RA ファミリ、RX ファミリ、RL78 ファミリ
FS3000 サンプルソフトウェアマニュアル

要旨

本アプリケーションノートでは、RA ファミリ、RX ファミリ、RL78 ファミリで動作する FS3000 フローセンサのサンプルソフトウェアについて説明します。

動作確認デバイス

RA6M4 グループ

RX65N グループ

RL78/G14 グループ

RL78/G23 グループ

商標・他社 TM

FreeRTOS™ と FreeRTOS.org™ は Amazon Web Serices, Inc.の登録商標です。

Microsoft® Azure RTOS は米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。

Contents

1.　概要

本ソフトウェアは、FS3000-1005 風速センサのデータの取得および、演算を行うためのサンプルプログラムです。MCU に内蔵されている I2C を用いて、FSP / FIT の I2C ドライバとの組み合わせにより、センサから ADC データの取得および風速値の演算を行います。

2. 動作確認環境

2.1 RA 動作確認環境

本ソフトウェアの動作確認環境を表 2-1 RA 動作環境に示します。

表 2-1 RA 動作環境

項目	内容
デモボード	RTK7EKA6M4S00001BE (EK-RA6M4)
使用マイコン	RA6M4 (R7FA6M4AF3CFB: 144pin)
動作周波数	200MHz
動作電圧	5V
統合開発環境	e^2 Studio 2023-01
C コンパイラ	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™ / Microsoft® Azure RTOS
エミュレータ	On board (J-LINK)
変換ボード	Interposer Board to convert Type2/3 to Type 6A PMOD standard (US082-INTERPEVZ)
センサボード	PMOD Daughter Card for FS3000 flow sensor (US082-FS3000EVZ)

表 2-2 RA 使用メモリ量

領域	サイズ(Non-OS)	サイズ(FreeRTOS)	サイズ(Azure RTOS)
ROM	1,295 bytes	1,628 bytes	1,572 bytes
RAM	73 bytes	253 bytes	422 bytes

メモリサイズは FS3000 に関連する関数と変数のみを使用して計算しています。RTOS の場合は、スレッドのメモリサイズは計算に含めていません。

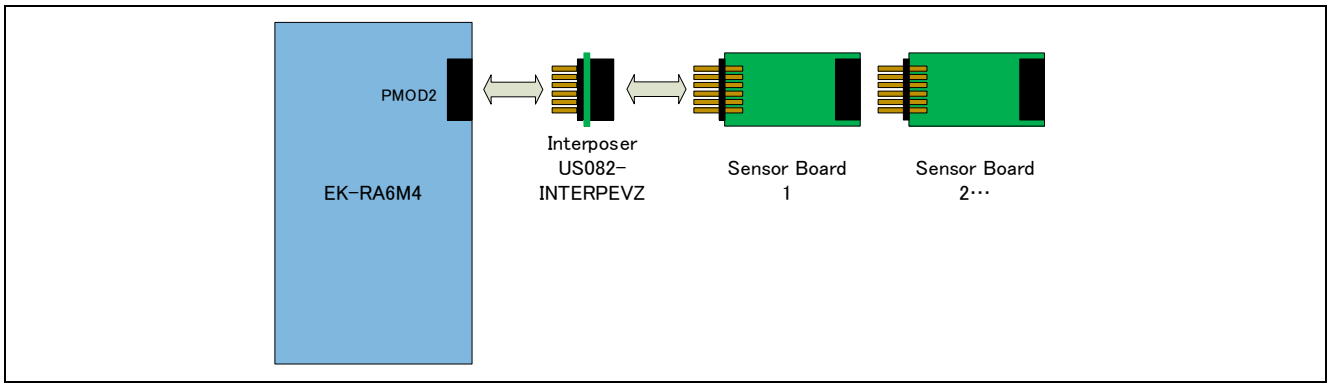


図 2-1 HW 接続図

2.2 RX 動作確認環境

本ソフトウェアの RX 動作確認環境を以下に示します。

表 2-3 RX 動作環境

項目	内容
デモボード	RPBRX65N (Envision Kit RX65N)

使用マイコン	RX65N (R5F565NEDDFB : 144pin)
動作周波数	12MHz
動作電圧	5V
統合開発環境	e^2 Studio 2023-01 IAR EW for RX 4.20.1
C コンパイラ	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
エミュレータ	On board (E2OB)
変換ボード	Interposer Board to convert Type2/3 to Type 6A PMOD standard (US082-INTERPEVZ)
センサボード	PMOD Daughter Card for FS3000 flow sensor (US082-FS3000EVZ)

表 2-4　RX 使用メモリ量

領域	サイズ(Non-OS)	サイズ(FreeRTOS)	サイズ(Azure RTOS)
ROM	1,467 bytes	1,685 bytes	1,745 bytes
RAM	145 bytes	205 bytes	414 bytes

メモリサイズは FS3000 に関連する関数と変数のみを使用して計算しています。RTOS の場合は、スレッドのメモリサイズは計算に含めていません。

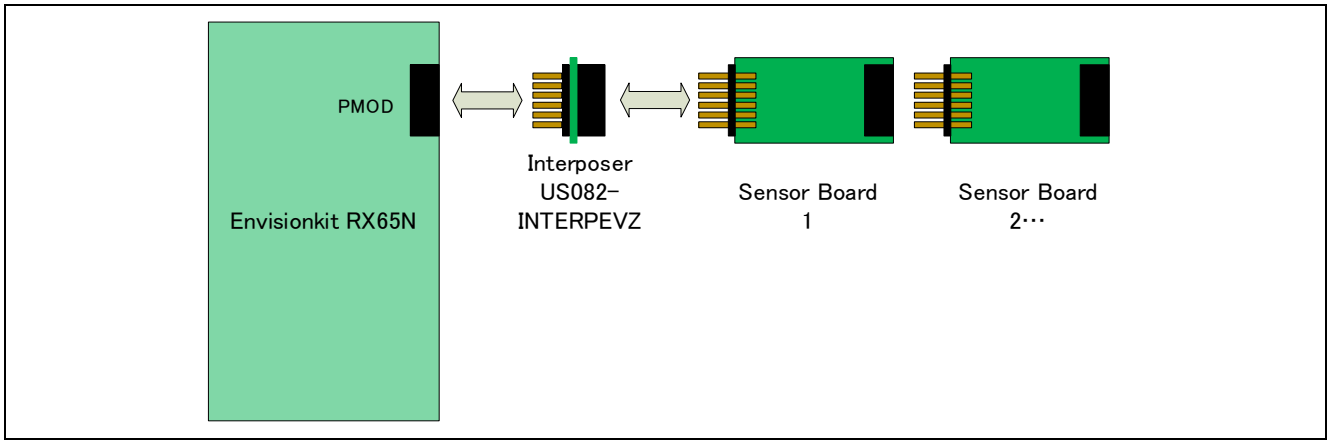


図 2-2　RX HW 接続図

2.3　RL78/G14 動作確認環境

本ソフトウェアの RL78/G14 動作確認環境を以下に示します。

表 2-5　RL78/G14 動作環境

項目	内容
デモボード	RTK5RLG140C00000BJ (RL78/G14 Fast Prototyping Board)
使用マイコン	RL78/G14 (R5F104MLAFB : 80pin)
動作周波数	32MHz
動作電圧	3.3V
統合開発環境	e^2 Studio 2023-01 IAR EW for RL78 4.21.1
C コンパイラ	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
エミュレータ	On board (E2OB)
センサボード	PMOD Daughter Card for FS3000 flow sensor (US082-FS3000EVZ)

表 2-6　RL78/G14 使用メモリ量

領域	サイズ
ROM	1,359 bytes
RAM	92 bytes

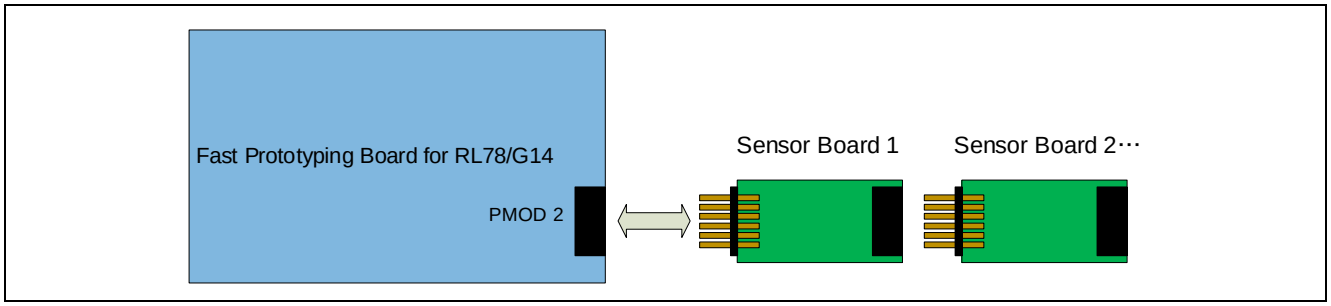


図 2-3　RL78/G14 HW 接続図

2.4　RL78/G23 動作確認環境

本ソフトウェアの RL78/G23 動作確認環境を以下に示します。

表 2-7　RL78/G23 動作環境

項目	内容
デモボード	RTK7RLG230CSN000BJ (RL78/G23-128p Fast Prototyping Board)
使用マイコン	(R7F100GSN2DFB :128pin)
動作周波数	32MHz
動作電圧	3.3V
統合開発環境	e^2 Studio 2023-01 IAR EW for RL78 4.21.1
C コンパイラ	C compiler package for RL78 family V1.10.00 LLVM for RL78 10.0.0.202209 IAR Toolchain for RL78 4.21.1.2409
エミュレータ	E2 Lite
センサボード	PMOD Daughter Card for FS3000 flow sensor (US082-FS3000EVZ)

表 2-8　RL78/G23 使用メモリ量

領域	サイズ
ROM	1,684 bytes
RAM	308 bytes

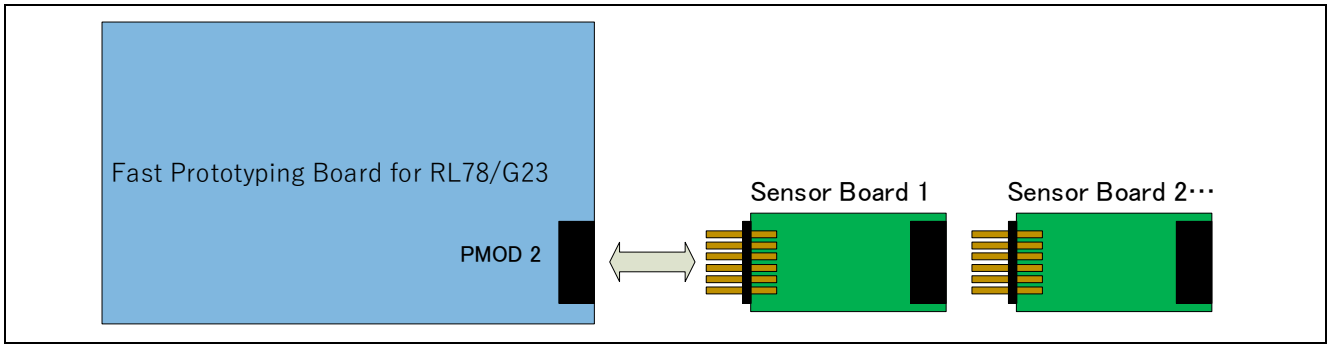


図 2-4　RL78/G23 HW 接続図

3.　センサ仕様

3.1　センサ仕様概要

FS3000-1005 風速センサの仕様概要は以下の通りです

表 3-1　センサ機能概要

項目	内容
風速の測定範囲	0 – 7.23 [m/sec]
カウント値の範囲	409 – 3686 [counts]
分解能	12 bit
精度	5% (at 25℃)
測定時間	125 [ms]
I2C クロック周波数	100kHz, 400kHz 対応
スレーブアドレス	0x28
アドレスモード	7-bit address のみ対応

風速は以下の曲線に従ってカウント値から計算を行います。

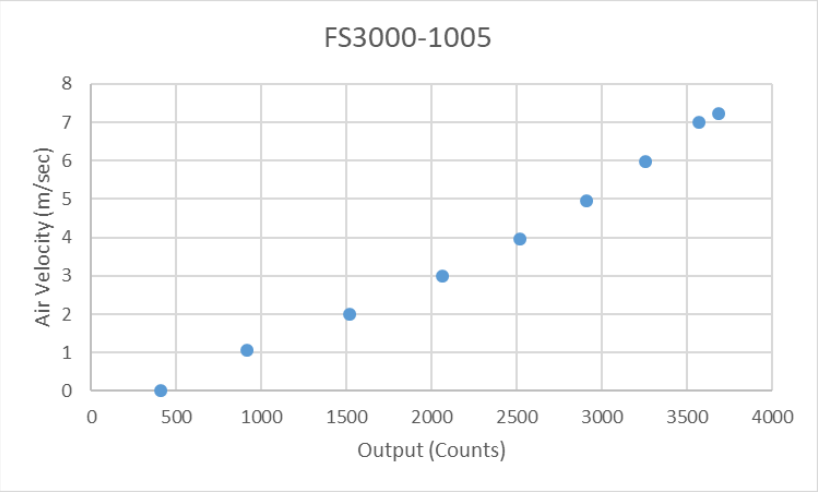


図 3-1　FS3000-1005 カウント値と風速の関係

カウント値と風速の関係は以下になります。

表 3-2　風速と計測カウント関係図

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　センサ機能

FS3000 サンプルソフトウェアではフローセンサのうち FS3000-1005 風速センサに対応しています。（FS3000-1015 センサは未対応）

センサは電源投入と同時に測定を行います。

センサデータを取得する場合、図のようなコマンドを送信し、それに続いて、5 バイトのデータを Read します。

風速データは 12bits で、2 バイト目(Byte 2)の下位 4bit が有効なデータになります。

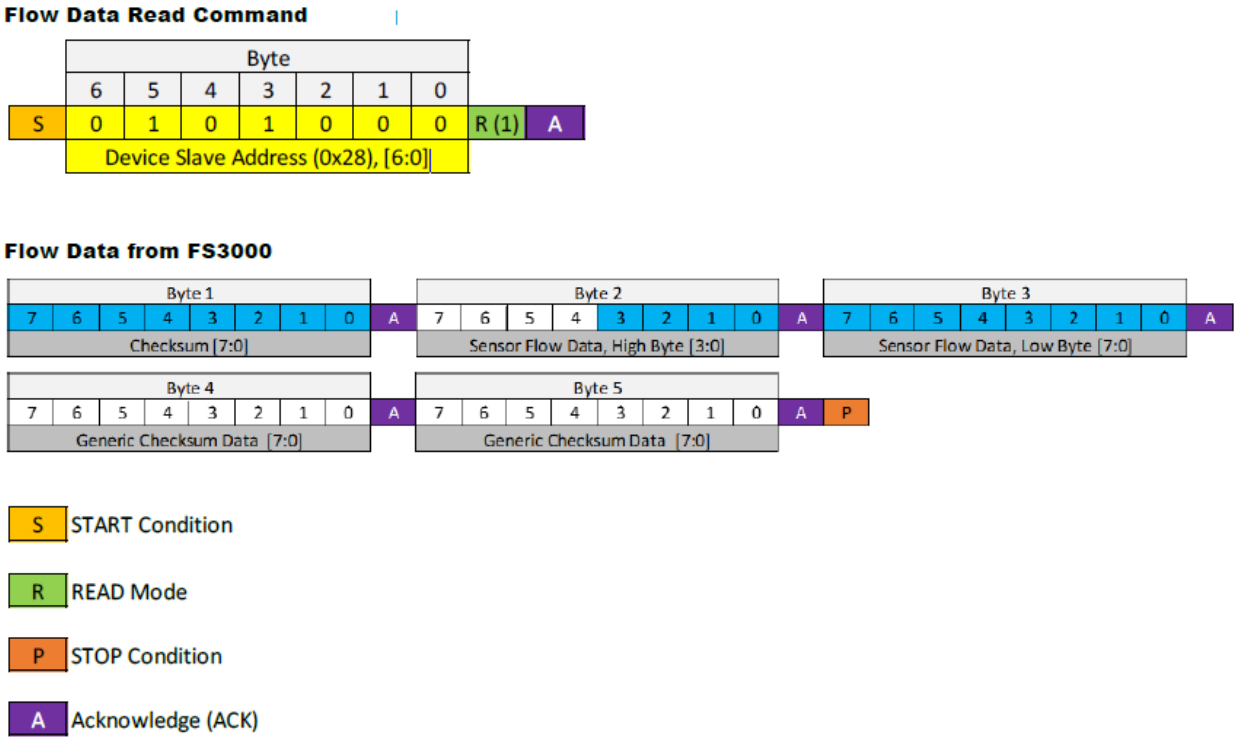


図 3-2　データ構造

また、チェックサムを用いて、以下のような計算で、有効なデータか判断することが可能です。

Example :

1 バイト目 : 0xCC（Checksum）

2 バイト目 : 0x01（Flow data, High byte）

3 バイト目 : 0x99（Flow data, Low byte）

4 バイト目 : 0x01（General checksum）

5 バイト目 : 0x99（General checksum）

Checksum 以外を足します。

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

Checksum と Sum を足したときに、下位 8bit が 0x00 の場合に、データは有効であると判断できます。

Checksum + Sum = 0xCC + 0x134 = 0x200 ← 下位 8bit が 0x00 のため、有効なデータ

4.　サンプルソフトウェア仕様

サンプルソフトウェアパッケージには RA の Non-OS 版と OS 版 (FreeRTOS / Azure RTOS)、RX の Non-OS 版と OS 版、RL78 の Non-OS 版の合計 6 つのプロジェクトが含まれます。それぞれのプロジェクトについて以降に説明します。

RX の FreeRTOS の設定方法については FAQ を参照してください。

4.1　サンプルソフトウェア構成

サンプルソフトウェアのブロック構成を図 4-1　ソフトウェアブロック図に示します。

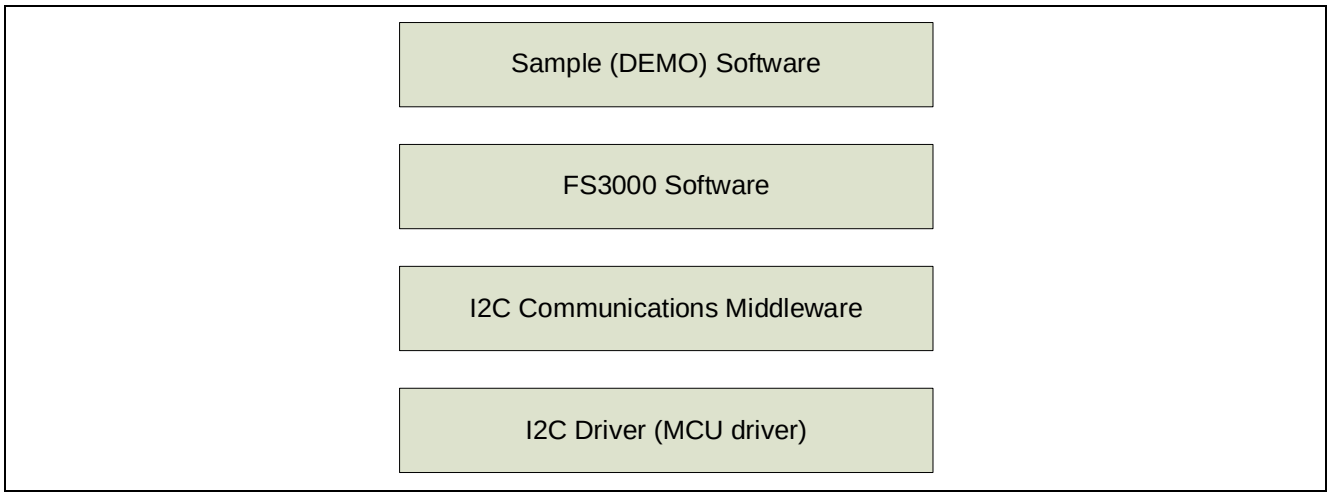


図 4-1　ソフトウェアブロック図

4.2　センサ API 関数仕様

4.2.1　センサ API 関数一覧

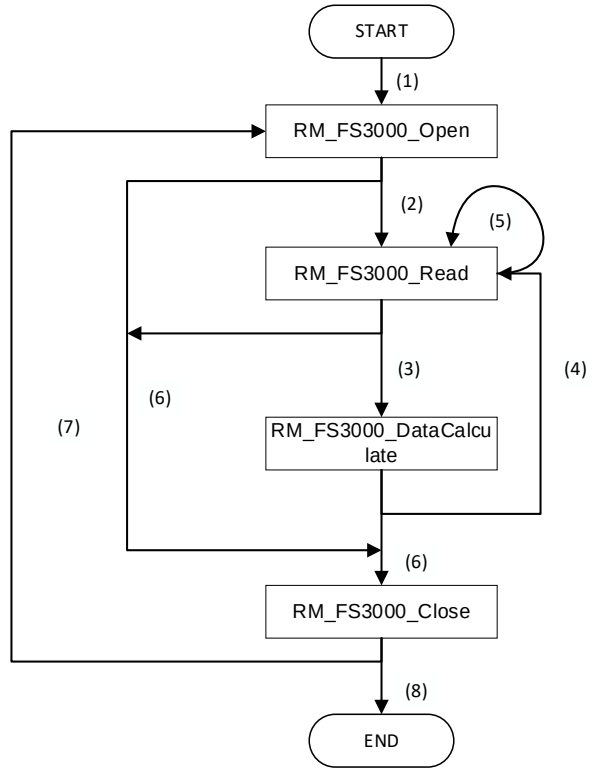
センサ API は以下の関数が含まれます。関数 API の詳細は別途 FS3000 Sensor API FIT Module アプリケーションノート (R01AN5894)、RL78 用ルネサスセンサーソフトウェア API 仕様アプリケーションノート(R01AN5896)を参照してください。

