RA Family, RX Family, RL78 Family

FS3000 Sample Software Manual

Introduction

This application note describes the sample software that is for use with the FS3000 flow sensor and runs on certain MCUs of the RA family, and RX family, RL78 family.

Target Devices

RA6M4 Group

RX65N Group

RL78/G14 Group

RL78/G23 Group

Trademarks

Contents

1. Overview

This sample software acquires data from the FS3000-1005 air flow sensor and handles calculations on the data. In combination with the I2C driver of the FSP or FIT, the sample software controls the FS3000-1005 through the I2C in the MCU to acquire ADC data from the sensor and calculate the air velocity.

2. Environment for Confirming Operation

2.1 Environment for Confirming Operation on the RA Family MCU

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

Table 2-1 Operating Environment for RA Family

Item	Description
Demonstration board	RTK7EKA6M4S00001BE (EK-RA6M4)
Microcontroller	RA6M4 (R7FA6M4AF3CFB: 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 Toolchain for ARM - (8.x) 8.1.0.202011101213 MDK-ARM Ver.5.34
FSP	v3.8.0
RTOS	FreeRTOS™ and Microsoft® Azure RTOS
Emulator	On-board debugger (J-LINK)
Interposer	Interposer Board to convert Type2/3 to Type 6A PMOD standard (US082-INTERPEVZ)
Sensor board	PMOD Daughter Card for FS3000 flow sensor (US082-FS3000EVZ)

Table 2-2 Amount of Memory Used in RA Family

Area	Size (Non-OS)	Size (FreeRTOS)	Size (Azure RTOS)
ROM	1,295 bytes	1,628 bytes	1,572 bytes
RAM	73 bytes	253 bytes	422 bytes

Calculation of these sizes in memory only takes functions and variables related to the FS3000 sensor into account. They do not include the sizes of RTOS threads for the RTOS versions.

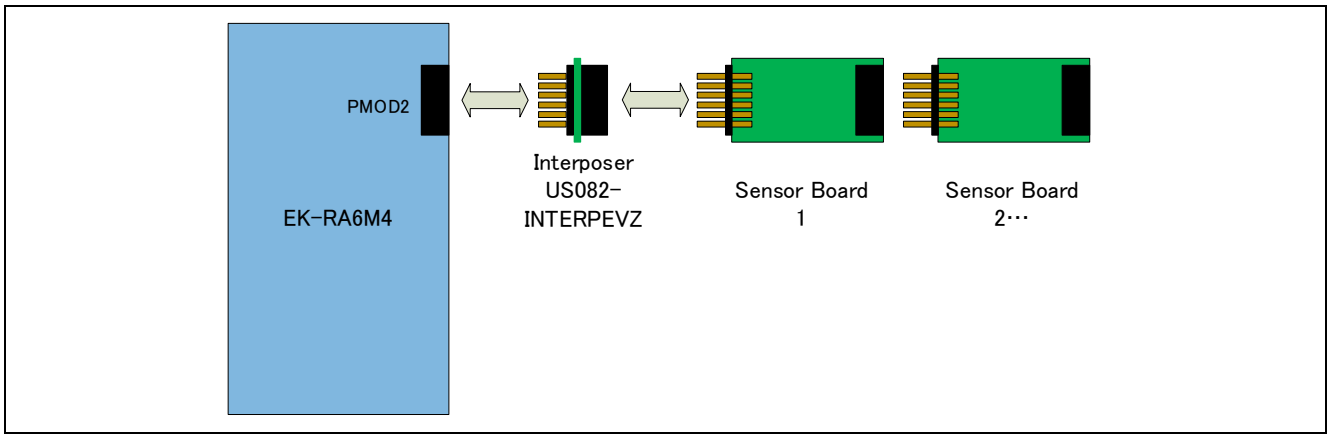


Figure 2-1 Hardware Connections for RX Family

2.2 Environment for Confirming Operation on the RX Family MCU

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

Table 2-3 Operating Environment for RX Family

Item	Description
Demonstration board	RPBRX65N (Envision Kit RX65N)
Microcontroller	RX65N (R5F565NEDDFB: 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.3.02.00 GCC 8.3.0.202004 IAR Toolchain for RX 8.4.10.7051
FIT	BSP v7.20
RTOS	FreeRTOS™ / Microsoft® Azure RTOS
Emulator	On-board debugger (E2OB)
Interposer	Interposer Board to convert Type2/3 to Type 6A PMOD standard (US082-INTERPEVZ)
Sensor board	PMOD Daughter Card for FS3000 flow sensor (US082-FS3000EVZ)

Table 2-4 Amount of Memory Used in RX Family

Area	Size (Non-OS)	Size (FreeRTOS)	Size (Azure RTOS)
ROM	1,467 bytes	1,685 bytes	1,745 bytes
RAM	145 bytes	205 bytes	414 bytes

Calculation of these sizes in memory only takes functions and variables related to the FS3000 sensor into account. They do not include the sizes of RTOS threads for the RTOS versions.

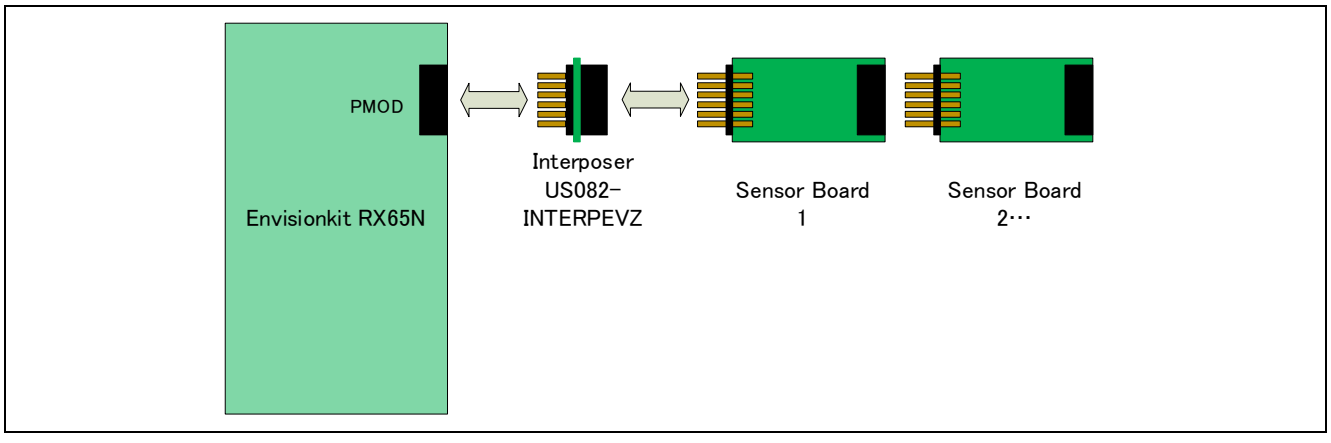


Figure 2-2 Hardware Connections for RX Family

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

The operation of this software has been confirmed on the MCU of the RL78 family 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: 80 pins)
Operating frequency	32 MHz
Operating voltage	3.3 V
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 debugger (E2OB)
Sensor board	PMOD Daughter Card for FS3000 flow sensor (US082-FS3000EVZ)

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

Area	Size
ROM	1,359 bytes
RAM	92 bytes

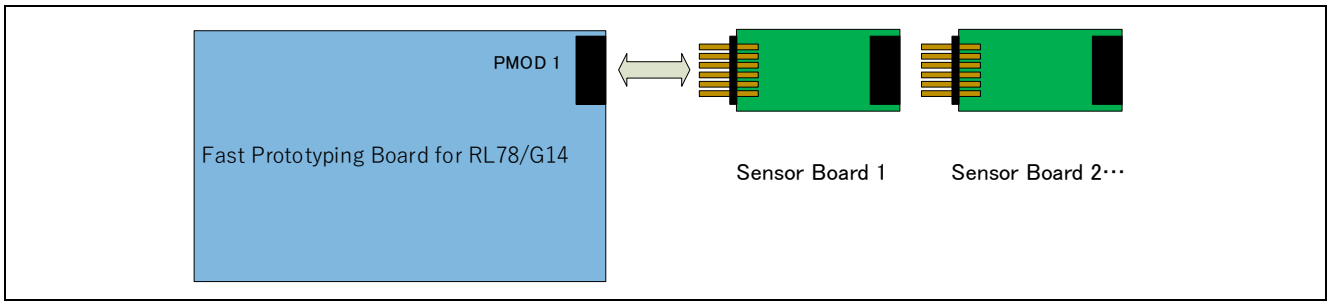


Figure 2-3 Hardware Connections for RL78/G14 Group

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

The operation of this software has been confirmed on the 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	RL78/G23 (R7F100GSN2DFB: 128 pins)
Operating frequency	32 MHz
Operating voltage	3.3 V
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	PMOD Daughter Card for FS3000 flow sensor (US082-FS3000EVZ)

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

Area	Size
ROM	1,684 bytes
RAM	92 bytes

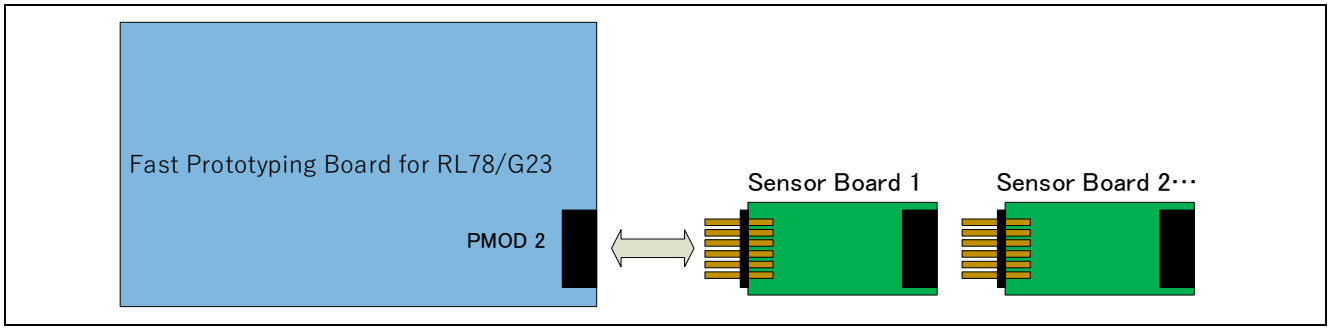


Figure 2-4 Hardware Connections for the RL78/G23 Group

3. Sensor Specifications

3.1 Overview of Sensor Specifications

The following gives an overview of the specifications of the FS3000-1005 air flow sensor.

Table 3-1 Overview of Sensor Specifications

Item	Description
Air velocity range	0 to 7.23 (m/sec)
Count range	409 to 3686 (numbers counted)
Resolution	12 bits
Accuracy	5% (at 25°C)
Measurement time	125 (ms)
I2C clock frequency	100 kHz and 400 kHz are supported.
Slave address	0x28
Addressing mode	Only 7-bit addressing is supported.

A value for air velocity is calculated from the value counted by the sensor according to the following curve.

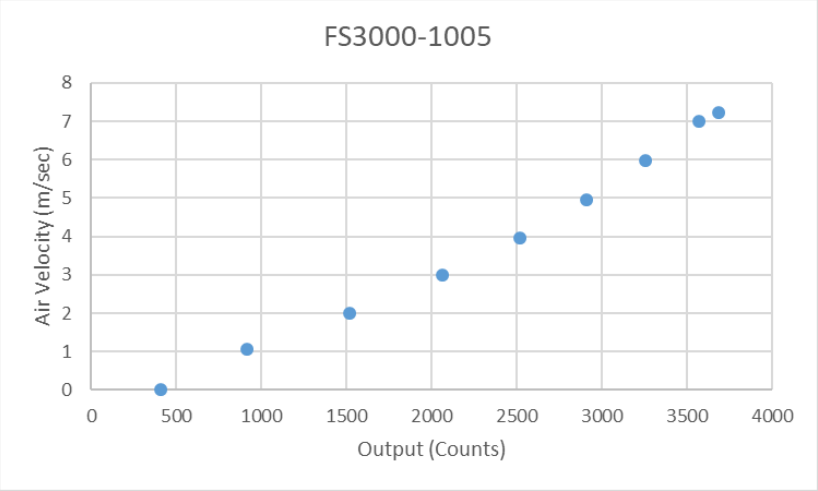


Figure 3-1 Relationship between the Values Counted by FS3000-1005 and Air Velocity

The following shows the relationship between the Air Velocity and Value Counted

Table 3-2 Relationship between the Air Velocity and Value Counted

Air Velocity (m/sec)	Output (Count)
0	409
1.07	915
2.01	1522
3.00	2066
3.97	2523
4.96	2908
5.98	3256
6.99	3572
7.23	3686

3.2 Sensor Functions

The FS3000 sample software supports the FS3000-1005 air flow sensor; it does not support the FS3000-1015 sensor.

The sensor begins measurement as soon as the power supply is turned on.

To obtain data from the sensor, send the command shown below and then read the five bytes of data.

The air velocity is represented by 12 bits; the four lower-order bits of the second byte (Byte 2) are valid.

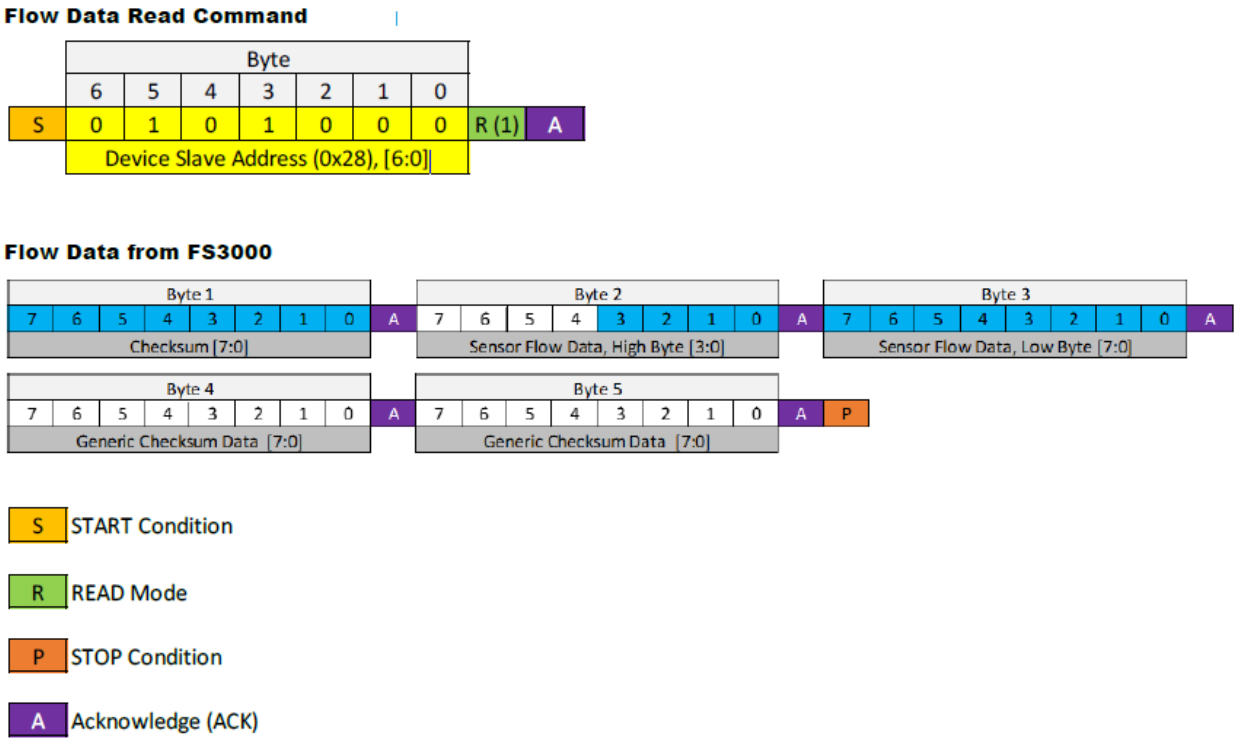


Figure 3-2 Data Format

Whether the value read is valid can be checked by calculating the checksum data as follows.

Example:

Byte 1: 0xCC (Checksum)

Byte 2: 0x01 (Flow data, high byte)

Byte 3: 0x99 (Flow data, low byte)

Byte 4: 0x01 (General checksum)

Byte 5: 0x99 (General checksum)

Add all values except the Checksum value.

Sum = 0x01 + 0x99 + 0x01 + 0x99 = 0x134

Add the Checksum value to the above Sum. When the eight lower-order bits of the result are 0x00, the value read is valid.

Checksum + Sum = 0xCC + 0x134 = 0x200 ← The eight lower-order bits are 0x00; the value read is valid.

4. Sample Software Specifications

This sample software package contains a total of six projects: non-OS and OS (FreeRTOS and Azure RTOS) versions for the RA family, non-OS and OS (FreeRTOS) versions for the RX family, and a non-OS version for the RL78 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.

