ルネサス RX ファミリ

Azure RTOS TraceX を用いた Azure RTOS ThreadX のデバッグ

要旨

Azure RTOS ThreadXはMicrosoft CorporationのRTOSで、高性能の組み込み型カーネルをベースとしています。
Azure RTOS TraceXはWindowsで動作する解析ツールです。リアルタイムシステムイベントをグラフィック表示する機能により、リアルタイムシステムの動作を可視化して、組み込み開発者がより良く理解できるようにします。
本アプリケーションノートでは、e2 studioでのアプリケーション開発におけるAzure RTOS ThreadXのスレッドとオブジェクト状態（リソース）をチェックする手順を説明します。Azure RTOS TraceXのインストールや起動についても説明します。

動作確認デバイス

RX65N グループ(R5F565NEHDFB)

動作環境

ターゲットボード	CK-RX65N
統合開発環境（IDE）	e2 studio バージョン2023-01
コンパイラ	CC-RX V3.04.00
トレースツール	Microsoft Azure RTOS TraceX v6.1.12.0
OS	Microsoft Azure RTOS ThreadX v6.2.0

【注】 あらかじめ、以下のURLからIDEをダウンロードしてください
統合開発環境e² studio RXファミリ向け情報 | Renesas

本アプリケーションノートは、下記アプリケーションノートのRX対応版となります
RA ファミリ Azure RTOS TraceX を用いた Azure RTOS ThreadX のデバッグ

下記の資料もご参照ください

- RX スマート・コンフィグレータ ユーザーガイド: e² studio 編のダウンロードサイト：
  RX Smart Configurator User's Guide: e² studio
- マイクロソフト WEB AzureRTOSThreadX のドキュメント
  Azure RTOS ThreadX とは | Microsoft Learn