表 4-1　センサ API 関数一覧

関数	機能
RM_FS3000_Open	センサ制御開始処理
RM_FS3000_Close	センサ制御終了処理
RM_FS3000_Read	センサデータ取得処理
RM_FS3000_DataCalculate	センサデータ結果演算処理

4.2.2　API 使用ガイド

FS3000 の API 関数の使用条件について、想定する関数コールの順番を遷移図として示します。



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

図 4-2　関数 API 遷移図

関数毎の呼び出し条件は以下の通りです。

・RM_FS3000_Open：　(1) FS3000 開始時、(7) RM_FS3000_Close 後の再開始

・RM_FS3000_Close：　(6) 各処理の正常終了または異常終了時

・RM_FS3000_Read：　(2) 開始後の測定データ取得時、

(5) データ取得応答待ちによる再試行

・RM_FS3000_DataCalculate：　(3) RM_FS3000_Read 後のデータ演算時

注意：

OS 使用時、複数のスレッド/タスクで同時にセンサを制御する場合はユーザーによるセマフォを用いた bus 制御が必要となります。セマフォの生成タイミング、ブロッキングの制御は 0

OS 版サンプルソフトウェアフローを参考にしてください。

4.3　Non-OS 版サンプルソフトウェアメイン処理フロー

サンプルソフトウェアはドライバの開始処理を行い、その後はセンサデータ取得、測定結果演算の処理を繰り返します。

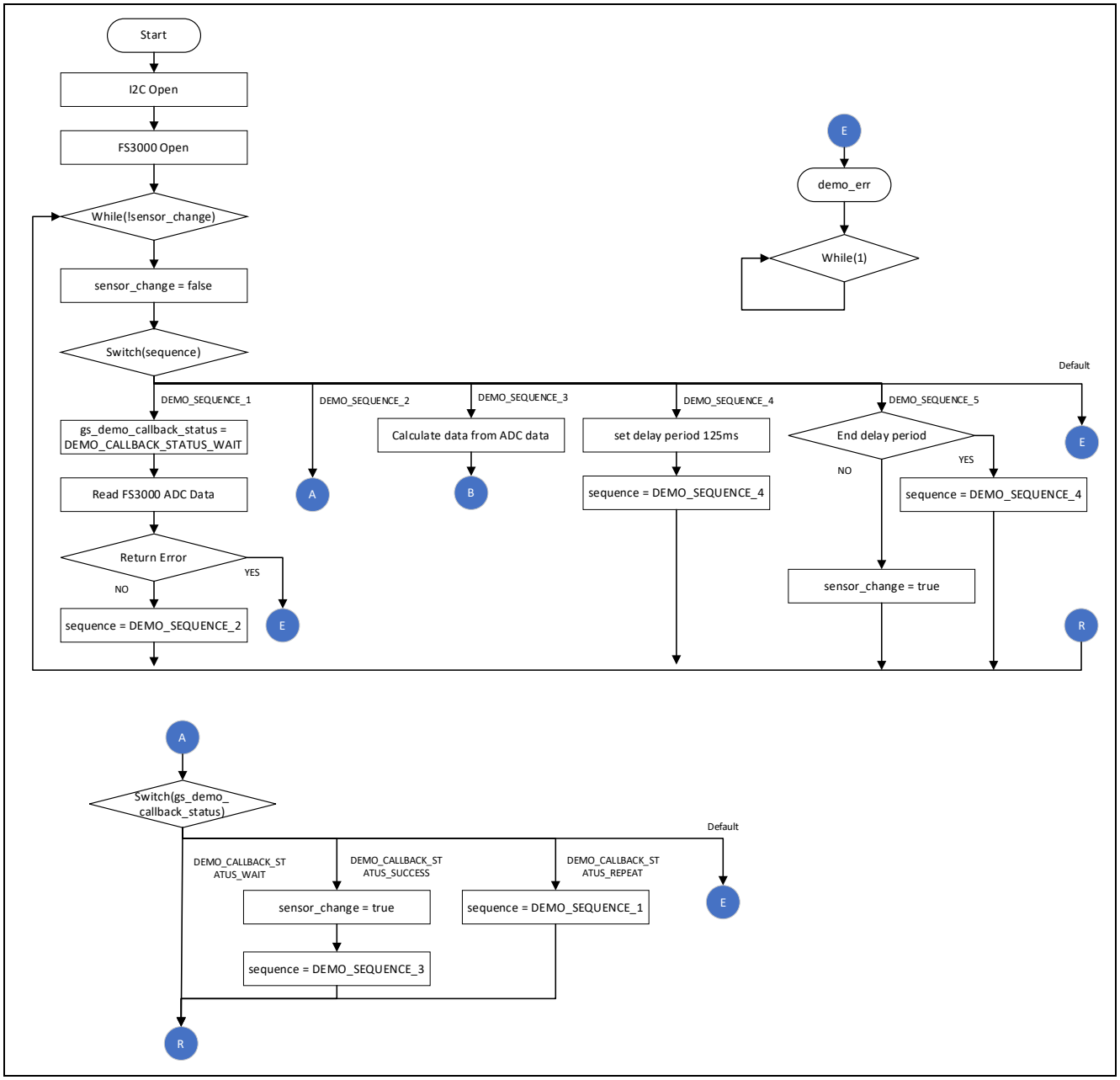


図 4-3　サンプルソフトウェア Non-OS 版メイン処理フロー 1

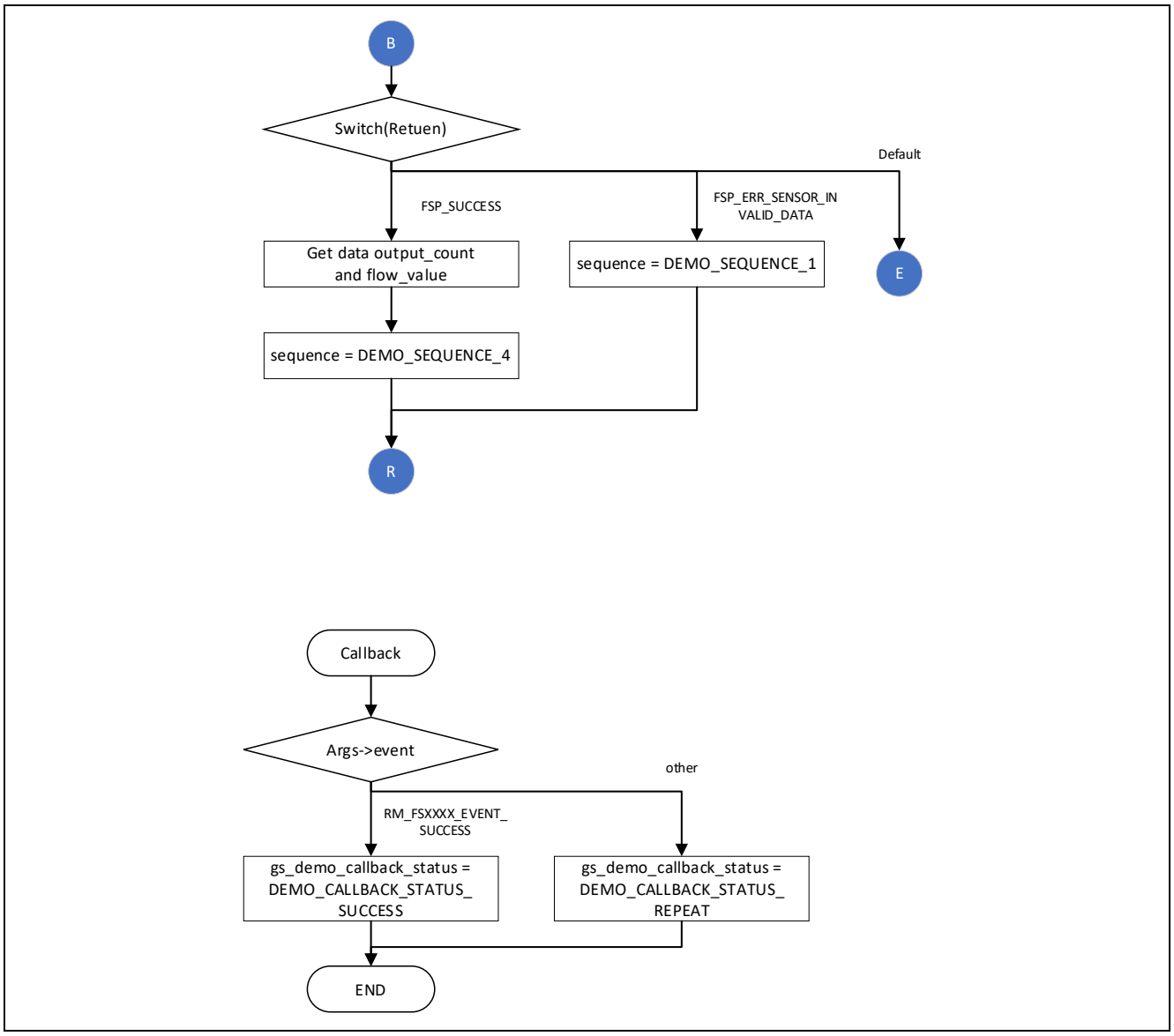


図 4-4　サンプルソフトウェア Non-OS 版メイン処理フロー2

4.4　OS 版サンプルソフトウェアフロー

OS 版ではセマフォによる制御を行い、センサ制御を行う 2 つのスレッドを並列で動作させます。

それぞれセンサの制御はドライバの開始処理を行い、その後はセンサデータ取得、測定結果演算の処理を繰り返します。

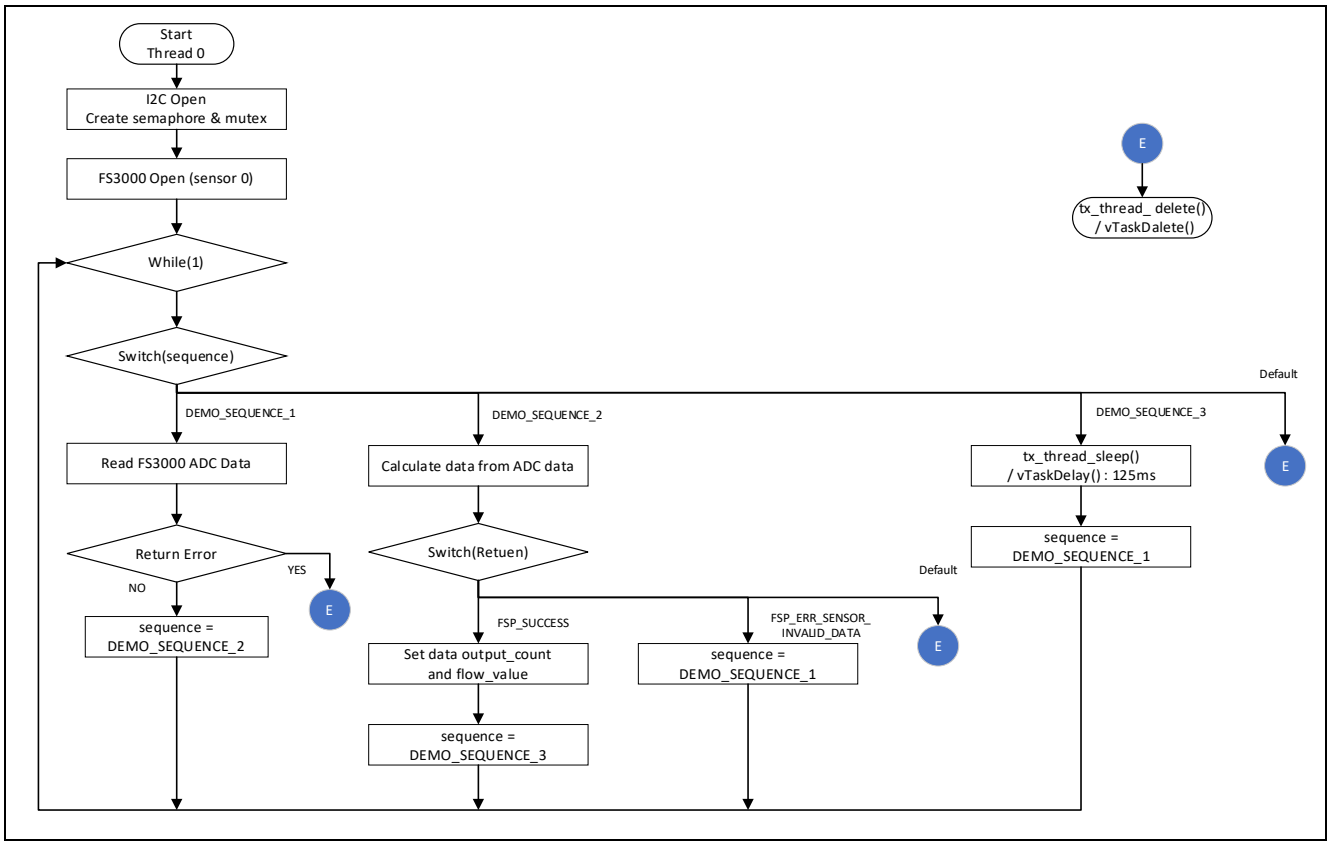


図 4-5　サンプルソフトウェア OS 版メイン処理フロー 1

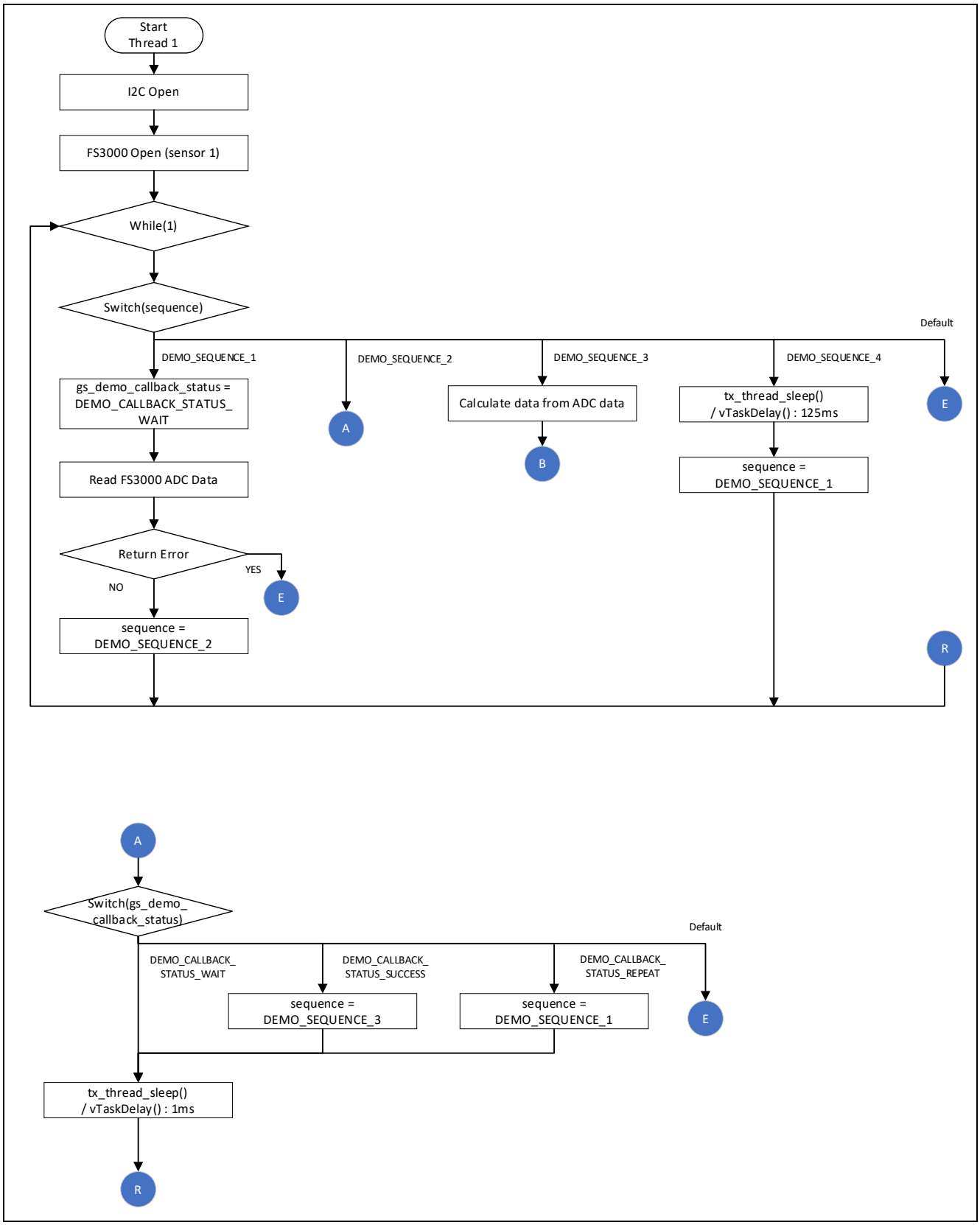


図 4-6　サンプルソフトウェア OS 版メイン処理フロー2

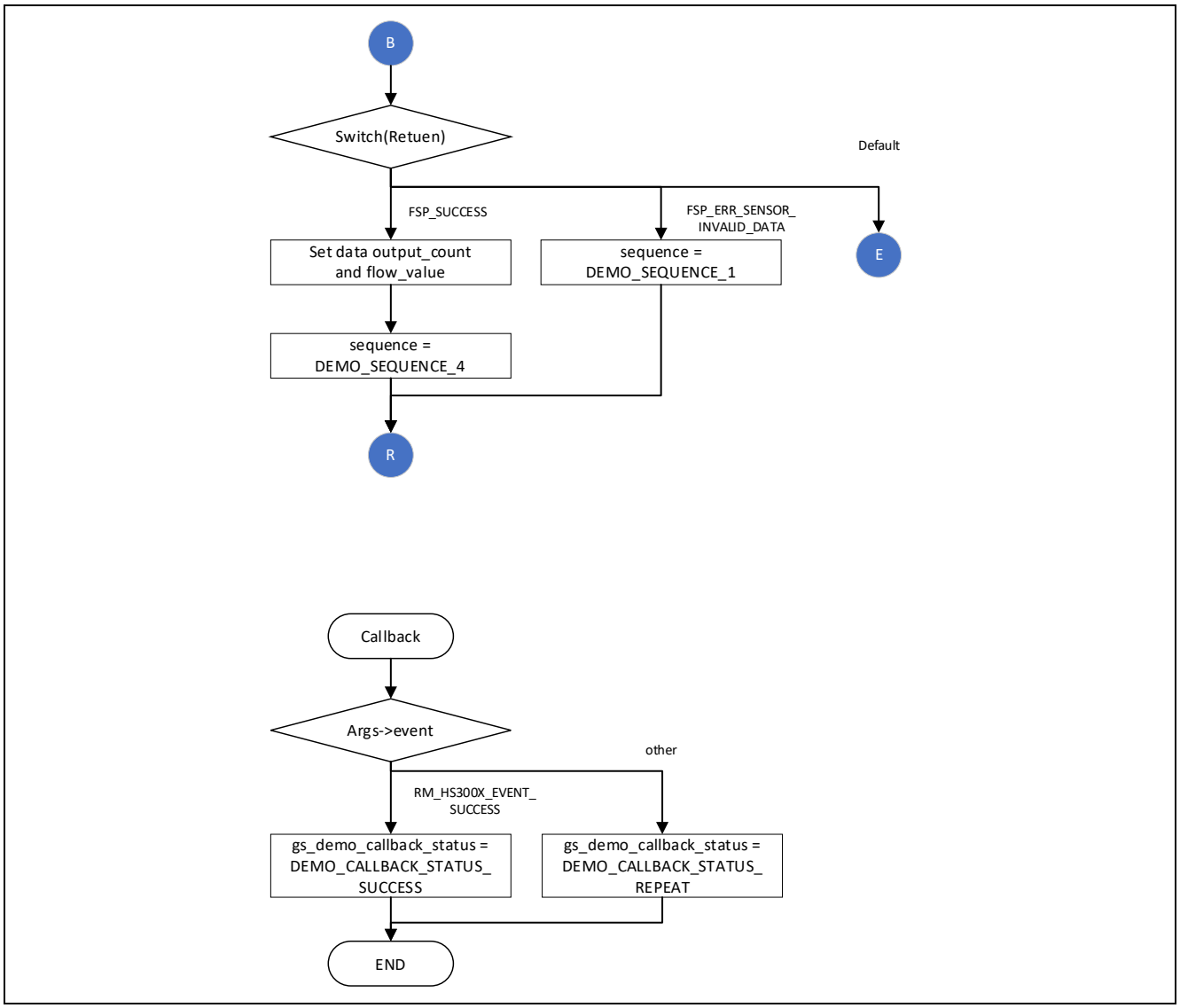


図 4-7　サンプルソフトウェア OS 版メイン処理フロー3

4.4.1　Azure RTOS プロジェクト

RX プロジェクトの Azure RTOS 版では、以下のような変更を行っています。

1. src/demo_thread.c

   57 行目：extern void tx_application_define_user (void);を追加

   178 行目：tx_application_define_user();を追加

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

   224 行目：TX_TIMER_TICKS_PER_SECOND ((ULONG) 1000)に変更

5. コンフィグ設定

5.1 FS3000 風速センサ設定

5.1.1 RA ファミリ

FSP Configurator の Stack タブで rm_fs3000 の Stack を選択することにより、Properties タブに設定可能な項目が表示されます。

設定可能な項目と設定値は以下の通りです。

表 5-1 RA FS3000 設定一覧

設定項目	設定値	説明
Common		
Parameter Checking	Default (BSP) Enabled Disabled	パラメータチェック処理をコードに含めるか選択できます。 “Disabled”の場合、パラメータチェック処理をコードから省略します。 “Enabled”の場合、パラメータチェック処理をコードに含めます。
Device type	FS3000-1005	センサの種別を設定します。 “FS3000-1005”のみ設定できます。
Module g_fs3000_sensor FS3000 on rm_fs3000		
Name	g_fs3000_sensor0	モジュール名を設定します。 設定可能なモジュール名は、C 言語規格に準拠します。
Callback	fs3000_callback	ユーザコールバック関数名を設定します。 設定可能なコールバック関数名は、C 言語規格に準拠します。 NULL を設定した場合は、コールバック関数は使用されません。