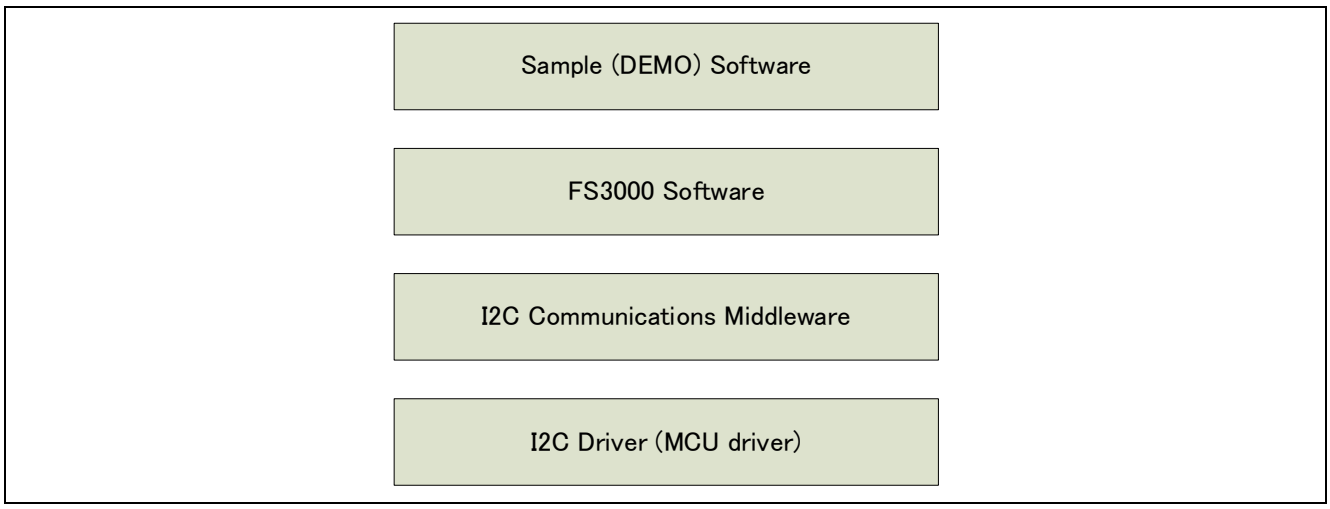


Figure 4-1 Block Diagram of the Sample Software

4.2 Specifications of Sensor API Functions

4.2.1 List of Sensor API functions

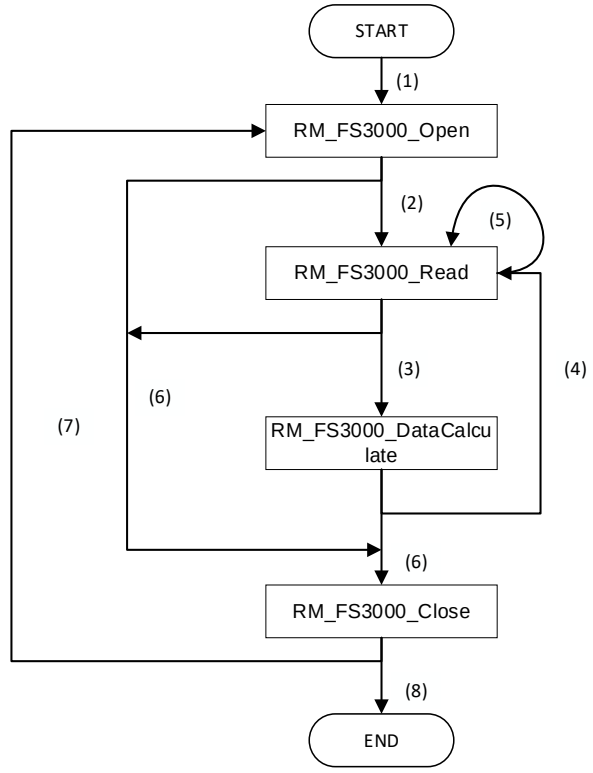
The following table lists the sensor API functions. For details of the API functions, refer to the separately provided RX Family FS3000 Sensor API FIT Module application note (R01AN5894) and RL78 Family Renesas Sensor Control Modules application note (R01AN5896)

Table 4-1 List of Sensor API Functions

Function	Description
RM_FS3000_Open	Starts control of the sensor.
RM_FS3000_Close	Terminates control of the sensor.
RM_FS3000_Read	Acquires data from the sensor.
RM_FS3000_DataCalculate	Calculates values from the data acquired from the sensor.

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 FS3000 API functions and the expected orders of function calls.



Flow No.	Function	Transition Case
(1)	Start FS3000	not open FS3000
(2)	ADC Read	open success
(3)	calculate	ADC read success (callback state success)
(4)	next read ADC	DataCalculate return success and next ADC data Get
(5)	get retry ADC data	user get retry ADC data or DataCalculate return invalid data
(6)	Close FS3000	Error state or user stop
(7)	Restart FS3000	Close success and user restart
(8)	Stop FS3000	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_FS3000_Open: (1) Activation of FS3000 or
  (7) restart after a call of RM_FS3000_Close
- RM_FS3000_Close: (6) Successful completion or abnormal end of individual processing
- RM_FS3000_Read: (2) Acquisition of measured data after the start of measurement or
  (5) retry after waiting for the response to the data acquisition request
- RM_FS3000_DataCalculate: (3) Calculation of data after a call of RM_FS3000_Read

Note:

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 section 4.4, Flowchart of the OS Version of the Sample Software.

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 acquiring data from the sensor and calculating values from the results of measurement.

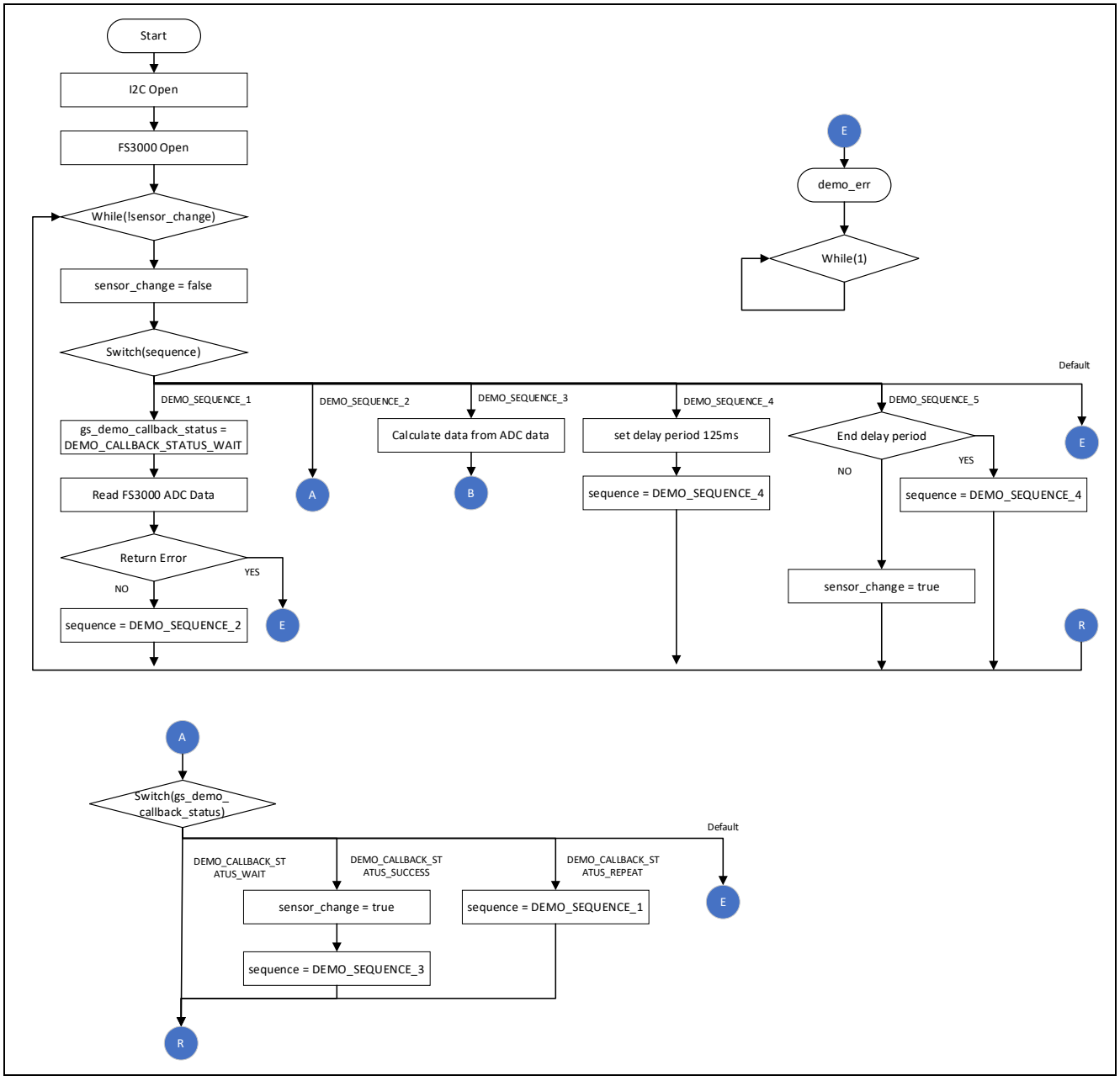


Figure 4-3 Flowchart of the Main Processing in the Non-OS Version of the Sample Software (1)

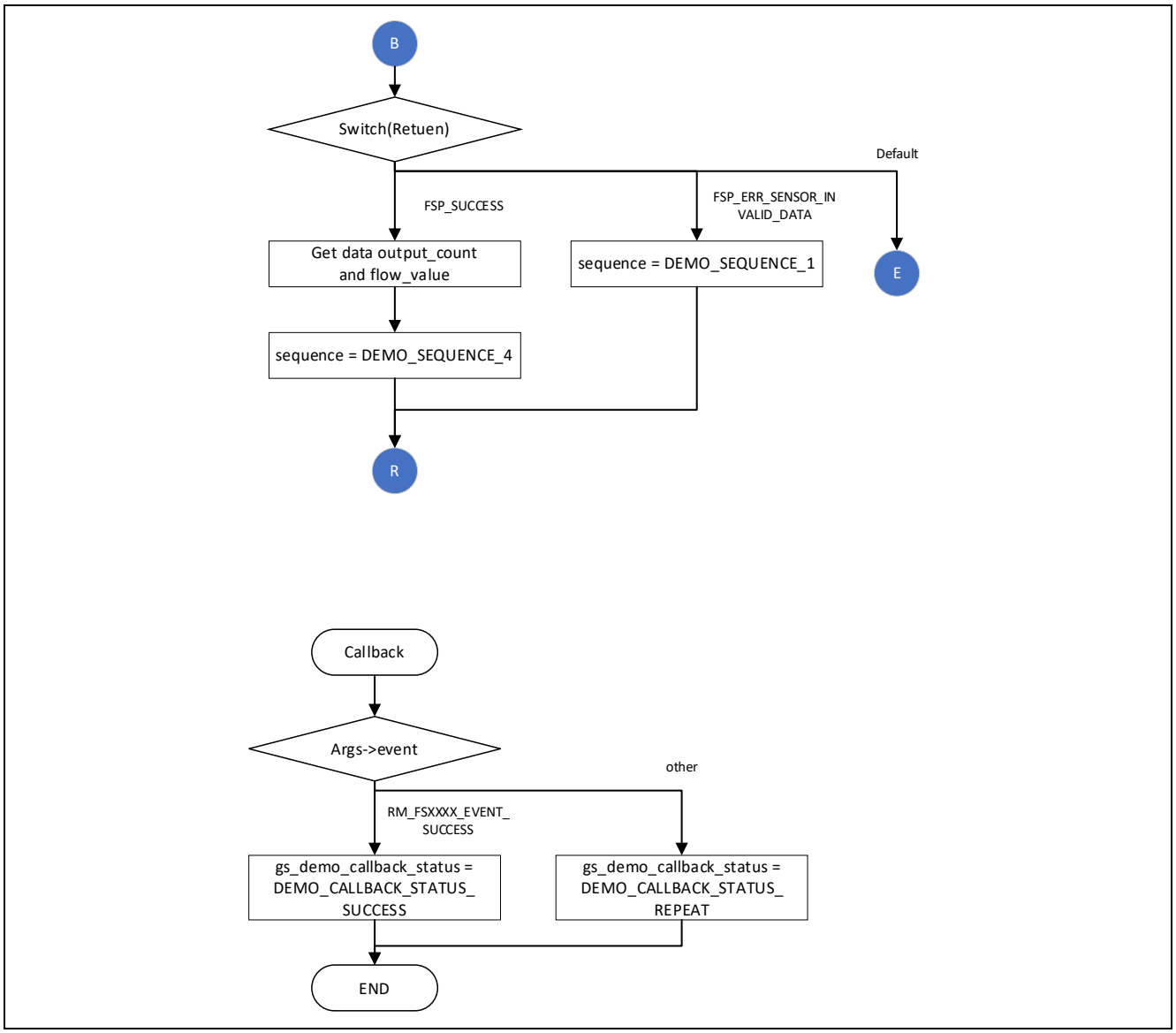


Figure 4-4 Flowchart of the Main Processing in the Non-OS Version of the Sample Software (2)

4.4 Flowchart of the OS Version of the Sample Software

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

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

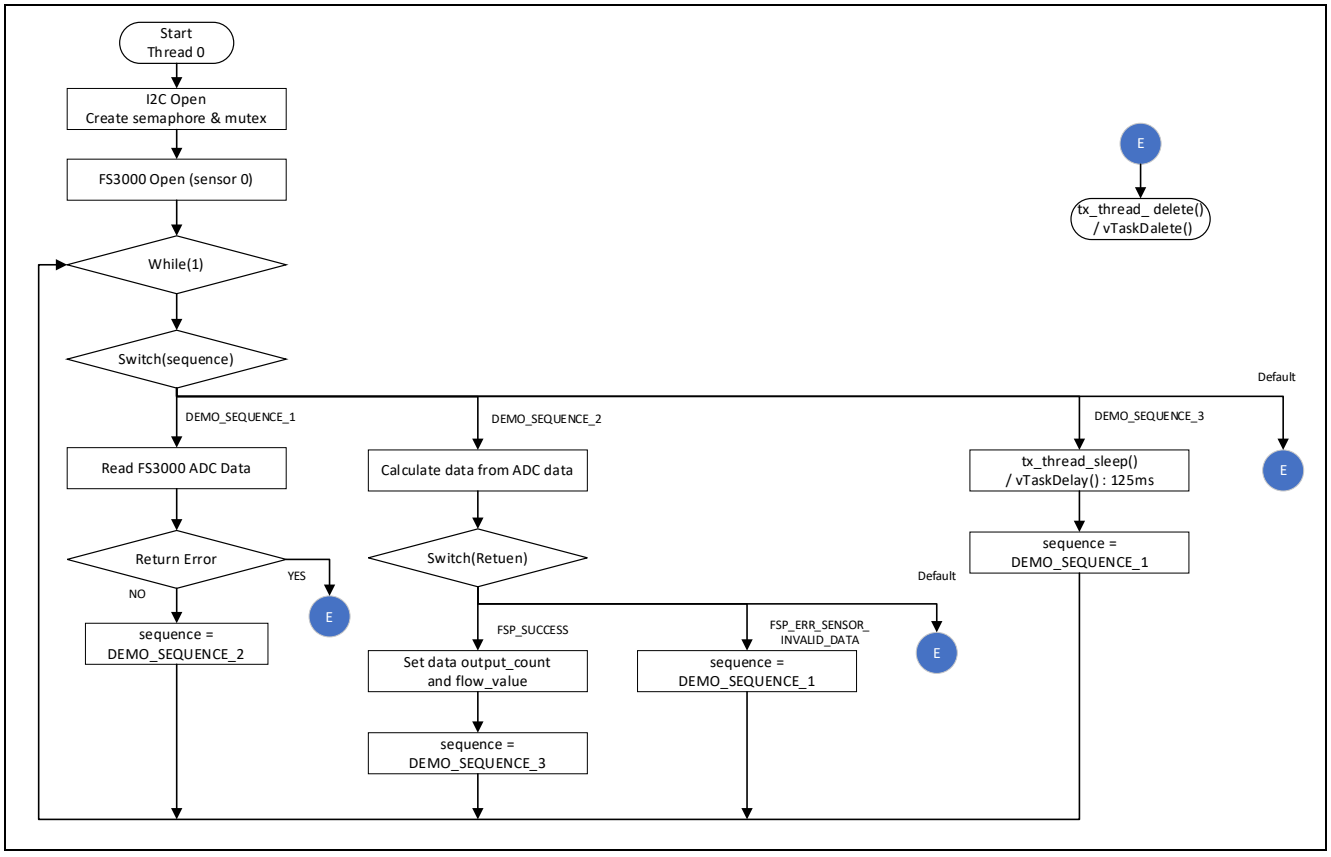


Figure 4-5 Flowchart of the Main Processing in the OS Version of the Sample Software (1)

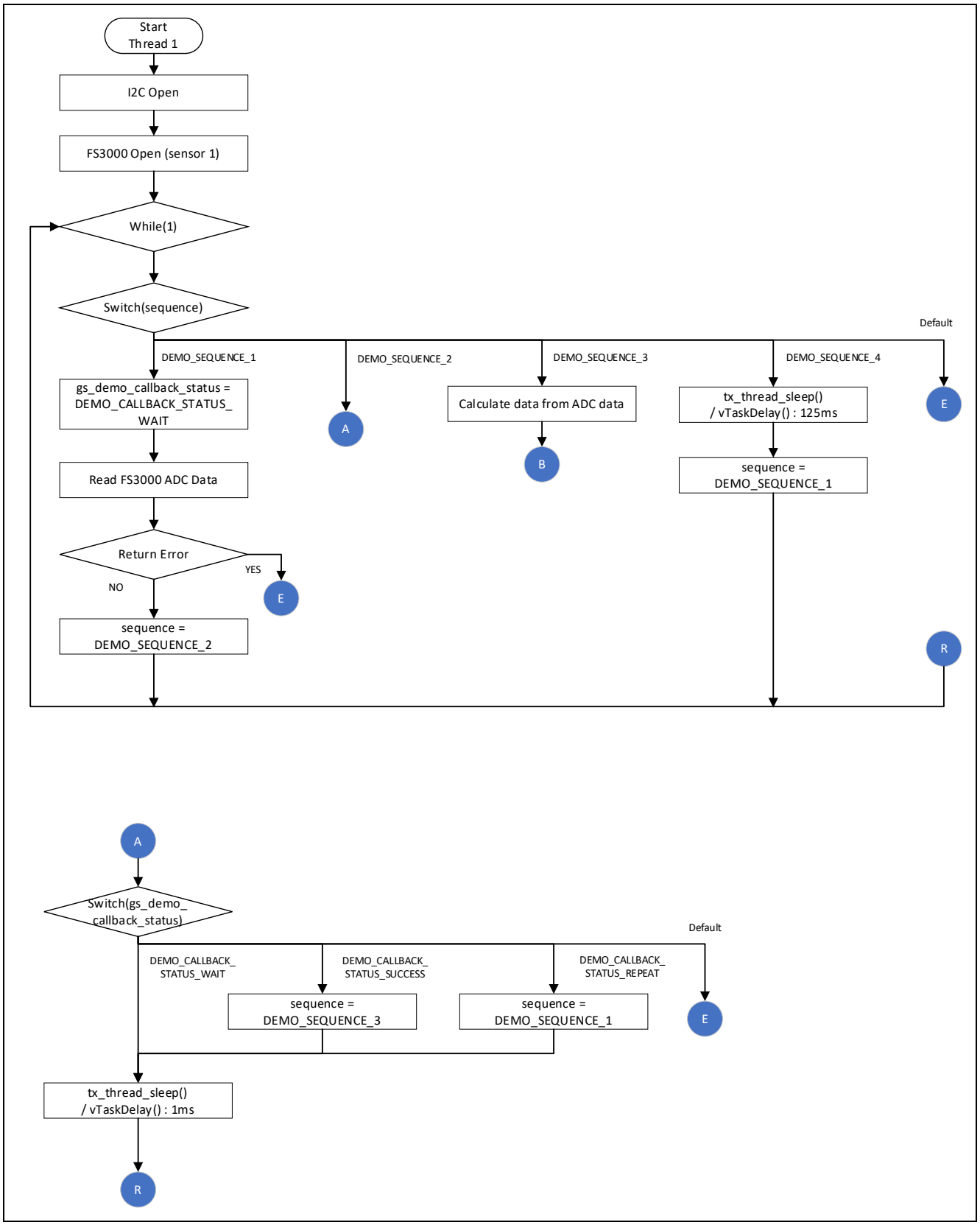


Figure 4-6 Flowchart of the Main Processing in the OS Version of the Sample Software (2)

