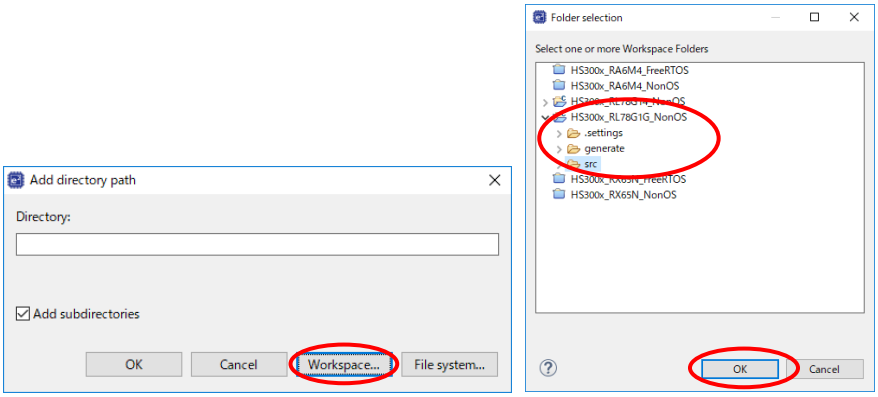


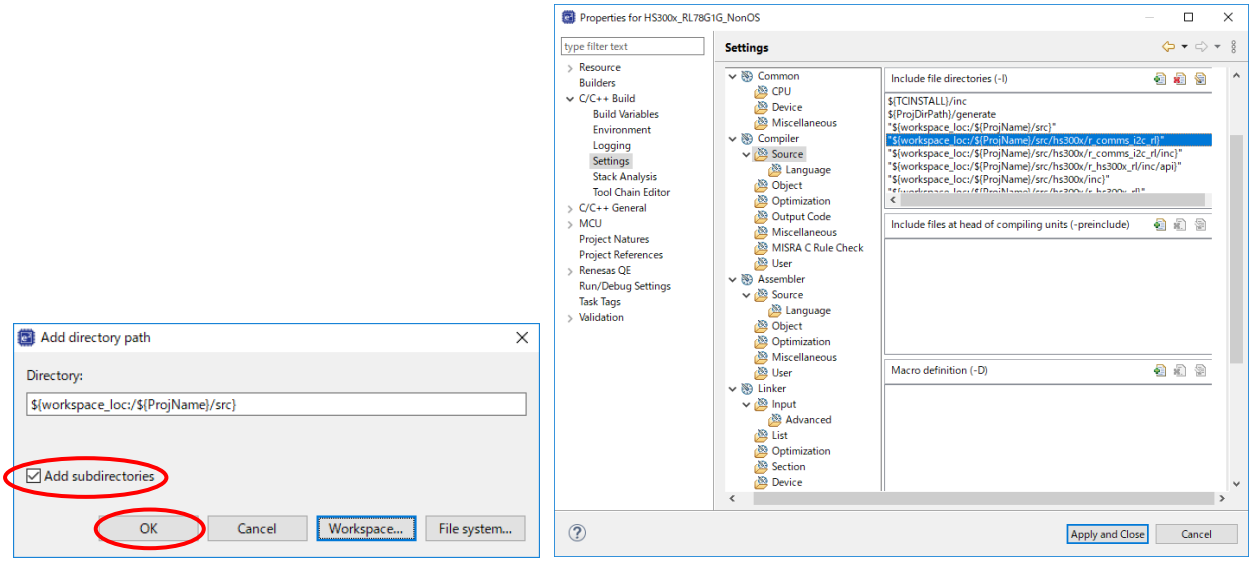
Select [Compiler] → [Source] on the [Tool Settings] tabbed page and press the [Add] icon.

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 from the list and press the [OK] button.