5.1.2　RX ファミリ

Smart Configurator の Component タブで r_fs3000_rx コンポーネントを選択することにより、Configure 領域に設定可能な項目が表示されます。

設定可能な項目と設定値は以下の通りです。

表 5-2　RX FS3000 設定一覧

設定項目	設定値	説明
Configurations		
Parameter Checking	System Default Enabled Disabled	パラメータチェック処理をコードに含めるか選択できます。 “Disabled”の場合、パラメータチェック処理をコードから省略します。 “Enabled”の場合、パラメータチェック処理をコードに含めます。
Number of FS3000 sensors	1 2	接続する FS3000 センサの数を設定します。
Device type of FS3000 Sensors	FS3000-1005	センサの種別を設定します。 “FS3000-1005”のみ設定できます。
I2C Communication device No. for FS3000 sensor device x (x = 0 or 1)	I2C Communication Device(y) (y = 0 - 15)	センサが使用する I2C Communication device を設定します。
Callback function for FS3000 sensor device x (x = 0 or 1)	fs3000_user_callback(x) (x = 0 or 1)	ユーザコールバック関数名を設定します。 設定可能なコールバック関数名は、C 言語規格に準拠します。

5.1.3　RL78 ファミリ

サンプルプロジェクトのプロジェクトツリー上の¥r_config¥r_fs3000_rl_config.h に定義されている定数の値を変更することにより、設定を変更することができます。

設定可能な項目と設定値は以下の通りです。

表 5-3　RL78 FS3000 設定一覧

定数名	設定値	説明
Configurations		
RM_FS3000_CFG_PARAM_CHECKING_ENABLE	0 1	パラメータチェック処理をコードに含めるか選択できます。 “0”の場合、パラメータチェック処理をコードから省略します。 “1”の場合、パラメータチェック処理をコードに含めます。
RM_FS300X_CFG_DEVICE_NUM_MAX	1 2	接続する FS3000 センサの数を設定します。
RM_FS3000_CFG_DEVICE_TYPE	0	センサの種別を設定します。 “0”(FS3000-1005)のみ設定できます。
RM_FS3000_CFG_DEVICEx_COMMS_INSTANCE (x = 0 or 1)	g_comms_i2c_device0	使用する communication line のインスタンス名を設定します。
RM_FS3000_CFG_DEVICEx_CALLBACK (x = 0 or 1)	NULL	ユーザコールバック関数名を設定します。 設定可能なコールバック関数名は、C 言語規格に準拠します。 “NULL”を設定した場合は、コールバック関数は使用されません。

5.2　通信ドライバミドルウェア設定

5.2.1　RA ファミリ

FSP Configurator の Stack タブで rm_comms_i2c の Stack を選択することにより、Properties タブに設定可能な項目が表示されます。

設定可能な項目と設定値は以下の通りです。

表 5-4　RA 通信ドライバ設定一覧

設定項目	設定値	説明
Common		
Parameter Checking	Default (BSP) Enabled Disabled	パラメータチェック処理をコードに含めるか選択できます。 “Disabled”の場合、パラメータチェック処理をコードから省略します。 “Enabled”の場合、パラメータチェック処理をコードに含めます。
Module g_comms_i2c_device I2C Communication Device on rm_comms_i2c		
Name	g_comms_i2c_device0	モジュール名を設定します。 設定可能なモジュール名は、C 言語規格に準拠します。
Semaphore Timeout	0xFFFFFFFF	RTOS プロジェクト時、semaphore のタイムアウト時間を設定します。
Slave Address	0x28	スレーブアドレスを設定します。 rm_fs3000 を使用する場合は、自動的に設定され変更できません。
Address Mode	7-Bit	スレーブアドレスのビット幅を設定します。 rm_fs3000 を使用する場合は、自動的に設定され変更できません。
Callback	rm_fs3000_callback	ユーザコールバック関数名を設定します。 rm_fs3000 を使用する場合は、自動的に設定され変更できません。
Module g_comms_i2c_bus0 I2C Shared Bus on rm_comms_i2c		
Name	g_comms_i2c_bus0	I2C モジュール名を設定します。
Bus Timeout	0xFFFFFFFF	I2C バスのタイムアウト時間を設定します
Semaphore for blocking	Unuse Use	RTOS プロジェクト時、Blocking 処理の有効/無効を設定します。
Recursive Mutex for Bus	Unuse Use	RTOS プロジェクトかつ、Blocking が有効の時、再帰動作の有効/無効を設定します。

5.2.2　RX ファミリ

Smart Configurator の Component タブで r_comms_i2c_rx コンポーネントを選択することにより、Configure 領域に設定可能な項目が表示されます。

設定可能な項目と設定値は以下の通りです。

表 5-5　RX 通信ドライバ設定一覧

設定項目	設定値	説明
Configurations		
Parameter Checking	System Default Enabled Disabled	パラメータチェック処理をコードに含めるか選択できます。 “Disabled”の場合、パラメータチェック処理をコードから省略します。 “Enabled”の場合、パラメータチェック処理をコードに含めます。
Number of I2C Shared Buses	Unused 1 2 - 16	接続可能とする I2C バス数を設定します。
Number of I2C Devices	Unused 1 2 - 16	接続可能とする I2C デバイスを設定します。
Blocking operation supporting with RTOS	Disabled Enabled	RTOS プロジェクト時のブロッキング動作を設定します。
Bus lock operation supporting with RTOS	Disabled Enabled	RTOS プロジェクト時のバスロック動作を設定します。
IIC Driver Type for I2C Shared bus(x) (x = 0 - 15)	RIIC SCI IIC Not selected	通信バスが使用する I2C バスの種別を設定します。 RIIC を使用する場合は、r_riic_rx, SCI IIC を使用する場合は、r_sci_iic_rx が必要となります。 使用しない FIT モジュールを削除すると、警告が表示されますが、動作に問題はありません。
Channel No. for I2C Shared bus(x) (x = 0 - 15)	0	通信バス使用する I2C バスのチャネル番号を設定します。
Timeout for the bus lock of the I2C bus for I2C Shared Bus(x) (x = 0 - 15)	0xFFFFFFFF	I2C バス(x)の I2C バスロックタイムアウト時間を設定します。 (x = 0 - 4)
I2C Shared Bus No. for I2C Communication Device(x) (x = 0 - 15)	I2C Shared Bus(x) (x = 0 - 15)	通信バス使用する I2C バスのコンフィグレーションを設定します。
Slave address for communication device(x) (x = 0 - 15)	0x28	通信バスに接続されるデバイスのスレーブアドレスを設定します。 r_fs3000_rx を使用する場合は、0x28 に設定してください。
Slave address mode for communication device(x) (x = 0 - 15)	7 bit address mode	スレーブアドレスモードを設定します。 r_fs3000_rx を使用する場合は、7 bit address mode に設定してください。

Callback function for Communication device(x) (x = 0 - 15)	comms_i2c_user_callback(x) (x = 0 - 15)	ユーザコールバック関数名を設定します。 r_fs3000_rx を使用する場合は、 rm_fs3000_callback(y) (y = 0)を設定します。

5.2.3　RL78 ファミリ

サンプルプロジェクトのプロジェクトツリー上の r_config¥r_comms_i2c_rl_config.h に定義されている定数の値を変更することにより、設定を変更することができます。

設定可能な項目と設定値は以下の通りです。

表 5-6　RL78 通信ドライバ設定一覧

定数名	設定値	説明
Configurations		
COMMS_I2C_CFG_PARAM_CHECKING_ENABLE	0	パラメータチェック処理をコードに含めるか選択できます。 “0”の場合、パラメータチェック処理をコードから省略します。 “1”の場合、パラメータチェック処理をコードに含めます。
	1	
COMMS_I2C_CFG_DEVICE_NUM_MAX	1	接続可能とする通信バス数を設定します。
	2	
	3	
	4	
	5	
COMMS_I2C_CFG_BUSx_DRIVER_TYPE (x = 0 – 4)	COMMS_DRIVER_I2C	通信バスが使用する I2C バスの種別を設定します。
	COMMS_DRIVER_SAU_I2C	
COMMS_I2C_CFG_BUSx_DRIVER_CH (x = 0 – 4)	g_comms_i2c_bus(x)_extended_cfg (x = 0 - 4)	通信バス使用する I2C バスのチャネル番号を設定します。
COMMS_I2C_CFG_BUSx_SLAVE_ADDR (x = 0 – 4)	0x28	通信バスに接続されるデバイスのスレーブアドレスを設定します。 rm_fs3000 を使用する場合は、0x28 に設定してください。
COMMS_I2C_CFG_DEVICE(x)_CALLBACK (x = 0 – 4)	rm_fs3000_callback0	ユーザコールバック関数名を設定します。 “NULL”を設定した場合は、コールバック関数は使用されません。

5.3　I2C ドライバ設定

5.3.1　RA ファミリ

FSP Configurator の Stack タブで r_iic_master もしくは、r_sci_i2c の Stack を選択することにより、Properties タブに設定可能な項目が表示されます。

設定可能な項目と設定値は以下の通りです。

表 5-7　RA r_iic_master 設定一覧

設定項目	設定値	説明
Common		
Parameter Checking	Default (BSP) Enabled Disabled	パラメータチェック処理をコードに含めるか選択できます。 “Disabled”の場合、パラメータチェック処理をコードから省略します。 “Enabled”の場合、パラメータチェック処理をコードに含めます。
DTC on Transmission and Reception	Enabled Disabled	送受信に DTC を使用するか設定する。
10-bit slave addressing	Enabled Disabled	スレーブアドレスに 10-bit addressing をサポートするか設定する。 rm_fs3000 を使用する場合は、Disabled に設定してください。
Module g_i2c_master0 I2C Master Driver on r_iic_master		
Name	g_i2c_master0	モジュール名を設定します。
Channel	0	使用するチャネル番号を設定します。
Rate	Standard Fast-mode Fast-mode plus	ボーレートを設定します。 Standard もしくは、Fast-mode に設定してください。
Rise Time (ns)	120	SCL の立ち上がり時間を設定します。使用するボードに合わせて設定してください。
Fall Time (ns)	120	SCL の立ち下がり時間を設定します。使用するボードに合わせて設定してください。
Duty Cycle (%)	50	SCL のデューティ比を設定します。
Slave Address	0x00	接続するデバイスのスレーブアドレスを設定します。 rm_comms_i2c により上書きされますので、設定は必要ありません。
Address Mode	7-Bit 10-Bit	接続するデバイスのスレーブアドレスモードを設定します。 rm_comms_i2c により上書きされますので、設定は必要ありません。
Timeout Mode	Short Mode Long Mode	I2C バスのタイムアウト時間を設定します
Callback	rm_comms_i2c_callback	ユーザコールバック関数名を設定します。 rm_comms_i2c により自動的に設定されます。

Interrupt Priority Level	Priority 0 (highest)	I2C バスドライバの割り込み優先レベルを設定します。
	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	ドライバが使用する端子番号が表示されます。端子の設定は、Pins タブで行います。
SCL	Pxxx	

表 5-8　RA r_sci_i2c 設定一覧

設定項目	設定値	説明
Common		
Parameter Checking	Default (BSP) Enabled Disabled	パラメータチェック処理をコードに含めるか選択できます。 “Disabled”の場合、パラメータチェック処理をコードから省略します。 “Enabled”の場合、パラメータチェック処理をコードに含めます。
DTC on Transmission and Reception	Enabled Disabled	送受信に DTC を使用するか設定する。
10-bit slave addressing	Enabled Disabled	スレーブアドレスに 10-bit addressing をサポートするか設定する。 Disabled に設定してください。
Module g_i2c0 I2C Master Driver on r_sci_i2c		
Name	g_i2c0	モジュール名を設定します。
Channel	0	RTOS プロジェクト時、semaphore のタイムアウト時間を設定します。
Slave Address	0x00	接続するデバイスのスレーブアドレスを設定します。 rm_comms_i2c により上書きされますので、設定は必要ありません。
Address Mode	7-Bit 10-bit	接続するデバイスのスレーブアドレスモードを設定します。 rm_comms_i2c により上書きされますので、設定は必要ありません。
Rate	Standard Fast-mode Fast-mode plus	ボーレートを設定します。 Standard もしくは、Fast-mode に設定してください。
SDA Output Delay (nano seconds)	300	SDA 出力遅延時間を設定します。
Noise filter setting	Use clock signal divided by 1 with noise filter 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	入力信号のノイズフィルタ使用を設定します
Bit Rate Modulation	Enable Disable	Bit Rate Modulation 機能の使用を設定します
Callback	rm_comms_i2c_callback	ユーザコールバック関数名を設定します。 rm_comms_i2c により自動的に設定されます。

Interrupt Priority Level	Priority 0 (highest)	I2C 割り込みの割り込み優先レベルを設定します。
	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)	DTC を使用した場合の受信割り込みの割り込み優先レベルを設定します。
	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	ドライバが使用する端子番号が表示されます。端子の設定は、Pins タブで行います。
SCL	Pxxx	

5.3.2　RX ファミリ

Smart Configurator の Component タブで r_riic_rx もしくは、r_sci_iic_rx コンポーネントを選択することにより、Configure 領域に設定可能な項目が表示されます。

設定可能な項目と設定値は以下の通りです。

表 5-9　RX r_riic_rx 設定一覧

設定項目	設定値	説明
Configurations		
Set parameter checking enable	System Default Not Include	パラメータチェック処理をコードに含めるか選択できます。 “Not”の場合、パラメータチェック処理をコードから省略します。 “Include”の場合、パラメータチェック処理をコードに含めます。
MCU supported channels for CHx (x = 0 – 2)	Not supported Supported	該当チャネルを使用するかを選択できます。 該当チャネルを使用しない場合は Not suppported に設定してください。 Not supported の場合、該当チャネルに関する処理をコードから省略します。 Supported の場合、該当チャネルに関する処理をコードに含めます。
CHx RIIC bps(kbps) (x = 0 – 2)	400	通信速度を設定できます。 rm_hs300x を使用する場合は、400 以下に設定してください
Digital filter for CHx (x = 0 – 2)	Not One IIC phi Two IIC phi Three IIC phi Four IIC phi	指定したチャネルのノイズフィルタの段数を選択できます。 “Not”の場合、ノイズフィルタは無効となります。
Setting port setting processing	Not include port setting Include port setting	ポートを SCL, SDA 端子として使用するための設定処理をコードに含めるかを選択します。 Not include port setting の場合、ポートの設定処理をコードから省略します。 Include port setting の場合、ポートの設定処理をコードに含めます。
Master arbitration lost detection function for CHx (x = 0 – 2)	Unused Used	指定したチャネル のマスタアービトレーションロスト検出機能の有効/無効を選択できます。 マルチマスタで使用する場合は、 “Used”(有効)にしてください。 “Unused”の場合、マスタアービトレーションロスト検出を無効にします。 “Used”の場合、マスタアービトレーションロスト検出を有効にします。
Address y format for CHx (x = 0 – 2, y = 0 – 2)	Not 7 bit address format 10 bit address format	指定した RIIC のスレーブアドレスのフォーマットを 7 ビット/10 ビットから選択できます。 rm_hs300x を使用する場合は、”7 bit address format”に設定してください
Slave Address y for CHx (x = 0 – 2, y = 0 – 2)	0x0025	指定したチャネルのスレーブアドレスを設定します。 rm_comms_i2c により上書きされますので、設定は必要ありません。

General call address for CHx	Unused Used	指定したチャネル のゼネラルコールアドレスの有効/無効が選択できます。 “Unused”の場合、ゼネラルコールアドレスを無効にします。 “Used”の場合、ゼネラルコールアドレスを有効にします。
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)	指定したチャネルの受信データフル割り込み(RXI)の優先レベルを選択できます。 “Level 1”～” Level 15”の範囲で設定してください。
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)	指定したチャネルの送信データエンプティ割り込み(TXI)の優先レベルを選択できます。 “Level 1”～”Level 15”の範囲で設定してください。
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)	指定したチャネルの通信エラー/イベント発生割り込み(EEI)の優先レベルを選択できます。 “Level 1” ～” Level 15” の範囲で設定してください。
CHx TEI INT Priority Level (x = 0 – 2)	Level 1 Level 2 Level 3 Level 4 Level 5	指定したチャネルの送信終了割り込み(TEI)の優先レベルを選択できます。 “Level 1” ～” Level 15” の範囲で設定してください。

	Level 6 Level 7 Level 8 Level 9 Level 10 Level 11 Level 12 Level 13 Level 14 Level 15 (highest)	
Timeout function for CHx (x = 0 – 2)	Unused Used	指定したチャネルのタイムアウト検出機能を有効にできます。 “Unused”の場合、タイムアウト検出機能無効 “Used”の場合、タイムアウト検出機能有効
Timeout detection time for CHx (x = 0 – 2)	Long mode Short mode	指定したチャネルのタイムアウト検出時間を選択できます。 “Long mode”の場合、ロングモードを選択。 “Short mode”の場合、ショートモードを選択。
Count up during low period of timeout detection for CHx (x = 0 – 2)	Unused Used	指定したチャネルのタイムアウト検出機能有効時、SCL ラインが Low 期間中にタイムアウト検出機能の内部カウンタのカウントアップを有効にできます。 “Unused”の場合、SCL ラインが Low 期間中のカウントアップ禁止。 “Used”の場合、SCL ラインが Low 期間中のカウントアップ有効。
Count up during high period of timeout detection for CHx (x = 0 – 2)	Unused Used	指定したチャネルのタイムアウト検出機能有効時、SCL ラインが High 期間中にタイムアウト検出機能の内部カウンタのカウントアップを有効にできます。 “Unused”の場合、SCL ラインが High 期間中のカウントアップ禁止。 “Used”の場合、SCL ラインが High 期間中のカウントアップ有効。
Set Counter of checking bus busy	1000	API 関数のバスチェック処理時に、ソフトウェアによりタイムアウトカウンタ(バス確認回数)を設定できます。
Resources		
SDAx Pins	Unchecked	使用する端子を設定します。 使用する端子のチェックボックスにチェックしてください。
SCLx Pins	Unchecked	

表 5-10　RX r_sci_iic_rx 設定一覧

設定項目	設定値	説明
Configurations		
Set parameter checking enable	System Default Not Include	パラメータチェック処理をコードに含めるか選択できます。 “Not”の場合、パラメータチェック処理をコードから省略します。 “Include”の場合、パラメータチェック処理をコードに含めます。
MCU supported channels for CHx (x = 0 – 12)	Not supported Supported	該当チャネルを使用するかを選択できます。

		“Not supported”の場合、該当チャネルに関する処理をコードから省略します。 “Supported”の場合、該当チャネルに関する処理をコードに含めます。
SCI IIC bitrate (bps) for CHx (x = 0 – 12)	384000	ビットレートを設定してください。 384000(384kbit/s)以下を設定してください。
Interrupt Priority for CHx (x = 0 – 12)	Level 1 Level 2 Level 3 Level 4 Level 5 Level 6 Level 7 Level 8 Level 9 Level 10 Level 11 Level 12 Level 13 Level 14 Level 15 (highest)	コンディション割り込み、受信割り込み、送信空割り込み、送信完了割り込みの優先レベルを設定してください。 “Level 1”～”Level 15”の範囲で設定してください。
Digital noise filter (NFEN bit) for CHx (x = 0 – 12)	Disable Enable	SSCL、SSDA 入力信号のノイズ除去機能を使用するか選択できます。 “Disable”の場合、ノイズ除去機能を無効にします。 “Enable”の場合、ノイズ除去機能を有効にします。
Noise Filter Setting Register (NFCS bit) for CHx (x = 0 – 12)	The clock divided by 1 The clock divided by 2 The clock divided by 4 The clock divided by 8	デジタルノイズフィルタのサンプリングクロックを選択します。 “The clock divided by 1”の場合、1 分周のクロックをノイズフィルタに使用します。 “The clock divided by 2”の場合、2 分周のクロックをノイズフィルタに使用します。 “The clock divided by 4”の場合、4 分周のクロックをノイズフィルタに使用します。 “The clock divided by 8”の場合、8 分周のクロックをノイズフィルタに使用します。
I2C Mode Registger 1 (IICDL bit) for CHx (x = 0 – 12)	18	SSCL 端子出力の立ち下がりに対する SSDA 端子出力の遅延を選択します。 “1”～“31”の範囲で設定してください。
Software bus busy ckeck counter	1000	バスビジー判定のカウント数を設定します。 簡易 I2C の API 関数のバスチェック処理時の、タイムアウトカウンタ(バス確認回数)を設定できます。
Seting port setting processing	Not include port setting Include port setting	ポートを SSCL、SSDA 端子として使用するための設定処理をコードに含めるか選択できます。 “Not include port setting”の場合、ポートの設定処理をコードから省略します。 “Include port setting”の場合、ポートの設定処理をコードに含めます。
Resources		
SSDAx Pins	Unchecked	