  Azure RTOS TraceX とは | Microsoft Learn

目次

1.　e² studio のインストール

e² studio のインストールについては、下記の WEB に掲載の e² studio インストールをご参照ください。

RX Family Software & Tool Course | Renesas

2.　Azure RTOS TraceX のインストール

マイクロソフト社の WEB サイトよりダウンロードしてインストールしてください

Azure RTOS TraceX - Microsoft Store アプリ

図 2-1　Microsoft Store アプリ

3.　CK-RX65N ボードの設定と PC の接続

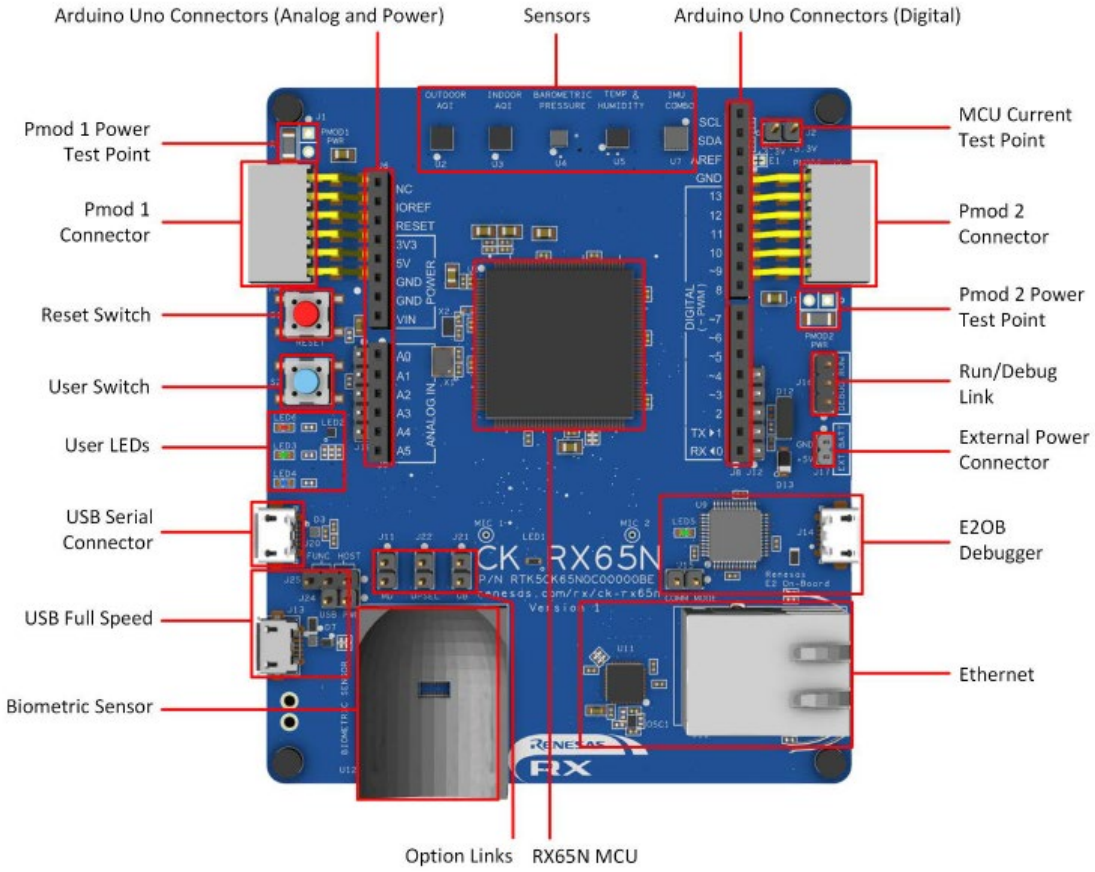


図 3-1　CK-RX65N ボード

ボードのジャンパ設定とケーブルの接続を下図のように行います

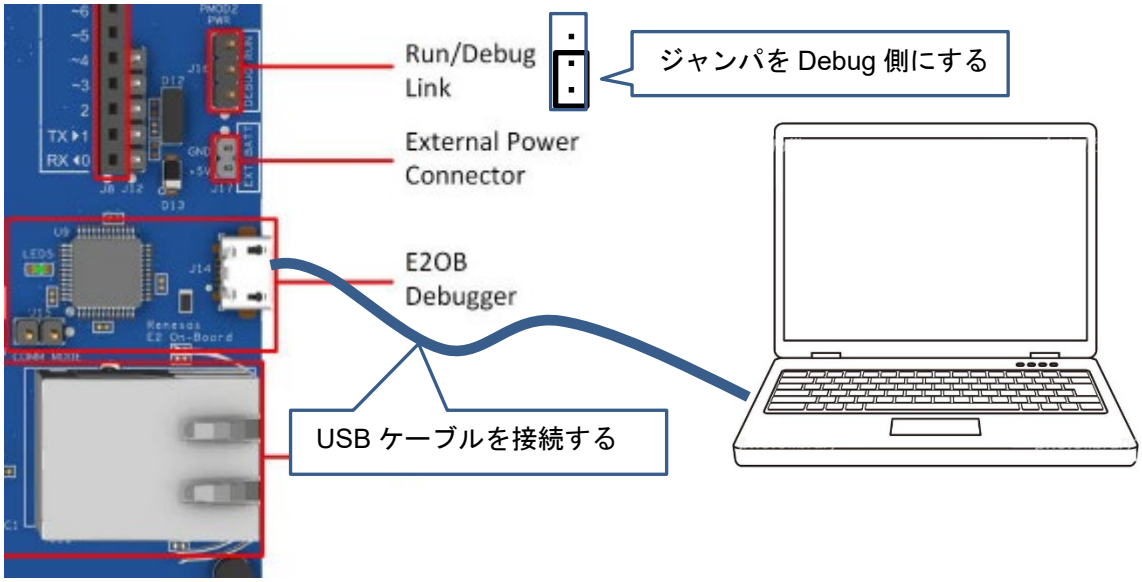


図 3-2 CK-RX65N を PC と接続する際のジャンパ設定

4.　e² studio で ThreadX のプロジェクト作成

下記の手順でプロジェクトを作成します

4.1　プロジェクトの作成

(1)　[ファイル]メニュー → [新規] → [Renesas C/C++ Project] → [Renesas RX]の順に選択

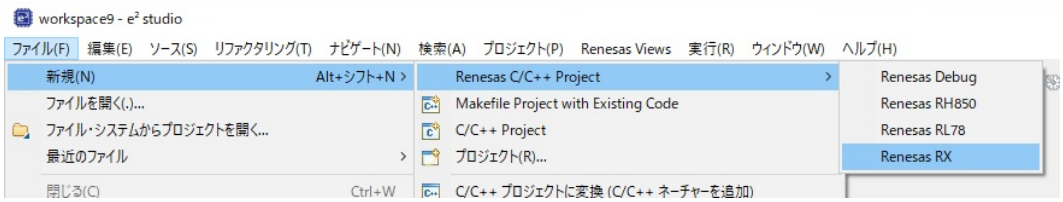


図 4-1 プロジェクトの種類とマイコンの選択

(2)　"Renesas CC-RX C/C++ Executable Project" テンプレートを選択し、[次へ]で次に進む

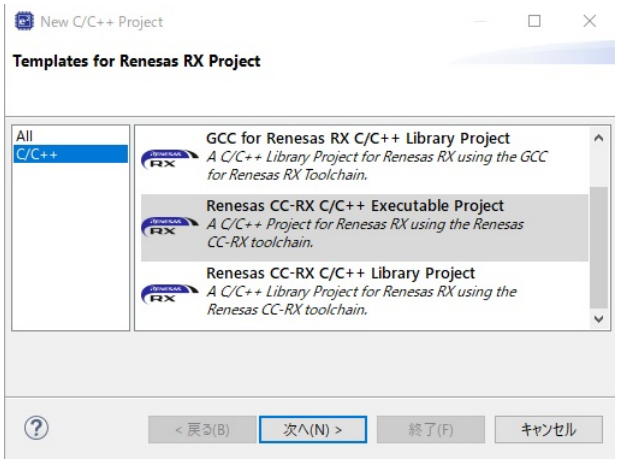


図 4-2 実行プロジェクト選択

(3)　　表示されたダイアログボックスでプロジェクト名を入力し、[次へ]をクリック

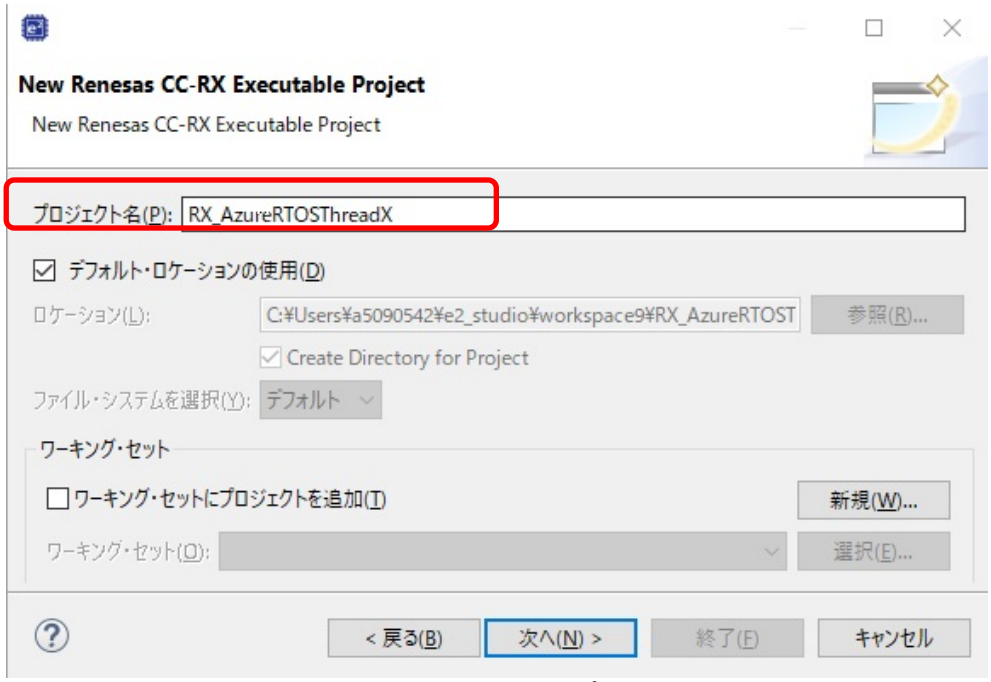


図 4-3 プロジェクト名入力

(4)　デバイス選択の画面で、デバイスとツールの情報を入力

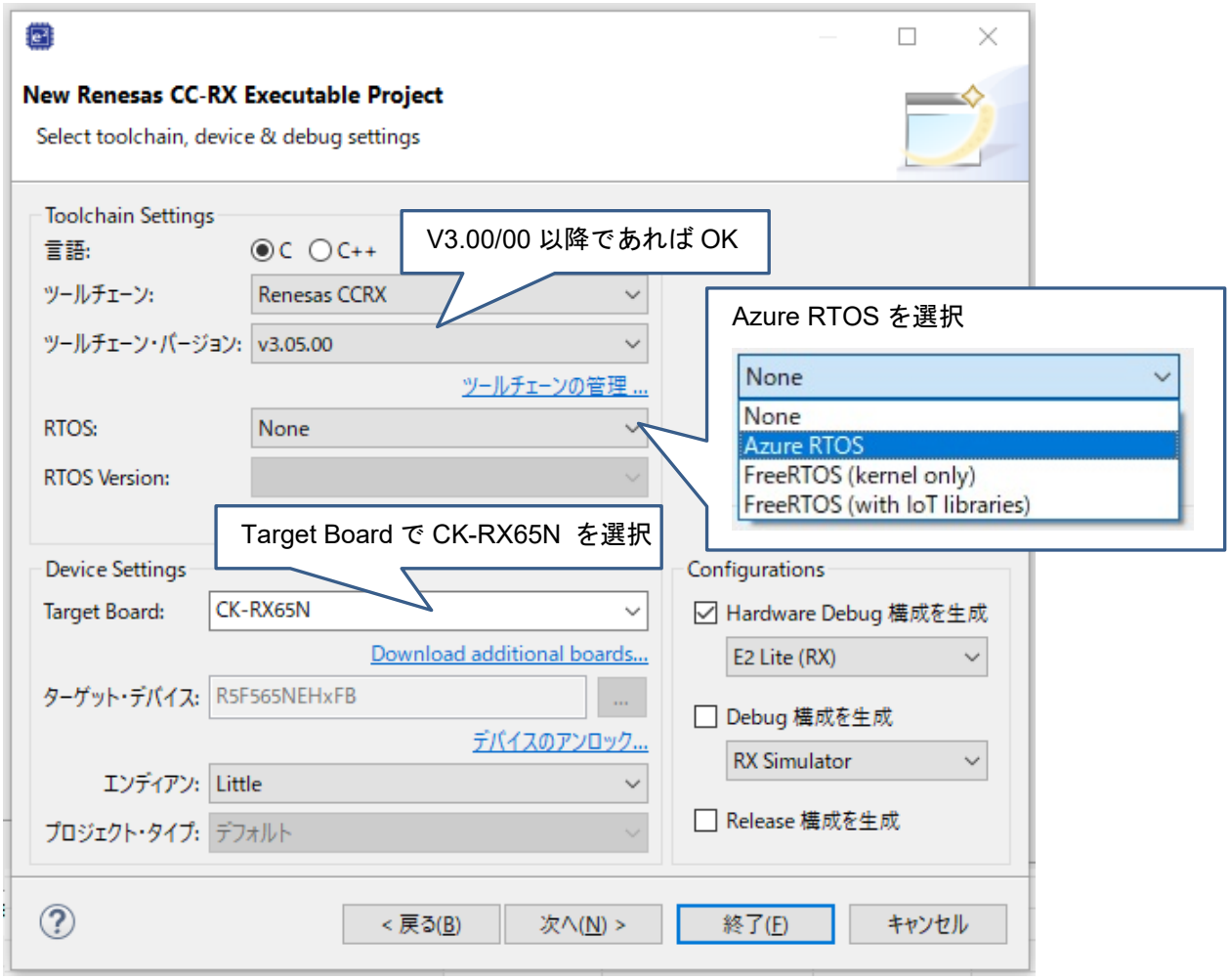


図 4-4 ボードと OS を選択