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

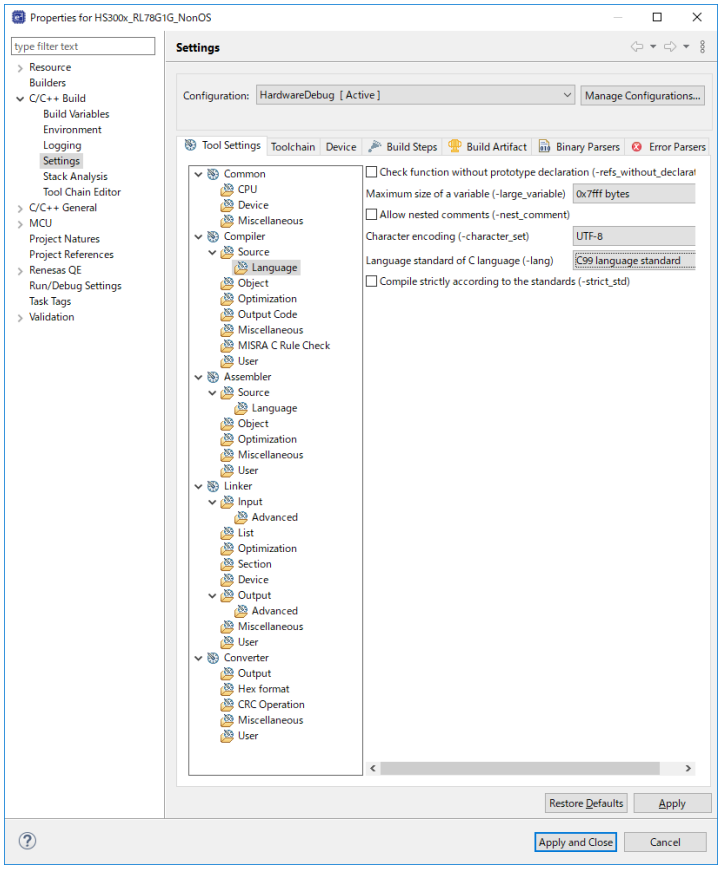






Select [Compiler] → [Source] → [Language] on the [Tool Settings] tabbed page and change the setting of "Language standard of C language" to "C99 language standard".

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



Build the project.

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

6.4 RZ Sample Project

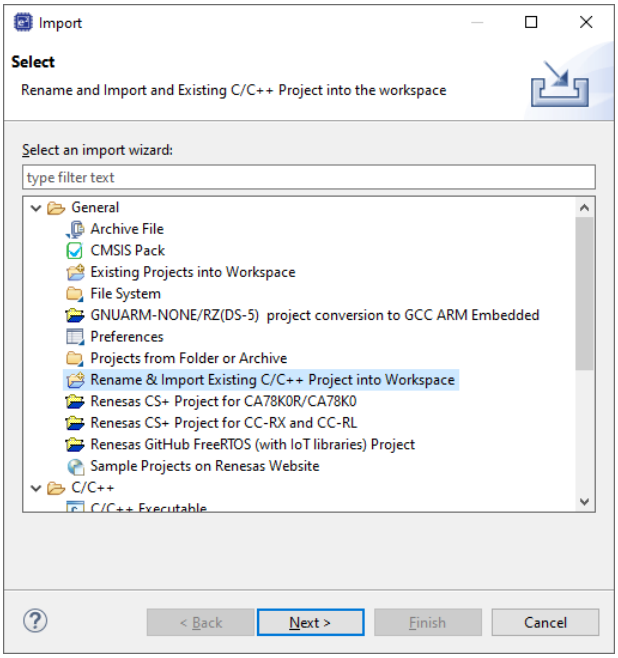
Use the following procedures to modify a sample project.

This section describes how to modify the "HS300x_RZG2L_NonOS" sample project so that it can be used on the RZ/G2L Evaluation Kit (SMARC) board as an example.