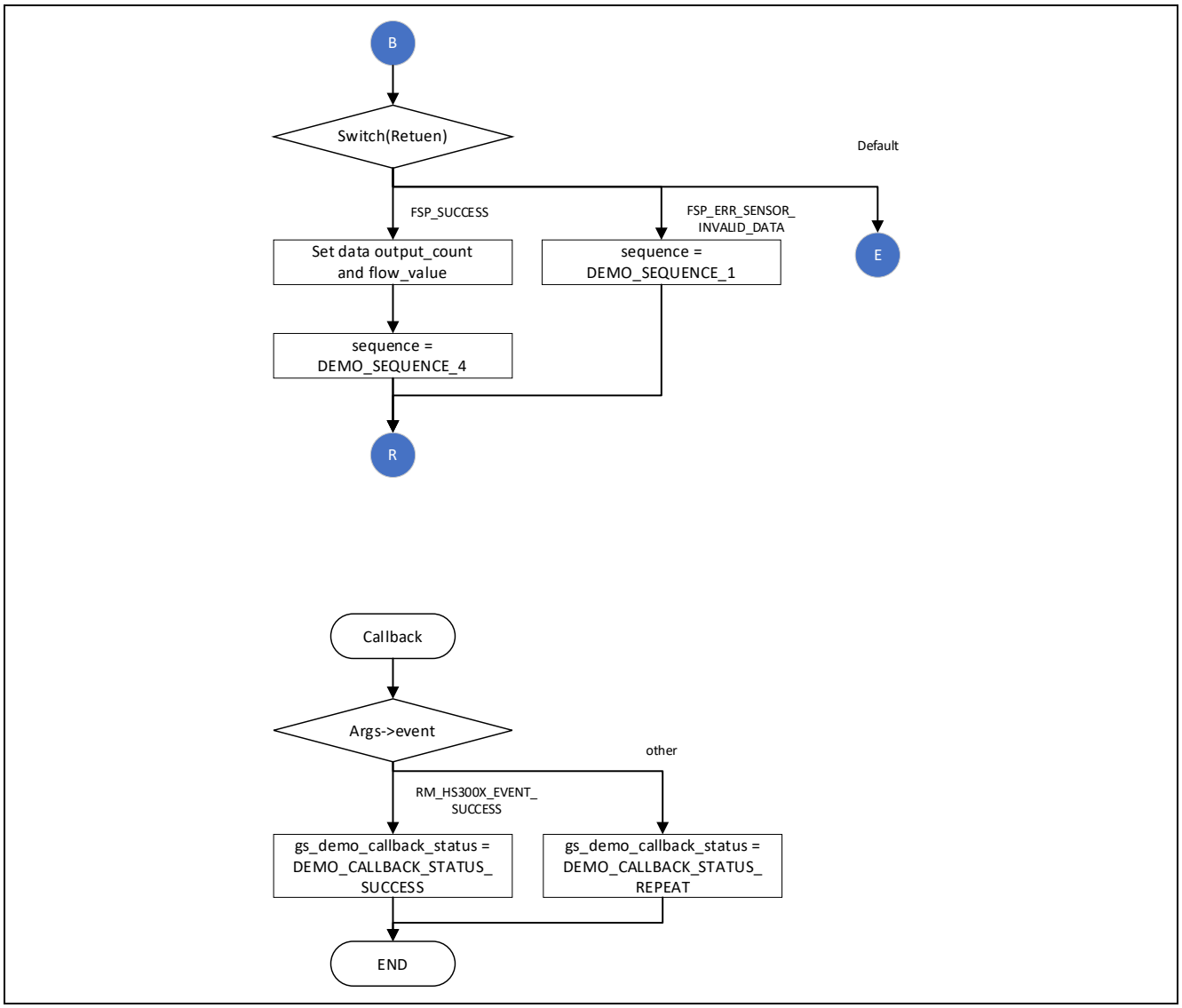


Figure 4-7 Flowchart of the Main Processing in the OS Version of the Sample Software (3)

4.4.1 Azure RTOS Project

The RX project for use with the Azure RTOS has the following changes from the default source files generated by the RX Smart Configurator.

1. src/demo_thread.c

   Line 57: Addition of extern void tx_application_define_user (void);

   Line 178: Addition of tx_application_define_user();

2. libs/threadx/common/inc/tx_api.h

   Line 224: Change of TX_TIMER_TICKS_PER_SECOND ((ULONG) 1000)

5. Configuration Settings

5.1 FS3000 Air Velocity Sensor Settings

5.1.1 RA Family

Select the rm_fs3000 stack in the "Stack" tabbed page of the FSP Configurator, and the configurable items are shown in the "Properties" tabbed page.

The following items and values can be specified.

Table 5-1 FS3000 Settings for RA Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP) Enabled Disabled	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
Device type	FS3000-1005	Specify the type of device to be acquired from the sensor. "FS3000-1005" only can be selected.
Module g_fs3000_sensor FS3000 on rm_fs3000		
Name	g_fs3000_sensor0	Specify the name of the module. A module name conforming to the C language standard can be specified.
Callback	fs3000_callback	Specify the name of the user callback function. A callback function name conforming to the C language standard can be specified. When "NULL" is specified, no callback function is used.

5.1.2 RX Family

Select the r_fs3000_rx component in the "Component" tabbed page of the Smart Configurator, and the configurable items are shown in the "Configure" panel.

The following items and values can be specified.

Table 5-2 FS3000 Settings for RX Family

Configurable Item	Value	Description
Configurations		
Parameter Checking	System Default Enabled Disabled	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
Number of FS3000 sensors	1 2	Specify the number of FS3000 sensors to be connected.
Device type of FS3000 Sensors	FS3000-1005	Specify the type of sensor. "FS3000-1005" can be selected.
Using communication line number for FS3000 sensor device x (x = 0 or 1)	I2C Communication Device(y) (y = 0 - 15)	Specify the communication line to be used by the sensor.
Callback function for FS3000 sensor device x (x = 0 or 1)	fs3000_user_callback(x) (x = 0 or 1)	Specify the name of the user callback function. A callback function name conforming to the C language standard can be specified.

5.1.3 RL78 Family

Settings can be modified by changing the values of the constants defined in the ¥r_config¥r_fs3000_rl_config.h file in the project tree of the sample project.

The following items and values can be specified.

Table 5-3 FS3000 Settings for RL78 Family

Constant Name	Value	Description
Configurations		
RM_FS3000_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_FS3000_CFG_DEVICE_NUM_MAX	1	Specify the number of FS3000 sensors to be connected.
	2	
RM_FS3000_CFG_DEVICE_TYPE	RM_FS3000_DEVICE_TYPE_1005	Specify the type of sensor. "FS3000-1005" can be selected.
RM_FS3000_CFG_DEVICEx_COMMS_INSTANCE (x = 0 or 1)	g_comms_i2c_device0	Specify the instance name of the communication line to be used.
RM_FS3000_CFG_DEVICEx_CALLBACK (x = 0 or 1)	fs3000_callback	Specify the name of the user callback function. A callback function name conforming to the C language standard can be specified. When "NULL" is specified, no callback function is used.

5.2 Communication Driver Middleware Settings

5.2.1 RA Family

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

The following items and values can be specified.

Table 5-4 Communication Driver Settings for RA Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP)	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
	Enabled	
	Disabled	
Module g_comms_i2c_device I2C Communication Device on rm_comms_i2c		
Name	g_comms_i2c_device0	Specify the name of the module. A module name conforming to the C language standard can be specified.
Semaphore Timeout	0xFFFFFFFF	For an RTOS project, specify the time of semaphore timeout.
Slave Address	0x28	Specify the slave address. When rm_fs3000 is used, this value is automatically specified and cannot be modified.
Address Mode	7-Bit	Specify the number of slave address bits. When rm_fs3000 is used, this value is automatically specified and cannot be modified.
Callback	rm_fs3000_callback	Specify the name of the user callback function. When rm_fs3000 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 the blocking processing.
	Use	
Recursive Mutex for Bus	Unuse	For an RTOS project, enable or disable the recursive operation when blocking is enabled.
	Use	

5.2.2 RX Family

Select the r_comms_i2c_rx component in the "Component" tabbed page of the Smart Configurator, and the configurable items are shown in the "Configure" panel.

The following items and values can be specified.

Table 5-5 Communication Driver Settings for RX Family

Configurable Item	Value	Description
Configurations		
Parameter Checking	System Default Enabled Disabled	Specify whether to include the processing to check parameters in the code to be generated. When "Disabled" is selected, the generated code does not include the processing to check parameters. When "Enabled" is selected, the generated code includes the processing to check parameters.
Number of I2C Shared Buses	Unused 1 2 – 16	Specify the number of I2C bus lines that can be connected.
Number of I2C Devices	Unused 1 2 – 16	Specify the number of I2C devices that can be connected.
Blocking operation supporting with RTOS	Disabled Enabled	For an RTOS project, enable or disable the blocking operation.
Bus lock operation supporting with RTOS	Disabled Enabled	For an RTOS project, enable or disable the bus lock operation.
IIC Driver Type for I2C Shared bus(x) (x = 0 – 15)	RIIC SCI IIC Not selected	Specify the I2C type to be used for the communications bus. Using the RIIC requires r_riic_rx. Using the SCI IIC requires r_sci_iic_rx. If an unused FIT module is deleted, a warning message will appear but this will not affect the operation.
Channel No. for I2C Shared bus(x) (x = 0 – 15)	0	Specify the I2C channel number to be used for the communications bus.
Timeout for the bus lock of the I2C bus for I2C Shared Bus(x) (x = 0 – 15)	0xFFFFFFFF	Specify the time of I2C bus lock timeout for I2C bus x. (x = 0 – 15)
I2C Shared Bus No. for I2C Communication Device(x) (x = 0 – 15)	I2C Shared Bus(x) (x = 0 – 15)	Specify the configuration of the I2C bus to be used for the communications bus.
Slave address for communication device(x) (x = 0 – 15)	0x28	Specify the slave address of the device to be connected to the communications bus. When using r_fs3000_rx, specify 0x28.
Slave address mode for communication device(x) (x = 0 – 15)	7 bit address mode	Specify the slave address mode. When using r_ fs3000_rx, specify the 7-bit address mode.
Callback function for Communication device(x) (x = 0 – 15)	comms_i2c_user_callback(x) (x = 0 – 15)	Specify the name of the user callback function. When using r_ fs3000_rx, specify rm_ fs3000_callback(y) (y = 0).

5.2.3 RL78 Family

Settings can be modified by changing the values of the 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 5-6 Communication Driver Settings for RL78 Family

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_DEVICE_NUM_MAX	1	Specify the number of communication bus lines that can be connected.
	2	
	3	
	4	
	5	
COMMS_I2C_CFG_BUSx_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_BUSx_DRIVER_CH (x = 0 – 4)	g_comms_i2c_bus(x)_extended_cfg (x = 0 - 4)	Specify the I2C channel number to be used for the communication bus.
COMMS_I2C_CFG_BUSx_SLAVE_ADDR (x = 0 – 4)	0x28	Specify the slave address of the device to be connected to the communication bus. When using rm_fs3000, specify 0x48.
COMMS_I2C_CFG_BUSx_ADDR_MODE (x = 0 – 4)	COMMS_I2C_ADDR_7BIT	Specify the slave address mode. The RL78 code generator only supports the 7-bit address mode.
COMMS_I2C_CFG_DEVICE(x)_CALLBACK (x = 0 – 4)	rm_fs3000_callback0	Specify the name of the user callback function. When "NULL" is specified, no callback function is used.

5.3 I2C Driver Settings

5.3.1 RA Family

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

The following items and values can be specified.

Table 5-7 r_iic_master Settings for RA Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP) Enabled Disabled	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
DTC on Transmission and Reception	Enabled Disabled	Specify whether to use the DTC for transmission and reception.
10-bit slave addressing	Enabled Disabled	Specify whether to support 10-bit addressing for the slave address. When using rm_fs3000, select "Disabled".
Module g_i2c_master0 I2C Master Driver on r_iic_master		
Name	g_i2c_master0	Specify the name of the module.
Channel	0	Specify the channel number to be used.
Rate	Standard Fast-mode Fast-mode plus	Specify the baud rate. When using rm_fs3000, select "Standard" or "Fast-mode".
Rise Time (ns)	120	Specify the SCL rise time according to the specifications of the target board to be used.
Fall Time (ns)	120	Specify the SCL fall time according to the specifications of the target board to be used.
Duty Cycle (%)	50	Specify the SCL duty cycle.
Slave Address	0x00	This item specifies the slave address of the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
Address Mode	7-Bit 10-Bit	This item specifies the salve address mode for the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
Timeout Mode	Short Mode Long Mode	Specify the time of I2C bus timeout.
Callback	rm_comms_i2c_callback	The name of the user callback function is automatically specified by rm_comms_i2c.

Interrupt Priority Level	Priority 0 (highest)	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.
SCL	Pxxx	

Table 5-8 r_sci_i2c Settings for RA Family

Configurable Item	Value	Description
Common		
Parameter Checking	Default (BSP)	Enable or disable the parameter check processing. When "Enabled" is selected, the project is built so that the generated code includes the parameter check processing.
	Enabled	
	Disabled	
DTC on Transmission and Reception	Enabled	Specify whether to use the DTC for transmission and reception.
	Disabled	
10-bit slave addressing	Enabled	Specify whether to support 10-bit addressing for the slave address. When using rm_fs3000, select "Disabled".
	Disabled	
Module g_i2c0 I2C Master Driver on r_sci_i2c		
Name	g_i2c0	Specify the name of the module.
Channel	0	For an RTOS project, specify the time of semaphore timeout.
Slave Address	0x00	This item specifies the slave address of the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
Address Mode	7-Bit	This item specifies the salve address mode for the device to be connected but the user does not need to make this setting because rm_comms_i2c overwrites it.
	10-bit	
Rate	Standard	Specify the baud rate. Select "Standard" or "Fast-mode".
	Fast-mode	
	Fast-mode plus	
SDA Output Delay (nano seconds)	300	Specify the SDA output delay time.
Noise filter setting	Use clock signal divided by 1 with noise filter	Specify the noise filter to be used for input signals.
	Use clock signal divided by 2 with noise filter	
	Use clock signal divided by 4 with noise filter	
	Use clock signal divided by 8 with noise filter	
Bit Rate Modulation	Enable	Enable or disable the bit rate modulation function.
	Disable	
Callback	rm_comms_i2c_callback	The name of the user callback function is automatically specified by rm_comms_i2c.

Interrupt Priority Level	Priority 0 (highest)	Specify the interrupt priority level of the I2C bus driver.
	Priority 1	
	Priority 2	
	Priority 3	
	Priority 4	
	Priority 5	
	Priority 6	
	Priority 7	
	Priority 8	
	Priority 9	
	Priority 10	
	Priority 11	
	Priority 12	
	Priority 13	
	Priority 14	
	Priority 15	
RX Interrupt Priority Level [Only used when DTC is enabled**]**	Priority 0 (highest)	When using the DTC, specify the priority level of the reception interrupt.
	Priority 1	
	Priority 2	
	Priority 3	
	Priority 4	
	Priority 5	
	Priority 6	
	Priority 7	
	Priority 8	
	Priority 9	
	Priority 10	
	Priority 11	
	Priority 12	
	Priority 13	
	Priority 14	
	Priority 15	
	Disabled	
Pins		
SDA	Pxxx	The pin numbers to be used by the driver are displayed Use the "Pins" tabbed page to modify the pin configuration.
SCL	Pxxx	

5.3.2 RX Family

Select the r_riic_rx or r_sci_iic_rx component in the "Component" tabbed page of the Smart Configurator, and the configurable items are shown in the "Configure" panel.

The following items and values can be specified.