(5)　スマートコンフィグレータはデフォルトのまま使用

図 4-5 スマートコンフィグレータの設定

(6)　ThreadX sample project を選択

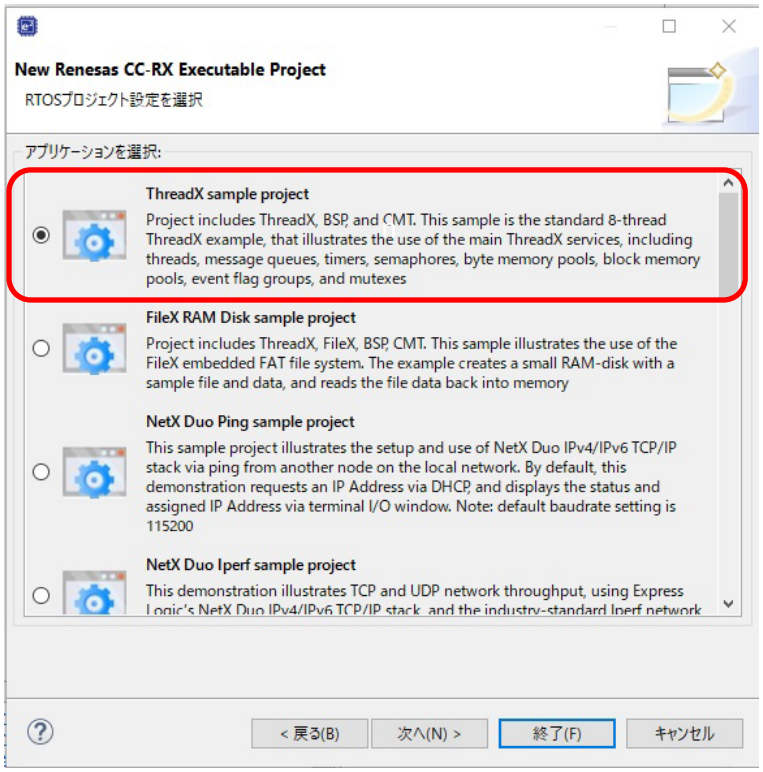


図 4-6 ThreadXsample project 選択

(7)　終了を押下してプロジェクト作成完了

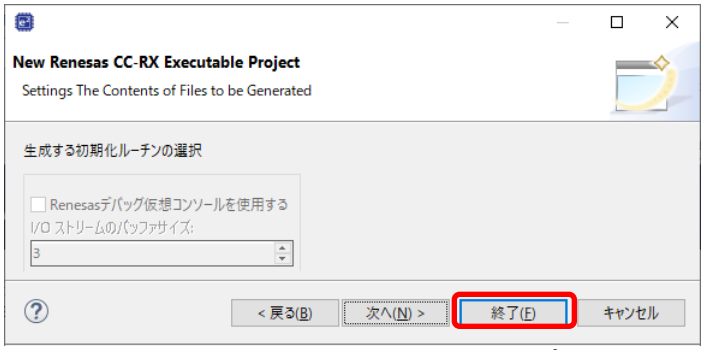


図 4-7　プロジェクト作成完了

(8)　カレントテキストエディターを選択

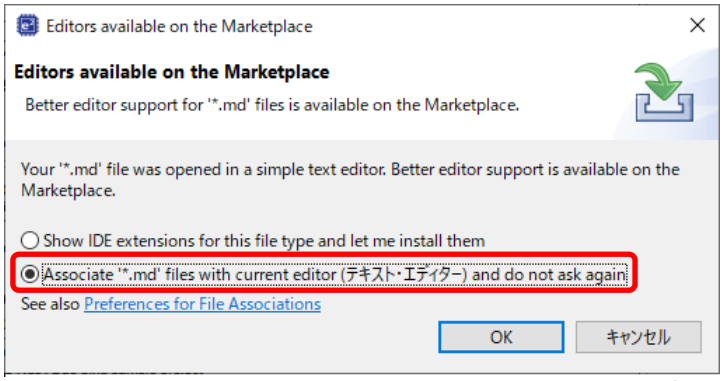


図 4-8　テキストエディタの選択

4.2　コンポーネントのダウンロード

RX_AzureRTOSThreadX.cfg をクリックして、ソフトウェアコンポーネントの設定を開きます。

r_cmt_rx が灰色の場合は、無効となっていますのでコンポーネントをダウンロードしてください

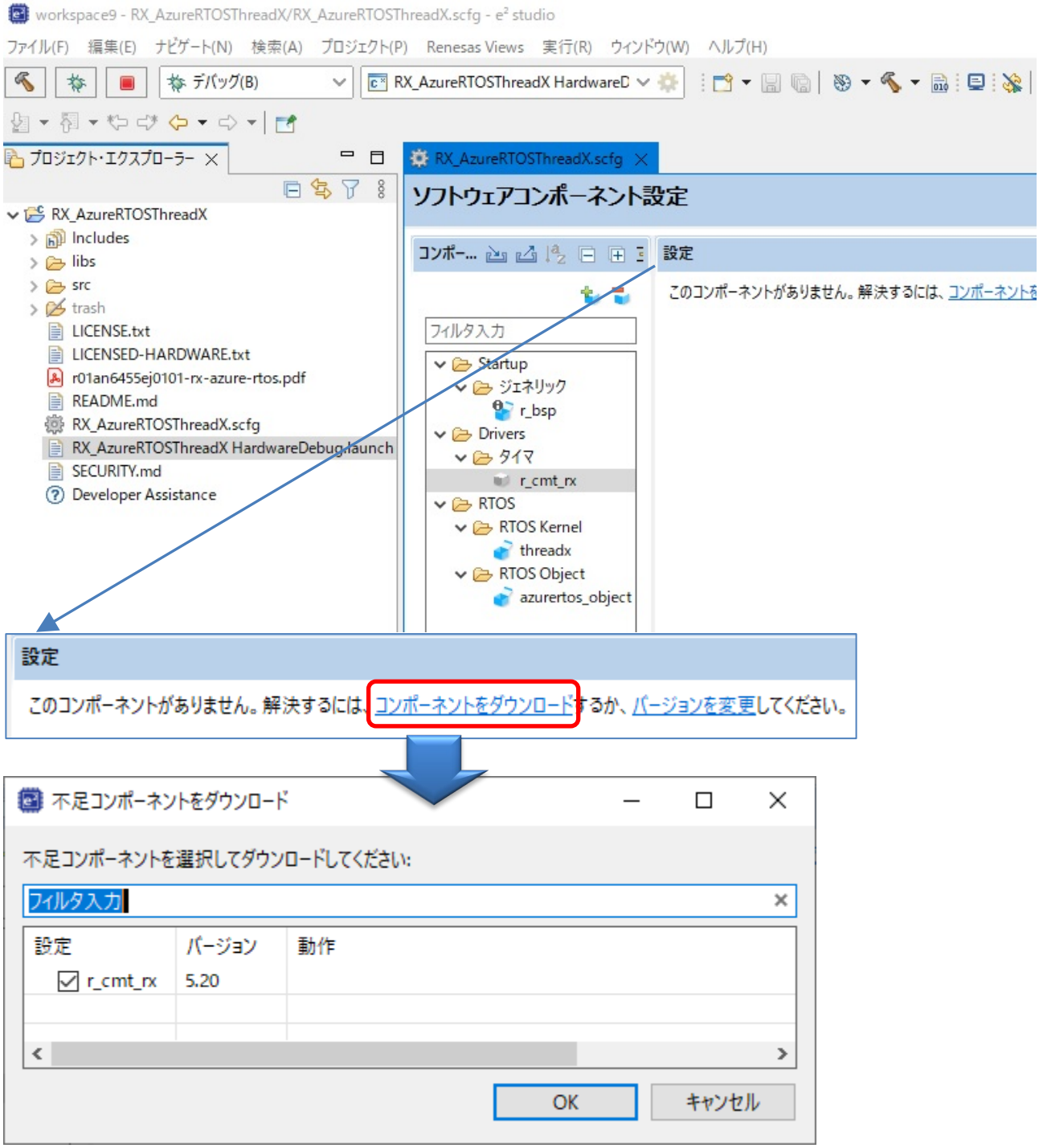


図 4-9 コンポーネントのダウンロード

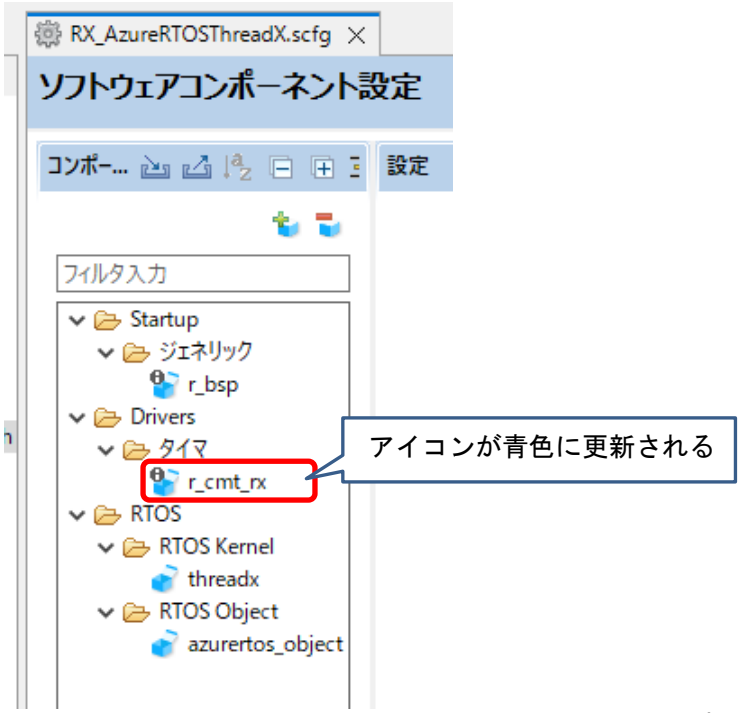


図 4-10 更新後のコンポーネント

概要タブでバージョンを確認します

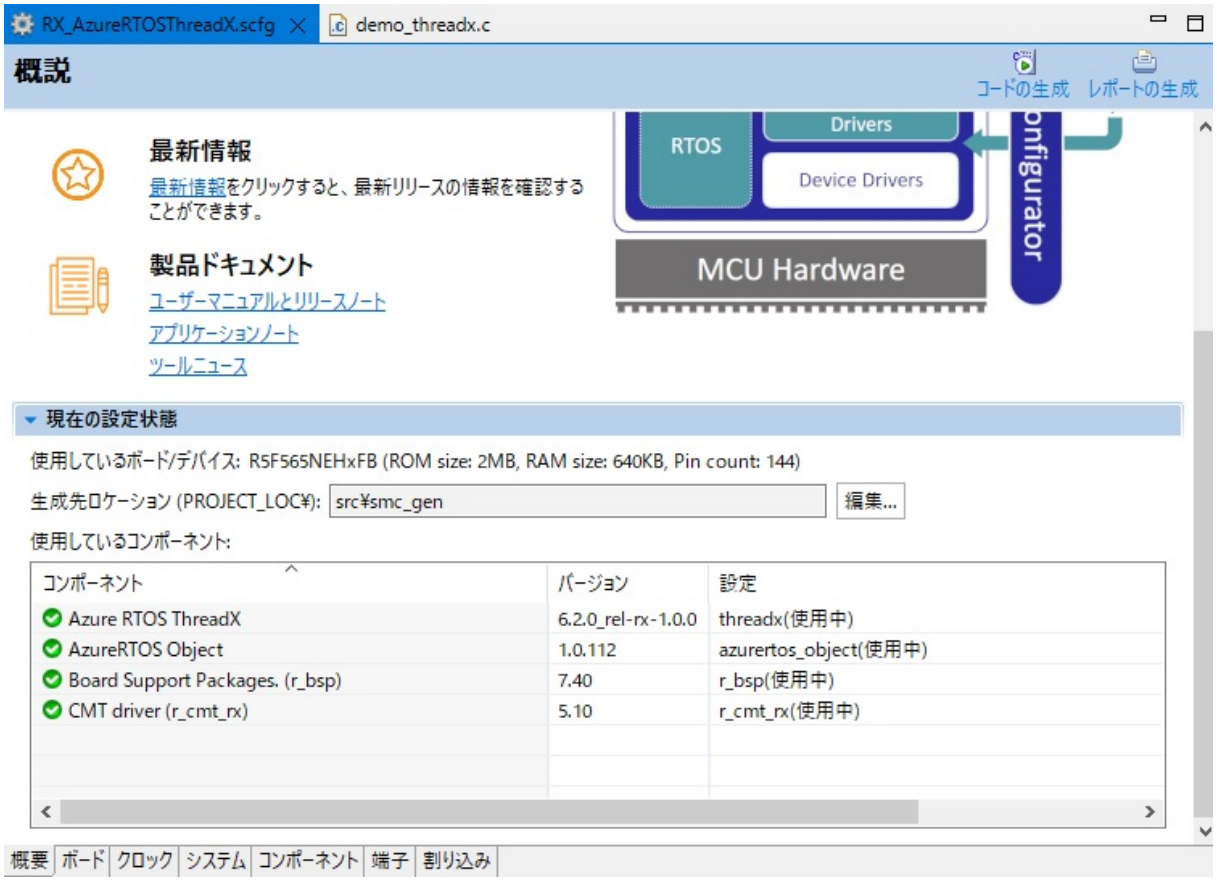


図 4-11 使用しているバージョンの確認

4.3　コンポーネントの設定

(1)　ThreadX のコンポーネント設定のプロパティで Trace イベントを有効にする

下図にある項目を Enable にします

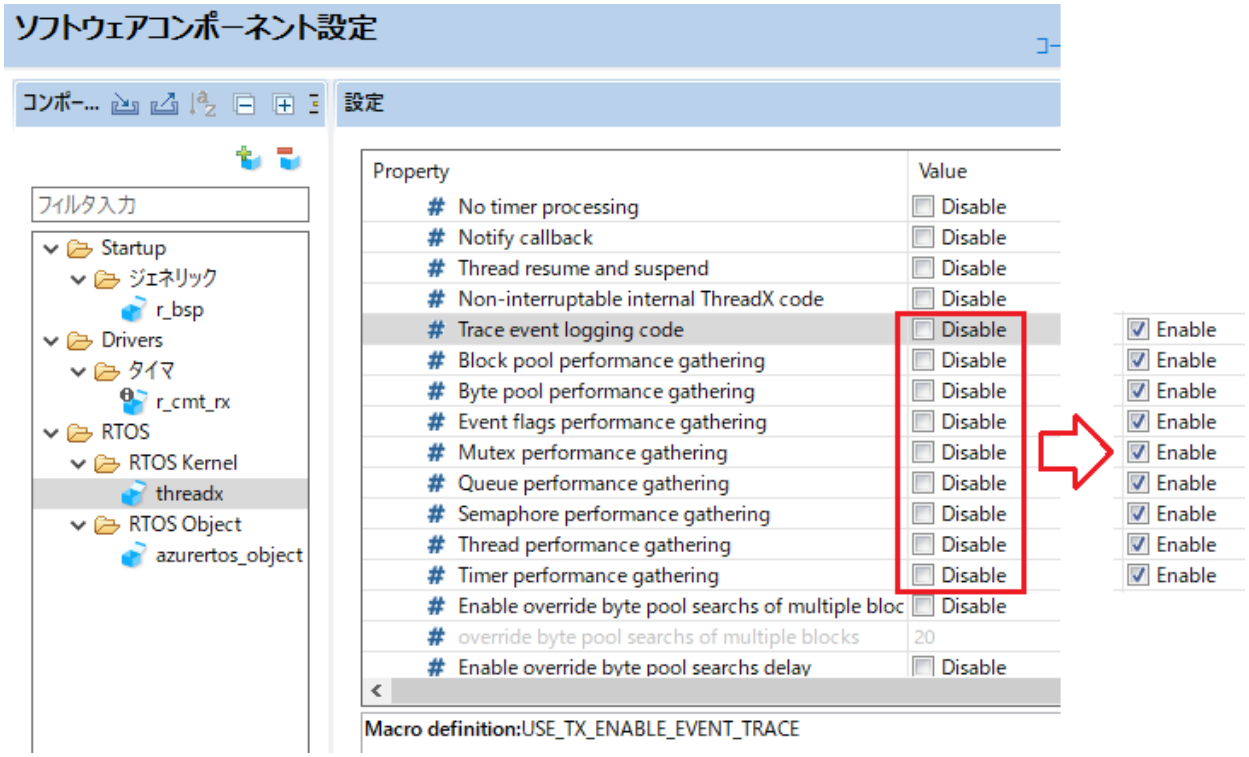


図 4-12 ThreadX のプロパティ設定

(2)　[ソフトウェアコンポーネントの選択] ダイアログボックスで「ポート」を追加

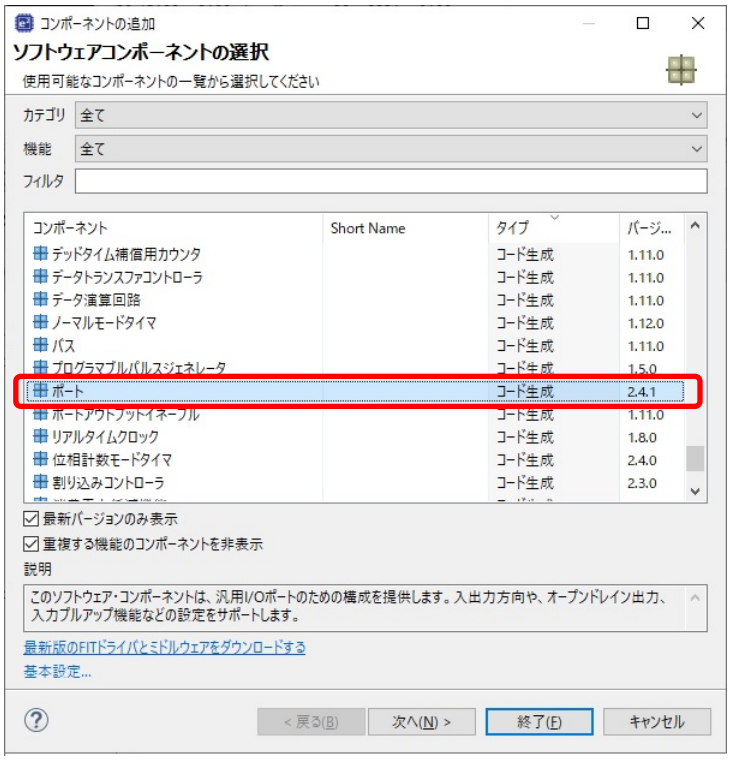
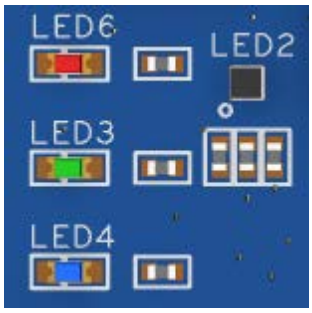


図 4-13 ポートの追加

(3)　ポートコンポーネントの設定

ボード上のLED6を制御するためスマート・コンフィグレータで該当ポートを設定する