6.4.1 Importing the Sample Project

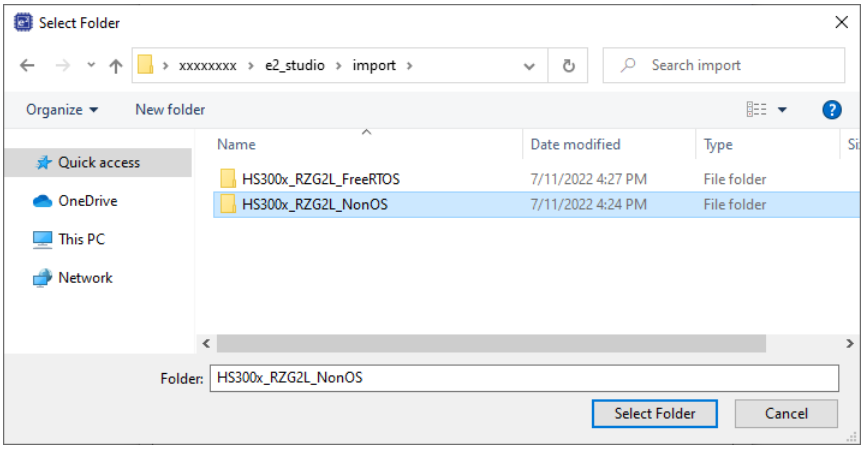
Select [Import] from the menu.

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

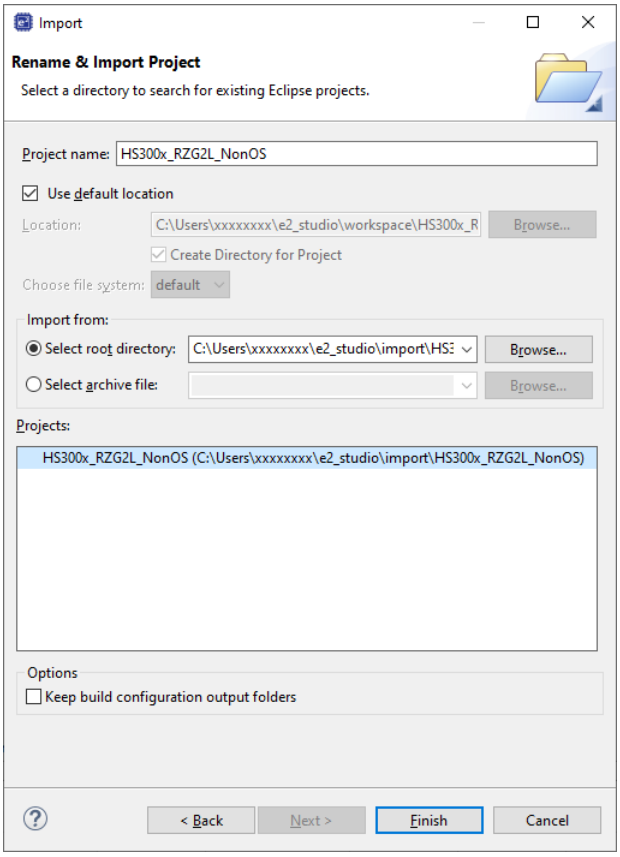


Press the [Browse] button to open the [Select Folder] window.

Select the folder of the original project for the current device from the 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.