Table 5-9 r_riic_rx Settings for RX Family

Configurable Item	Value	Description
Configurations		
Set parameter checking enable	System Default	Enable or disable the parameter check processing. When "Include" is selected, the project is built so that the generated code includes the parameter check processing.
	Not	
	Include	
MCU supported channels for CHx (x = 0 – 2)	Not supported	Specify whether to support the operation of channel x.
	Supported	
CHx RIIC bps(kbps) (x = 0 – 2)	400	Specify the baud rate. Set to 400 or a smaller value.
Digital filter for CHx (x = 0 – 2)	Not	Specify the digital filter for input signals.
	One IIC phi	
	Two IIC phi	
	Three IIC phi	
	Four IIC phi	
Setting port setting processing	Not include port setting	Specify whether to include the pin function settings in the code to be generated.
	Include port setting	
Master arbitration lost detection function for CHx (x = 0 – 2)	Unused	Specify whether to use the master arbitration lost detection function.
	Used	
Address y format for CHx (x = 0 – 2, y = 0 – 2)	Not	This item specifies the slave address mode for slave address y but the user does not need to make this setting because rm_comms_i2c overwrites it.
	7 bit address format	
	10 bit address format	
Slave Address y for CHx (x = 0 – 2, y = 0 – 2)	0x0025	This item specifies slave address y but the user does not need to make this setting because rm_comms_i2c overwrites it.
General call address for CHx	Unused	Specify whether to use the general call function.
	Used	
CHx RXI INT Priority Level (x = 0 – 2)	Level 1	Specify the priority level of the reception interrupt.
	Level 2	
	Level 3	
	Level 4	
	Level 5	
	Level 6	
	Level 7	
	Level 8	
	Level 9	
	Level 10	
	Level 11	
	Level 12	
	Level 13	
	Level 14	
	Level 15 (highest)	

CHx RXI INT Priority Level (x = 0 – 2)	Level 1 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 count 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 count for detecting a timeout while SCL is at the high level.

Set Counter of checking bus busy	1000	Specify the count to detect 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 5-10 r_sci_iic_rx Settings for RX Family

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 baud 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 count to detect 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 are shown in the "Peripheral Functions" tabbed page.

The following items and values can be specified.

Table 5-11 Serial Settings for RL78 Family

Configurable Item	Value	Description
SAUx		
Channel		
Channel x	Unused	Specify the communication function of the channel to be used. When using r_fs3000, select IICxx.
	UARTxx	
	CSIxx	
	IICxx	
IICxx		
Transfer rate	1000000	Specify the baud rate. When using rm_fs3000, 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 call back function when master transmission ends.
Master reception end	Checked	Specify whether to use the call back function when master reception ends.
Master error	Checked	Specify whether to use the call back 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 for 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 baud 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 call back function when master transmission ends.
Master reception end	Checked	Specify whether to use the call back function when master reception ends.
Master error	Checked	Specify whether to use the call back 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 a callback. Deselect the checkbox.

6. Guide for Changing the Target Device

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

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

6.1 RA Sample Project

Use the following procedures to modify a sample project.

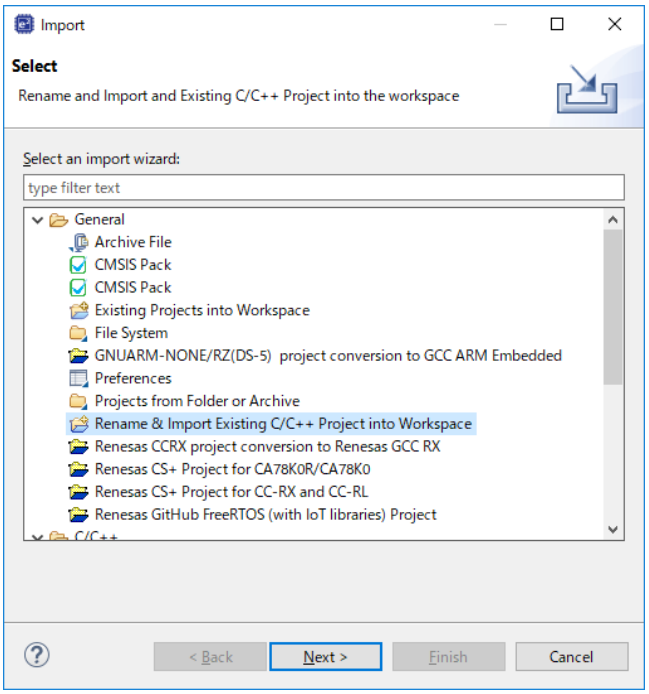
This section describes an example of modifying the sample project "FS3000_RA6M4_NonOS" so that it can be used on the EK-RA2E1 board.

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

6.1.1 Importing the Sample Project

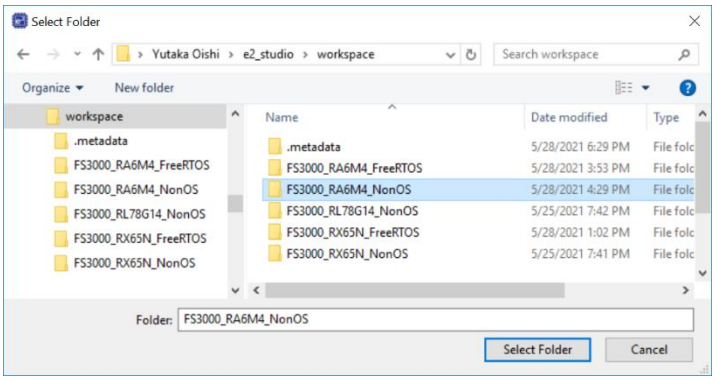
Select [Import] from the menu.

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

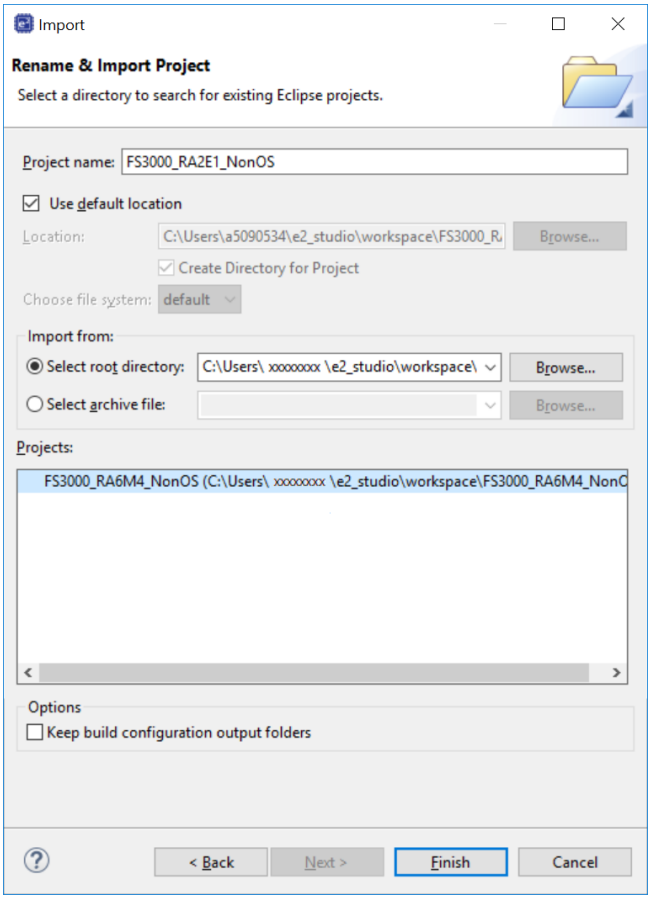


Press the [Browse] button to open the "Select Folder" window.

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



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



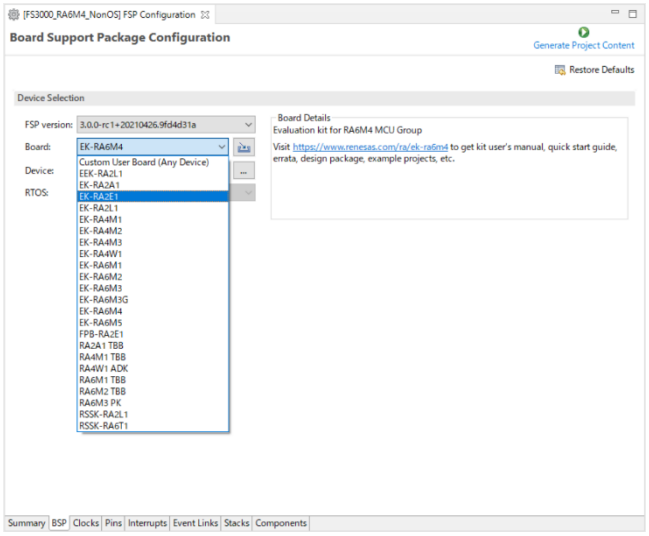
6.1.2 Modifying Settings of the FSP Configurator

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

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

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

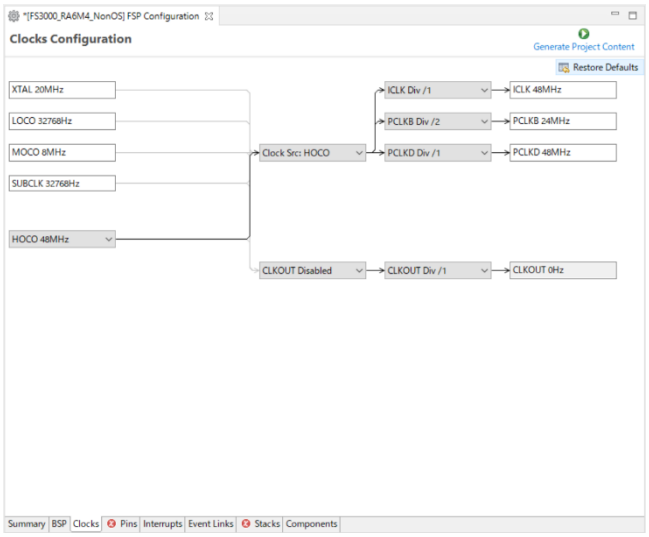
When selecting a board provided from other companies, change the "Board" setting to "Custom User Board (Any Device)" and then change the "Device" setting to the new device to be used.



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

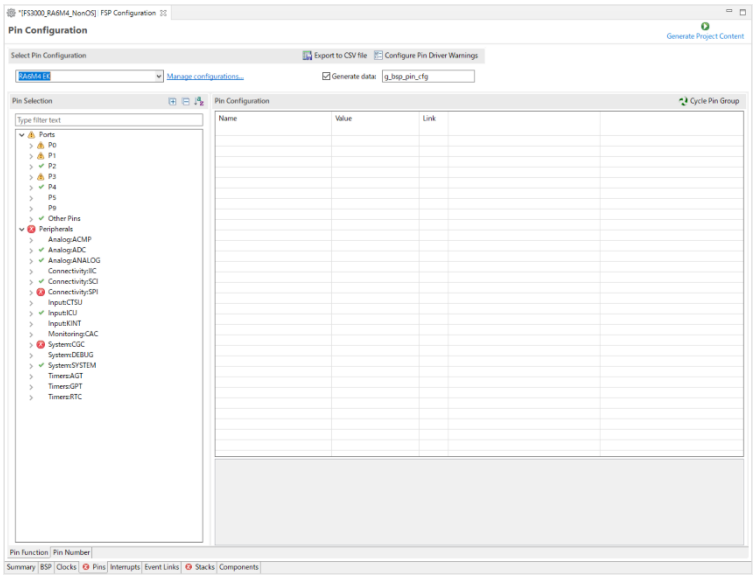
When "Custom User Board (Any Device)" is selected for "Board", set up the clocks according to the specifications of the target board to be used.

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

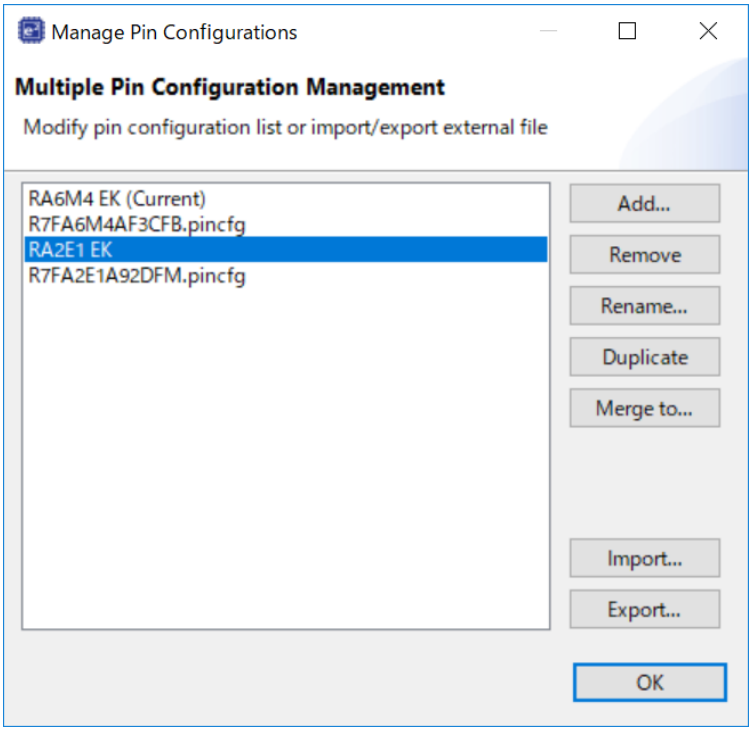


In the “Pins” tabbed page, modify the pin configuration according to the specifications of the target board to be used.

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



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



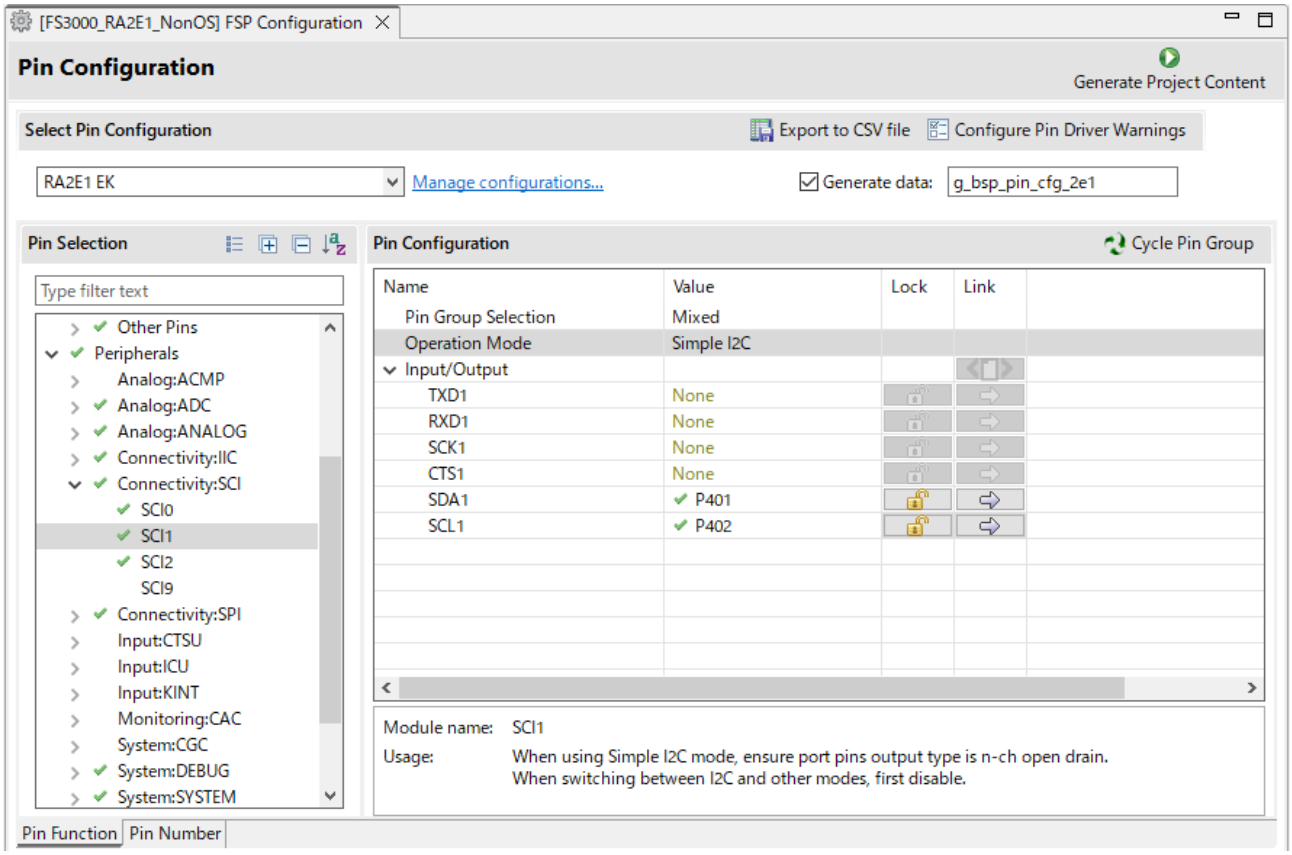
However, the "Select Pin Configuration" assignment will apply the SPI communication pin settings that support PMOD Type 2A on the EK-RA2E1 board.