シルク	LED色	接続ポート
LED2	カラーLED	P17, PA5, PA7
LED6	赤LED	P25
LED3	緑LED	P22
LED4	青LED	PA3

図 4-14 ボード上で制御する LED

LED6はP25に接続されています

LED6の接続図を示します。この回路では0出力で点灯するため初期値を1へ設定します

図 4-15 LED6 の回路図

(4)　PORT2 の設定

PORT2をチェックする

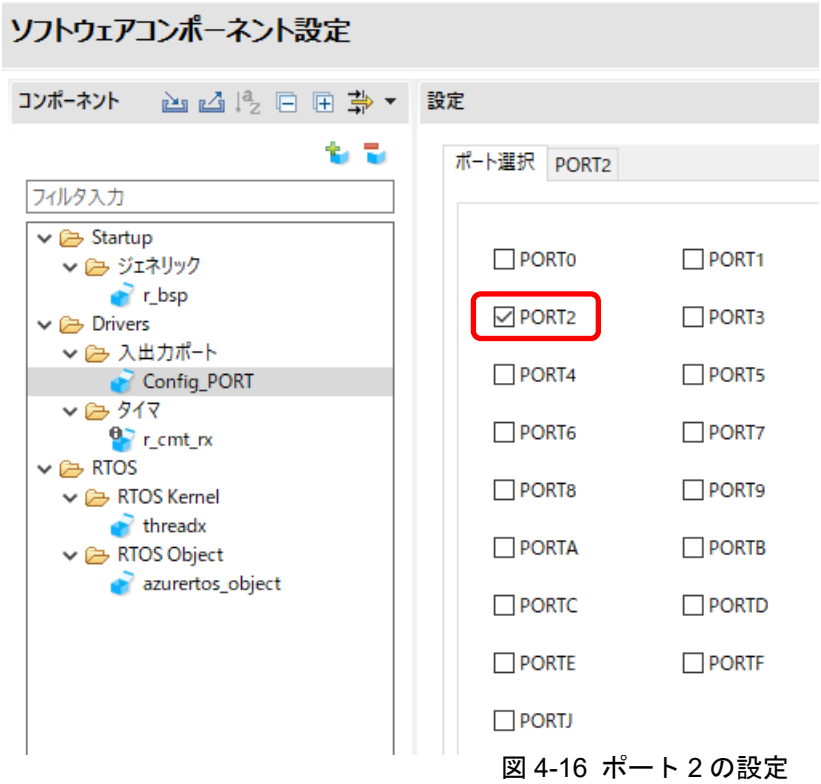


図 4-16 ポート 2 の設定

(5)　P25 を出力に設定

0 出力で点灯するため初期値を 1 を出力へ設定

図 4-17 P25 の設定

4.4　ソースファイルの生成

コードの生成を押下してソースファイルを生成します

図 4-18 設定項目をソースファイルへ反映

4.5　ソースファイルの編集

プロジェクトツリーから hardware_setup.c を開きます

図4-19 ソースファイルを開く

hardware_setup.c を開き下記の枠内を追記します

```
#include "r_smc_entry.h"
#include <platform.h>
#include <r_cmt_rx_if.h>
#include "tx_api.h"
#include "hardware_setup.h"

volatile uint8_t gTimer;

void _tx_timer_interrupt(void);

/* CMT Timer callback used as the system tick. */
void timer_callback(void * pdata)
{
    _tx_timer_interrupt();
    gTimer ^= 1;
    PORT2.PODR.BYTE = gTimer * 0x20;
}

void platform_setup(void)
{
    uint32_t chan;
    /* Create periodic timer for the system tick. */
    R_CMT_CreatePeriodic(TX_TIMER_TICKS_PER_SECOND, timer_callback, &chan);
}
```