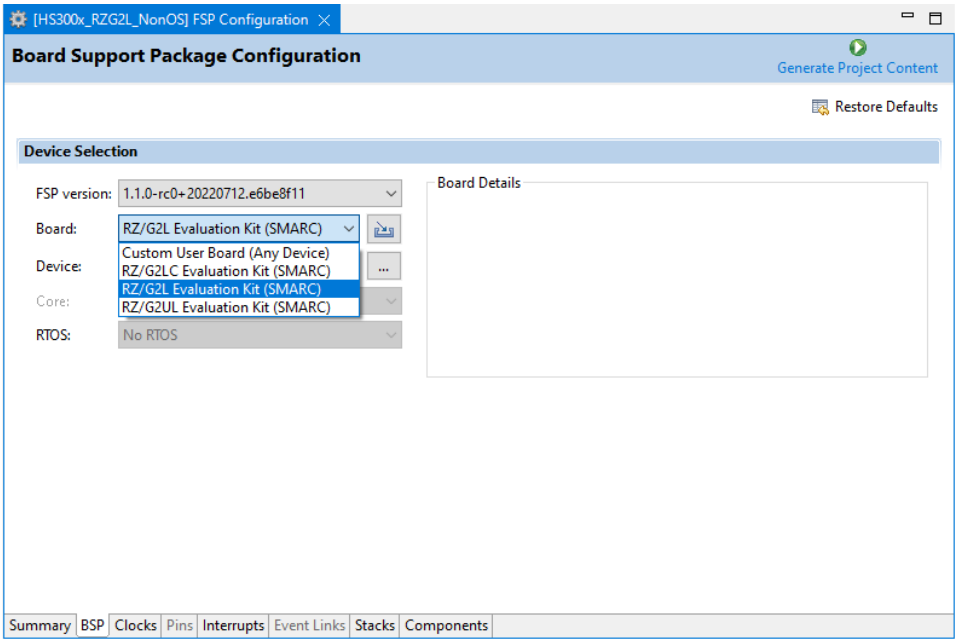
6.4.2 Modifying Settings of the FSP Configurator

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

Change the settings of "Board" and "Device" on the [BSP] tabbed page.

When selecting a Renesas board, you will only need to modify the "Board" setting.

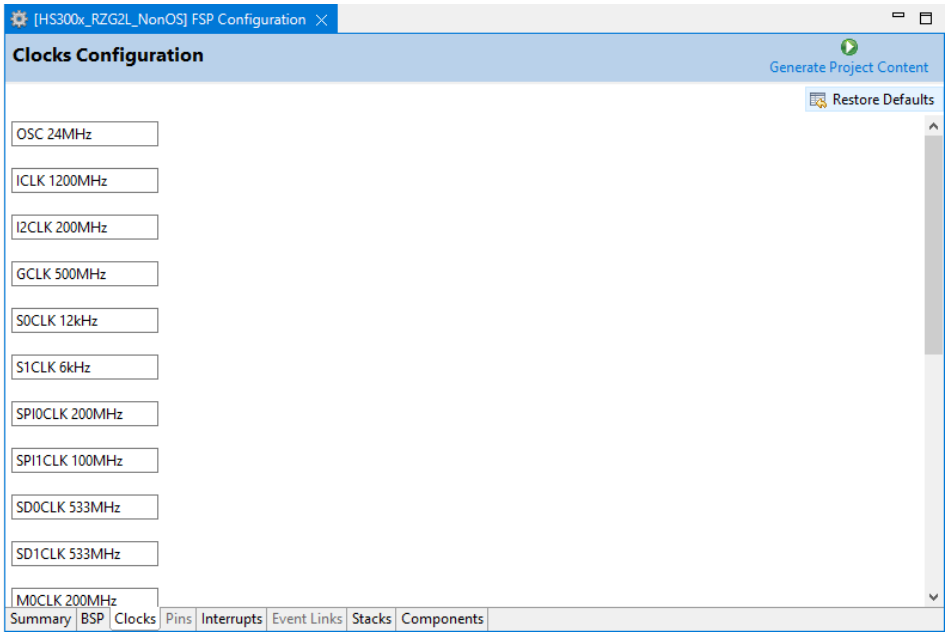
When selecting a board provided from other companies, change the "Board" setting to "Custom User Board (Any Device)" and then change the "Device" setting to the new device to be used.



Set up the clocks on the [Clocks] tabbed page.