SSCLx Pins	Unchecked	使用する端子を設定します。 使用する端子のチェックボックスにチェックしてください。

5.3.3　RL78 ファミリ

Code Generator の周辺機能にあるシリアルを選択することで、Peripheral Functions タブに設定可能な項目が表示されます。

設定可能な項目と設定値は以下の通りです。

表 5-11　RL78 シリアル設定一覧

設定項目	設定値	説明
SAUx		
チャネル		
チャネル x	使用しない	使用するチャネルの通信機能を設定します。 r_fs3000 を使用する場合は、IICxx を選択してください。
	UARTxx	
	CSIxx	
	IICxx	
IICxx		
転送レート	1000000	ボーレートを設定します。 rm_fs3000 を使用する場合は、100000 に設定してください。
転送完了割り込み優先順位（INTIICxx)	高	転送完了割り込みの割り込み優先順位を設定します。
	レベル 1	
	レベル 2	
	低	
マスタ送信完了	Checked	マスタ送信完了によるコールバック機能を設定します。
マスタ受信完了	Checked	マスタ受信完了によるコールバック機能を設定します。
マスタエラー	Checked	通信エラーによるコールバック機能を設定します。
IICAx		
転送モード		
転送モード	使用しない	使用するチャネルの通信機能を設定します。 シングルマスタを選択してください。
	シングルマスタ	
	スレーブ	
設定		
カウントクロック	fCLK	カウント・クロックを設定します。
	fCLK/2	
アドレス	16	自局アドレスを設定します。
動作モード	標準	動作モードを設定します。
	ファスト・モード/ ファスト・モード・プラス	
転送クロック（fSCL)	100000	ボーレートを設定します。 400000 以下に設定してください。
通信完了割り込み優先順位(INTIICAx)	高	通信完了割り込みの割り込み優先順位を設定します。
	レベル 1	
	レベル 2	
	低	
マスタ送信完了	Checked	マスタ送信完了によるコールバック機能を設定します。
マスタ受信完了	Checked	マスタ受信完了によるコールバック機能を設定します。

マスタエラー	Checked	通信エラーによるコールバック機能を設定します。
マスタ送信完/受信完了コールバック時にストップ・コンディションを生成	Checked	コールバック時のストップ・コンディション生成を設定します。 チェックを外してください。

6.　デバイス変更ガイド

サンプルプロジェクトを異なるデバイスで動作させるには、以下の手順に従ってください。

移行元デバイスのサンプルプロジェクトは、事前に Workspace にインポートしてください。

6.1　RA サンプルプロジェクト

サンプルプロジェクトを変更する場合の手順は以下の通りです。

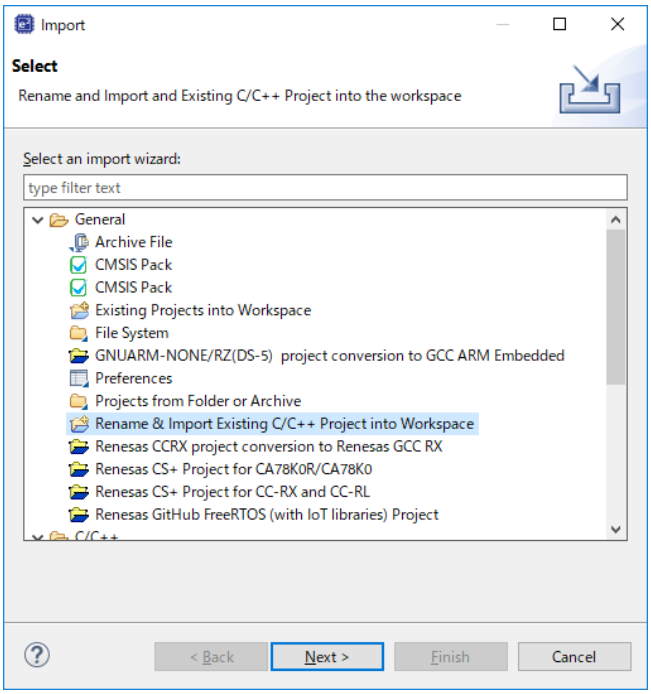
本章解説では、例としてサンプルプロジェクト”FS3000_RA6M4_NonOS”から、EK-RA2E1 ボードで使用できるプロジェクトへの変更手順を記載します。

なお、PMOD1 の記述については”OptionType6A”を適用したボードを使用する場合の手順となります。

6.1.1　サンプルプロジェクトのインポート

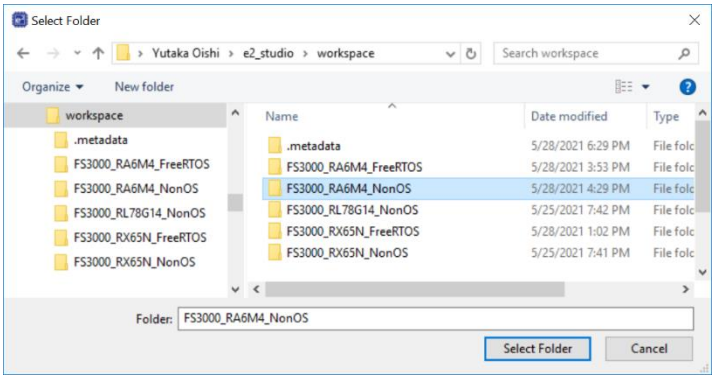
メニューから、インポートを選択します。

表示されたインポートウィンドウで、”Rename & Import Existing C/C++ Project into Workspace”を選択し、[Next]ボタンを押下します。

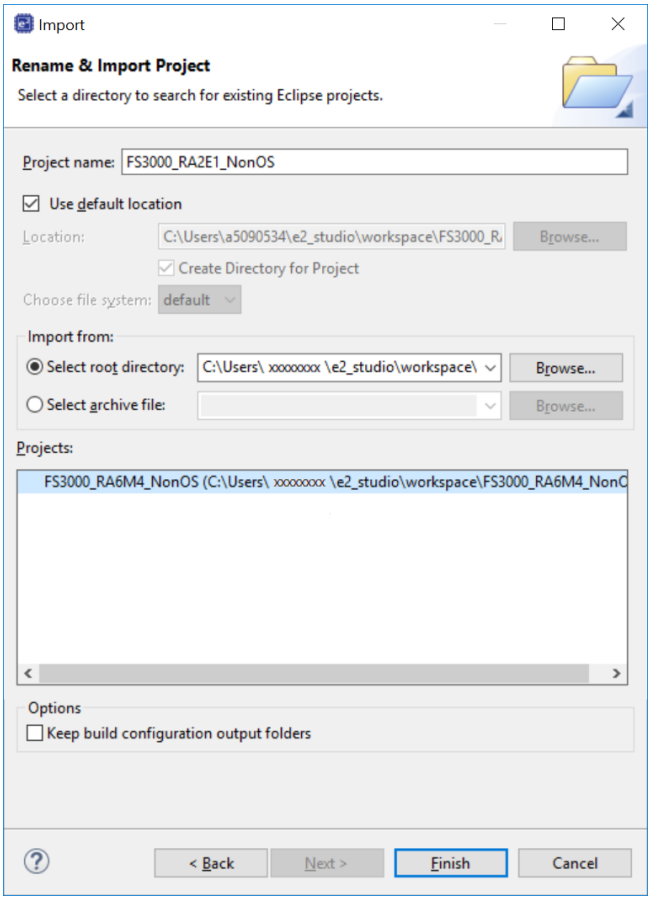


[Browse]ボタンを押下し、フォルダの選択ウィンドウを表示します。

インポート済みのサンプルプロジェクトから、移行元デバイスのプロジェクトのフォルダを選択し、[フォルダの選択]ボタンを押下します。



プロジェクト名の入力および、移行元デバイスのプロジェクトを選択し、[Finish]ボタンを押下します。



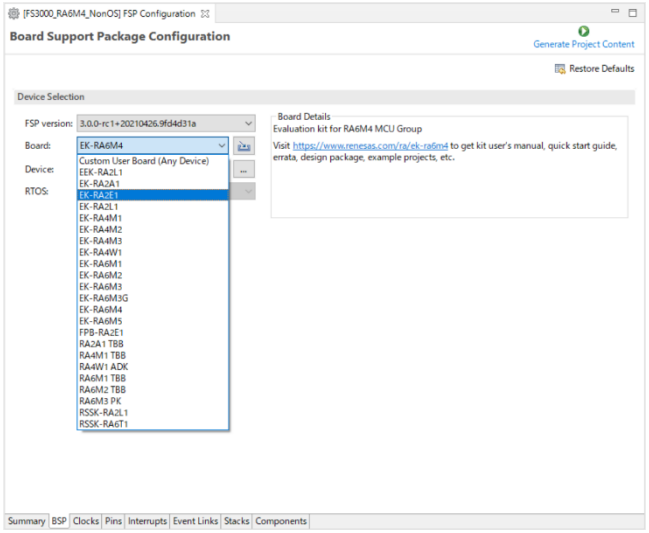
6.1.2　FSP Configurator の設定変更

プロジェクトツリーの Configuratorn.xml をダブルクリックし、FSP Configurator を開きます。

BSP タブで Board および Device を変更します。

ルネサス製ボードに変更する場合は、Board の設定のみ変更してください。

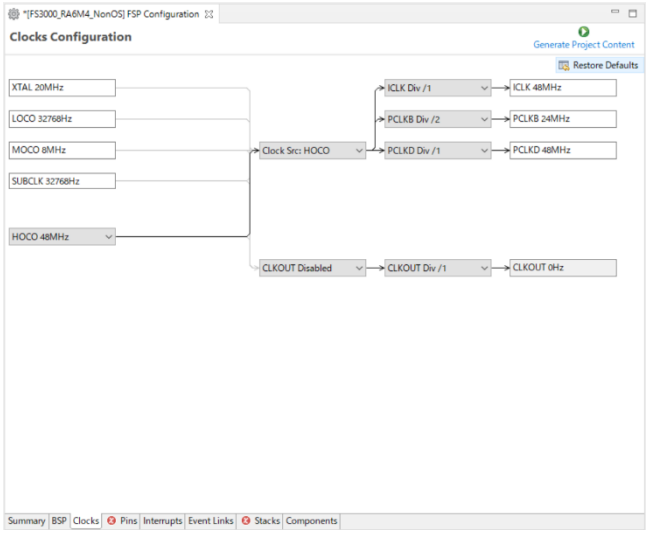
ルネサス製以外のボードに変更する場合は、Board を”Custom User Board (Any Device)”に変更後、Device を使用するデバイスに変更してください。



Clocks タブで、クロック設定を変更します。

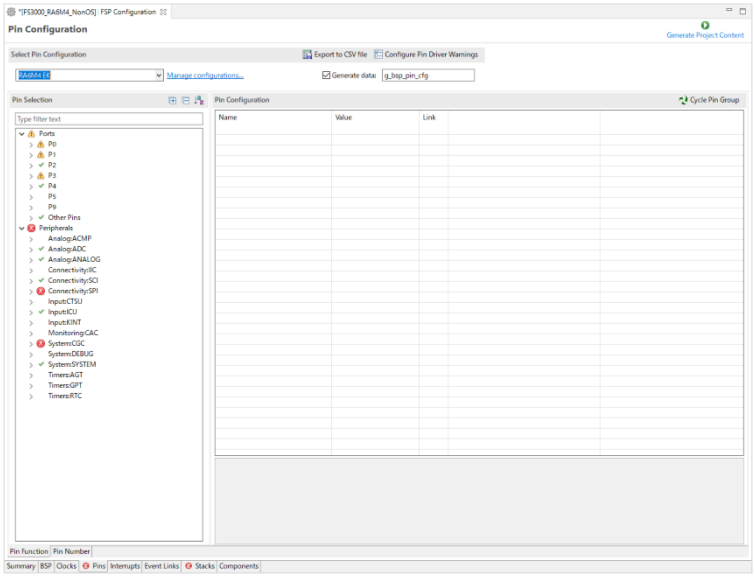
Board を”Custom User Board (Any Device)”に変更した場合は、使用するボードに合わせてクロック設定を変更してください。

Board をルネサス製ボードに変更した場合は、自動的に設定が変更されます。

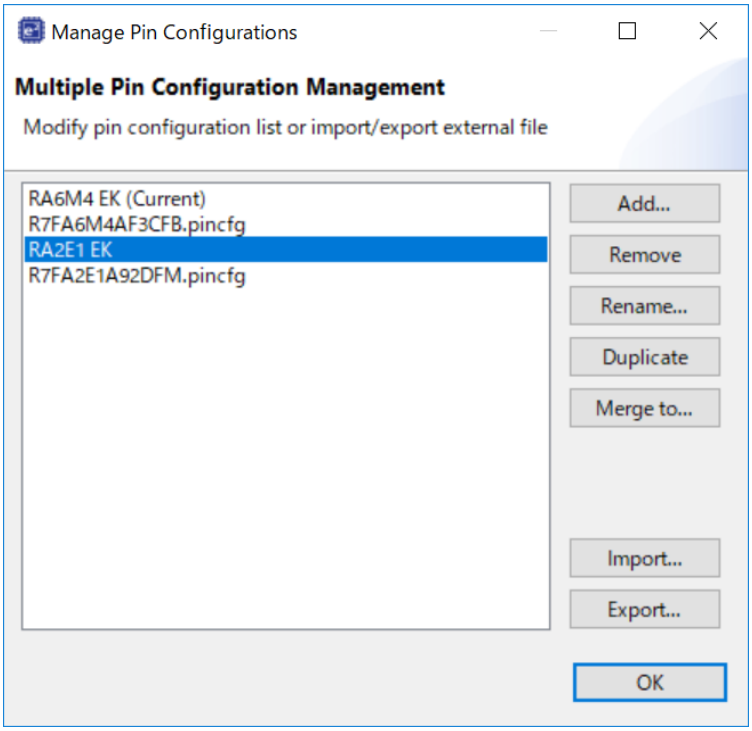


Pins タブで、使用するボードに合わせて、端子設定を変更します。

ルネサス製ボードを使用する場合は、Select Pin Configuration を”RA6M4 EK”から使用するボードに変更することで、自動的に割り当てが行われます。



Select Pin Configuration のドロップダウンリストに変更したいボードが表示されない場合は、[Manage Configuration]をクリックし、表示された Manage Pin Configuration ウィンドウで、使用したいボードを選択してください。

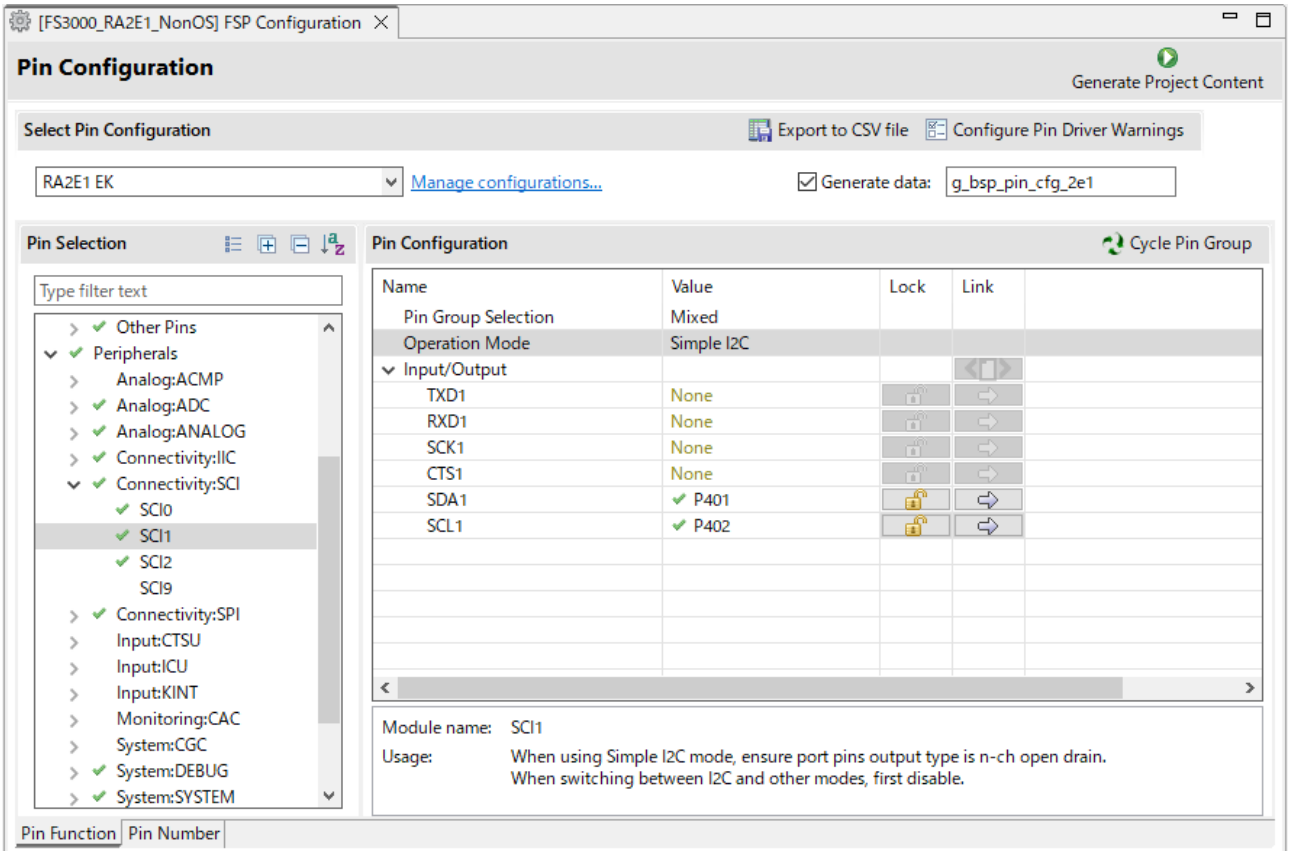


ただし、EK-RA2E1 ボードの場合、上記の割り当てでは PMOD Type 2A に対応した SPI 通信の端子設定が適用されます。

このサンプルソフトウェアでは、PMOD Type 6A を使用するため、PMOD Type 6A に対応した I2C 通信の端子設定への変更が必要です。

EK-RA2E1 ボードでは、PMOD1 に SCI2、PMOD2 に SCI1 が割り当てられています。

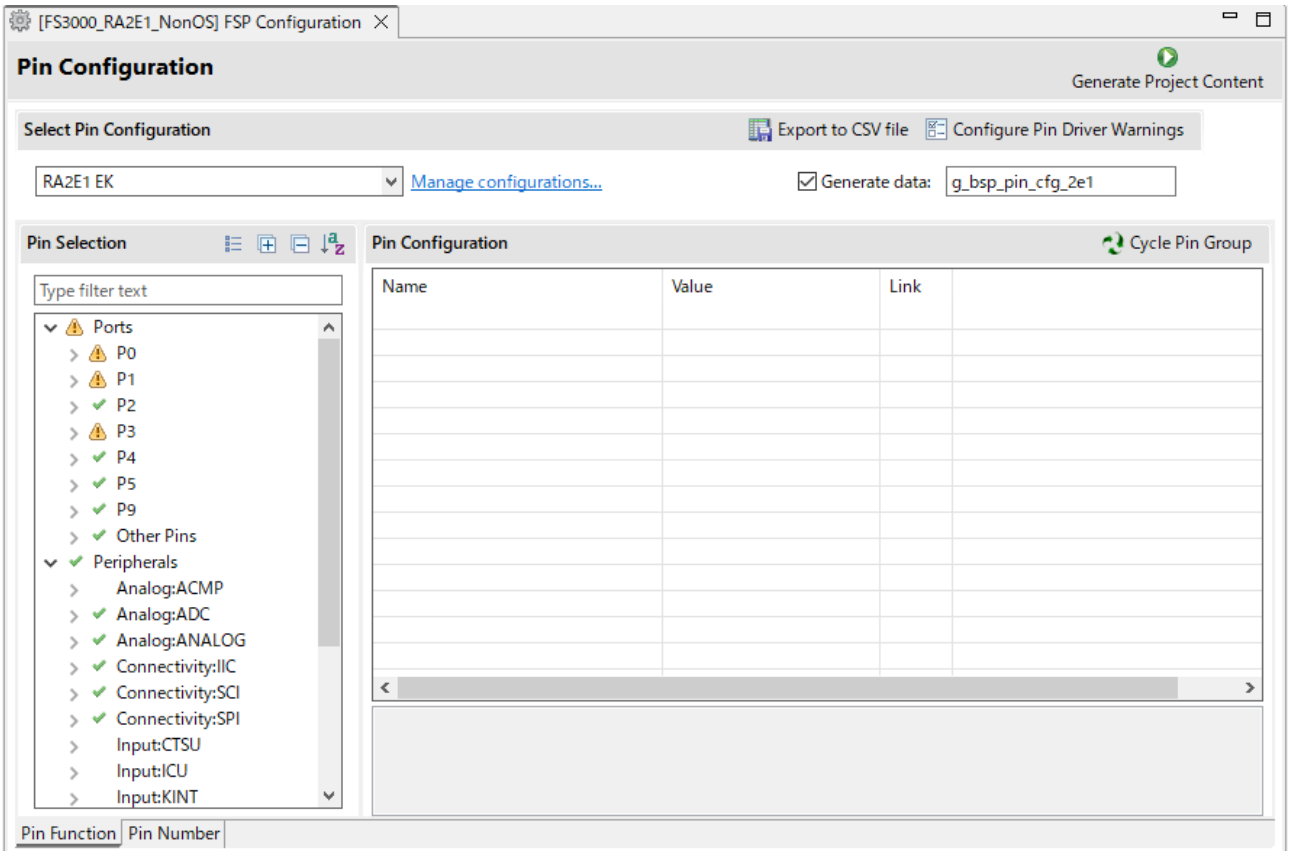
PMOD1(OptionType6A)を使用する場合は P301 と P302 が、PMOD2 を使用する場合は P401 と P402 が I2C 通信を行う端子となるため、Select Pin Configuration の自動割り当て後、Pin Configuration にて再設定を行います。