タイマの割り込みハンドラ内の処理で gTimer の値を 0 と 1 に交互に切り替える。
P25 に接続された LED を点滅させる

tx_api.h を編集する

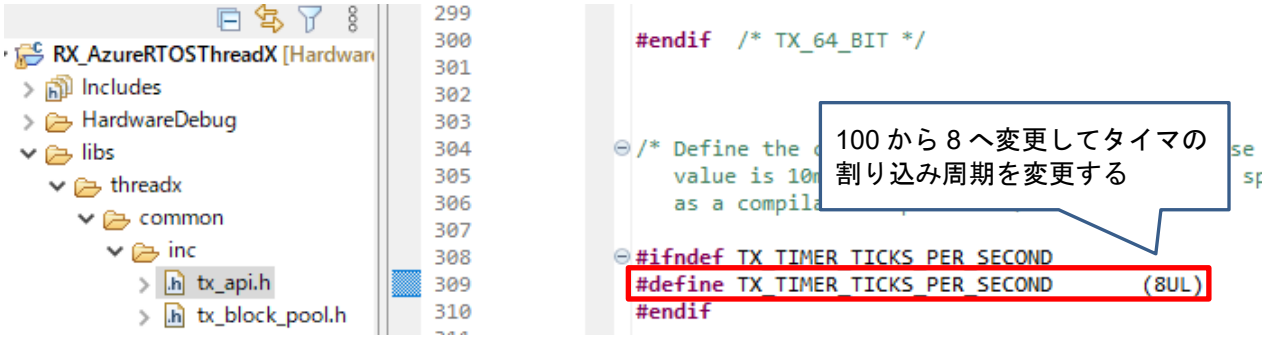


図 4-20 編集するファイルと変更内容

4.6　プロジェクトのビルド

プロジェクトのビルドを行います

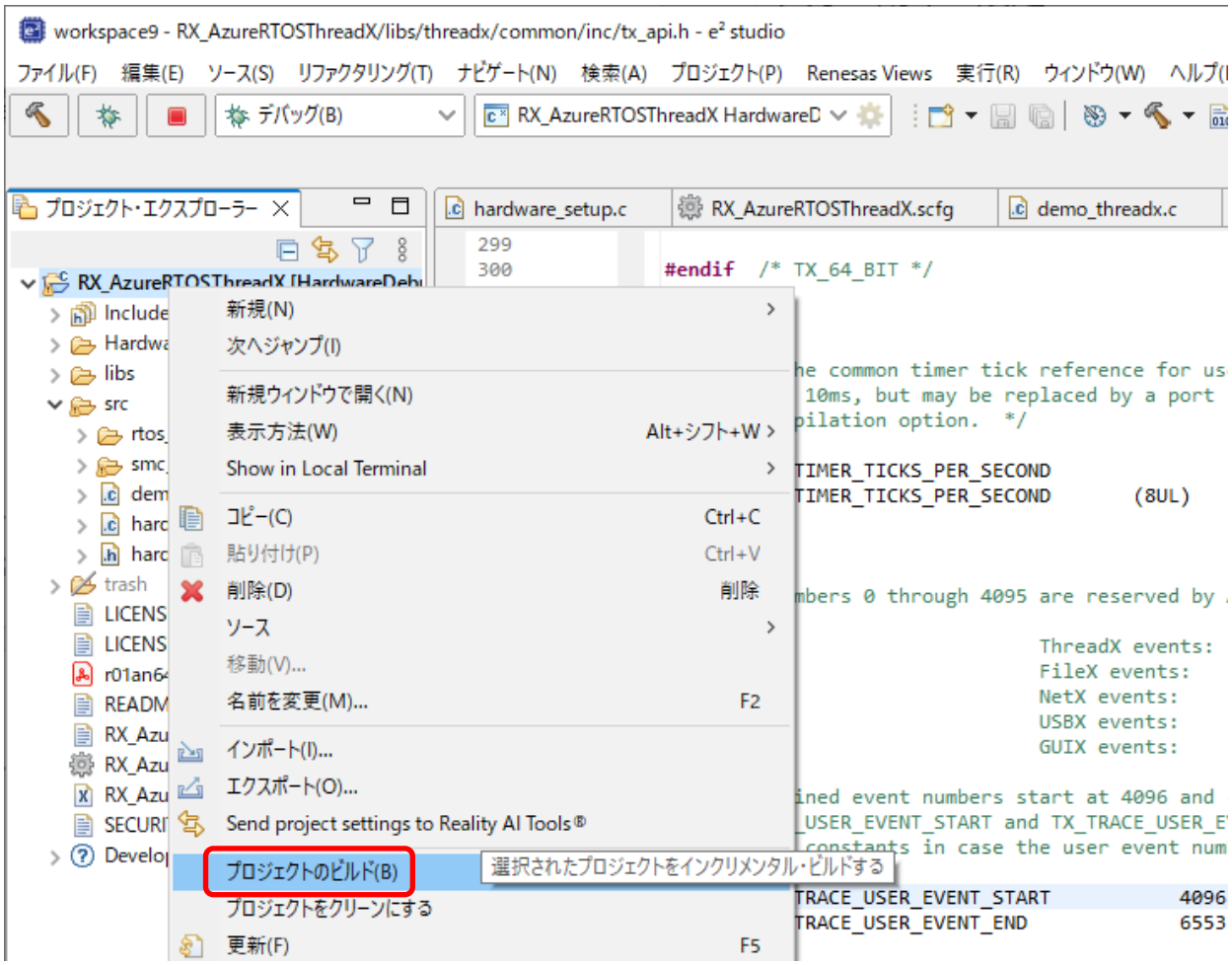


図 4-21　プロジェクトのビルド

ビルド終了し、エラーが発生していないことを確認する

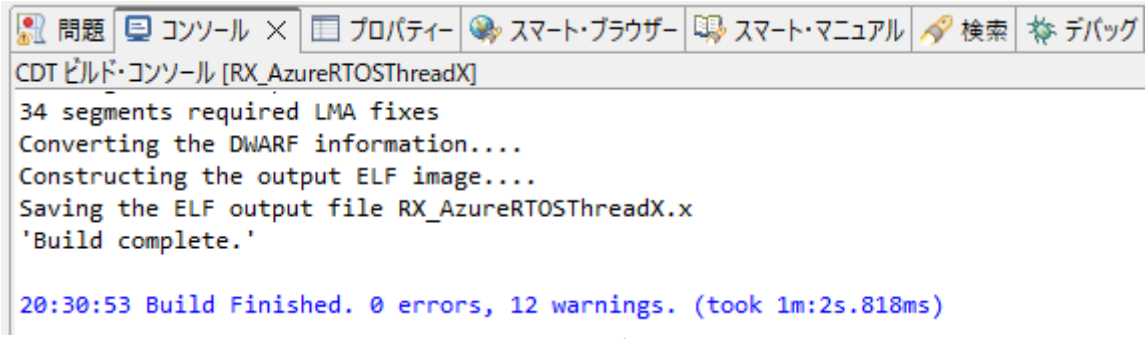


図 4-22　ビルド結果

4.7　プロジェクトの実行

デバッグの構成を選択して、デバッグを実行します

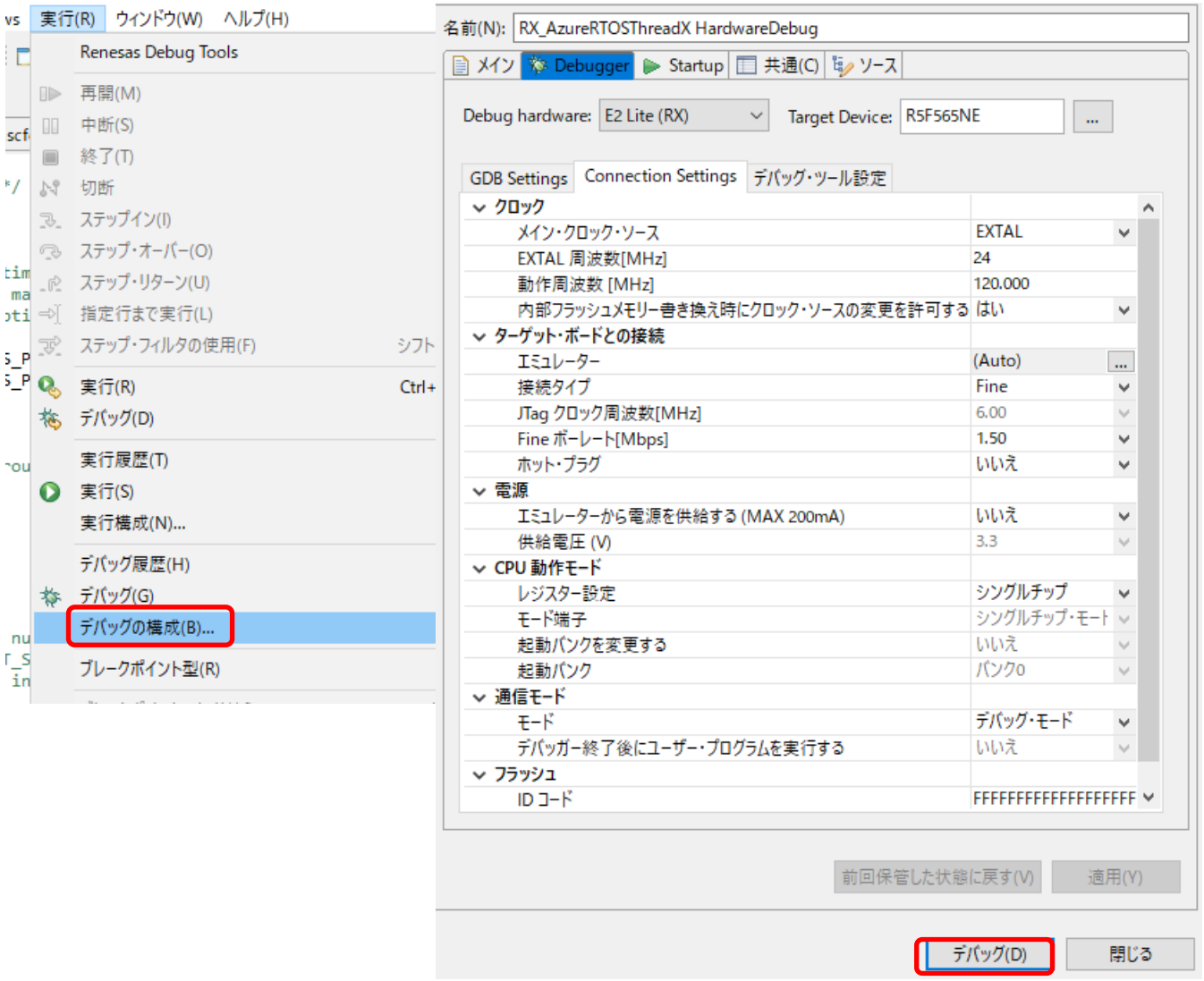


図 4-23 デバッグ構成の選択とプログラム実行

図 4-24　実行時のボード

5.　サンプルプロジェクトの解説

demo_thread.c は、8 スレッドの標準的な ThreadX のサンプルプロジェクトファイルです。このファイルには、スレッド、メッセージ キュー、タイマー、セマフォ、バイト メモリ プール、ブロック メモリ プール、イベント フラグ グループ、ミューテックスなど、ThreadX の主要なサービスの使用方法が記述されています。