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

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

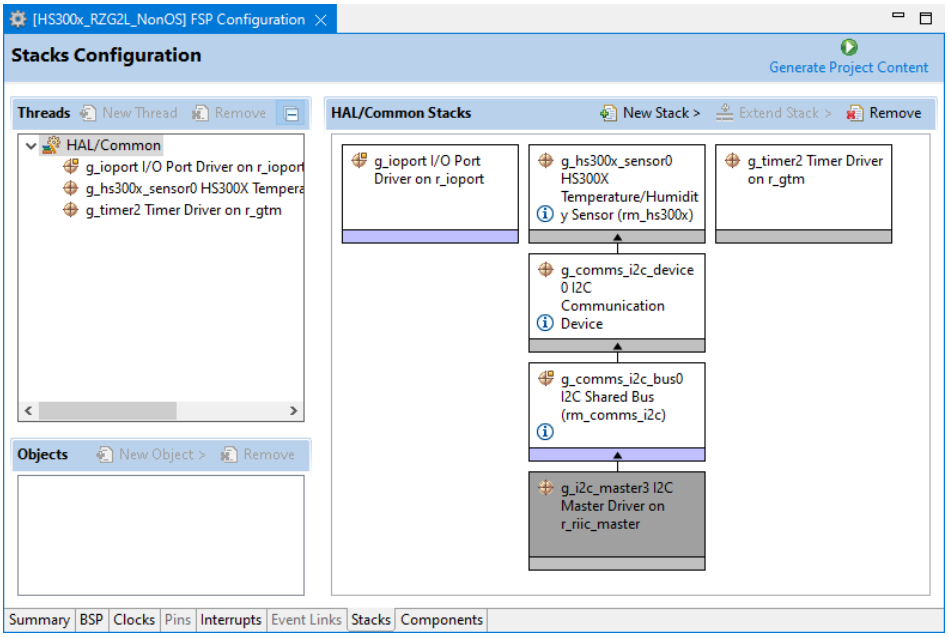


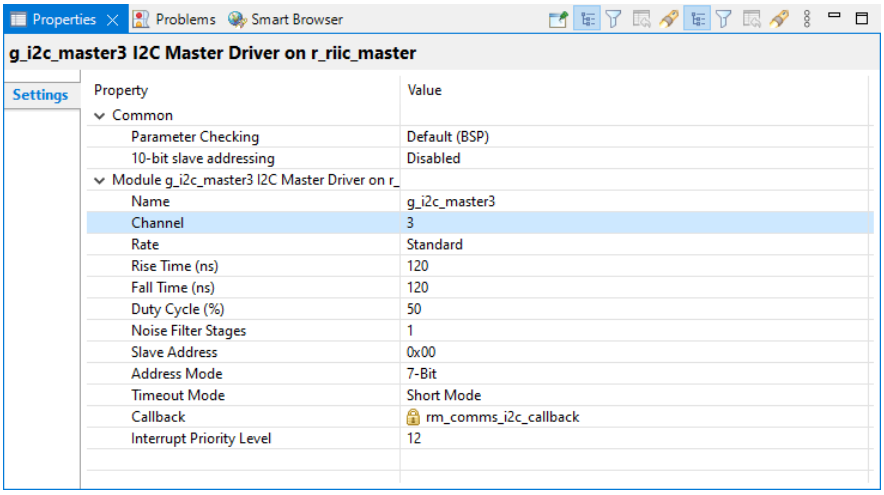
Modify the configuration of individual components on the [Stacks] tabbed page.

Modify the settings for r_iic_master to suit the specifications of the target board.

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

If you are using PMOD1, set channel to 3.





Press [Generate Project Content] to generate files.

Build the project.

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

6.4.1 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.