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

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

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

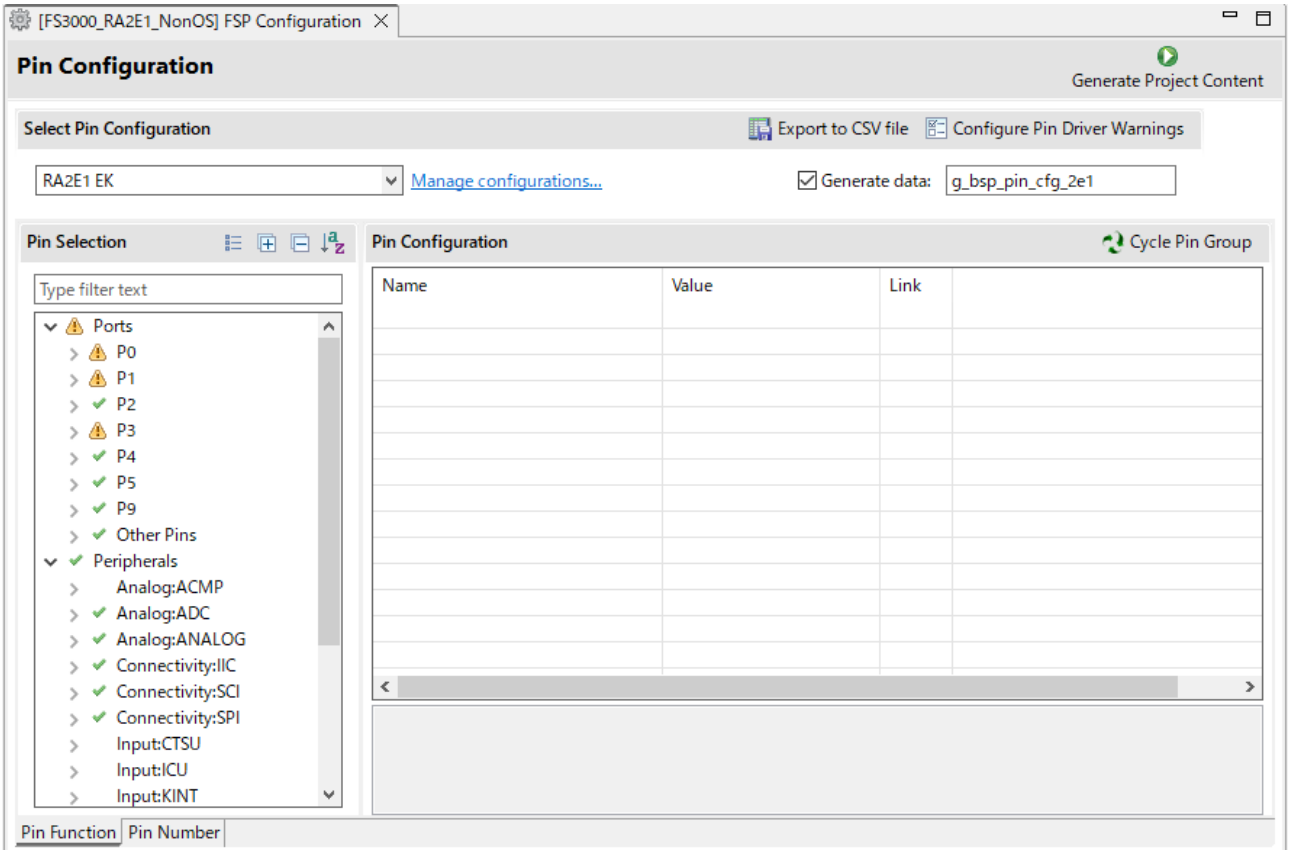
After automatic assignment of "Select Pin Configuration", reconfigure in "Pin Configuration".



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

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

In our example, it is ”g_bsp_pin_cfg_2e1”.



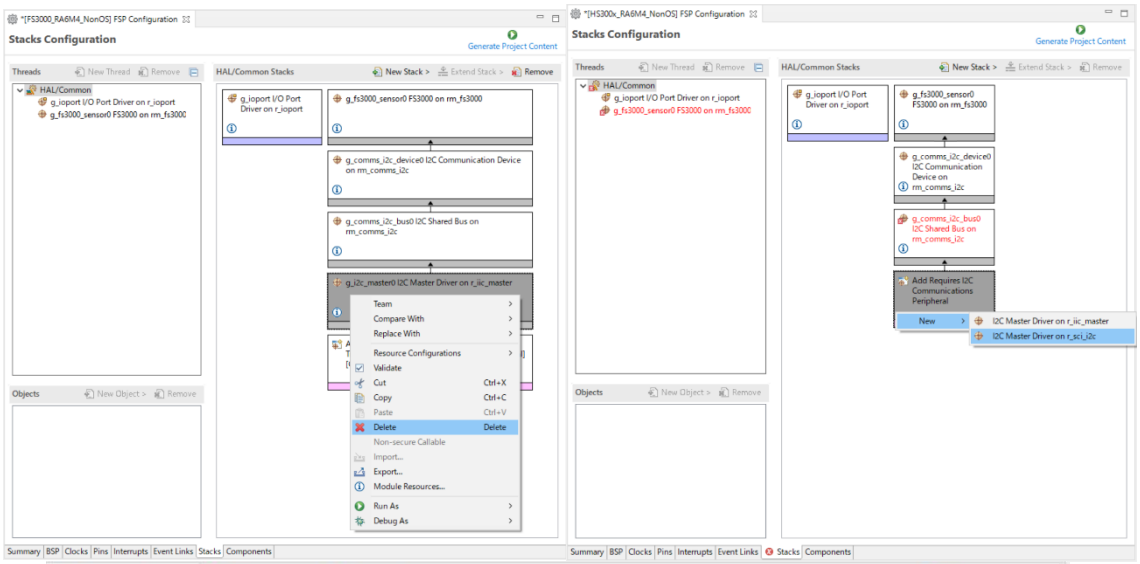
Modify the configuration of individual components in the "Stacks" tabbed page.

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

To use the pins of the IIC, delete the "I2C Master Driver on r_sci_i2c" stack and then add the "I2C Master Driver on r_iic_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 to "1".



Properties | Problems | Smart Browser

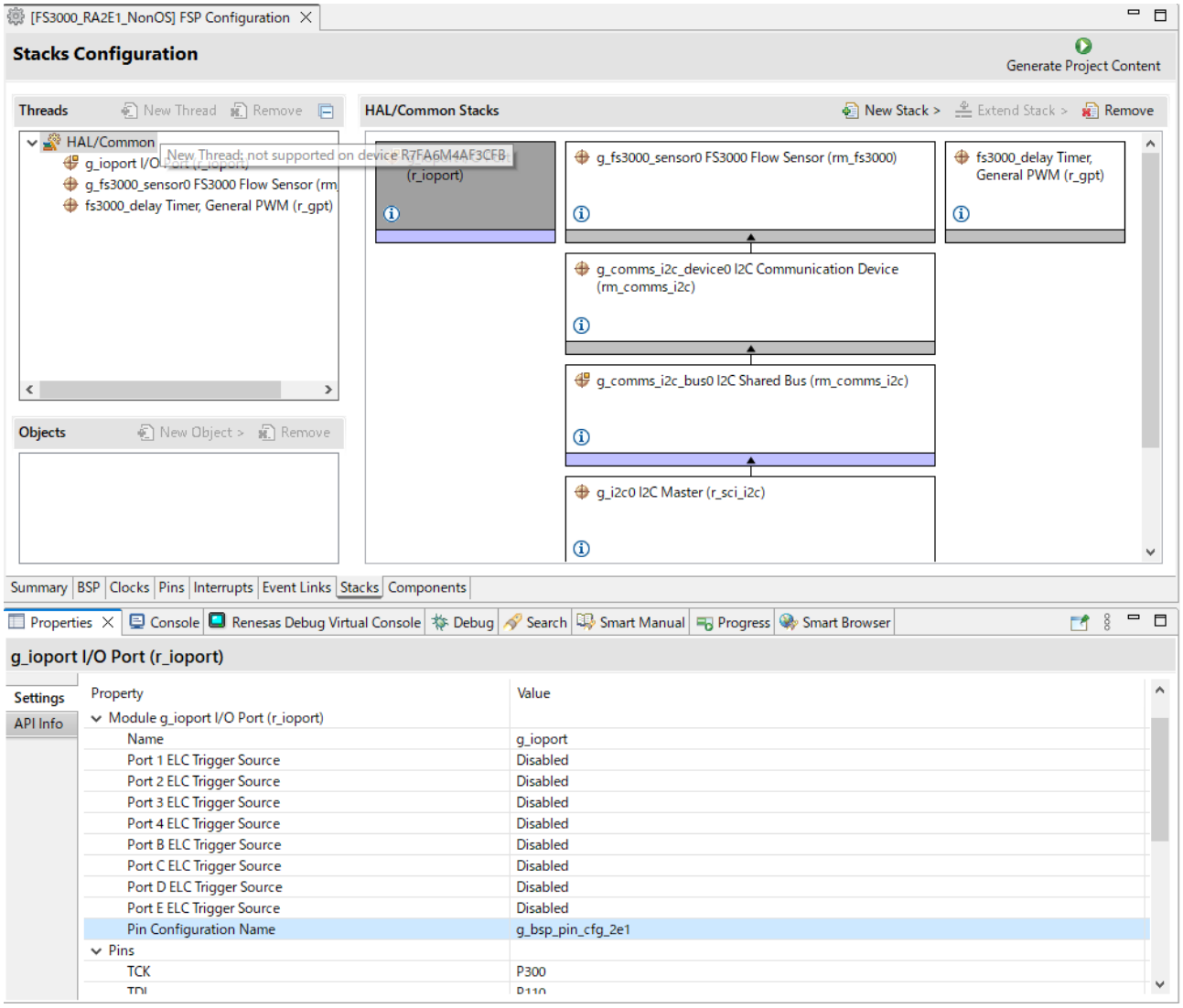
g_i2c0 I2C Master Driver on r_sci_i2c

Settings | API Info

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

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

In our example, it is ”g_bsp_pin_cfg_2e1”.



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 according to 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_FS3000.c (Non-OS) or fs3000_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

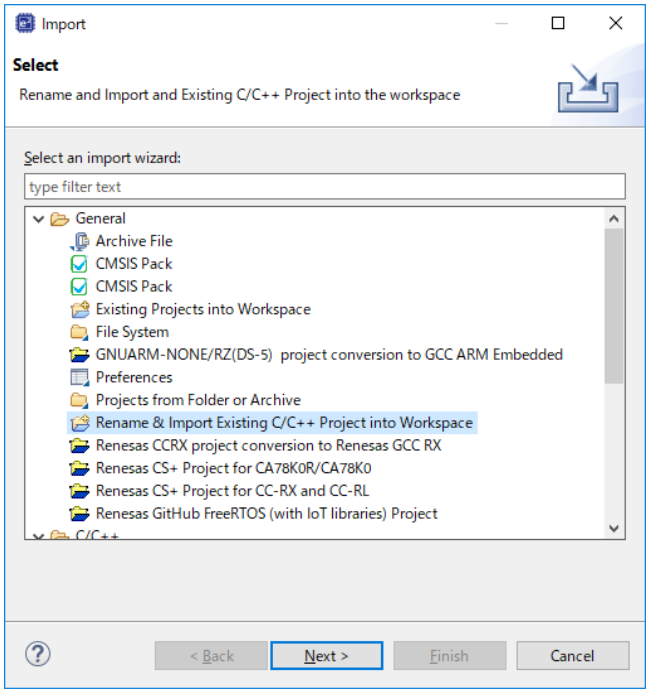
Use the following procedures to modify a sample project.

This section describes an example of modifying the sample project "FS3000_RX65N_NonOS" so that it can be used on the RSKRX231 board.

6.2.1 Importing the Sample Project

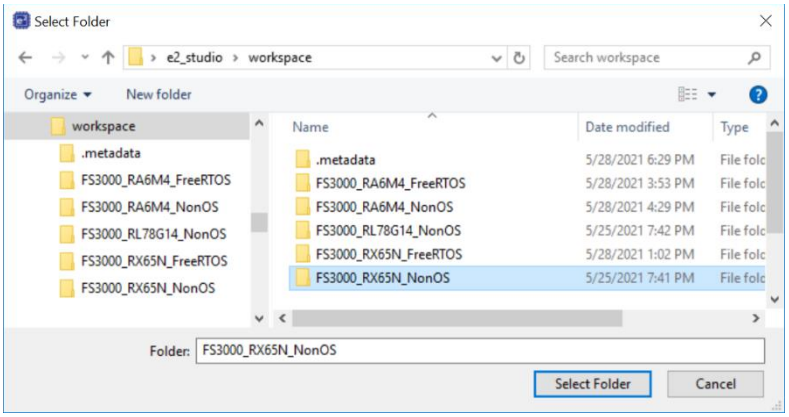
Select [Import] from the menu.

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

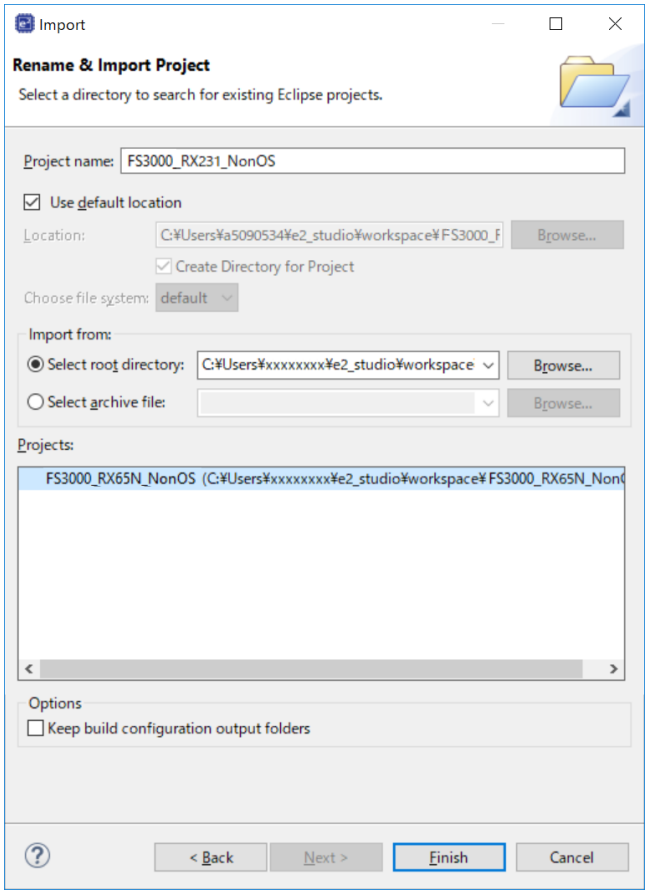


Press the [Browse] button to open the "Select Folder" window.

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

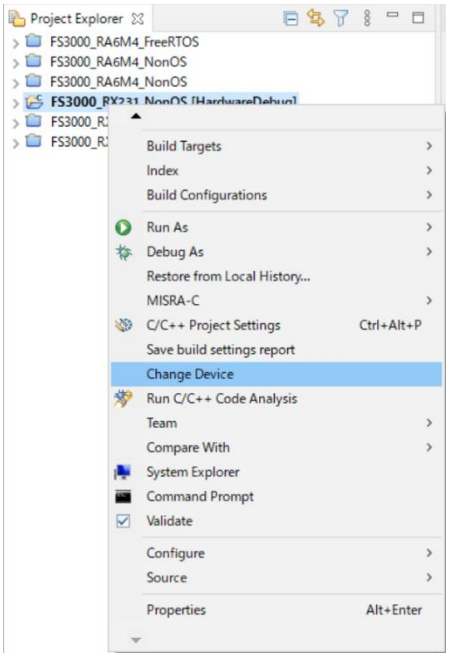


Enter the project name, select the original project for the current device, and press the [Finish] button.

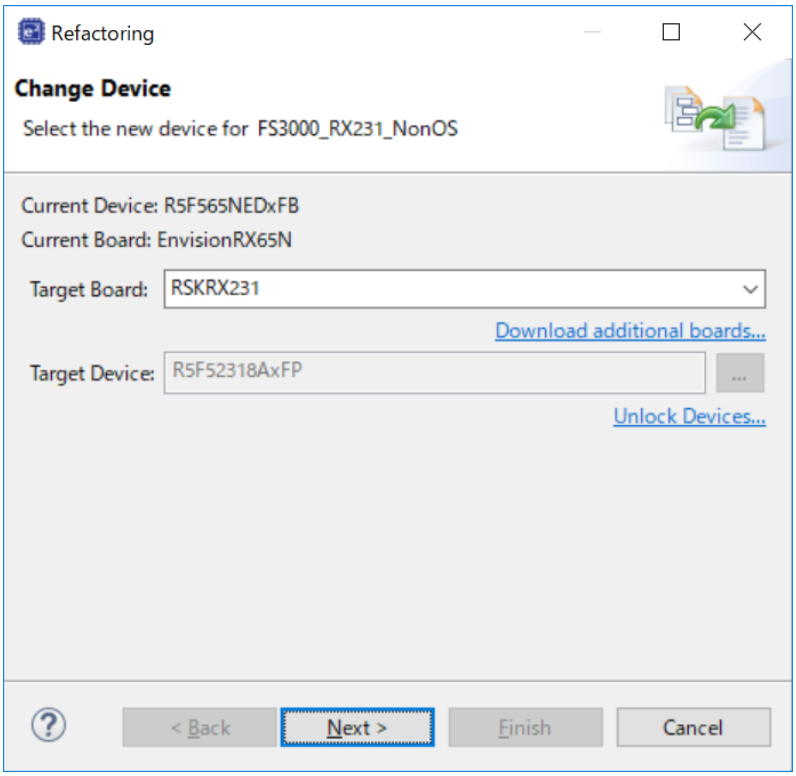


6.2.2 Changing the Device

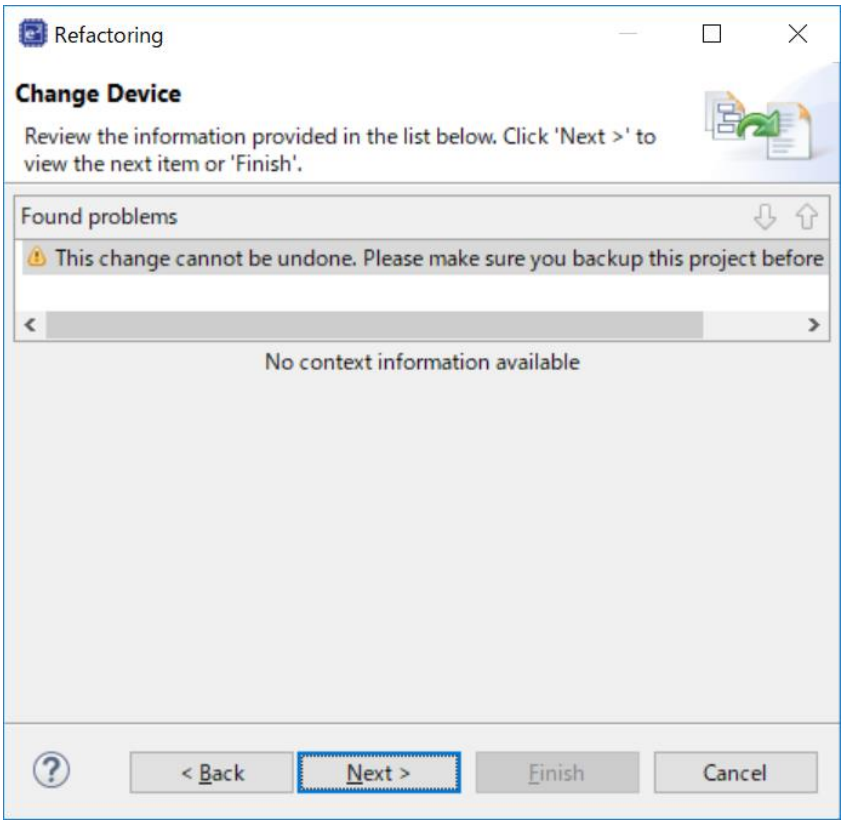
Select the imported project from the project tree and right-click on int to open the context menu. Select "Change Device" from the menu.