e² studio ウィンドウ -> ビューの表示 -> アウトラインを選択

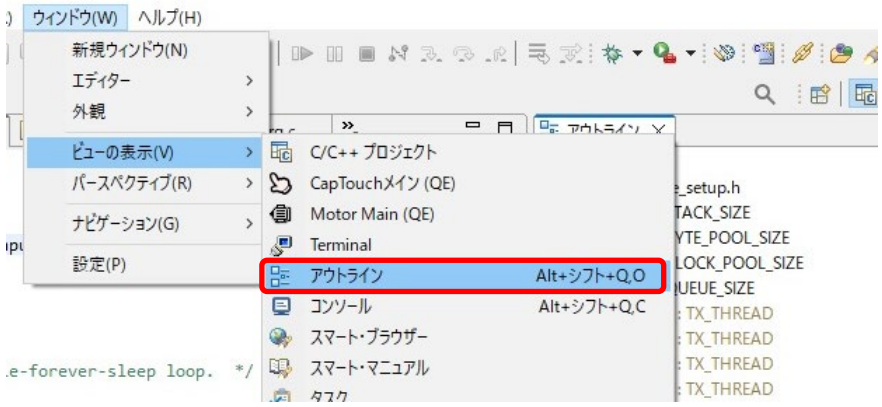


図 5-1 アウトラインの選択

demo_thread.cのアウトラインでスレッドを選択すると、選択したスレッドの関数へジャンプする

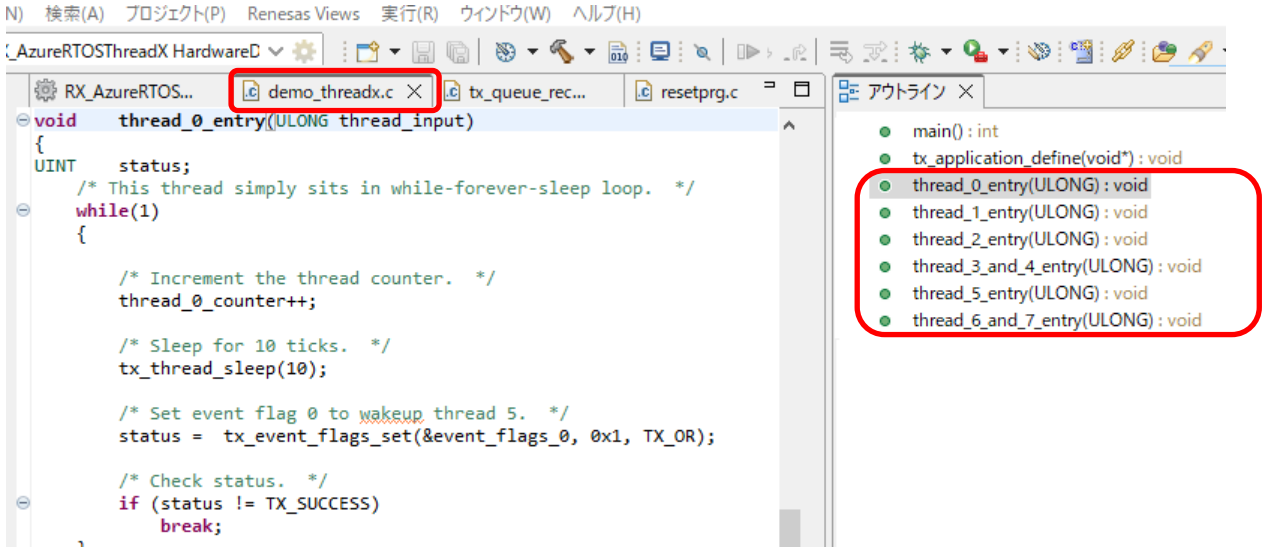


図 5-2 アウトラインに表示されるスレッド

関数の先頭に各スレッドの処理がコメントされています

スレッド番号	処理内容
0	このスレッドは単に無限ループを行います
1	このスレッドは、スレッド2が共有するキューにメッセージを送信するだけです
2	このスレッドは、スレッド1によってキューに入れられたメッセージを取得します
3	この関数は、スレッド3とスレッド4から実行されます。以下のループが示すように、
4	これらの関数は semaphore_0 の所有を巡って競争します。
5	このスレッドは、永久ループでイベントを待ちます
6	この関数は、スレッド6とスレッド7から実行されます。以下のループが示すように
7	これらの関数は mutex_0 の所有を巡って競争します。

表 5-1 サンプルプロジェクトのスレッド内容

6.　TraceX 用のバッファを設定

Azure RTOS TraceX は、組み込みターゲット上で実行されている ThreadX によって収集されたシステムイベント情報を表示します。
TraceX で解析を行うためには、プログラム中に解析用のバッファ設定が必要です。

```
tx_trace_enable (VOID *trace_buffer_start, ULONG trace_buffer_size, ULONG registry_entries);
```

を使用します。

このAPIにより、ThreadX 内のイベント トレースが有効になります。 アプリケーションでは、トレースバッファと ThreadX オブジェクトの最大数を指定します

demo_threadx.c を開きグローバル変数を追記します

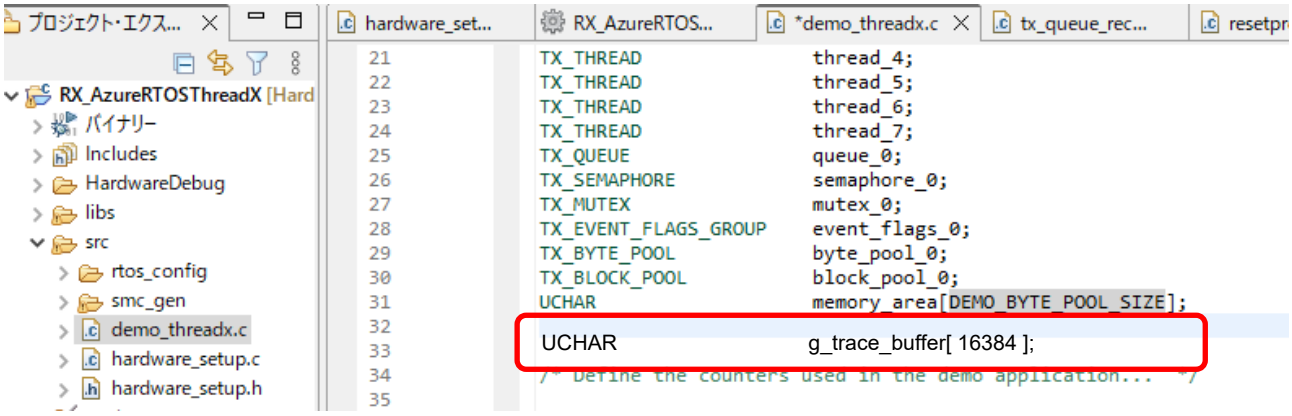


図 6-1 追記するグローバル変数

同じく demo_threadx.c の下記関数に追記します

```
void    tx_application_define(void *first_unused_memory)
{
    CHAR *pointer = TX_NULL;

    _tx_trace_enable( &g_trace_buffer, 16384, 30 );
```

図 6-2 追記する API 関数

※　この例ではバッファの大きさを16384にしていますが、32768などマイコンのRAM容量に空きがあれば大きくできます。

※　demo_threadx.c にグローバル変数とAPIを追加した後にビルドを行ってください。

詳細については、マイクロソフト社のWEBをご参照ください。
第 5 章 - トレース バッファーの生成 | Microsoft Learn

7. RTOS リソースビューの使用

e² studio には[RTOS リソース]ビュー機能があり、Azure RTOS ThreadX のリソースの状態を表示できます。この章では、[RTOS リソース]ビューの使用手順を説明します。

7.1 RTOS リソースビューの表示

RTOSリソース]ビュー機能が使用できるのはデバッガ実行中のみです。デバッガを起動してから[Renesas Views] → [パートナーOS] → [RTOS リソース]の順に選択します。[OS選択]ダイアログボックスが表示されたら図11のように"ThreadX"を選択してください。図12のように[RTOSリソース]ビューが開きます。

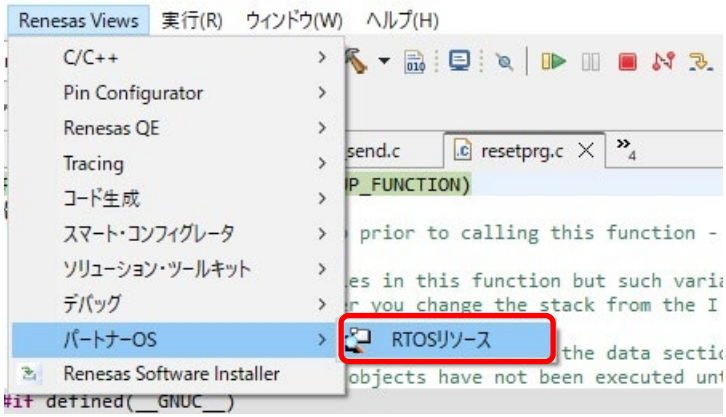


図 7-1 RTOS リソースの選択

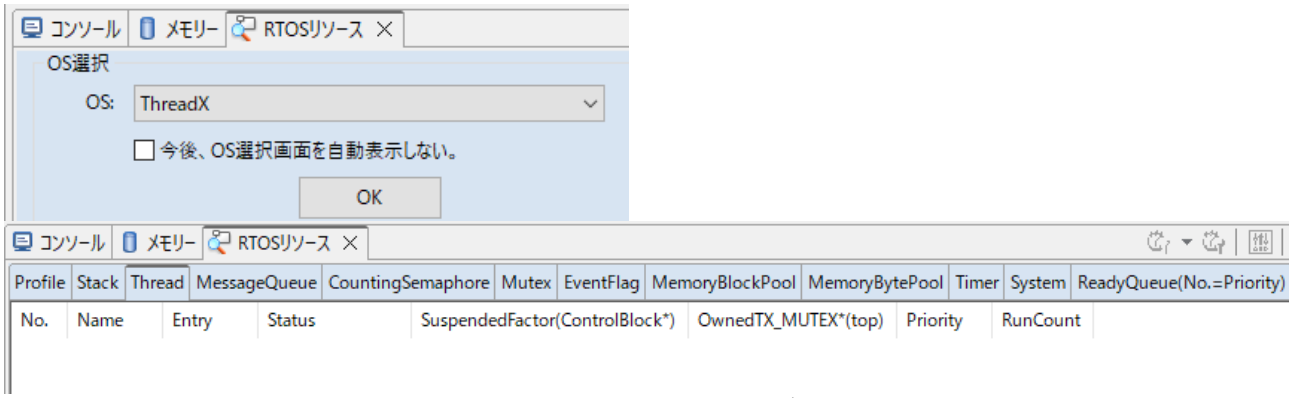


図 7-2 OS の選択と RTOS リソースビュー

7.2　コンテキストメニュー

RTOSリソース]ビュー上でマウスの右ボタンをクリックすると、コンテキストメニューが開きます。

図 7-3 コンテキストメニュー

- **リアルタイム・リフレッシュカラム**

表示されている項目の内容をリアルタイムに更新できます。
このメニューは、プログラム実行中はグレー表示で選択できません。

- **リアルタイム・リフレッシュ間隔**

リアルタイム表示を更新する間隔を指定します。設定できる間隔は500 msから10000 msの範囲です。
このメニューは、プログラム実行中はグレー表示で選択できません

- **スタック設定**

スタックデータロードを有効または無効にし、スタック警告のしきい値を設定します。
このメニューは、プログラム実行中はグレー表示で選択できません

- **最新の情報へ更新**

表示を最新の情報に更新します。

- **ソースへジャンプ**

[エディタ]ビューを開き、タスク/スレッドまたはハンドラのソースコードを表示します。[エディタ]ビューは、タスク/スレッドまたはハンドラをダブルクリックしても開くことができます。
このメニューは、プログラム実行中はグレー表示で選択できません