端子設定の生成を有効にするには、[Generate data]にチェックを入れ、テキストボックスに任意の名称を入力してください。

入力された名称は設定した端子構成に紐付けられるため、他の端子構成と重複しないよう、固有の名称としてください。

例の場合、”g_bsp_pin_cfg_2e1”です。



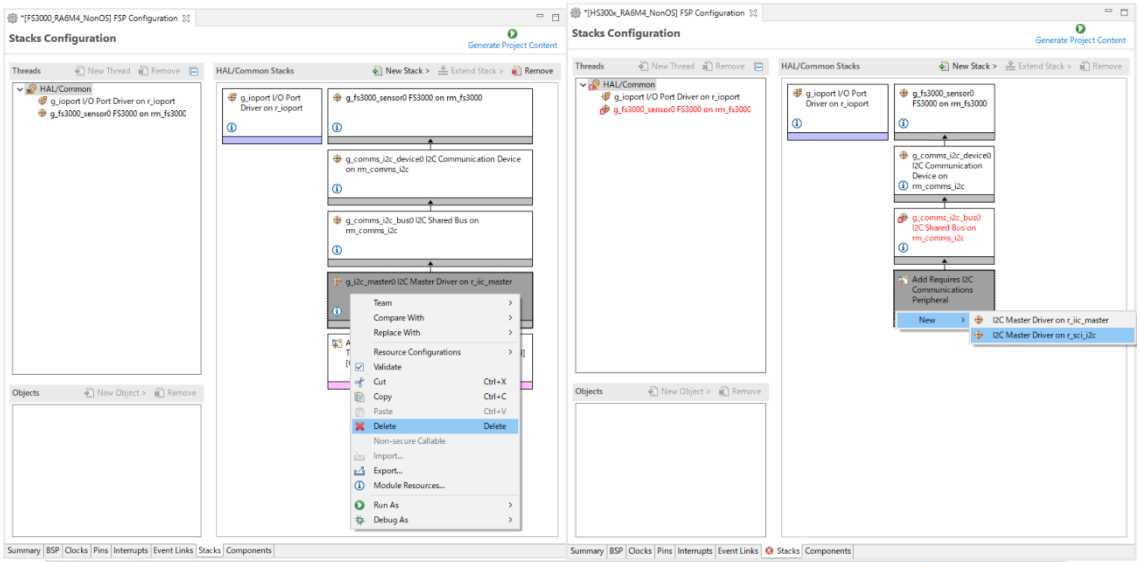
Stacks タブで、各コンポーネント設定を変更します。

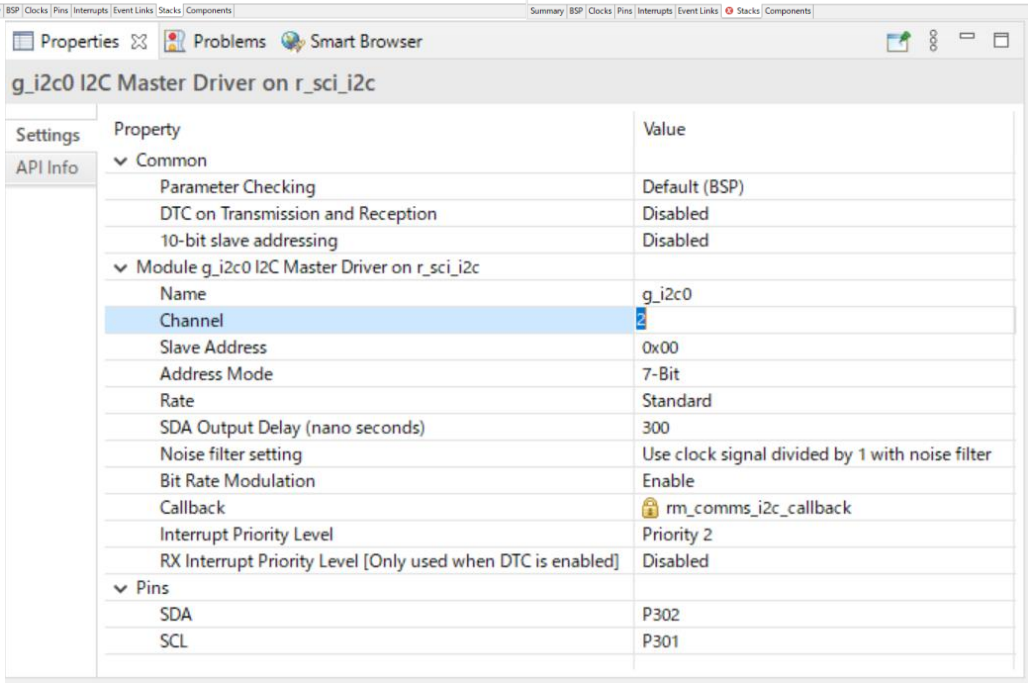
使用するボードに合わせて、r_iic_master もしくは r_sci_i2c の設定を変更してください。

IIC の端子を使用する場合は、I2C Master Driver on r_sci_i2c”の stack を削除してから、”I2C Master Driver on r_iic_mster”の stack を追加してください。

EK-RA2E1 ボードでは、PMOD1 に SCI2、PMOD2 に SCI1 が割り当てられています。

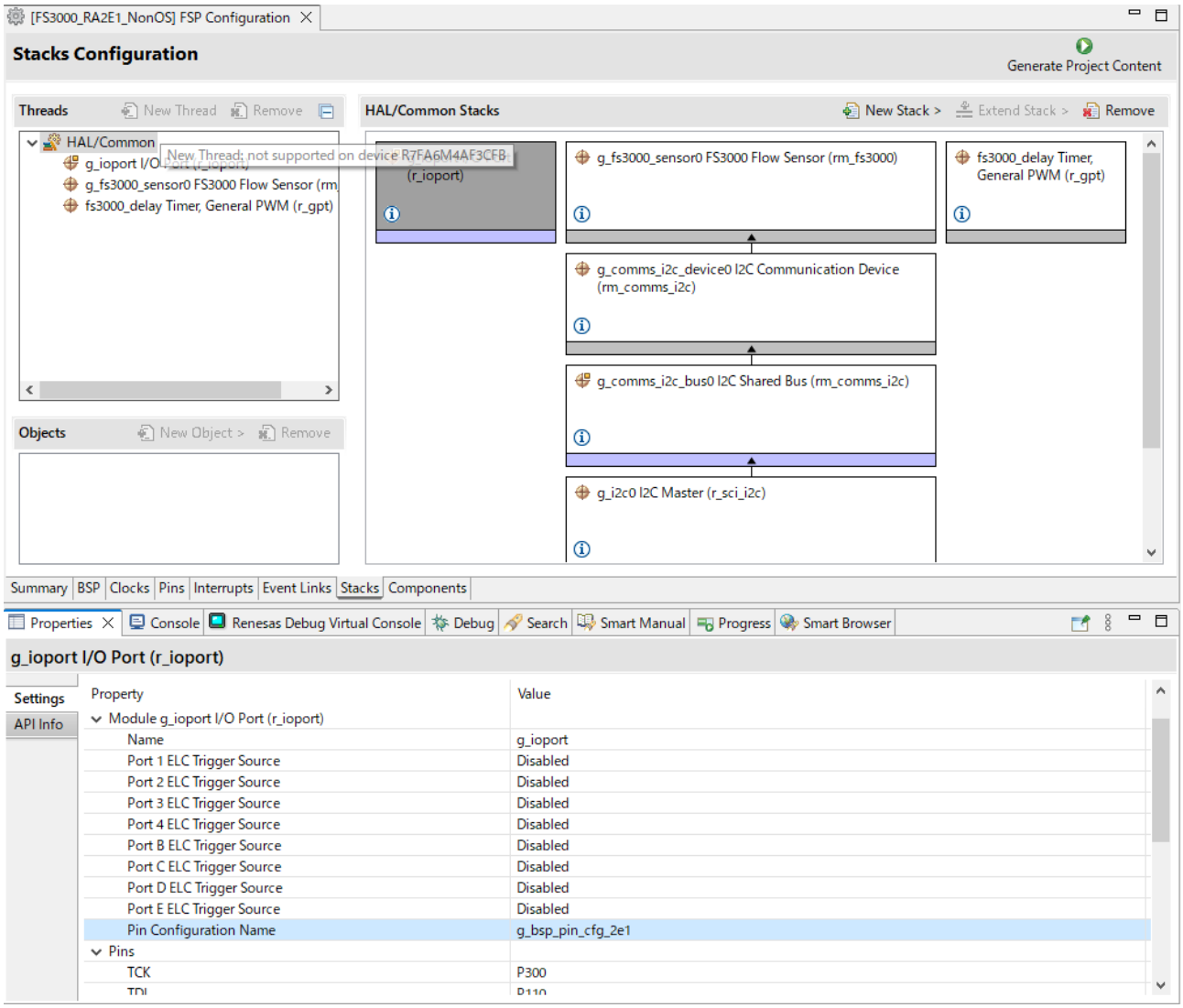
PMOD1 を使用する場合は channel を 2 に、PMOD2 を使用する場合は channel を 1 に設定します。





“g_ioport I/O Port”の Pin Configration Name に、使用する端子構成の名称を入力してください。

例の場合、”g_bsp_pin_cfg_2e1”です。



他 stack でエラーが表示されている場合は、表示されたエラーに従って指定の項目を変更してください。

[Generate Project Content]を押下して、ファイルを生成します。

プロジェクトをビルドします。

メニューから[Debug Configurations]を選択し、使用するボードに接続するエミュレータに合わせて、Debugger の設定を変更してください。

6.1.3　ツールチェイン設定変更

GCC ARM Embedded ツールチェイン以外のツールチェインを使用する場合は、本プロジェクトから RA_FS3000.c (Non-OS)または、fs3000_sensor_thread_entry.c 及び sensor_thread_common.c、sensor_thread_common.h (FreeRTOS、Azure)をコピーしてプロジェクトを作成してください。

6.2　RX サンプルプロジェクト

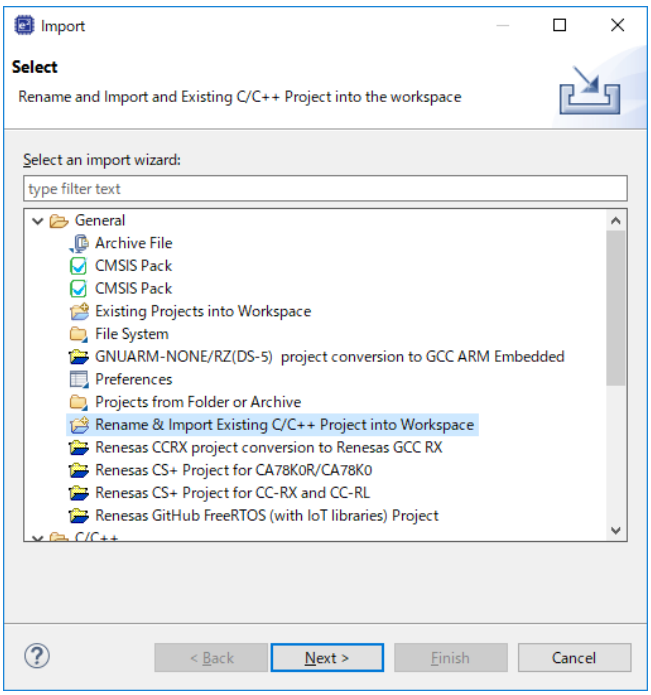
サンプルプロジェクトを変更する場合の手順は以下の通りです。

本章解説では、例としてサンプルプロジェクト”FS3000_RX65N_NonOS”から、RSKRX231 ボードで使用できるプロジェクトへの変更手順を記載します。

6.2.1　サンプルプロジェクトのインポート

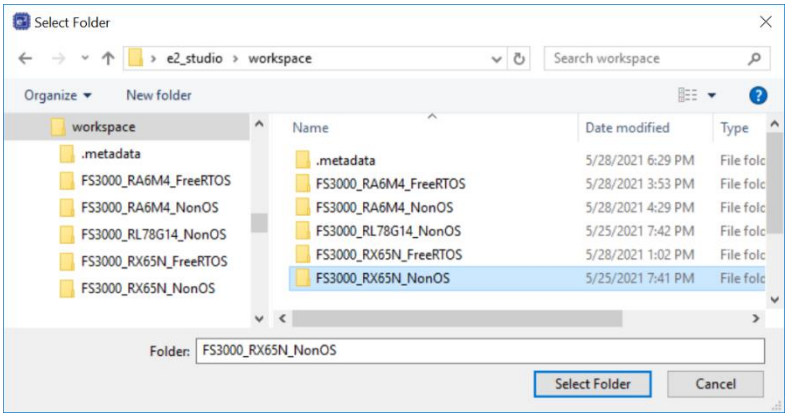
メニューから、インポートを選択します。

表示されたインポートウィンドウで、”Rename & Import Existing C/C++ Project into Workspace”を選択し、[Next]ボタンを押下します。

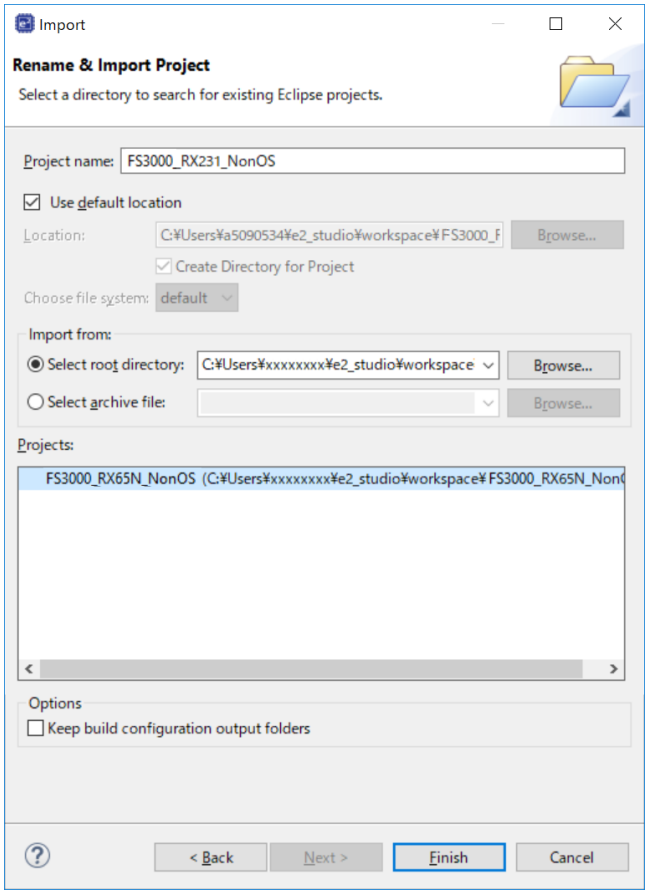


[Browse]ボタンを押下し、フォルダの選択ウィンドウを表示します。

インポート済みのサンプルプロジェクトから、移行元デバイスのプロジェクトのフォルダを選択し、[フォルダの選択]ボタンを押下します。

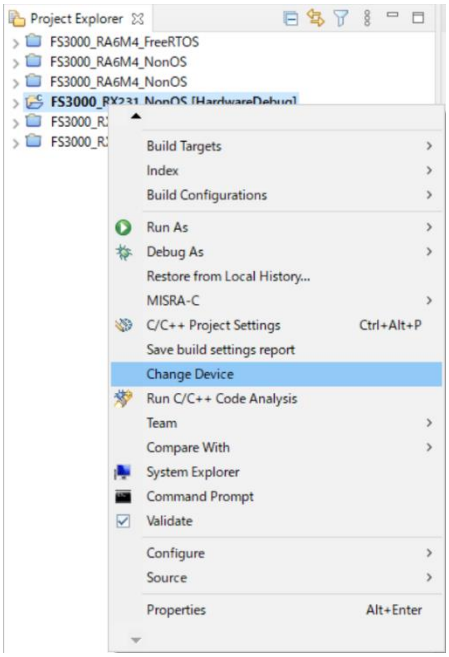


プロジェクト名の入力および、移行元デバイスのプロジェクトを選択し、[Finish]ボタンを押下します。

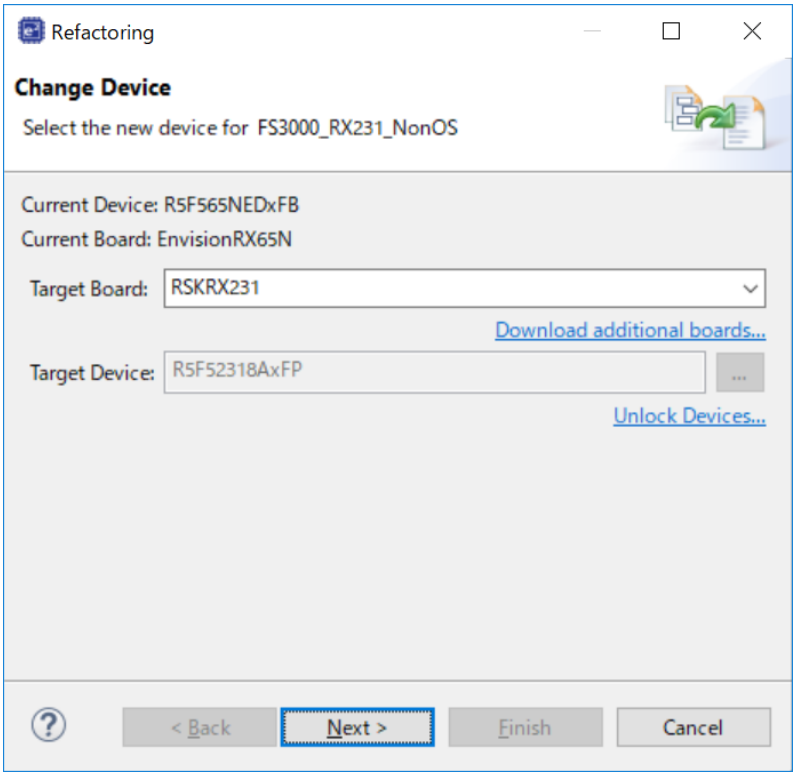


6.2.2　デバイスの変更

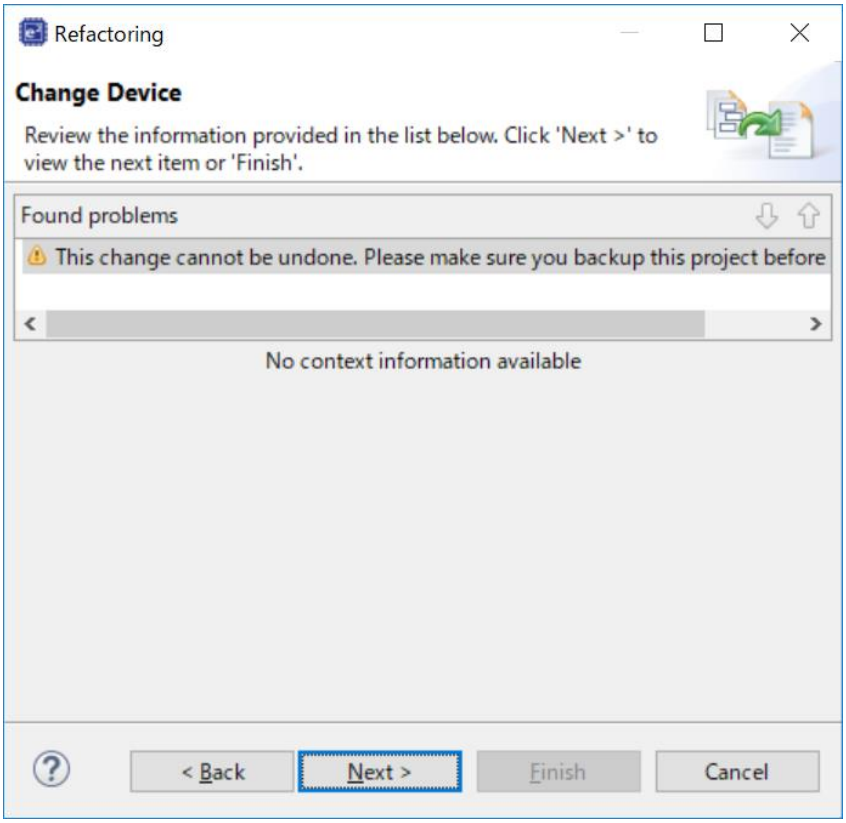
プロジェクトツリーでインポートしたプロジェクトを選択し、右クリックでコンテキストメニューを表示します。表示されたメニューから、”Change Device”を選択します。



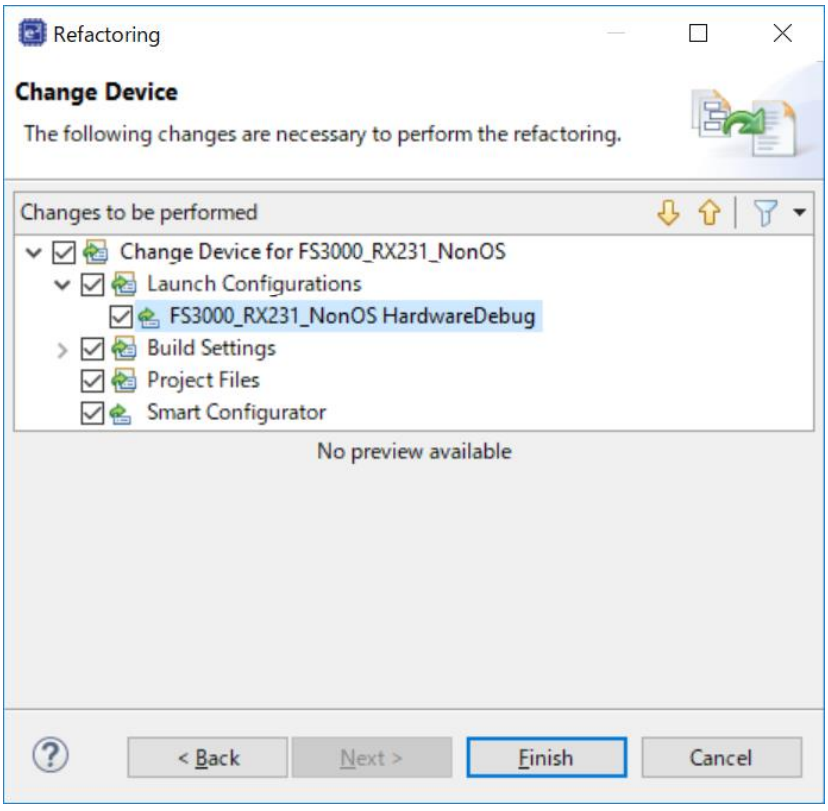
Change Device ウィンドウで、変更したいボードもしくは、デバイスを選択し、[Next]ボタンを押下します。