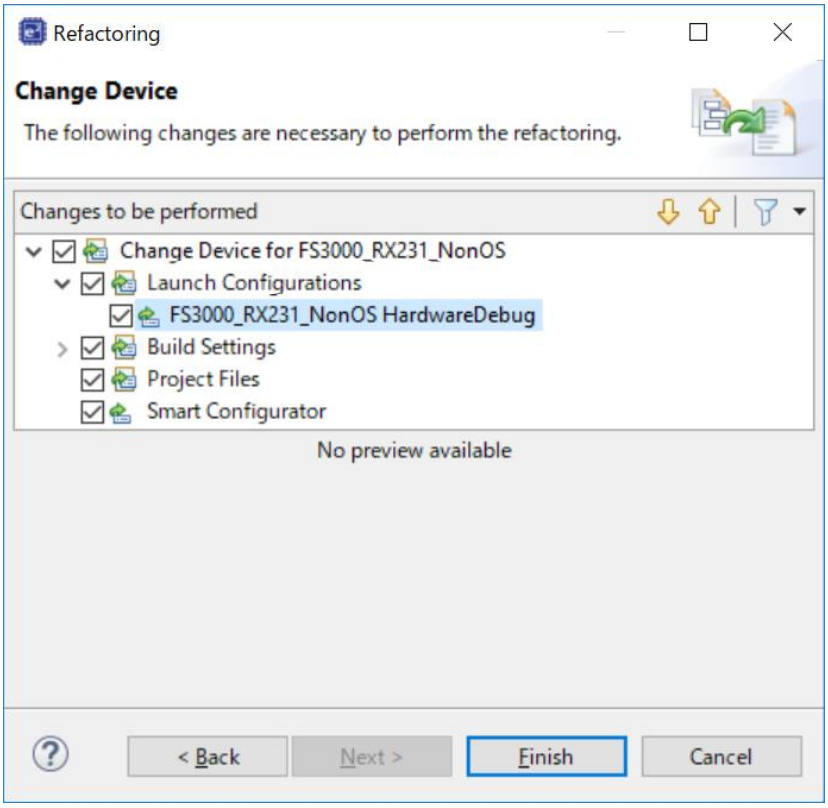
Select a desired board or device in the "Change Device" window and press the [Next] button.



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

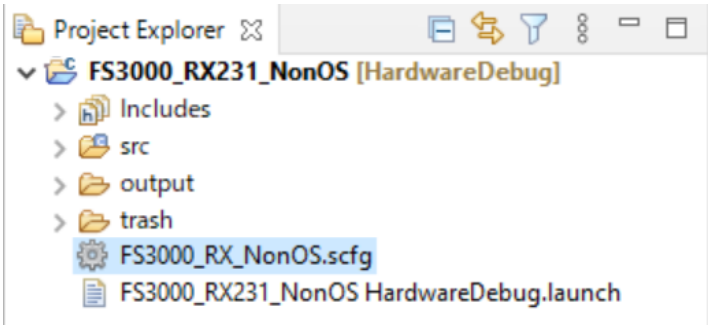


The changes you have made in the settings will be displayed. Press the [Finish] button to apply the changes to the project.

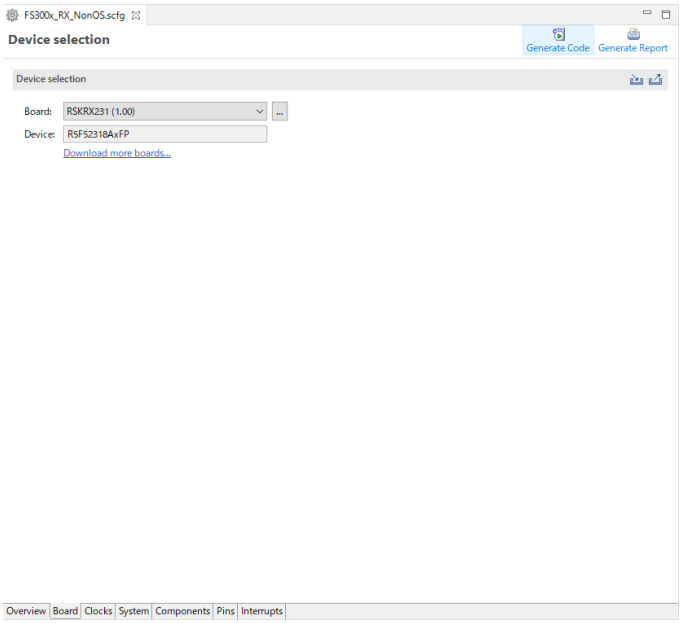


6.2.3 Modifying Settings of the Smart Configurator

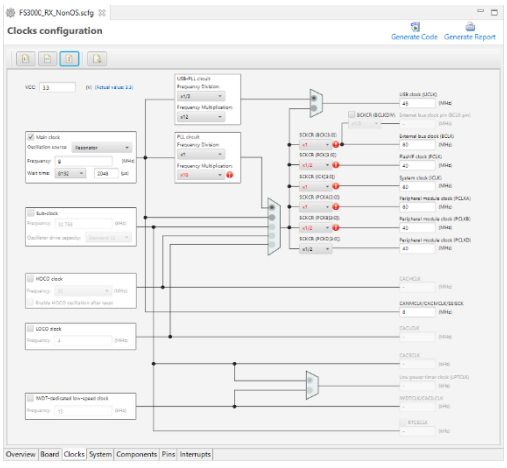
On the project tree, double-click on the .scfg file of the imported project in which the target device has been changed; the Smart Configurator window will open.

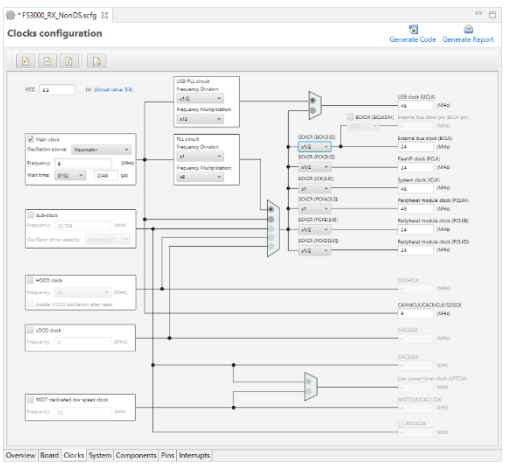


Select the "Board" tabbed page to check that the board and device have been changed correctly.



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

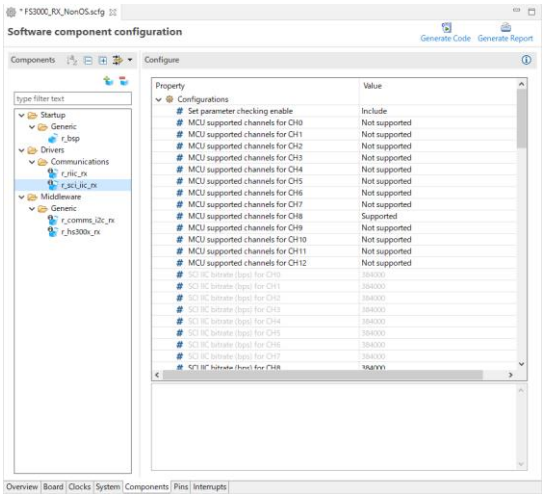


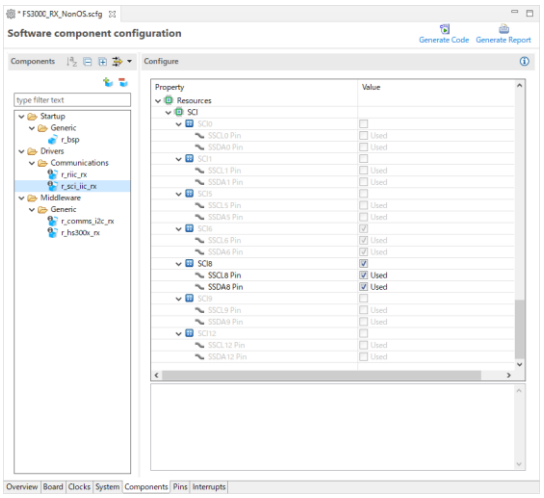


Modify the settings of individual components in the "Components" tabbed page according to the specifications of the target board.

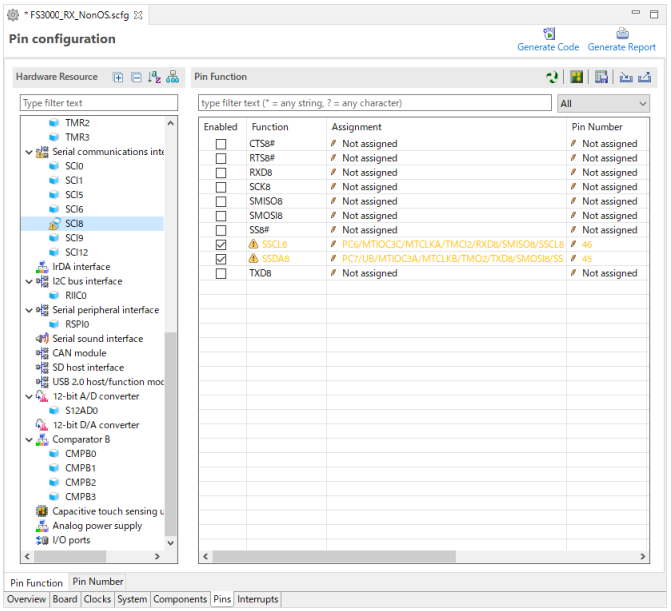
As SCI8 is assigned to PMOD on the RSK RX231 board, change the setting of "MCU supported channels for CH2" to "Not supported" and "MCU supported channels for CH8" to "Supported" in r_sci_iic_rx.

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





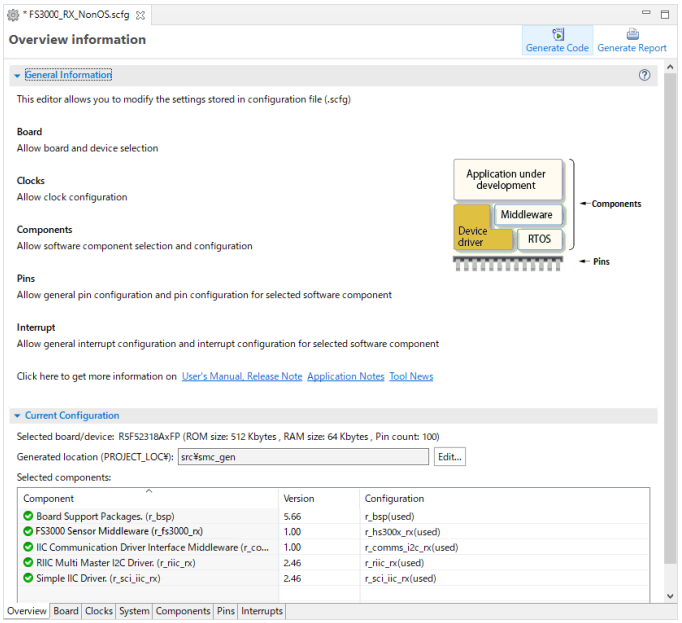
Open the "Pins" tabbed page and check that functions are assigned to the SCI8 pins in the "Pin function" panel.



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

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

Press the [Generate Code] icon to generate code.



Build the project.

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

6.2.4 Changing toolchain setting

If you want to use a toolchain other than the CC-RX toolchain, copy RA_FS3000.c (Non-OS), or main.c and fs3000_sensor_thread_entry.c (FreeRTOS), or fs3000_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

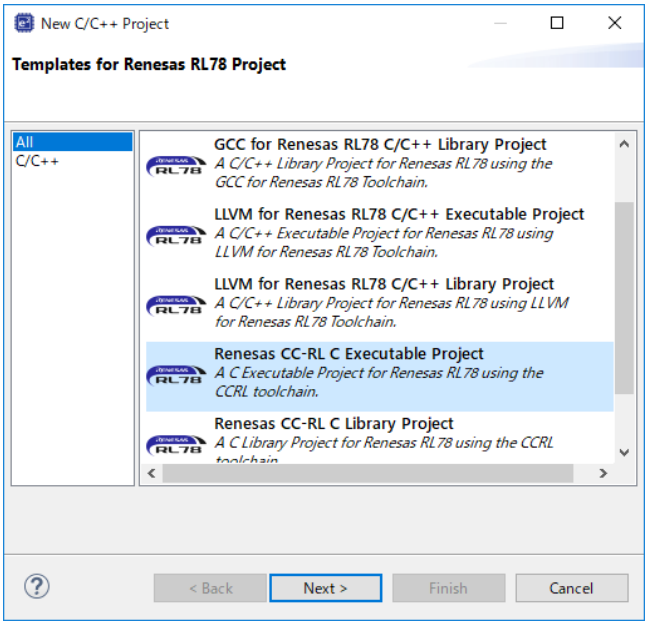
To change the target device of RL78 family, a new project should be created.

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

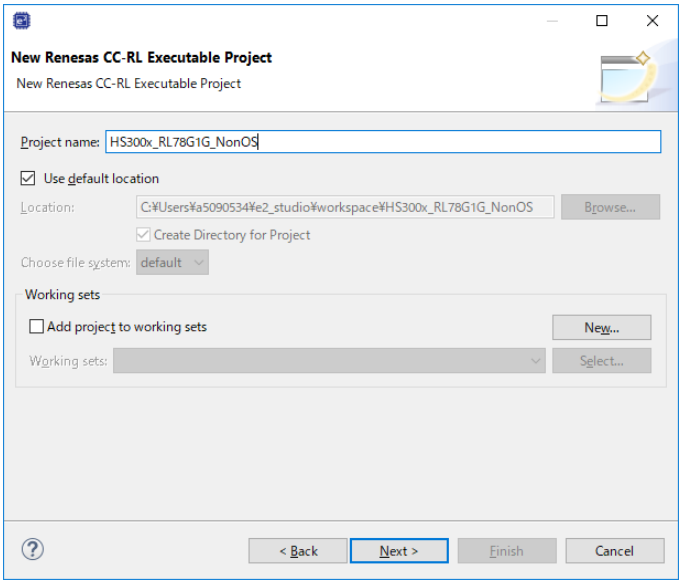
6.3.1 Creating a New Project

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

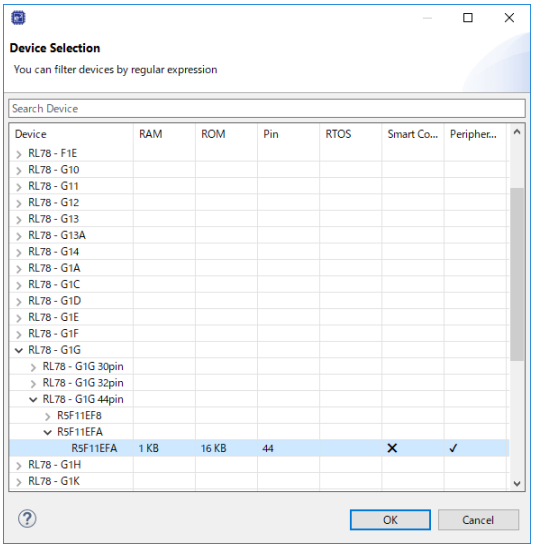
Select the template "Renesas CC-RL C Executable Project" and press the [Next] button.

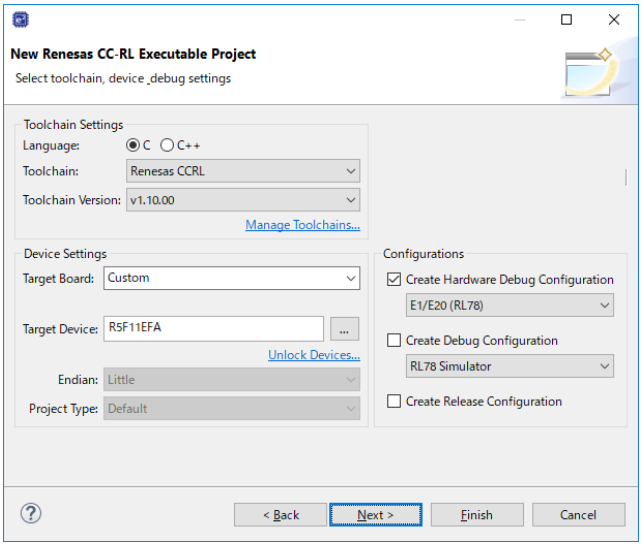


Enter the project name (example: "FS3000_RL78G1G_NonOS") and press the [Next] button.