- **ファイルへ保存**

現在選択されているタブのデータをテキストファイル（*.txt）に保存します。
このメニューは、プログラム実行中はグレー表示で選択できません

- **OSの選択**

[OS選択]ダイアログボックスを開きます。
このメニューは、プログラム実行中はグレー表示で選択できません

7.3　スタック設定

スタックデータロードの有効化とスタックしきい値の設定します

a. コンテキストメニューを開き、[スタック設定]を選択します。

b. スタックデータを[RTOSリソース]ビューにロードするには、[スタック設定]ダイアログボックスの"スタックデータロードを有効にする"のチェックボックスを選択します。これを有効にしていないと、次回のデバッグセッションでスタックデータがロードされません。

c. 任意のスタックしきい値を[スタックしきい値(%)]テキストボックスに設定できます。[OK]で設定を保存します。ここでは 80% に設定します。

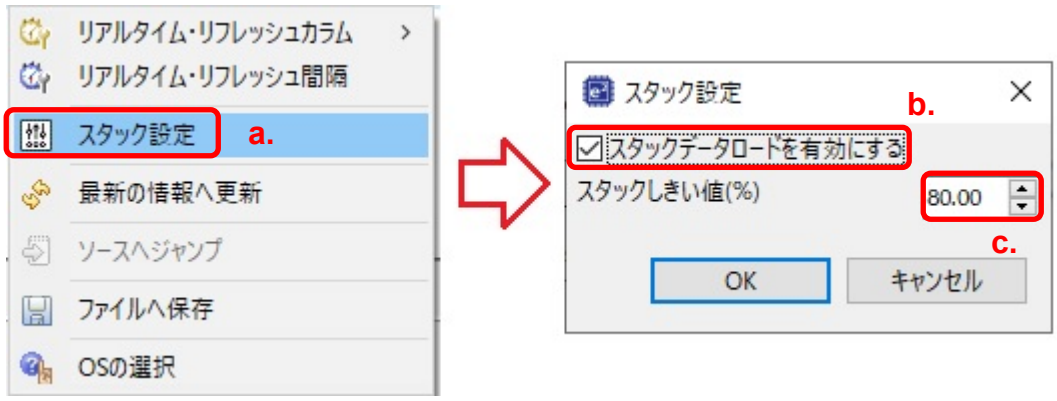


図 7-4 スタック設定

7.4　タブメニュー

各タブに表示する項目を表1に示します。

表 7-1 各タブに表示する内容

RTOSビューのタブの名称	表示する情報と選択項目の名称	表示内容
Profile	[Profile]タブの機能は、 現在使用できません	ー
Stack	Name	スレッド名
	Entry	各スレッドを開始した関数名
	StackPointer	現在のスタックポインタ
	StackStart	スタックの開始アドレス
	StackEnd	スタックの終了アドレス
	StackSize(bytes)	スタックサイズ
	MaxStackUsage(bytes)	現在使用中の最大スタックサイズ
Thread	Name	スレッド名
	Entry	各スレッドを開始した関数名
	Status	スレッドの状態
	Suspended Factor (Control Block*)	中断の要因となったリソース
	OwnedTX_MUTEX*(top)	獲得した上位のミューテックス
	Priority	優先度
	RunCount	スレッドを実行した回数
MessageQueue	Name	メッセージキュー名
	UsedCount	使用中のメッセージキュー数
	FreeCount	使用可能なメッセージキュー数
	TotalCount	メッセージキューの総数
	MessageSize	メッセージサイズ
	SuspendedTX_THREAD*(top)	キューに待機中の先頭スレッド
	SuspendedCount	中断中のスレッド数
	StartAddress	メッセージキューの開始アドレス
	EndAddress	メッセージキューの終了アドレス
CountingSemaphore	Name	セマフォ名
	SemaphoreCount	セマフォ数
	SuspendedTX_THREAD*(top)	キューに待機中の先頭スレッド
	SuspendedCount	中断中のスレッド数
Mutex	Name	ミューテックス名
	OwnerTX_THREAD*	獲得しているスレッド
	OwnerCount	所有者の数

	SuspendedTX_THREAD*(top)	キューに待機中の先頭スレッド
	SuspendedCount	中断中のスレッド数
EventFlag	Name	イベントフラグ名
	Flag	現在のフラグパターン
	SuspendedTX_THREAD*(top)	キューに待機中の先頭スレッド
	SuspendedCount	中断中のスレッド数
MemoryBlockPool	Name	メモリブロック名
	FreeCount	使用可能なブロック数
	TotalCount	ブロックの総数
	BlockSize(bytes)	ブロックサイズ
	TotalSize(bytes)	メモリブロックプールの合計サイズ
	SuspendedTX_THREAD*(top)	キューに待機中の先頭スレッド
	SuspendedCount	中断中のスレッド数
	StartAddress	メモリブロックプールの先頭アドレス
MemoryBytePool	Name	メモリプール名
	Free(bytes)	使用可能なバイト数
	Total(bytes)	メモリバイトプールの合計サイズ
	FragmentCount	フラグメント数
	SuspendedTX_THREAD*(top)	キューに待機中の先頭スレッド
	SuspendedCount	中断中のスレッド数
	StartAddress	メモリバイトプールの先頭アドレス
Timer	Name	タイマ名
	Remaining Tick	残り時間
	Re-initialization Tick	サイクル時間
System	SystemClock	システムクロック
ReadyQueue(No. = Priority)	QueuedTX_THREAD*(top)	上位の使用可能なスレッド

8. Azure RTOS TraceX を使用したプロジェクトデバッグの開始

メニューから[実行] → [デバッグ]の順に選択してデバッガを起動します。
以下の手順で、Azure RTOS TraceXのデータファイルを作成します。

8.1 TraceX データバッファアドレスの取得

”demo_threadx.c”の”g_trace_buffer”を”監視式を追加”で追加し、Azure RTOS TraceXのデータバッファアドレスを取得します。下図の例では、アドレスは 0x736c です。

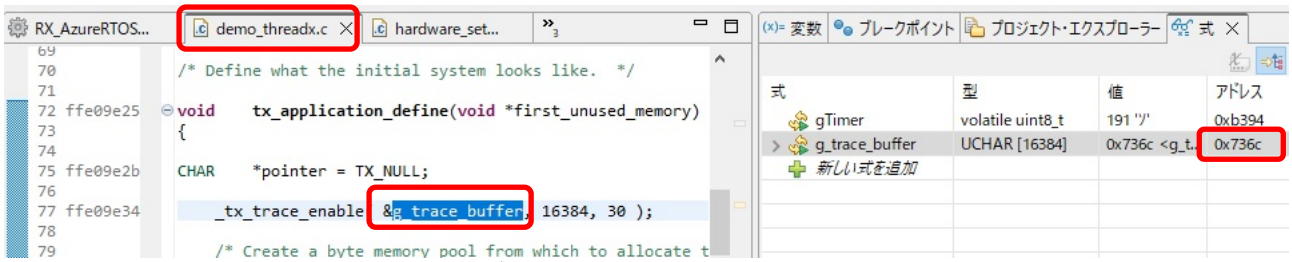


図 8-1 データバッファアドレスの取得

8.2 TraceX データファイルのエクスポート

TraceXのデータファイルをエクスポートします

プログラムを実行し、停止させる

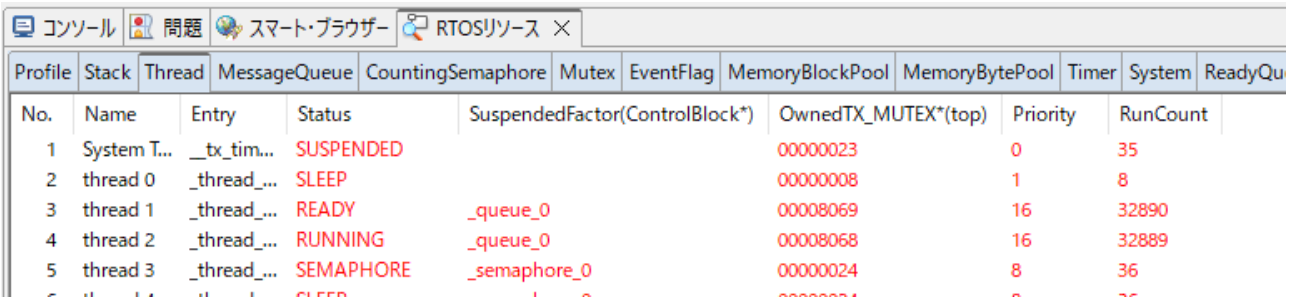


図 8-2 Thread に情報が表示される

a, ウィンドウ->ビューの表示からメモリーを選択

b.[モニター・メモリー]ダイアログボックスに、1.で取得したAzure RTOS TraceXのデータバッファアドレスを入力

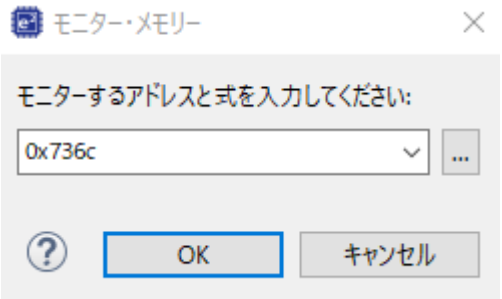


図 8-3 モニターするアドレスの入力

c. [メモリー]タブの右にある[エクスポート]ボタンをクリック
d. [メモリーをエクスポート]ダイアログボックスが開く
e. “Format”に”RAW Binary”を選択
f. "Start address"に、1.で取得したTraceXデータバッファアドレスを入力
g. "Length"に、TraceXデータバッファ"g_trace_buffer"のサイズとして16384※を入力
h. "File name"に、拡張子.trxの任意のファイル名を指定
i. [OK]をクリック(TraceXのデータファイルがh.で指定したファイル名でエクスポートされる)