Warning が表示された場合、内容を確認し問題無ければ、[Next]を押下します。

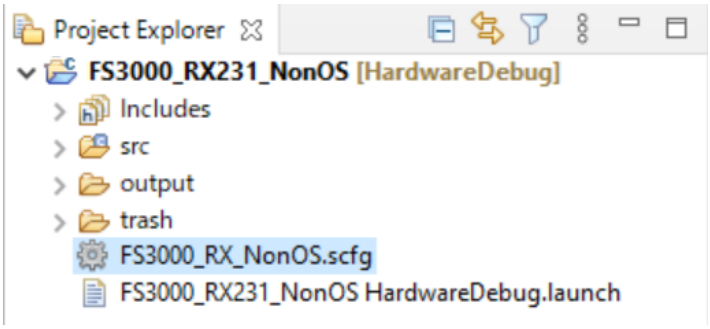


変更内容が表示されますので、[Finish]ボタンを押下して、変更を実行します。

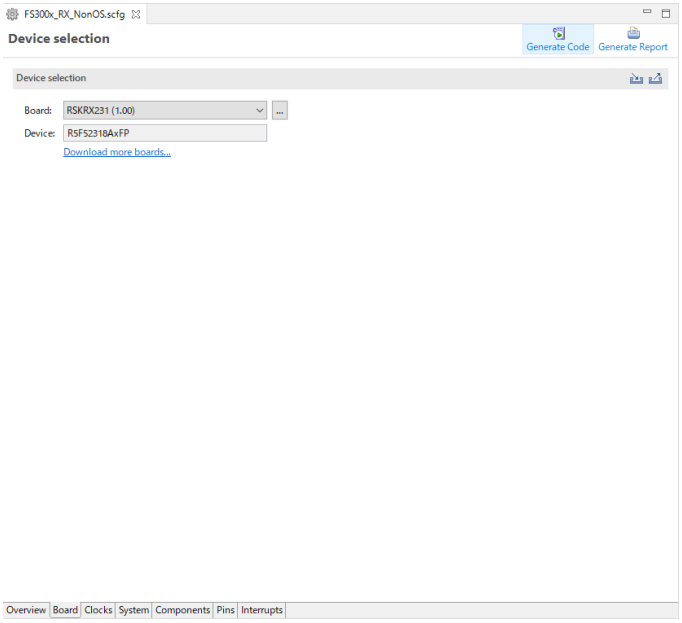


6.2.3　Smart Configurator 設定の変更

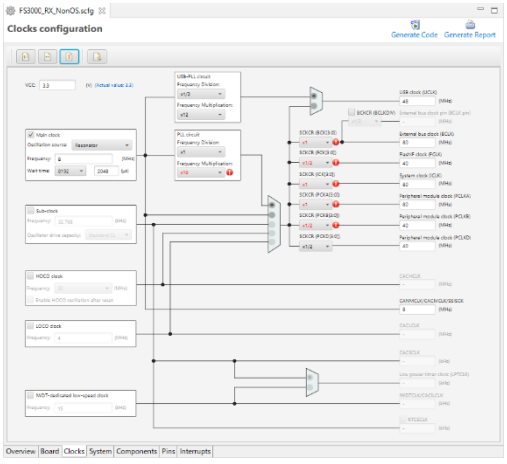
プロジェクトツリーで、デバイス変更したインポートしたプロジェクトの.scfg ファイルをダブルクリックし、Smart Configurator を表示します。

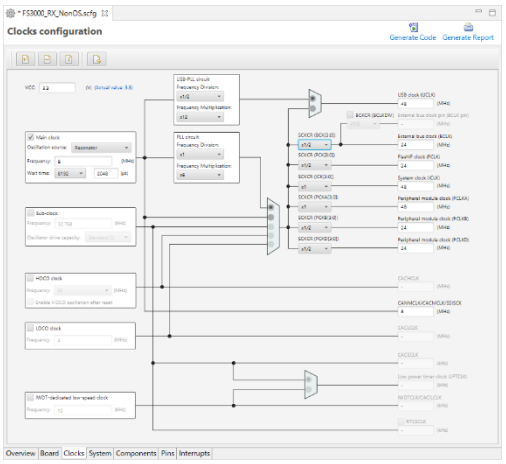


Board タブで、変更したボード、デバイスに変更されていることを確認します。



Clocks タブで、使用するボードに合わせてクロックを設定します。

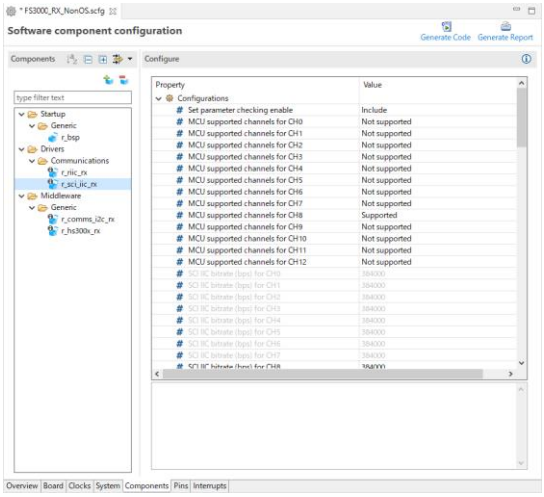


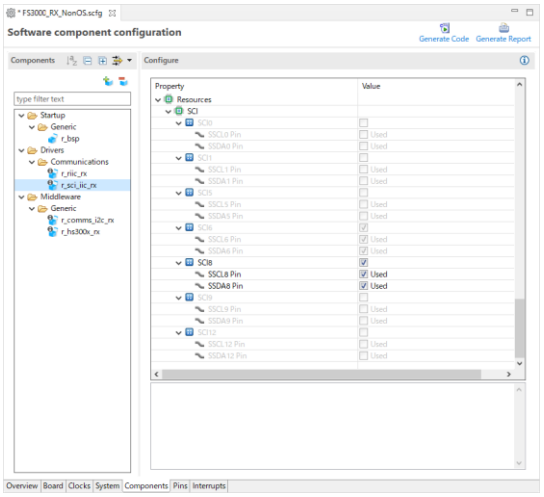


Components タブで、使用するボードに合わせて、各 component の設定を変更します。

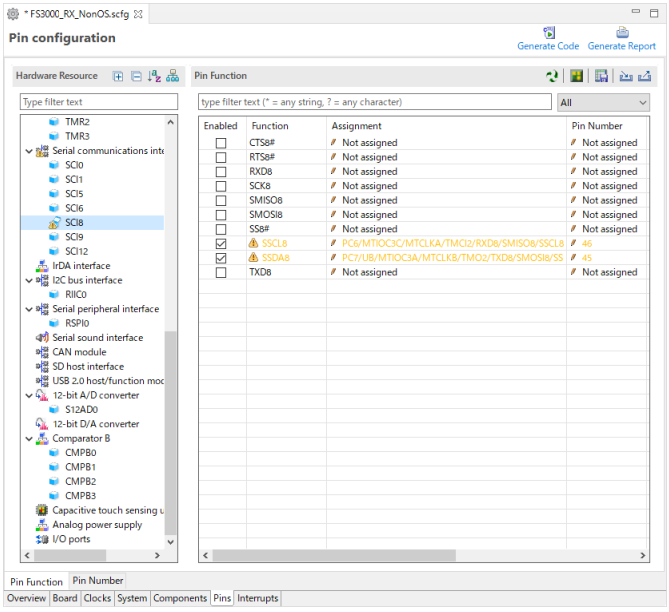
RSK RX231 では、PMOD に SCI8 が割り当てられていますので、r_sci_iic_rx の MCU supported channels for CH2 を”Not supported”に、MCU supported channels for CH8 を”Supported”に変更します。

Resources の SCI8, SSCL8 Pin, SSDA8 pin をチェックします。





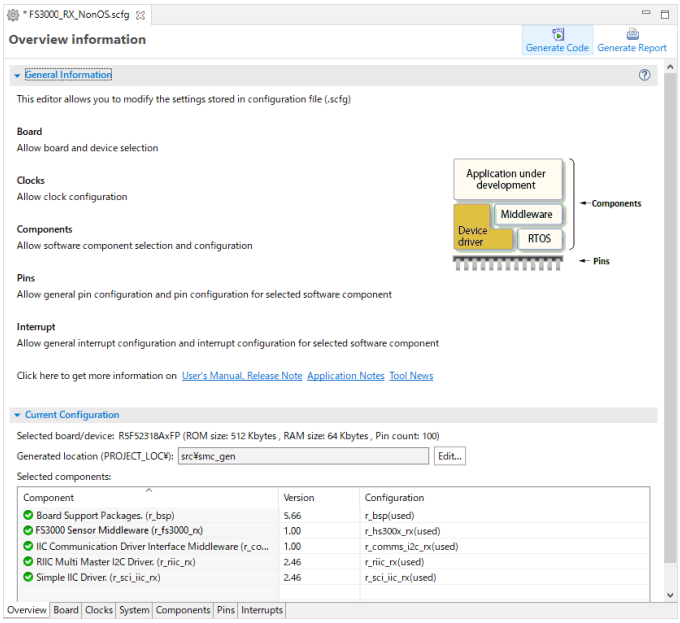
Pins タブの Pin function で、SCI8 端子に端子機能が割り当てられていることを確認します。



RSK RX231 のボード情報は、PMOD Type 2A(Extend SPI)で使用するように割り当てられていますので、I2C で使用する場合は、Warning が表示されますが問題ありません。

また、センサボードを接続するには、PMOD Type 2A を PMOD Type 6A に変換するボードが必要となります。

[Generate Code]アイコンを押下して、コード生成を行います。



プロジェクトをビルドします。

メニューから[Debug Configurations]を選択し、使用するボードに接続するエミュレータに合わせて、Debugger の設定を変更してください。

6.2.4　ツールチェイン設定変更

CC-RX ツールチェイン以外のツールチェインを使用する場合は、本プロジェクトから RX_FS3000.c (Non-OS)または、main.c と fs3000_sensor_thread_entry.c (FreeRTOS)または、fs3000_sensor_thread_entry.c 及び sensor_thread_common.c、sensor_thread_common.h (Azure)をコピーしてプロジェクトを作成してください。

6.3　RL78 サンプルプロジェクト

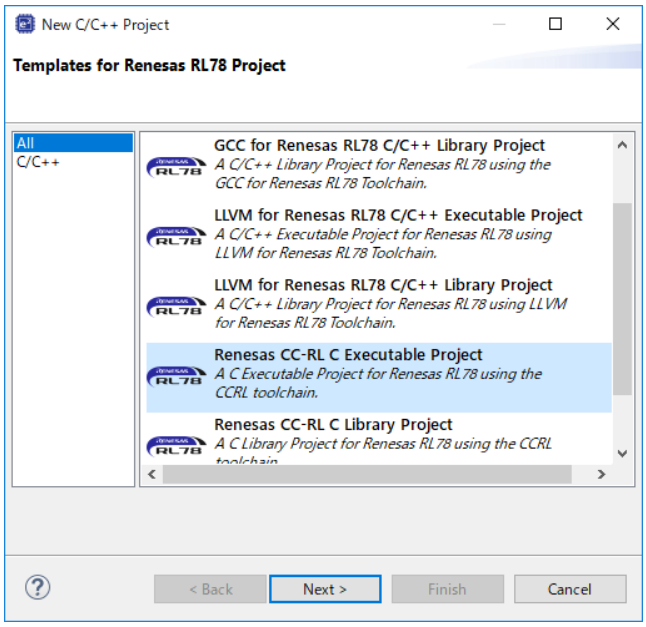
RL78 の場合、新規にサンプルプロジェクトを作成する手順となります。

本章解説では、例として RSK RL78/G1G ボードで使用できるサンプルプロジェクトの作成手順を記載します。

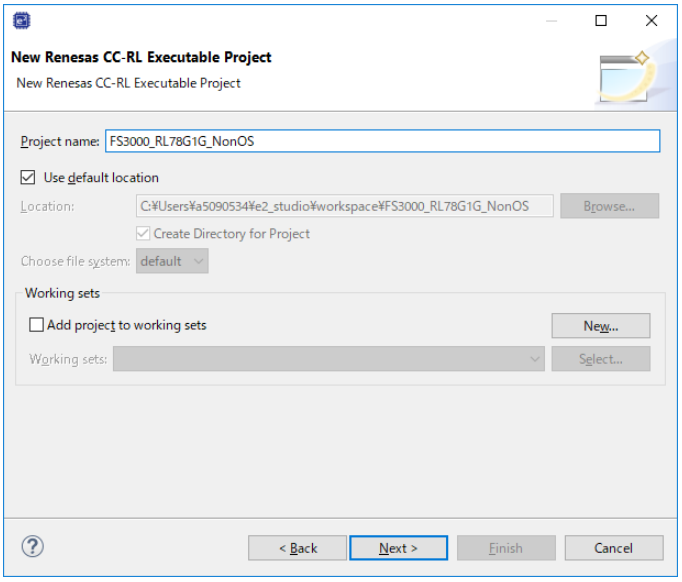
6.3.1　新規プロジェクトの作成

メニューから、[File]-[New]-[Renesas C/C++ project] – [Renesas RL78]を選択します。

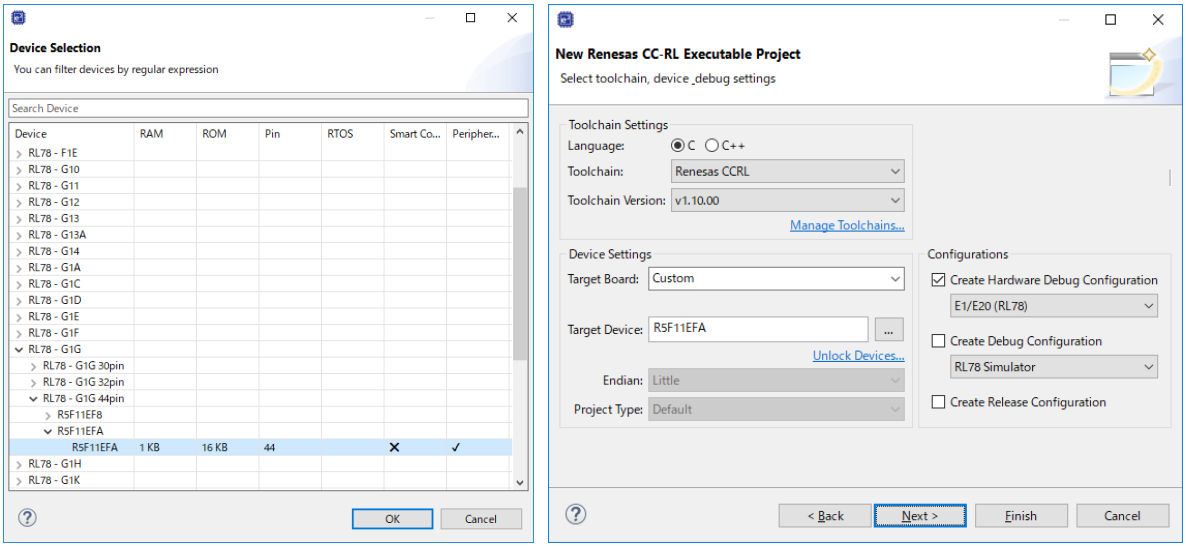
テンプレートから、"Renesas CC-RL C Executable Project”を選択し、[Next]ボタンを押下します。



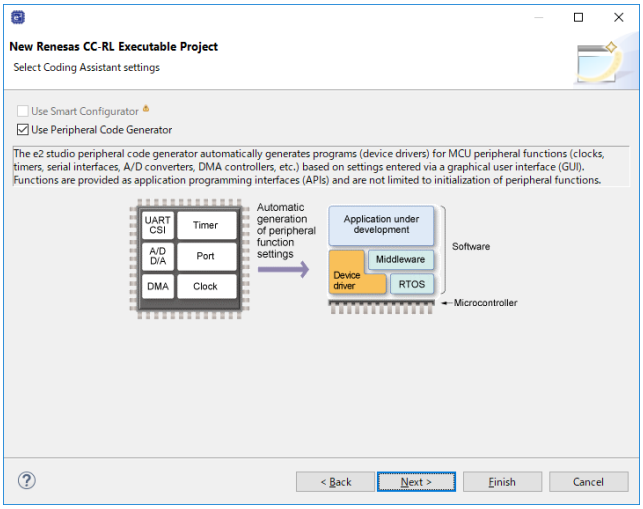
プロジェクト名(例:”FS3000_RL78G1G_NonOS”)を入力し、[Next]ボタンを押下します。



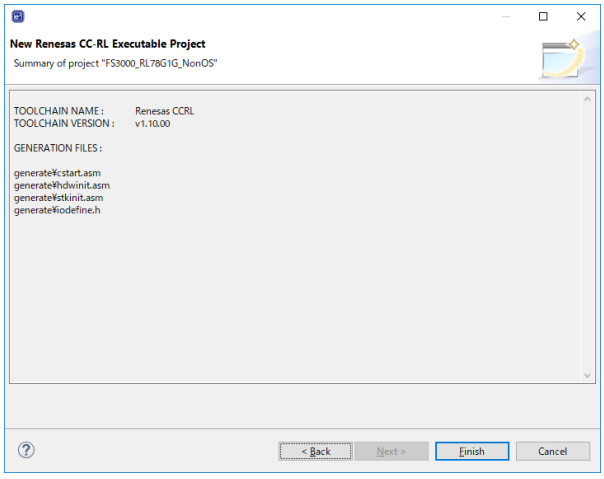
Target Device を変更したいデバイス(例:R5F11EFA)に変更し、[Next]ボタンを押下します。



Use Periphearl Code Generator をチェックし、[Next]ボタンを押下します。

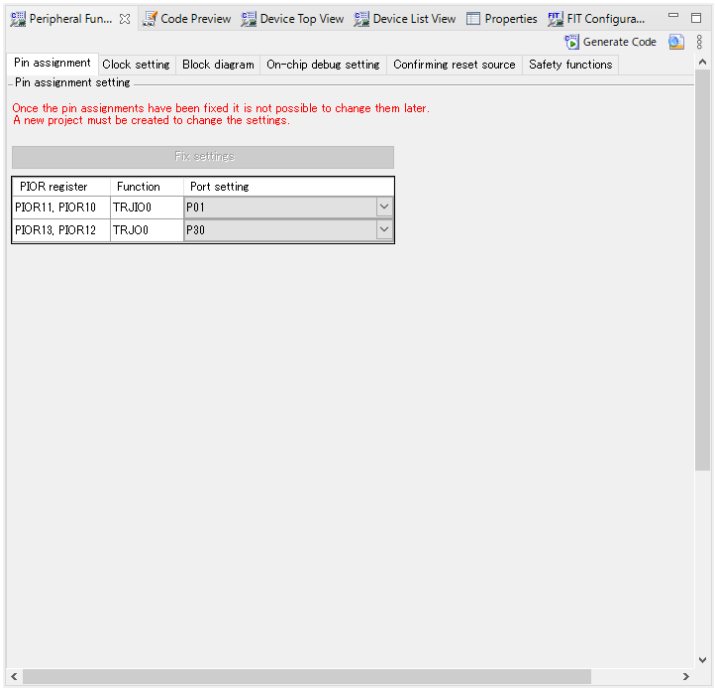


[Finish]ボタンを押下します。

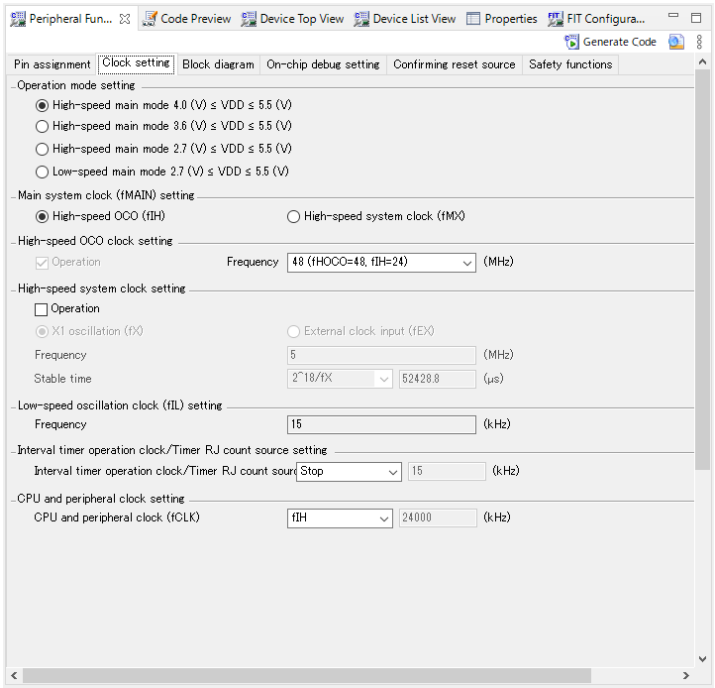


6.3.2　Code Generator の設定

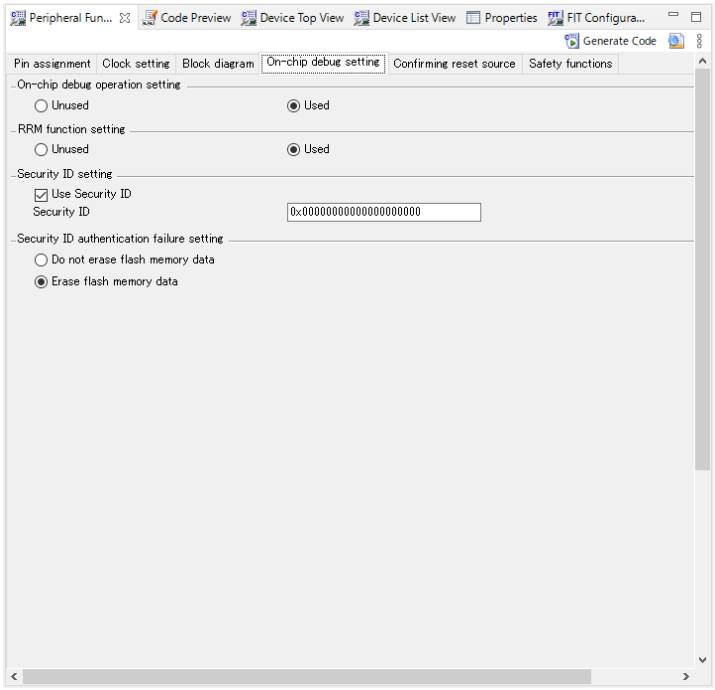
共通/クロック発生回路の端子割り当て設定タブで、使用するボードに合わせて端子割り当てを変更します。