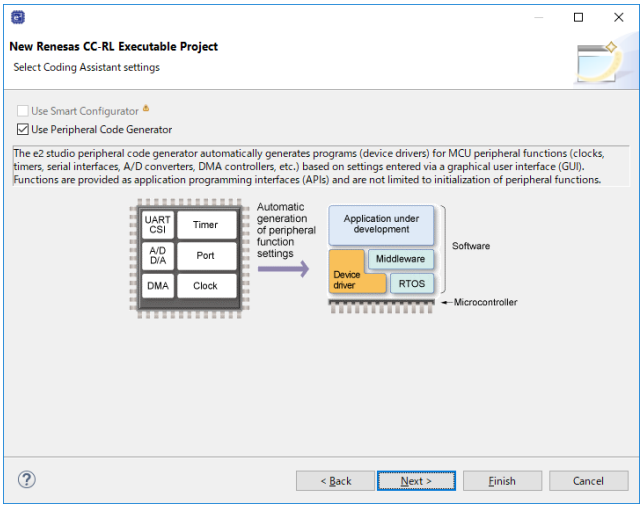


Change "Target Device" to a desired device (example: R5F11EFA) and press the [Next] button.

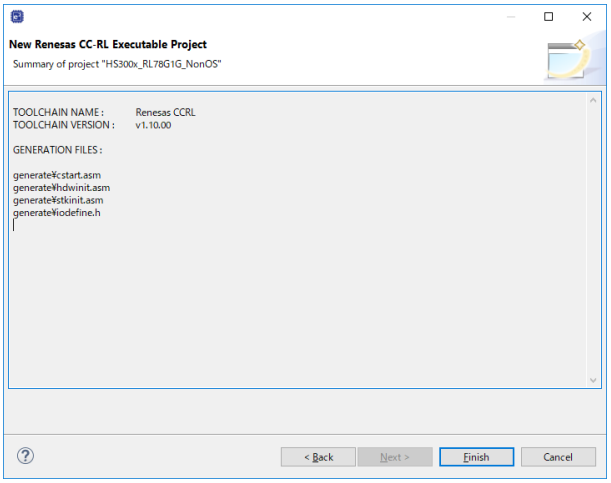




Select the checkbox for "Use Peripheral Code Generator" and press the [Next] button.

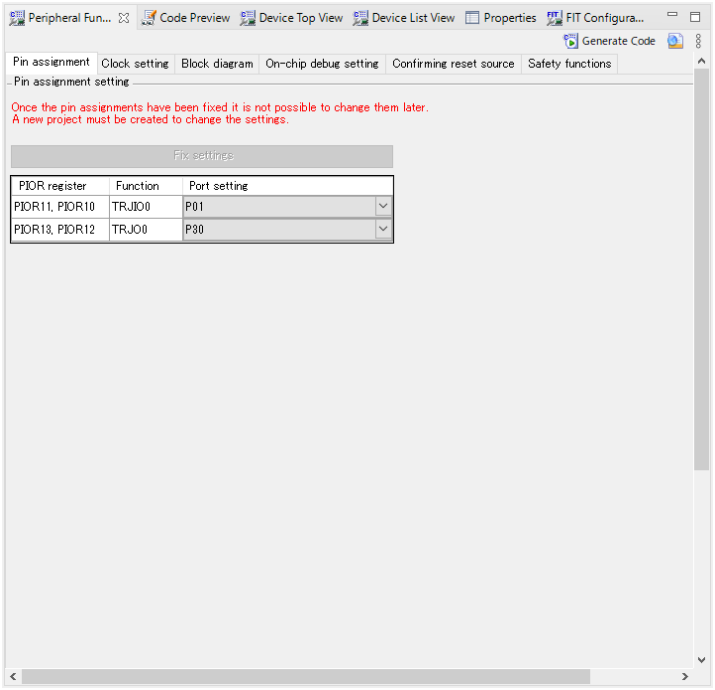


Press the [Finish] button.

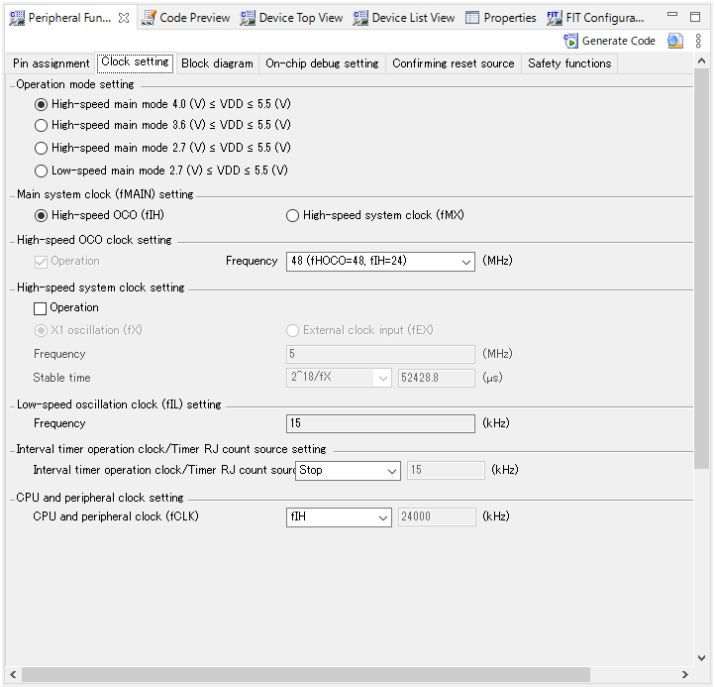


6.3.2 Settings of the Code Generator

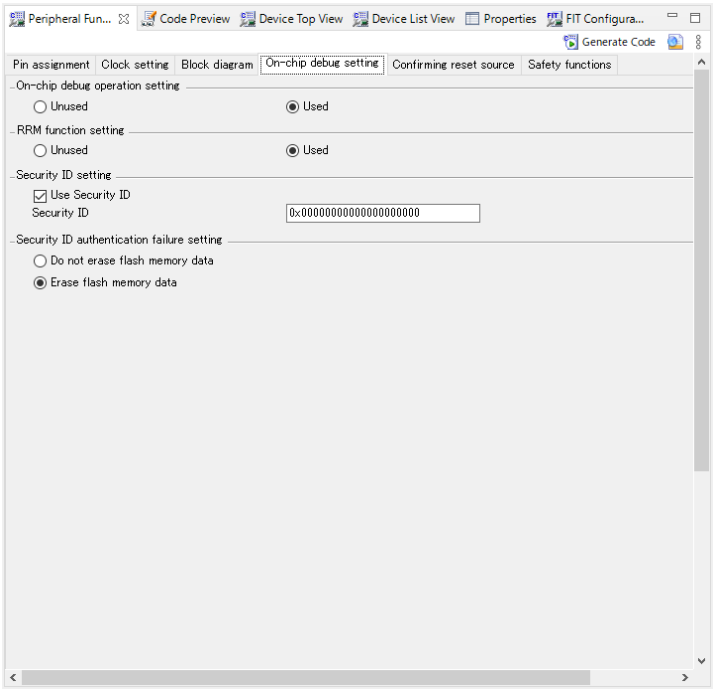
Modify the pin assignment in the "Pin assignment" tabbed page for "Common/Clock Generator" according to the specifications of the target board to be used.



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

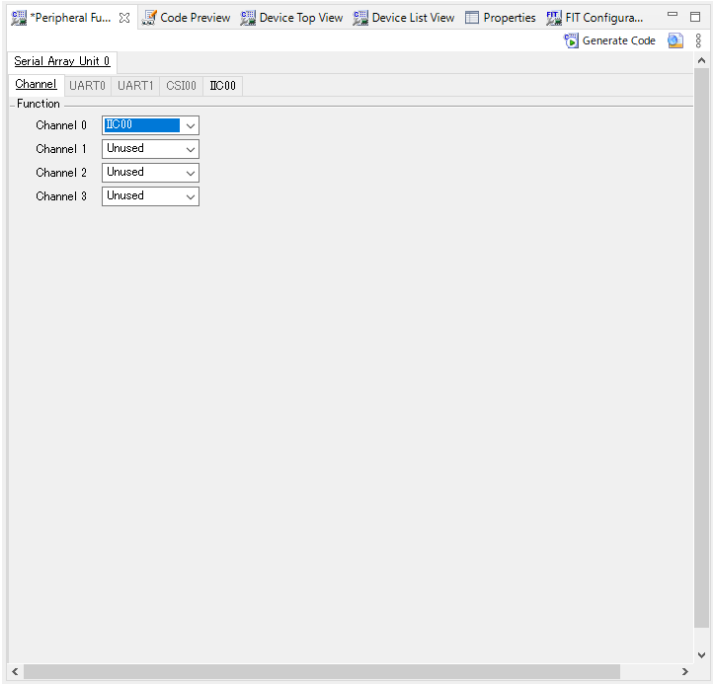


Select "Used" for "On-chip debug operation setting" in the "On-chip debug setting" tabbed page for "Common/Clock Generator".

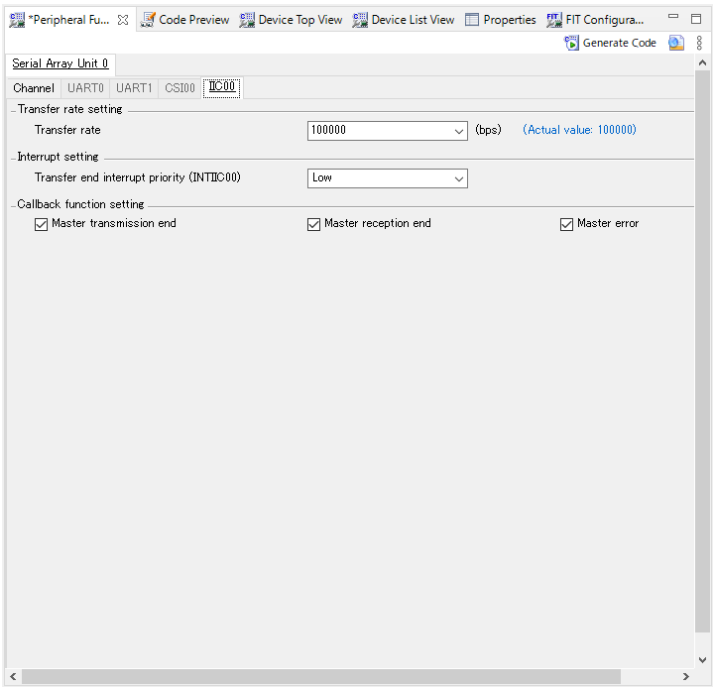


To use the serial array unit, set the channel assigned to PMOD on the target board to "IICxx" in the "Serial Array Unit" or "Serial" tabbed page.

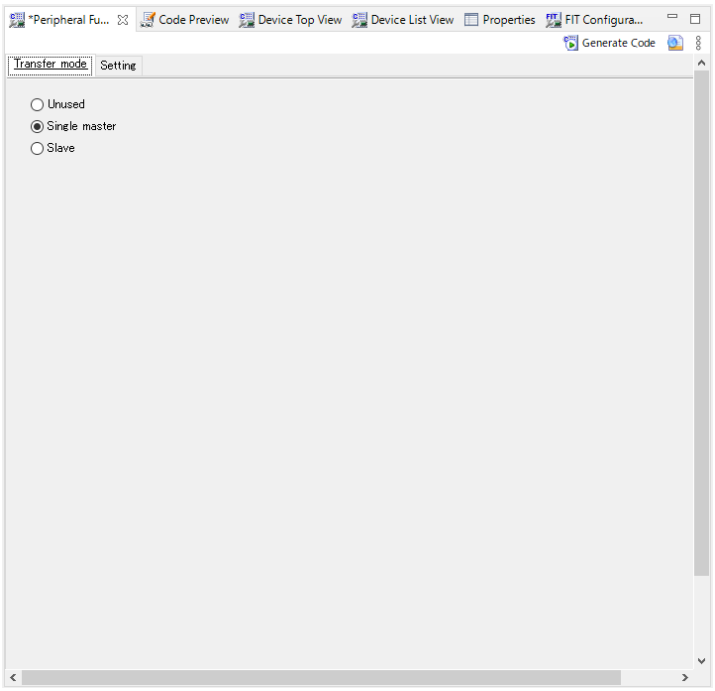
Note: The corresponding pin must be selected as N-ch by "Port".



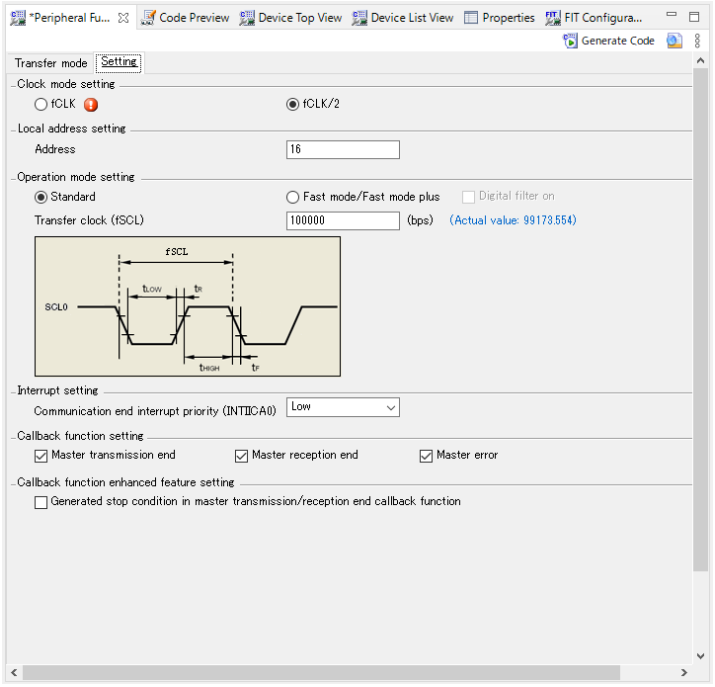
In the tabbed page for IICxx enabled in the serial array unit, set "Transfer rate" to 400000 or 100000, set "Transfer end interrupt priority" to a desired value, and enable all functions under "Callback function setting".
Note : When using a serial array unit, the Nch open drain of the pin to be used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



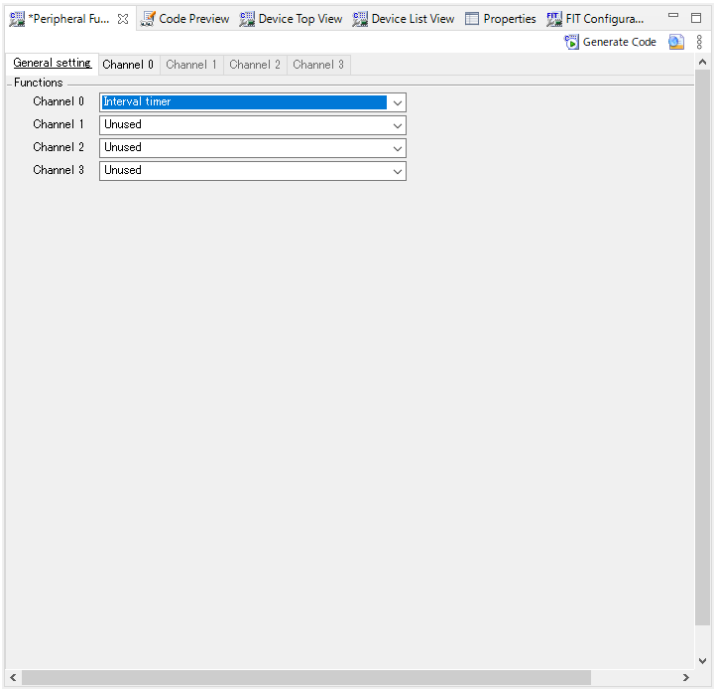
To use the serial interface IICA, select "Single master" in the "Transfer mode" tabbed page for the channel assigned to PMOD on the target board in the "Serial Interface IICA" or "Serial" setting window.



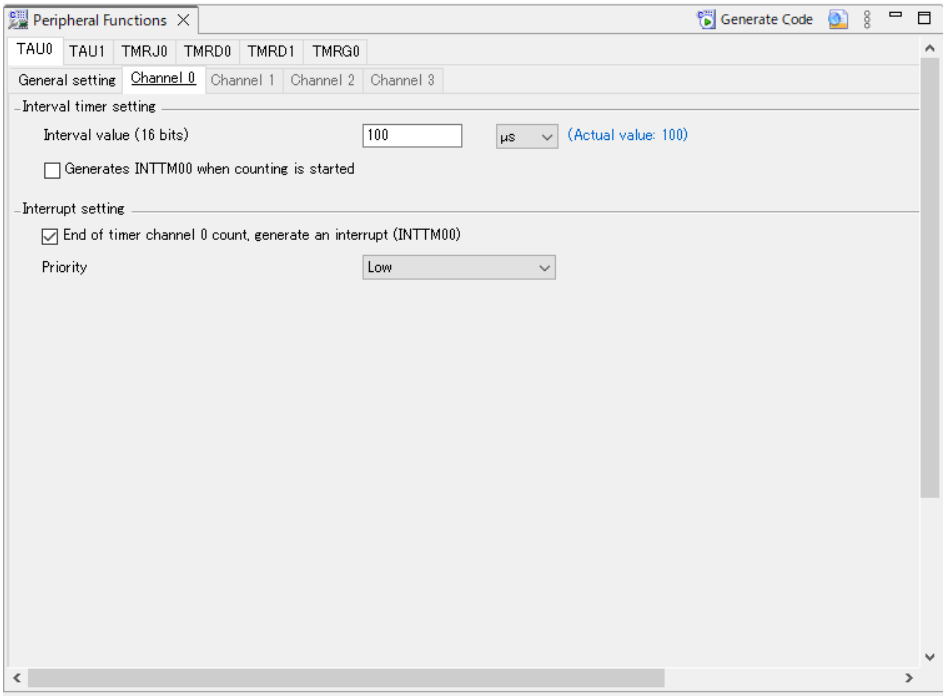
In the "Setting" tabbed page for the channel set to the single master, set "Operation mode setting" to either a combination of "Fast mode" and "400000" or a combination of "Standard" and 100000, set the interrupt priority to a desired level, enable all functions under "Callback function setting", and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



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 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.