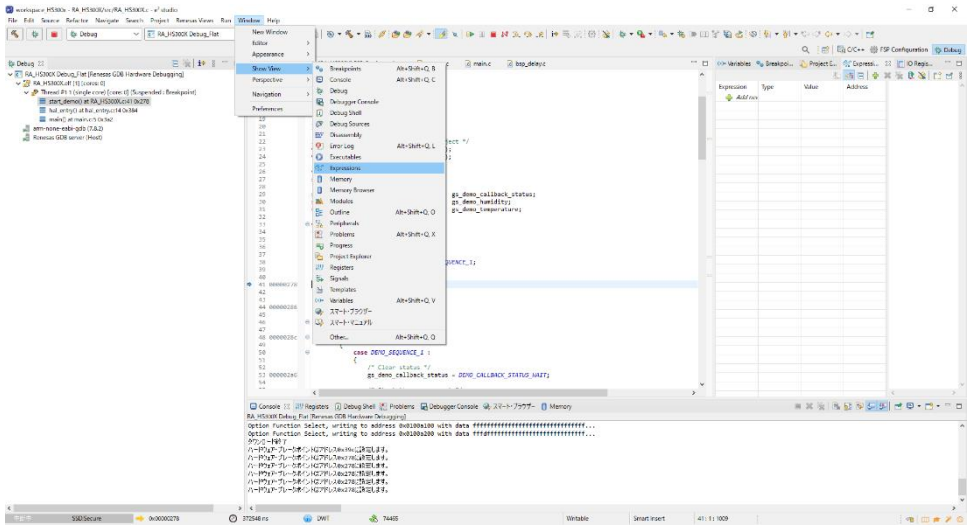
If you are using PMOD1, set the P18_0 to RIIC3_SDA (Function 3) and the P18_1 to RIIC3_SCL (Function 3).

```
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)},
};
```

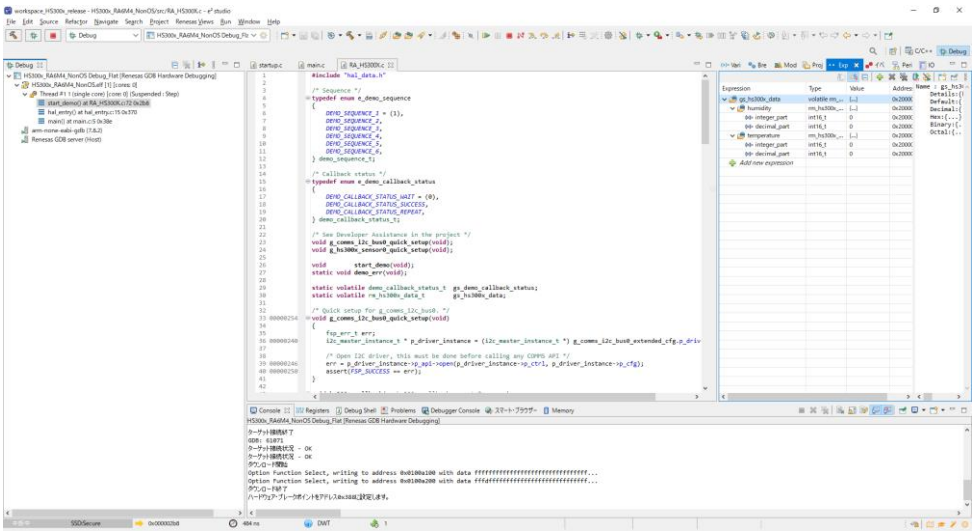
7. Viewing Temperature and Humidity Data

Use the following procedure to view temperature and humidity data in real time.