共通/クロック発生回路のクロック設定タブで、使用するボードに合わせてクロック設定を変更します。

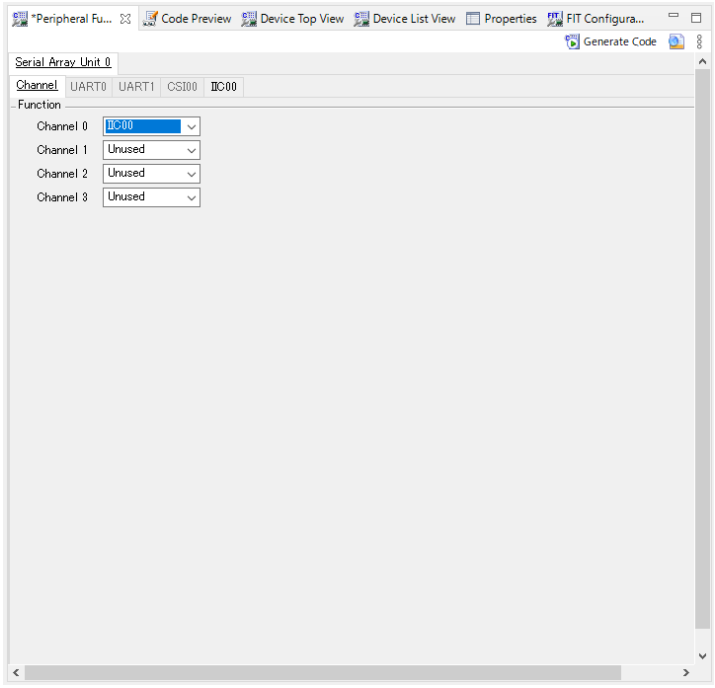


共通/クロック発生回路のオンチップ・デバッグ設定タブで、オンチップ・デバッグ動作設定を”使用する”に設定します。

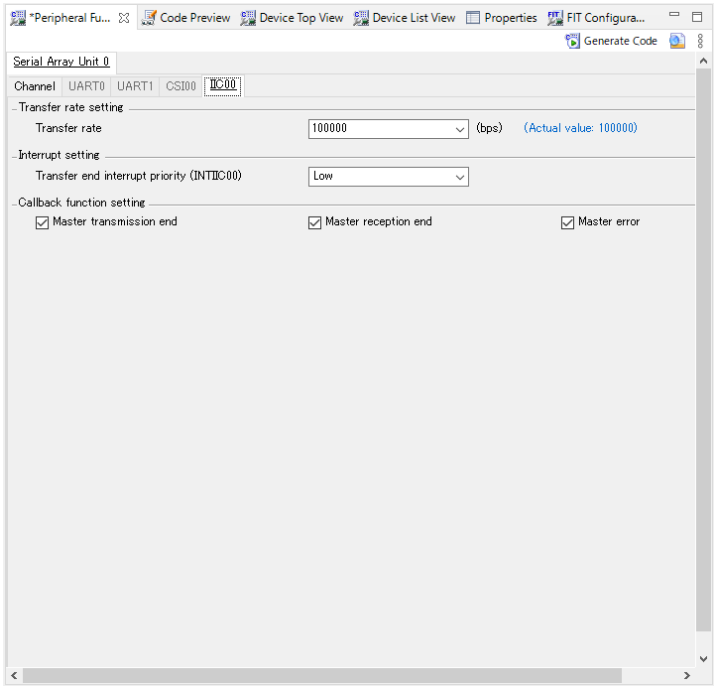


シリアル・アレイ・ユニットを使用する場合は、シリアル・アレイ・ユニットもしくは、シリアルのチャネルタブで、使用するボードの PMOD に割り当てられているチャネルを”IICxx”に設定します。

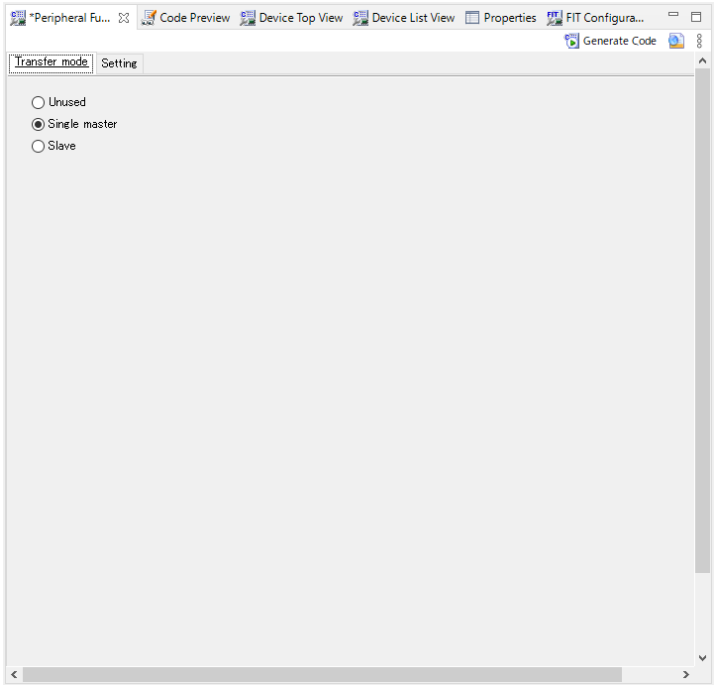
【注】該当端子はポート機能で N-ch が選択されている必要があります。



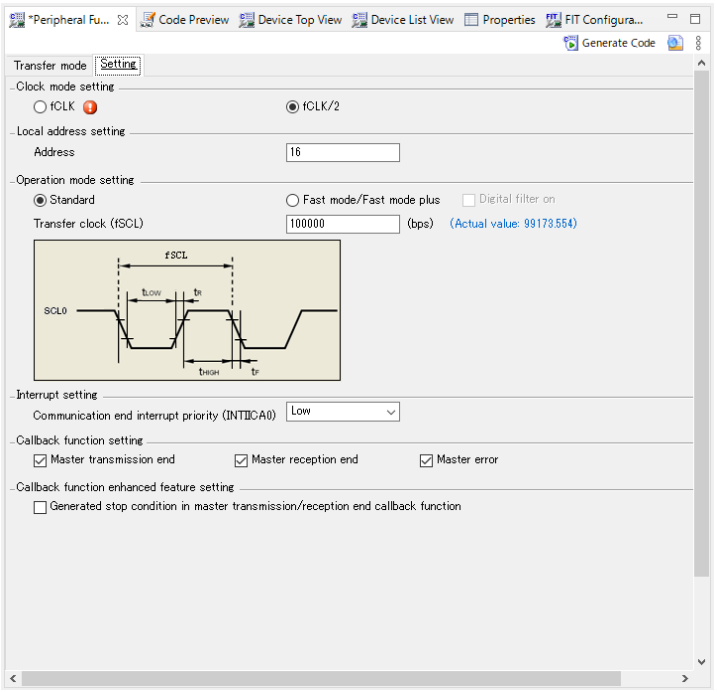
有効にしたシリアル・アレイ・ユニットの IICxx タブで、転送レートを 400000 または、100000 に、転送完了割り込み優先順位を任意の値に、コールバック機能設定を全て有効に設定します。



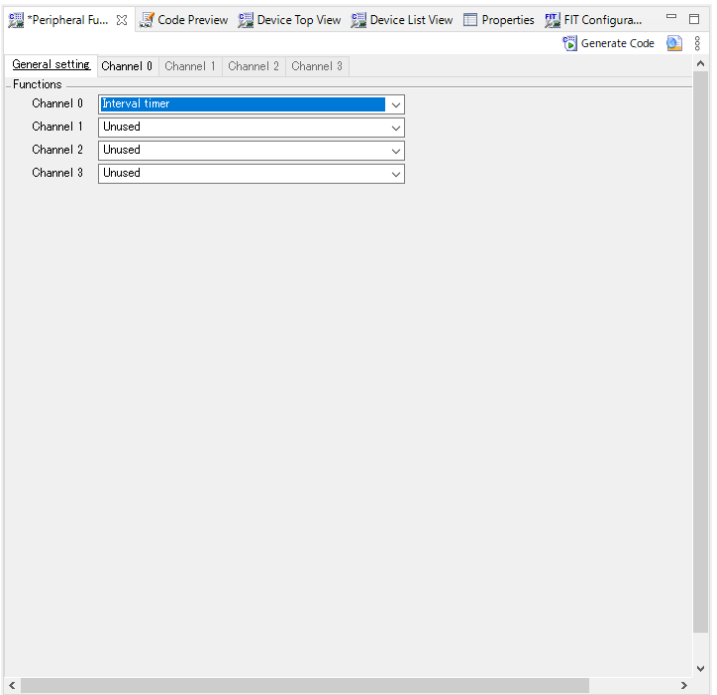
シリアル・インタフェース IICA を使用する場合は、シリアル・インタフェース IICA もしくは、シリアルチャネルで、使用するボードの PMOD に割り当てられているチャネルの転送モードを、”シングルマスタ”に設定します。



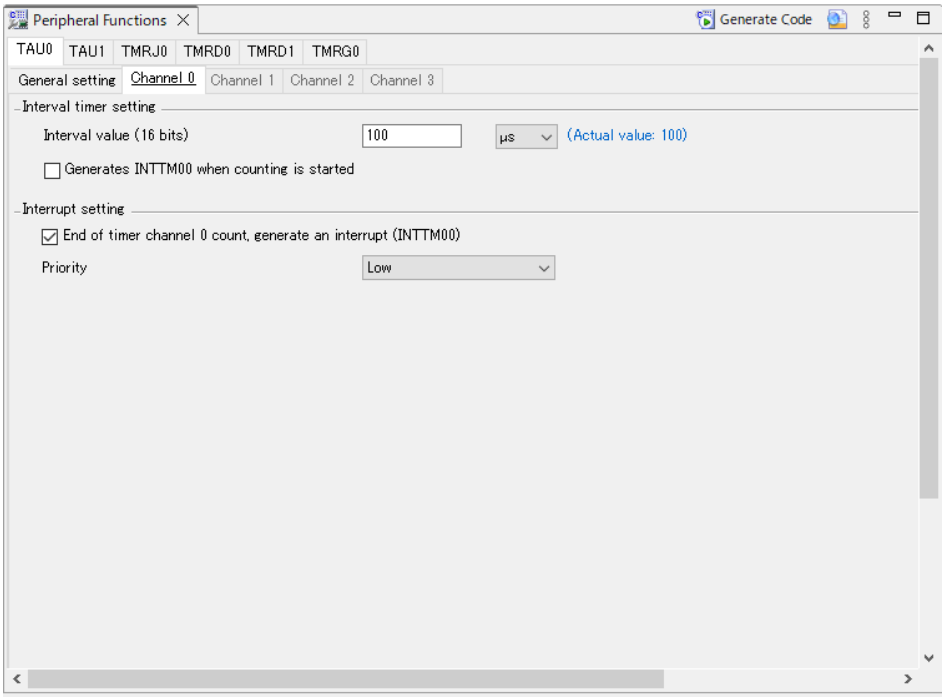
シングルマスタに設定したチャネルの設定で、動作モード設定をファスト・モード、400000 または、標準 100000 に、割り込み優先設定を任意の値に、コールバック機能設定を全て有効に、コールバック拡張機能設定を無効に設定します。



タイマ・アレイ・ユニットの任意のチャネルもしくは、タイマの任意の TAU の一般設定で、機能を”インターバル・タイマ”に設定します。



インターバル・タイマに設定したチャネルのインターバル時間を”100μs”に、タイマ割り込みを有効に、割り込み優先レベルを任意の値に設定します。



[Code Generate]ボタンを押下し、コードを生成します。

6.3.3　生成コードの変更

使用する MCU によりコード生成のバージョンが異なるため、コード生成の出力先がこのサンプルソフトウェアとは変更されている場合があります。

r_cg_sau_user.c、r_cg_iica_user.c もしくは、r_cg_serial_user.c を開き、以下のコードを追加します。

r_comms_i2c_if.h のインクルード定義

```
/***********************************************************************
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"
```

コールバック関数への、rm_comms_i2c_bus0_callback()関数の追加

送受信完了コールバックは、引数を false に、エラーコールバックは、引数を true に設定してください。

```
/***********************************************************************
* 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 */
}
```

t_cg_tau_user.c もしくは、r_cg_timer_user.c を開き、以下のコードを追加します。

(sensor_name)_delay_callback()関数の external 宣言

```
/***********************************************************************
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 */
```

タイマ割り込みのコールバック関数に、(sensor_name)_delay_callback()関数のコール処理

```
/***********************************************************************
* 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 */
}
```

r_cg_tau.c もしくは、r_cg_timer.c を開き、以下のコードを追加します。

上記ファイルのユーザコード記述部分に R_TAU0_Channel0_Reset()関数の定義

```
void R_TAU0_Channel0_Reset(void)

{
      /* function not supported by this module */
}
```

r_cg_tau.h もしくは、r_cg_timer.h を開き、以下のコードを追加します。

R_TAU0_Channel0_Reset()関数のプロトタイプ宣言

```
/***********************************************************************
  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 */
```

r_cg_main.c もしくは、r_main.c を開き、以下のコードを追加します。

各関数のプロトタイプ宣言

```
/***********************************************************************
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 */
```

main()関数へのコード追加

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

    /* Open FS3000 */
    g_fs3000_sensor0_quick_setup();

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

g_comms_i2c_bus0_quick_setup()関数、demo_err()関数の定義

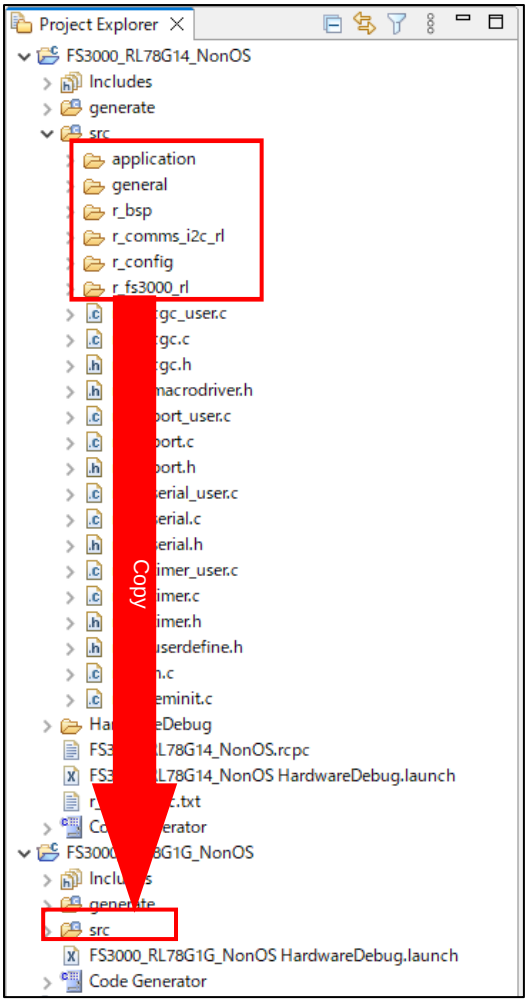
```
void g_comms_i2c_bus0_quick_setup(void)
{
    /* bus has been opened by startup process */
}

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

6.3.4　サンプルソースの変更

サンプルプロジェクト”FS3000_RL78G14_NonOS”のプロジェクトツリーから、"application" "general" "r_bsp" "r_comms_i2c_rl" "r_config" "r_fs3000_rl"フォルダを選択し、右クリックで表示されるコンテキストメニューから”Copy”を選択してください。

その後、新しく作成したプロジェクトの”src”フォルダを選択し、右クリックで表示されるコンテキストメニューから”paste”を選択し、ファイルをコピーしてください。



r_config フォルダにある r_comms_i2c_rl_config.h を開き、以下の定義の値を変更します。

・COMMS_I2C_CFG_BUSx_DRIVER_TYPE

・COMMS_I2C_CFG_BUSx_DRIVER_CH

シリアル・アレイ・ユニット チャネル 0 を使用する場合

```
/* 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. */
```

シリアル・インタフェース IICA チャネル 0 を使用する場合

```
/* 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. */
```

その他の定義については、5.コンフィグ設定を参照してください。

コード生成の周辺機能名が、シリアル・アレイ・ユニット、シリアル・インタフェース IICA および、タイマ・アレイ・ユニットとなっている場合は、以下の箇所についてもサンプルソースを修正してください。

src/general/r_smc_entry.h

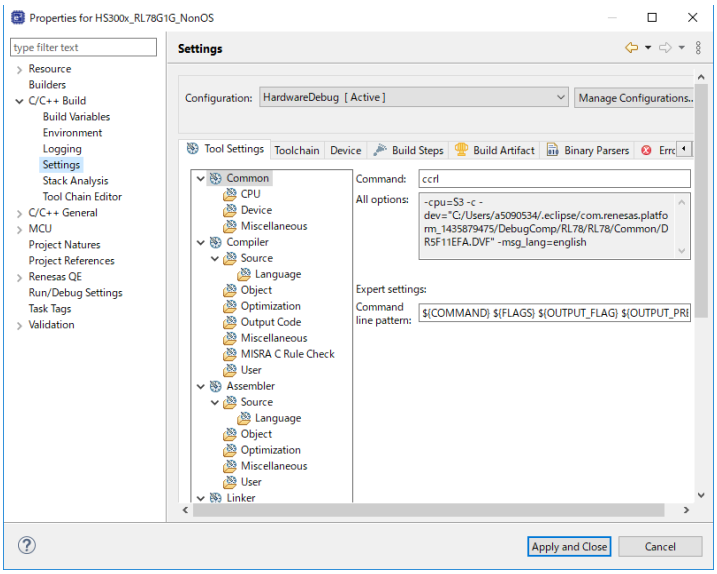
“r_cg_serial.h”を”r_cg_sau.h”もしくは、”r_cg_iica.h”に変更

“r_cg_timer.h”を”r_cg_tau.h”に変更