Definition for including r_comms_i2c_if.h:

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

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

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

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

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

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

```
/***********************************************************************
Global variables and functions
***********************************************************************/
/* Start user code for global. Do not edit comment generated here */
extern void fs3000_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 */
    fs3000_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_fs3000_sensor0_quick_setup(void);
void start_fs3000_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 FS3000 */
    g_fs3000_sensor0_quick_setup();

    while (1U)
    {
        start_fs3000_demo();
    }
```

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

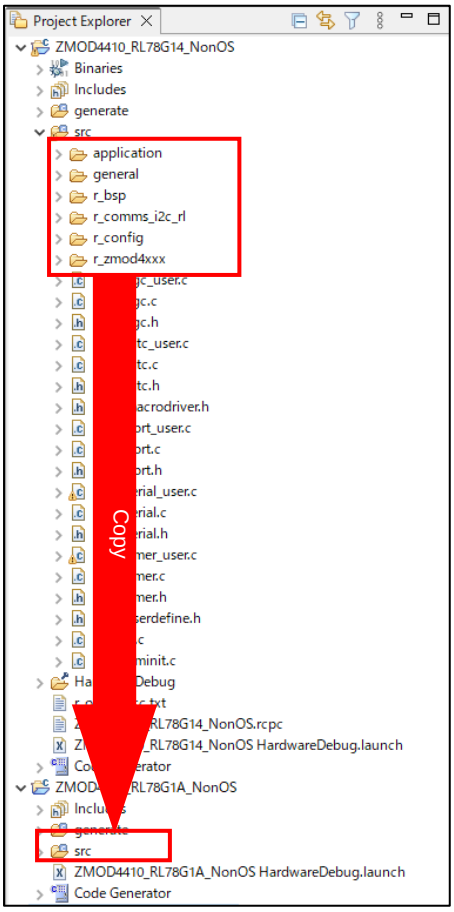
```
void g_comms_i2c_bus0_quick_setup(void)
{
    /* bus has been opened by startup process */
}

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_fs3000_rl" folder in the project tree of the sample project "FS3000_RL78G14_NonOS" and select "Copy" from the context menu.

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



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

- COMMS_I2C_CFG_BUSx_DRIVER_TYPE
- COMMS_I2C_CFG_BUSx_DRIVER_CH

When channel 0 of the serial array unit is used:

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

When channel 0 of the serial interface IICA is used:

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

For the other definitions, refer to section 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

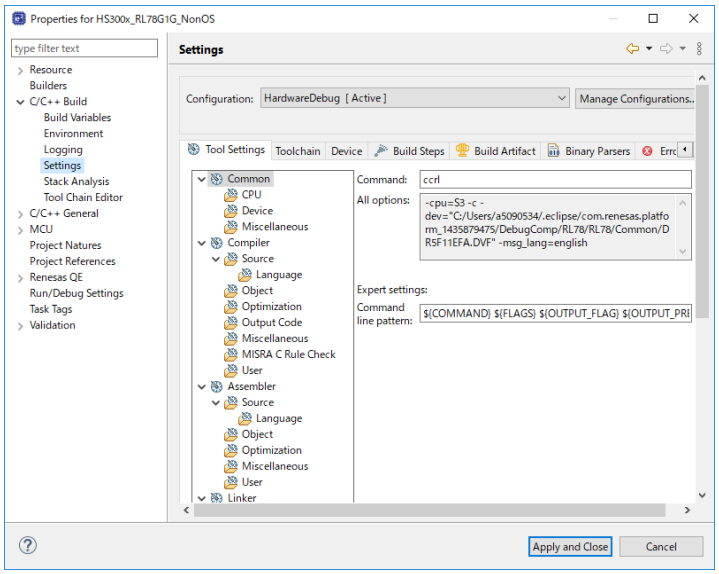
Modify "r_cg_serial.h" to "r_cg_sau.h" or "r_cg_iica.h":

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.

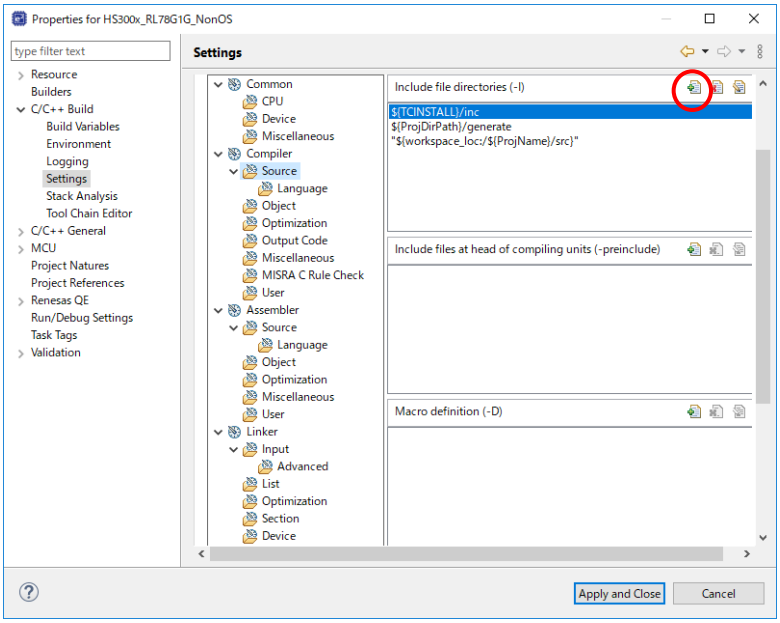
Select [C/C++ Build] → [Settings] in the "Properties" window to open the "Settings" panel.

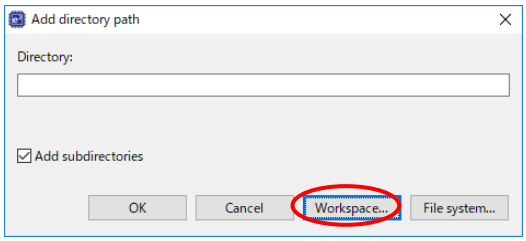


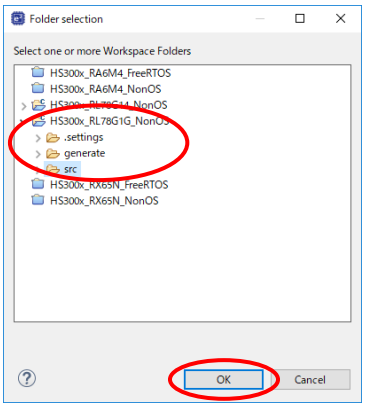
Select [Compiler] → [Source] in the "Tool Settings" tabbed page and press the [Add] icon.

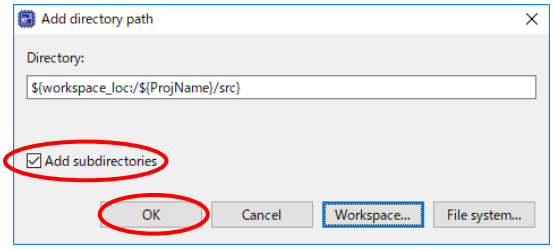
Press the [Workspace] button in the [Add directory path] dialog box and a list of projects will appear. Select the "src" folder for the newly created project in the list and press the [OK] button.

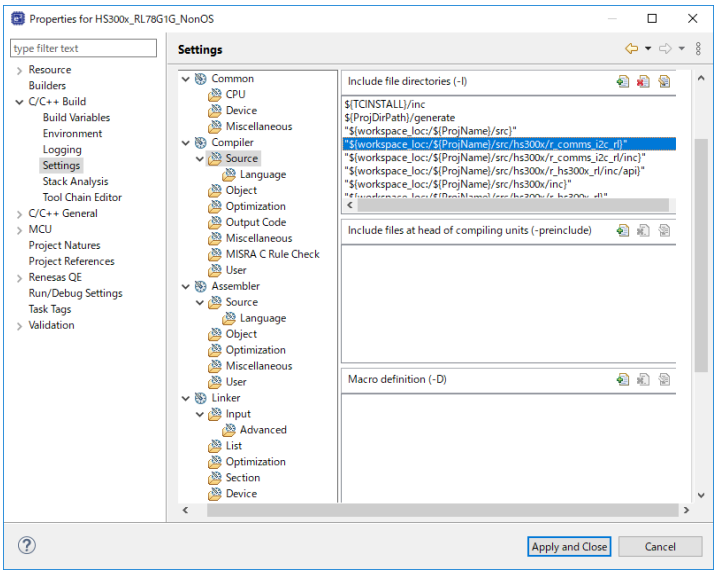
Select the checkbox for "Add subdirectories" and press the [OK] button.





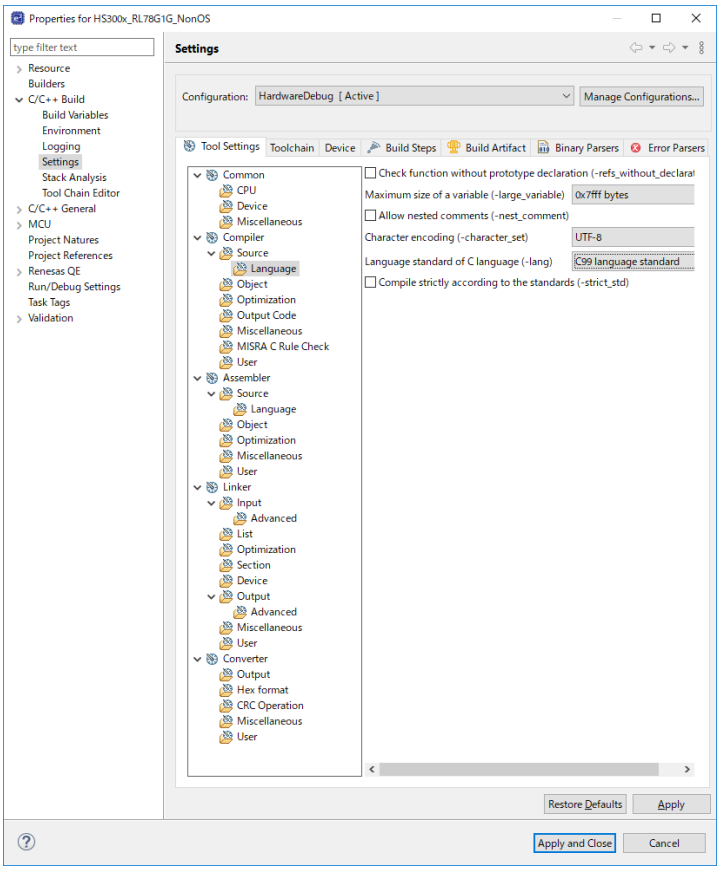






Select [Compiler] → [Source] → [Language] in the "Tool Settings" tabbed page and change the setting of "Language standard of C language" to "C99 language standard".

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



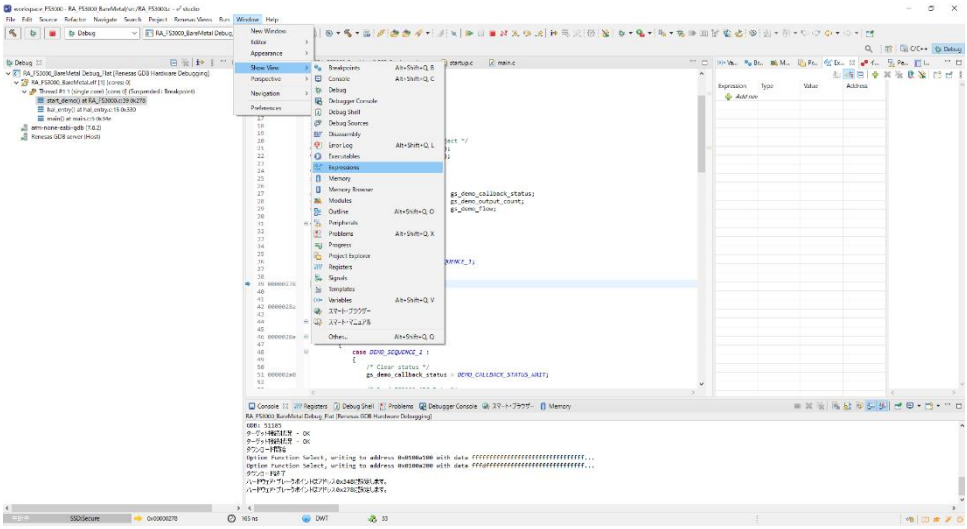
Build the project.

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

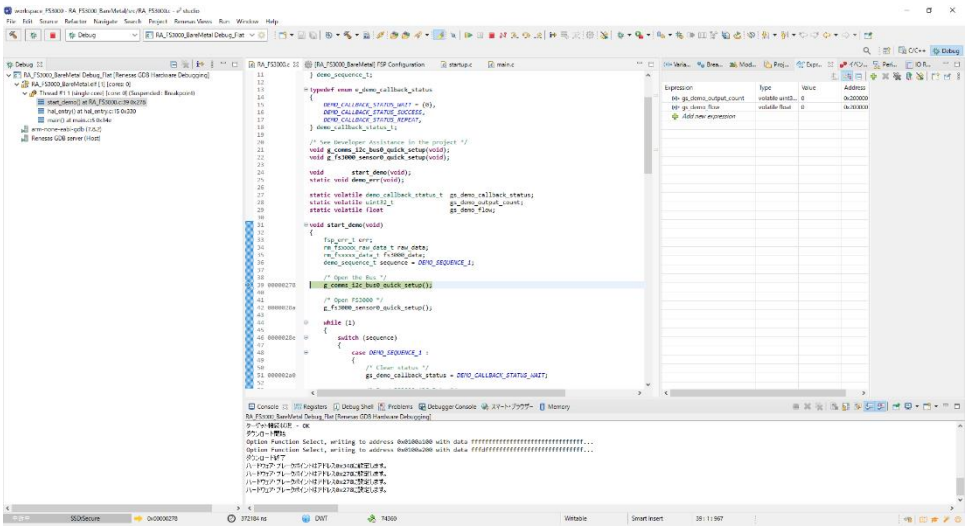
7. Viewing Air Velocity Data

Use the following procedure to view air velocity data in real time.

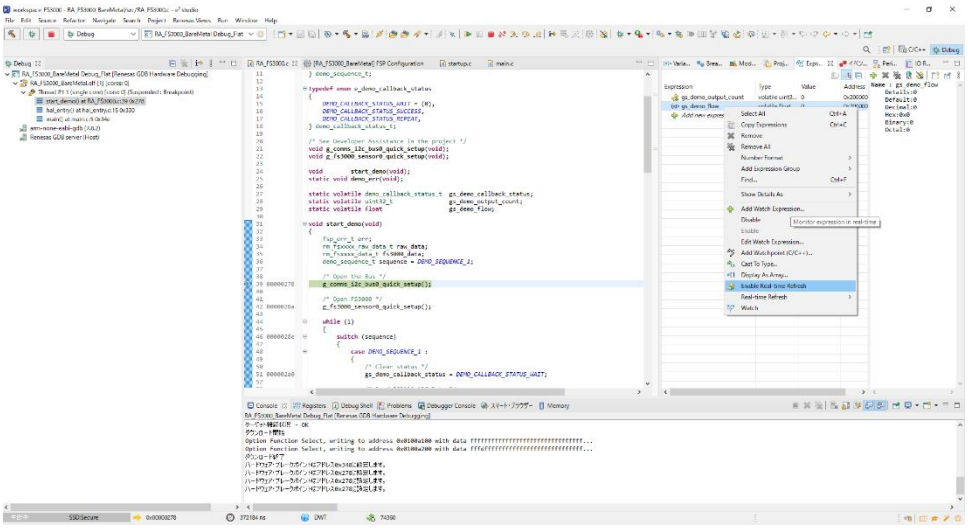
After executing debugging, select [Window] → [Show View] → [Expressions] to open the [Expressions] tabbed page.



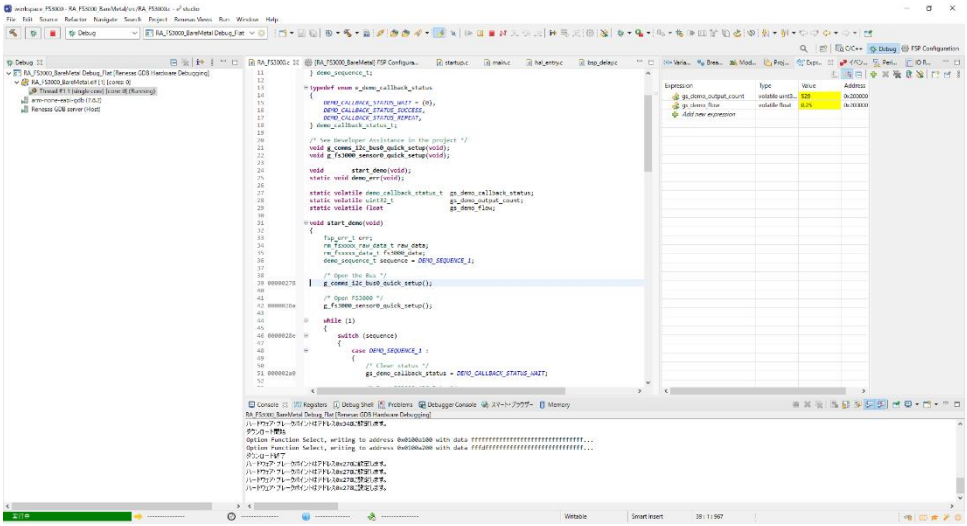
Click on [Add new expression] on the [Expressions] tabbed page and add "gs_fs3000_data".



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



Start debugging, and the values of the variables will be updated in real time.



Revision History

Rev.	Date	Description	
		Page	Summary
1.00	June 30, 2022	-	First Release
1.01	March 3, 2023	-	Updated: environments for RL78
1.02	March 29, 2023	-	Updated: Environments for RA, RX, RL78, RZ Updated: Main Processing Flow of Sample Software Updated: Guide for Changing the Target Device
1.03	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/.