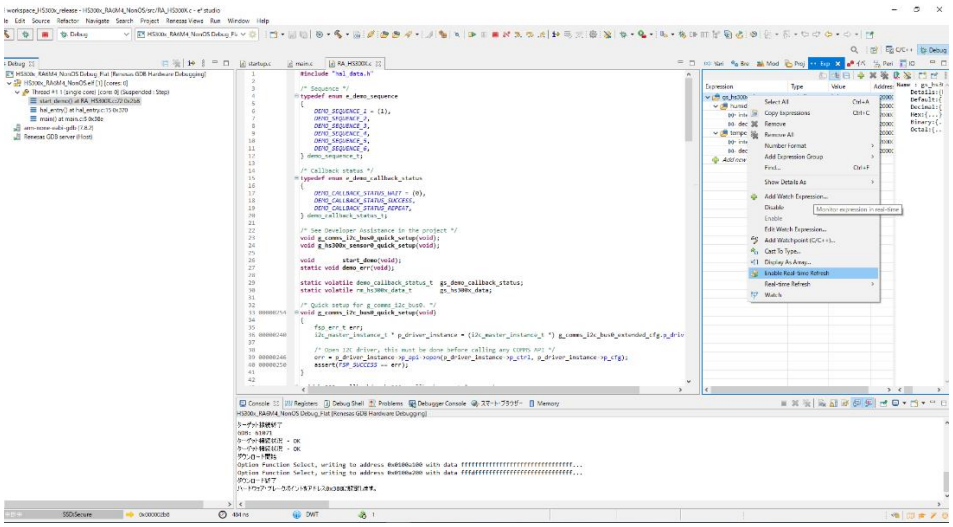
After executing debugging, select [Window] → [Show View] → [Expressions] to open the [Expressions] tabbed page.



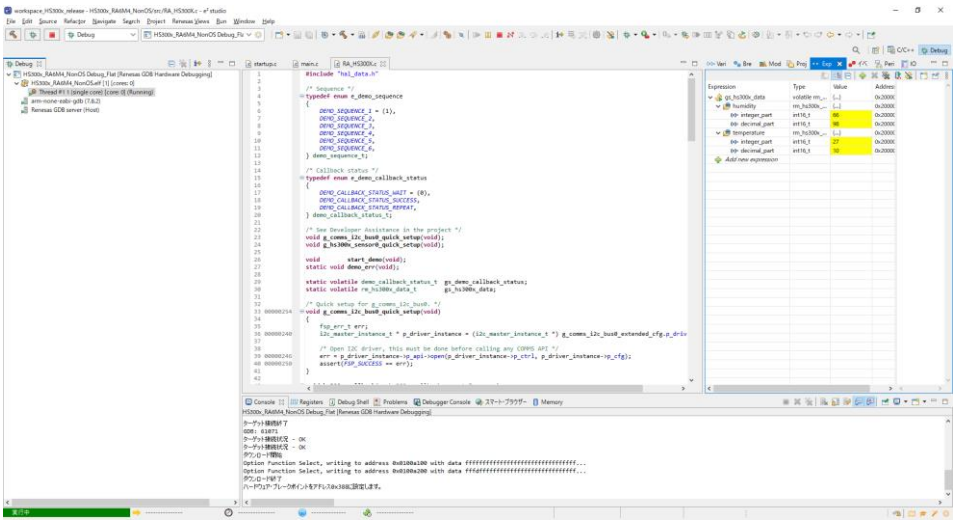
Click on [Add new expression] on the [Expressions] tabbed page and add "gs_hs300x_data".



Right-click on the added variables and select [Enable Real-time Refresh].



Start debugging, and the temperature and humidity values will be updated in real time.



Revision History

Rev.	Date	Description	
		Page	Summary
1.00	June 30, 2021	-	First Release
1.10	September 30, 2021	P10, P13 P7, P14, P36, P41, P45, P56	Add: programming mode API Add: RE01 256KB items
1.20	December 20, 2021	-	Add: Support multiple ZMOD sensors usage Add: Supoort RX Azure Other minor changes
1.30	March 1, 2022	P31	Add: Support multiple I2C devices
1.40	May 12, 2022	P4,P5, P9,P10	Changed: RA tables Changes: RX tables Changed: programming mode Changed: RE01 256KB / 1500KB tables
1.50	August 30, 2022	-	Add: RZ items
1.51	March 3, 2023	-	Updated: Environments for RL78
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
1.53	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/.