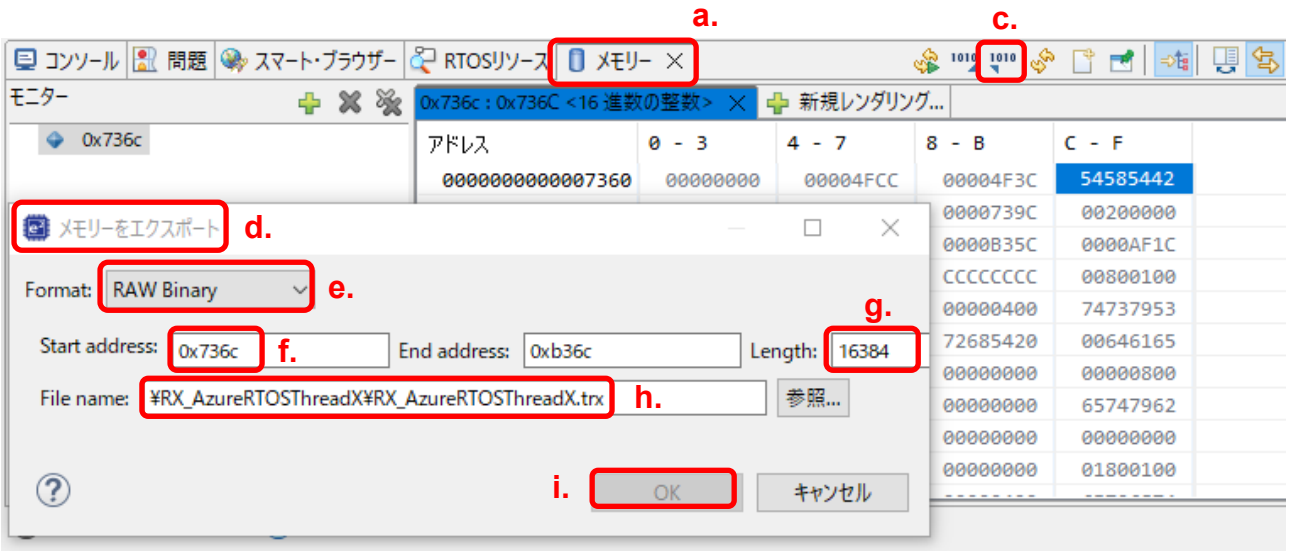


図 8-4 データファイルのエクスポート

※　6章のTraceX用のデータ設定で指定したサイズを指定してください

9.　Azure RTOS TraceX の起動

PCにインストールしたAzure RTOS TraceXを起動します。
Azure RTOS TraceXのメニューから[File] → [Open]の順にクリックします。
8.2章の手順でエクスポートしたTraceXデータファイル"****.trx"を選択します。
TraceX上にトレース情報が表示されます。

9.1　トレース情報の表示

トレース情報の表示モードは次の2種類があります。

- シーケンシャルビュー（Sequential View）モード
- タイムビュー（Time View）モード

9.1.1　シーケンシャルビューモード

シーケンシャルビューは、イベント間の経過時間に関係なくイベントが相互に連結されて表示されます
システムの動作の概要を把握するのに役立ちます。

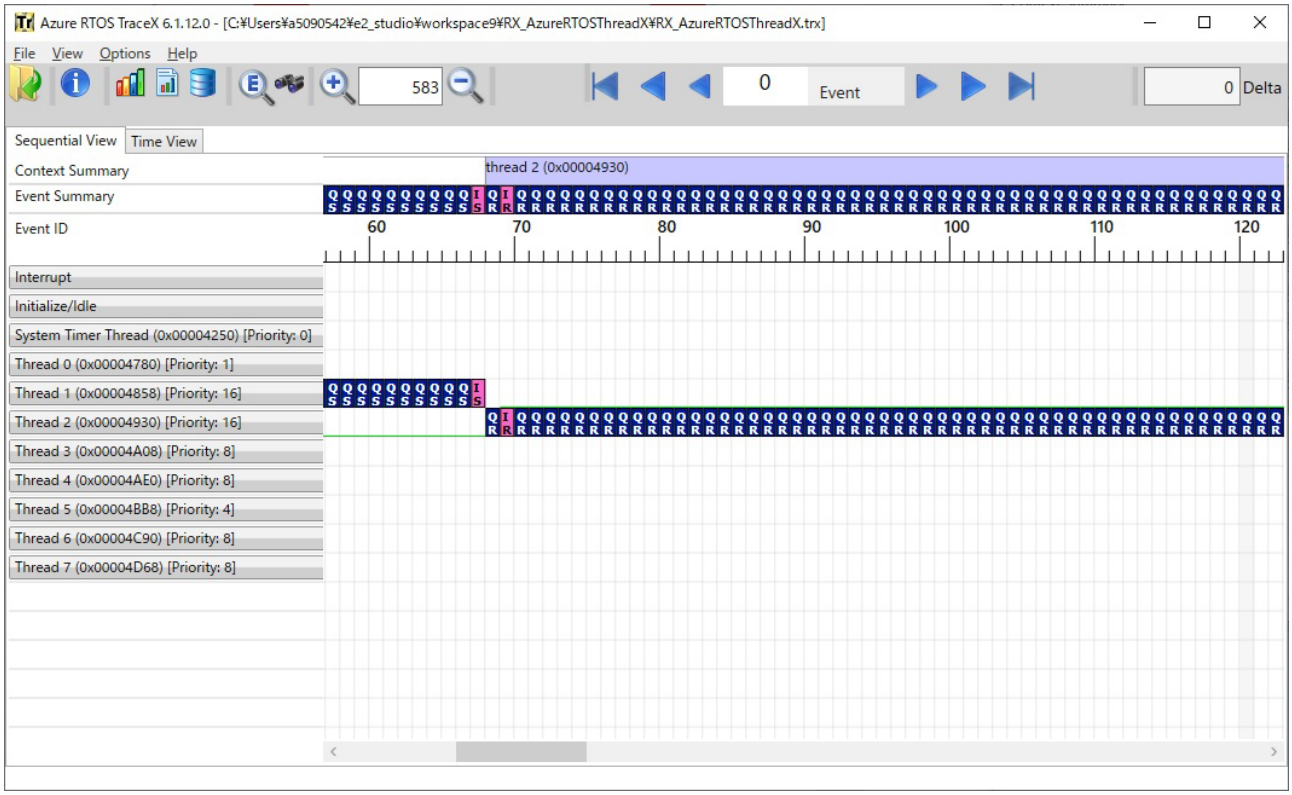


図 9-1 シーケンシャルビューの表示

9.1.2　時間ビューモード

時間ビューモードは、システム内のどこで大量の処理が行われているかを確認するのに便利です。
システムをチューニングしてパフォーマンスや応答性を向上させるのに役立ちます。

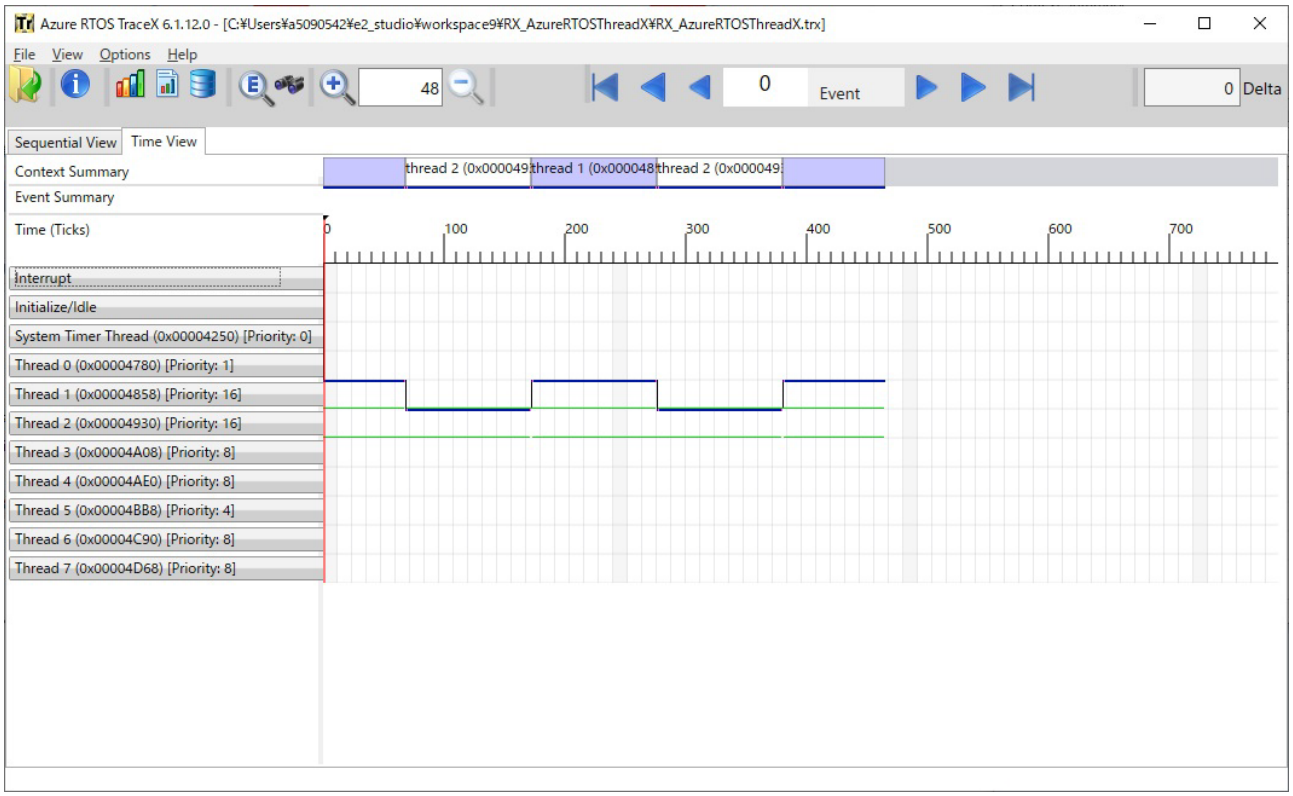


図 9-2 時間ビューの表示

9.2　TraceX の分析機能

分析機能の詳細は、[Help] → [Manual]の順に選択して情報を参照してください。

TraceX ユーザガイド | Microsoft Docsの「第3章 Azure RTOS TraceXの説明」の以下の説明も参照してください。

- シーケンシャルビュー（Sequential View）モード
- タイムビュー（Time View）モード

TraceX ユーザガイド | Microsoft Docsの「第4章 Azure RTOS TraceXのパフォーマンス分析」の以下の説明も参照してください。

- 実行プロファイル（Execution Profile）
- 人気のあるサービス（Popular Services）
- スレッド スタックの使用状況（Thread Stack Usage）
- パフォーマンス統計（Performance Statistics）
- FileX 統計（FileX Statistics）
- NetX 統計（NetX Statistics）
- トレース ファイル情報（Trace File Information）

ホームページとサポート窓口

RA ファミリの主な機能、ダウンロードコンポーネント、関連ドキュメント、サポートについては、下記のサイトにアクセスしてください。

RX ファミリ製品情報	www.renesas.com/rx
RX ファミリサポートフォーラム	www.renesas.com/rx/forum
ルネサスサポートサイト	www.renesas.com/support

改訂記録

Rev.	発行日	改訂内容	
		ページ	ポイント
1.00	Mar.1.23	—	初版発行

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。