```
/***********************************************************************
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"
```

プロジェクトのプロパティを開きます。

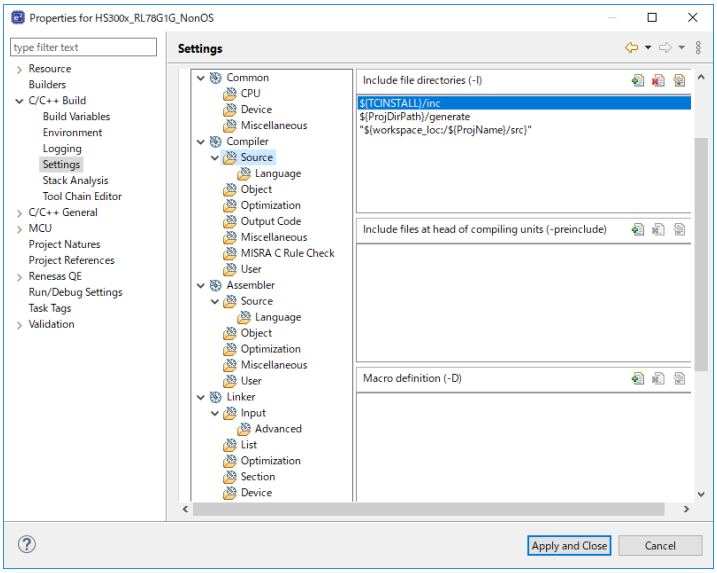
プロパティの[C/C++ Build]-[Settings]を選択し、settings を開きます。

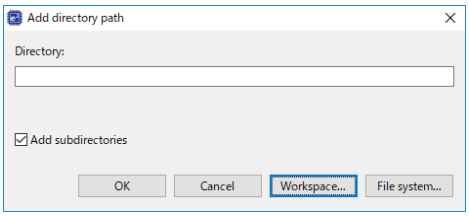


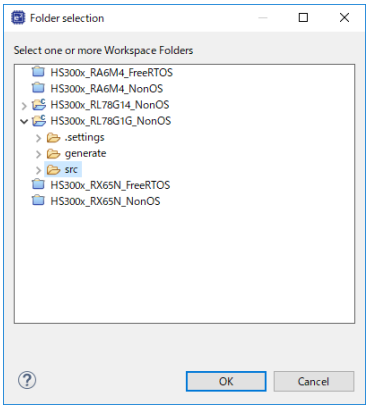
Tool Settings タブの[Compiler]-[Source]を選択し、[Add]アイコンを押下します。

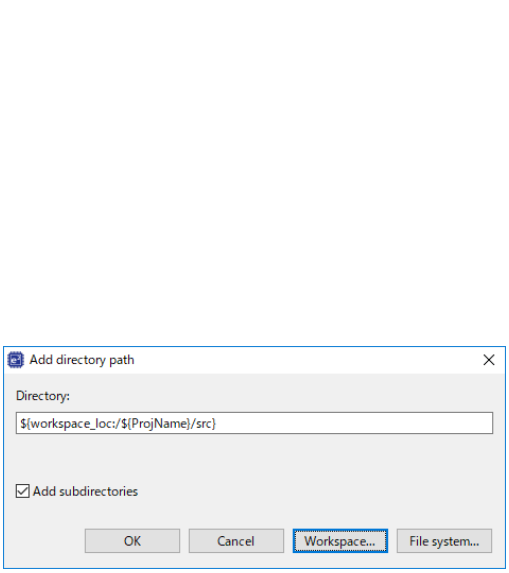
[Add directory path]ダイアログで、[Workspace]ボタンを押下し、表示されたプロジェクトの一覧から、新しく作成したプロジェクトの”src”フォルダを選択し、[OK]ボタンを押下します。

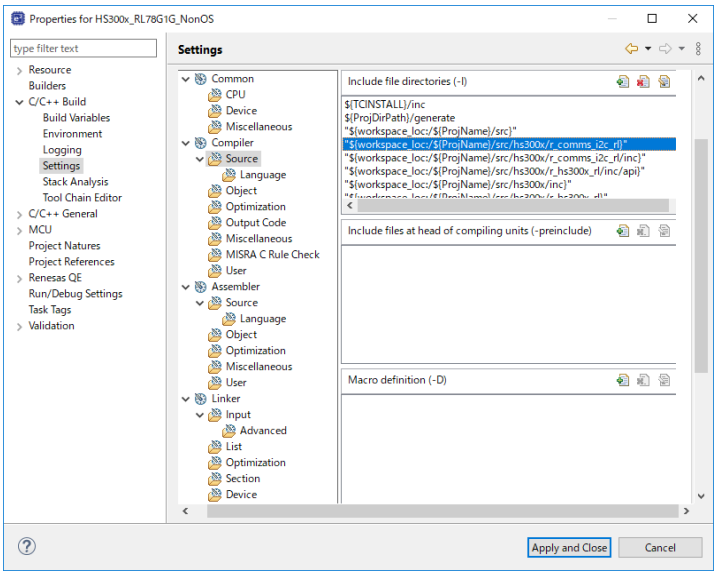
”Add subdirectroies”にチェックを入れて、[OK]ボタンを押下します。





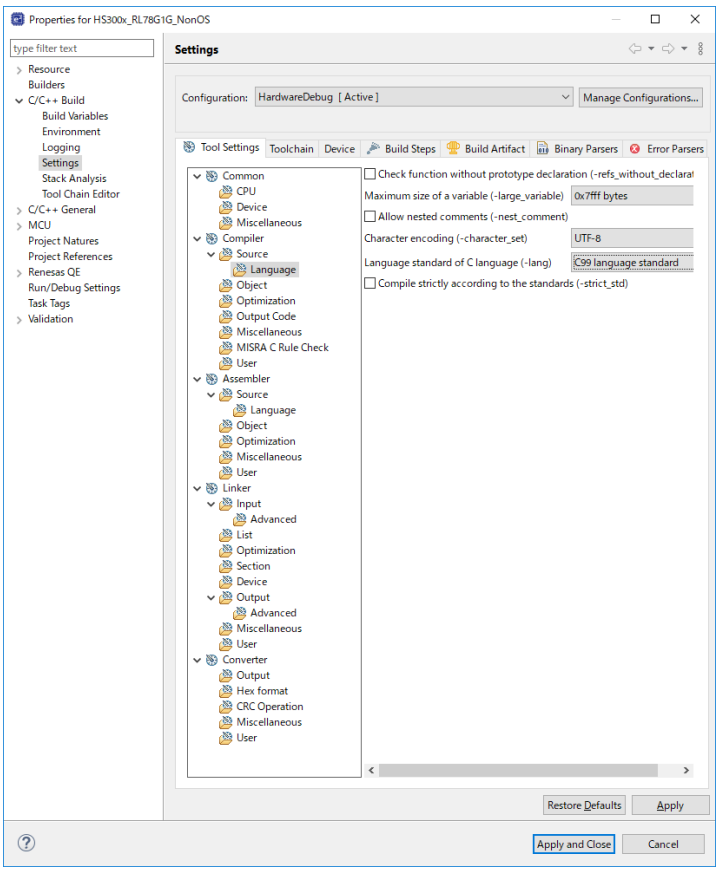






Tool settings タブの[Compiler]-[Source]-[Language]を選択し、Language standard of C language を”C99 language standard”に変更します。

[Apply and Close]ボタンを押下して、プロパティを閉じます。



プロジェクトをビルドします。

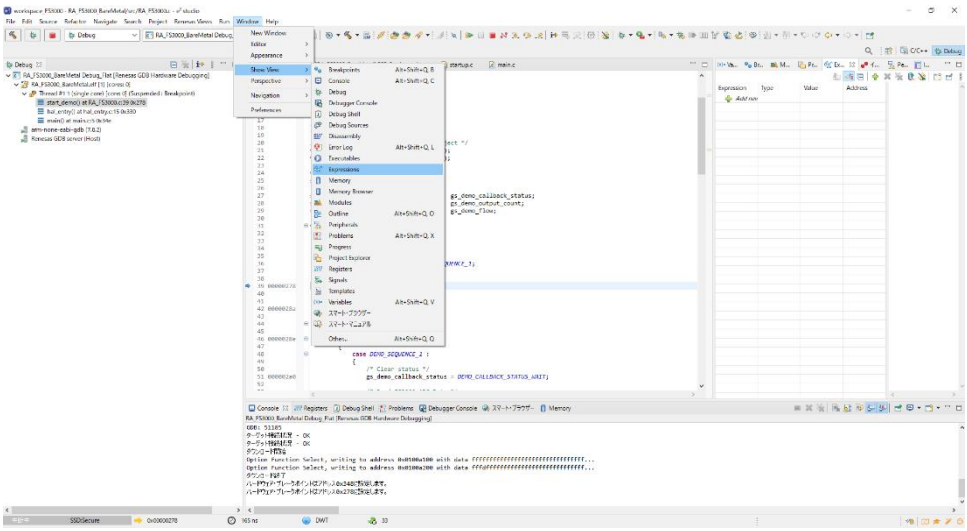
メニューから[Debug Configurations]を選択し、使用するボードに接続するエミュレータに合わせて、Debugger の設定を変更してください。

7.　風速データの確認方法

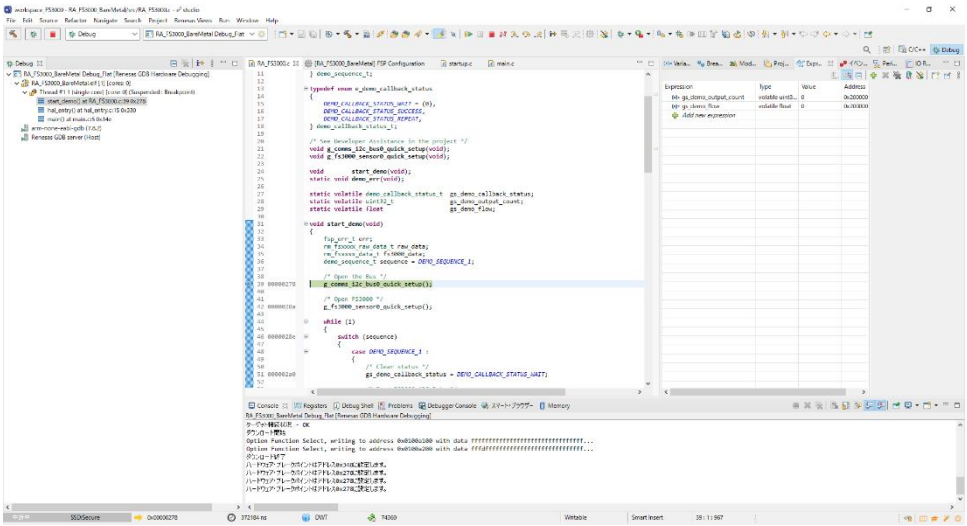
リアルタイムの風速データは、以下の手順に従って確認することができます。

Debug を実行後、Expressions ウィンドウを開いてください。

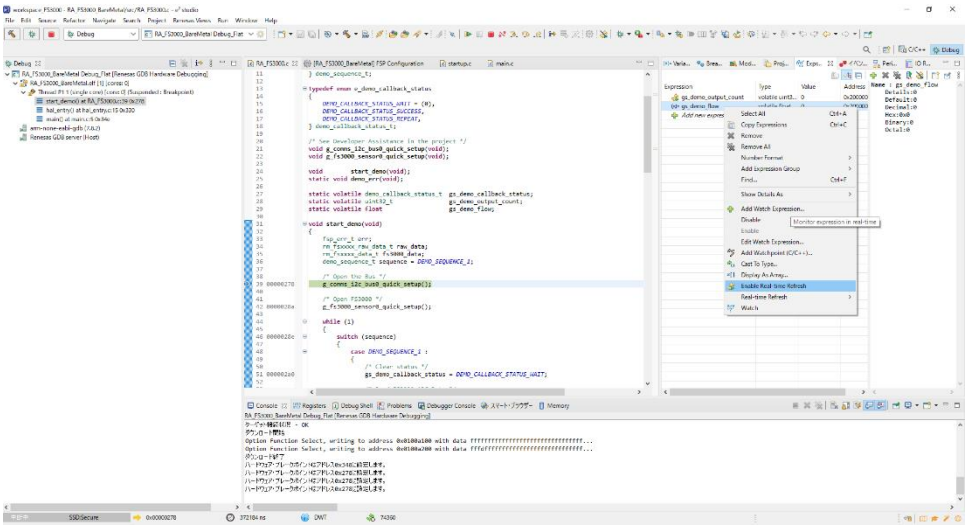
Expressions ウィンドウは[Window]→[Show View]→[Expressions]から開くことができます。



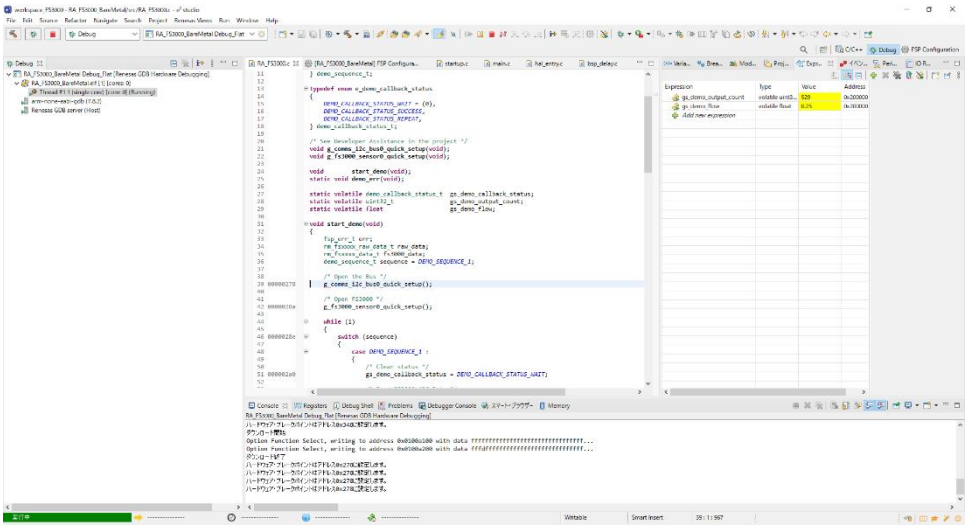
Expressions 内の Add new expression をクリックして、gs_fs3000_data を追加してください。



追加した変数を右クリックすると、Enable Real-time Refresh を選択することができます。



Debug をスタートすると、リアルタイムの値を確認することができます。



改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
Rev.1.00	June 30, 2022	-	初版リリース
Rev.1.01	March 3, 2023	-	更新：RL78 の動作環境
Rev.1.02	March 29, 2023	-	更新：RA、RX、RL78、RZ の動作環境 更新：サンプルソフトウェアメインフロー 更新：デバイス変更ガイド
Rev.1.03	September 7, 2023	-	更新：デバイス変更ガイド 削除：RE01 の項目

製品ご使用上の注意事項

ここでは、マイコン製品全体に適用する「使用上の注意事項」について説明します。個別の使用上の注意事項については、本ドキュメントおよびテクニカルアップデートを参照してください。

1. 静電気対策
   CMOS 製品の取り扱いの際は静電気防止を心がけてください。CMOS 製品は強い静電気によってゲート絶縁破壊を生じることがあります。運搬や保存の際には、当社が出荷梱包に使用している導電性のトレーやマガジンケース、導電性の緩衝材、金属ケースなどを利用し、組み立て工程にはアースを施してください。プラスチック板上に放置したり、端子を触ったりしないでください。また、CMOS 製品を実装したボードについても同様の扱いをしてください。
2. 電源投入時の処置
   電源投入時は、製品の状態は不定です。電源投入時には、LSI の内部回路の状態は不確定であり、レジスタの設定や各端子の状態は不定です。外部リセット端子でリセットする製品の場合、電源投入からリセットが有効になるまでの期間、端子の状態は保証できません。同様に、内蔵パワーオンリセット機能を使用してリセットする製品の場合、電源投入からリセットのかかる一定電圧に達するまでの期間、端子の状態は保証できません。
3. 電源オフ時における入力信号
   当該製品の電源がオフ状態のときに、入力信号や入出力プルアップ電源を入れないでください。入力信号や入出力プルアップ電源からの電流注入により、誤動作を引き起こしたり、異常電流が流れ内部素子を劣化させたりする場合があります。資料中に「電源オフ時における入力信号」についての記載のある製品は、その内容を守ってください。
4. 未使用端子の処理
   未使用端子は、「未使用端子の処理」に従って処理してください。CMOS 製品の入力端子のインピーダンスは、一般に、ハイインピーダンスとなっています。未使用端子を開放状態で動作させると、誘導現象により、LSI 周辺のノイズが印加され、LSI 内部で貫通電流が流れたり、入力信号と認識されて誤動作を起こす恐れがあります。
5. クロックについて
   リセット時は、クロックが安定した後、リセットを解除してください。プログラム実行中のクロック切り替え時は、切り替え先クロックが安定した後に切り替えてください。リセット時、外部発振子（または外部発振回路）を用いたクロックで動作を開始するシステムでは、クロックが十分安定した後、リセットを解除してください。また、プログラムの途中で外部発振子（または外部発振回路）を用いたクロックに切り替える場合は、切り替え先のクロックが十分安定してから切り替えてください。
6. 入力端子の印加波形
   入力ノイズや反射波による波形歪みは誤動作の原因になりますので注意してください。CMOS 製品の入力がノイズなどに起因して、V_{IL}（Max.）から V_{IH}（Min.）までの領域にとどまるような場合は、誤動作を引き起こす恐れがあります。入力レベルが固定の場合はもちろん、V_{IL}（Max.）から V_{IH}（Min.）までの領域を通過する遷移期間中にチャタリングノイズなどが入らないように使用してください。
7. リザーブアドレス（予約領域）のアクセス禁止
   リザーブアドレス（予約領域）のアクセスを禁止します。アドレス領域には、将来の拡張機能用に割り付けられている リザーブアドレス（予約領域）があります。これらのアドレスをアクセスしたときの動作については、保証できませんので、アクセスしないようにしてください。
8. 製品間の相違について
   型名の異なる製品に変更する場合は、製品型名ごとにシステム評価試験を実施してください。同じグループのマイコンでも型名が違うと、フラッシュメモリ、レイアウトパターンの相違などにより、電気的特性の範囲で、特性値、動作マージン、ノイズ耐量、ノイズ幅射量などが異なる場合があります。型名が違う製品に変更する場合は、個々の製品ごとにシステム評価試験を実施してください。

ご注意書き

1. 本資料に記載された回路、ソフトウェアおよびこれらに関連する情報は、半導体製品の動作例、応用例を説明するものです。回路、ソフトウェアおよびこれらに関連する情報を使用する場合、お客様の責任において、お客様の機器・システムを設計ください。これらの使用に起因して生じた損害（お客様または第三者いずれに生じた損害も含みます。以下同じです。）に関し、当社は、一切その責任を負いません。
2. 当社製品または本資料に記載された製品データ、図、表、プログラム、アルゴリズム、応用回路例等の情報の使用に起因して発生した第三者の特許権、著作権その他の知的財産権に対する侵害またはこれらに関する紛争について、当社は、何らの保証を行うものではなく、また責任を負うものではありません。
3. 当社は、本資料に基づき当社または第三者の特許権、著作権その他の知的財産権を何ら許諾するものではありません。
4. 当社製品を組み込んだ製品の輸出入、製造、販売、利用、配布その他の行為を行うにあたり、第三者保有の技術の利用に関するライセンスが必要となる場合、当該ライセンス取得の判断および取得はお客様の責任において行ってください。
5. 当社製品を、全部または一部を問わず、改造、改変、複製、リバースエンジニアリング、その他、不適切に使用しないでください。かかる改造、改変、複製、リバースエンジニアリング等により生じた損害に関し、当社は、一切その責任を負いません。
6. 当社は、当社製品の品質水準を「標準水準」および「高品質水準」に分類しており、各品質水準は、以下に示す用途に製品が使用されることを意図しております。

   標準水準：　コンピュータ、OA 機器、通信機器、計測機器、AV 機器、家電、工作機械、パーソナル機器、産業用ロボット等

   高品質水準：輸送機器（自動車、電車、船舶等）、交通制御（信号）、大規模通信機器、金融端末基幹システム、各種安全制御装置等

   当社製品は、データシート等により高信頼性、Harsh environment 向け製品と定義しているものを除き、直接生命・身体に危害を及ぼす可能性のある機器・システム（生命維持装置、人体に埋め込み使用するもの等）、もしくは多大な物的損害を発生させるおそれのある機器・システム（宇宙機器と、海底中継器、原子力制御システム、航空機制御システム、プラント基幹システム、軍事機器等）に使用されることを意図しておらず、これらの用途に使用することは想定していません。たとえ、当社が想定していない用途に当社製品を使用したことにより損害が生じても、当社は一切その責任を負いません。
7. あらゆる半導体製品は、外部攻撃からの安全性を 100%保証されているわけではありません。当社ハードウェア／ソフトウェア製品にはセキュリティ対策が組み込まれているものもありますが、これによって、当社は、セキュリティ脆弱性または侵害（当社製品または当社製品が使用されているシステムに対する不正アクセス・不正使用を含みますが、これに限りません。）から生じる責任を負うものではありません。当社は、当社製品または当社製品が使用されたあらゆるシステムが、不正な改変、攻撃、ウイルス、干渉、ハッキング、データの破壊または窃盗その他の不正な侵入行為（「脆弱性問題」といいます。）によって影響を受けないことを保証しません。当社は、脆弱性問題に起因しまたはこれに関連して生じた損害について、一切責任を負いません。また、法令において認められる限りにおいて、本資料および当社ハードウェア／ソフトウェア製品について、商品性および特定目的との合致に関する保証ならびに第三者の権利を侵害しないことの保証を含め、明示または黙示のいかなる保証も行いません。
8. 当社製品をご使用の際は、最新の製品情報（データシート、ユーザーズマニュアル、アプリケーションノート、信頼性ハンドブックに記載の「半導体デバイスの使用上の一般的な注意事項」等）をご確認の上、当社が指定する最大定格、動作電源電圧範囲、放熱特性、実装条件その他指定条件の範囲内でご使用ください。指定条件の範囲を超えて当社製品をご使用された場合の故障、誤動作の不具合および事故につきましては、当社は、一切その責任を負いません。
9. 当社は、当社製品の品質および信頼性の向上に努めていますが、半導体製品はある確率で故障が発生したり、使用条件によっては誤動作したりする場合があります。また、当社製品は、データシート等において高信頼性、Harsh environment 向け製品と定義しているものを除き、耐放射線設計を行っておりません。仮に当社製品の故障または誤動作が生じた場合であっても、人身事故、火災事故その他社会的損害等を生じさせないよう、お客様の責任において、冗長設計、延焼対策設計、誤動作防止設計等の安全設計およびエージング処理等、お客様の機器・システムとしての出荷保証を行ってください。特に、マイコンソフトウェアは、単独での検証は困難なため、お客様の機器・システムとしての安全検証をお客様の責任で行ってください。
10. 当社製品の環境適合性等の詳細につきましては、製品個別に必ず当社営業窓口までお問合せください。ご使用に際しては、特定の物質の含有・使用を規制する RoHS 指令等、適用される環境関連法令を十分調査のうえ、かかる法令に適合するようご使用ください。かかる法令を遵守しないことにより生じた損害に関して、当社は、一切その責任を負いません。
11. 当社製品および技術を国内外の法令および規則により製造・使用・販売を禁止されている機器・システムに使用することはできません。当社製品および技術を輸出、販売または移転等する場合は、「外国為替及び外国貿易法」その他日本国および適用される外国の輸出管理関連法規を遵守し、それらの定めるところに従い必要な手続きを行ってください。
12. お客様が当社製品を第三者に転売等される場合には、事前に当該第三者に対して、本ご注意書き記載の諸条件を通知する責任を負うものといたします。
13. 本資料の全部または一部を当社の文書による事前の承諾を得ることなく転載または複製することを禁じます。
14. 本資料に記載されている内容または当社製品についてご不明な点がございましたら、当社の営業担当者までお問合せください。

注 1. 本資料において使用されている「当社」とは、ルネサス エレクトロニクス株式会社およびルネサス エレクトロニクス株式会社が直接的、間接的に支配する会社をいいます。

注 2. 本資料において使用されている「当社製品」とは、注 1 において定義された当社の開発、製造製品をいいます。

(Rev.5.0-1　2020.10)

本社所在地

〒135-0061 東京都江東区豊洲 3-2-24（豊洲フォレシア）

www.renesas.com

お問合せ窓口

弊社の製品や技術、ドキュメントの最新情報、最寄の営業お問合せ窓口に関する情報などは、弊社ウェブサイトをご覧ください。

www.renesas.com/contact/

商標について

ルネサスおよびルネサスロゴはルネサス エレクトロニクス株式会社の商標です。すべての商標および登録商標は、それぞれの所有者に帰属します。