AP-RZT2-0A (RZ/T2M CPU BOARD)
DualCore サンプルプログラム解説

第 1.2 版　2023 年 10 月 02 日

1.　概要

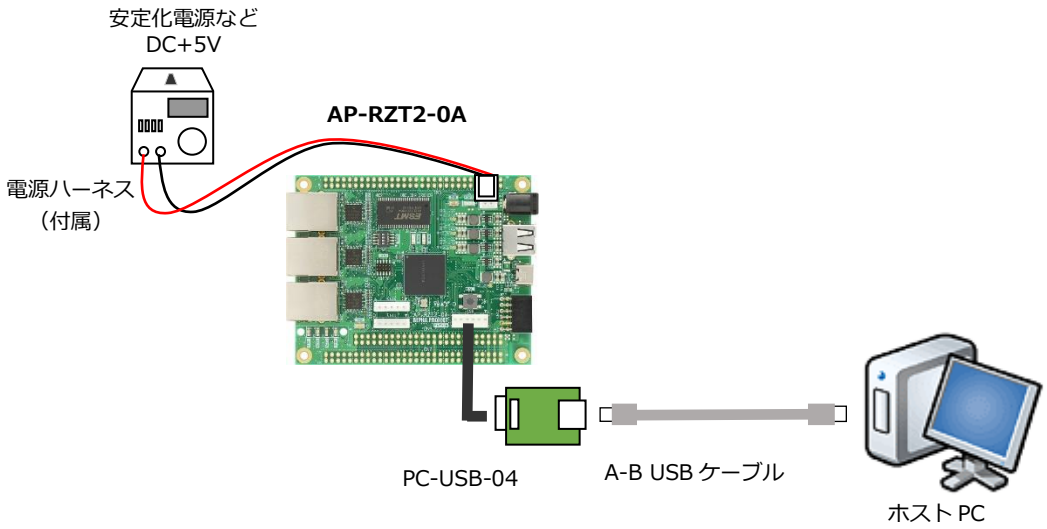
1.1　概要

本アプリケーションノートでは、AP-RZT2-0A に付属する「DualCore サンプルプログラム」について解説します。
解説するサンプルプログラムは下記のものになります。

サンプルプログラム	動作内容
AP-RZT2-0A DualCore サンプルプログラム	DualCore 動作

1.2　接続概要

「DualCore サンプルプログラム」の動作を確認する上で必要な CPU ボードとホスト PC 間の接続例を以下に示します。
詳細な接続に関しては後述の「３．動作説明」を参照してください。



1.3　本サンプルプログラムについて

本サンプルプログラムおよび本書含むアプリケーションノートは、弊社 Web サイトのボード紹介ページで公開されています。

株式会社アルファプロジェクト

AP-RZT2-0A 製品ページ　https://www.apnet.co.jp/product/rza/ap-rzt2-0a.html

1.4　開発環境について

本サンプルプログラムは統合開発環境「e2 studio」と「FSP」を用いて開発されています。
本サンプルプログラムに対応する開発環境、FSP、コンパイラ、デバッガのバージョンは次の通りです。

ソフトウェア	バージョン	備考
e2studio	2023-01	–
GCC for Renesas RZ	9.3.1.20200408	–
FSP	1.2.0	Flexible Support Package for Renesas RZ/T シリーズ

デバッガ	ハードウェアバージョン	備考
J-Link	V11	Segger Microcontroller Systems 社 ハードウェアバージョン V10 以下はご使用になれませんのでご注意ください。

※AP-RZT2-0A と J-Link を直接接続することはできません。
AP-RZT2-0A 側(ハーフピッチコネクタ)と J-Link 側(フルピッチコネクタ)を接続するための変換アダプタが必要となります。
変換アダプタについては、J-Link 取扱店へご確認ください。

1.5　ワークスペースについて

本サンプルプログラムのプロジェクトファイルは次のフォルダに格納されています。
ご使用のワークスペースにコピーして使用してください。

サンプルプログラム	フォルダ
DualCore サンプルプログラム プロジェクトフォルダ（CPU0）	¥sample¥ap_rzt2_0a_dualcore_sample_CPU0
DualCore サンプルプログラム プロジェクトフォルダ（CPU1）	¥sample¥ap_rzt2_0a_dualcore_sample_CPU1

2.　サンプルプログラムの構成

2.1　フォルダ構成

本サンプルプログラムは以下のフォルダで構成されています。

¥ sample	AP-RZT2-0A サンプルプログラムフォルダ
├─¥ ap_rzt2_0a_dualcore_sample_CPU0	DualCore サンプルプログラムフォルダ（CPU0）
│　├─¥ .settings	設定ファイルフォルダ
│　├─¥ Debug	デバッグビルド用フォルダ
│　├─¥ Release	リリースビルド用フォルダ
│　├─¥ script	スクリプト用フォルダ
│　└─¥ src	ソースファイル用フォルダ
└─¥ ap_rzt2_0a_dualcore_sample_CPU1	DualCore サンプルプログラムフォルダ（CPU1）
├─¥ .settings	設定ファイルフォルダ
├─¥ Debug	デバッグビルド用フォルダ
├─¥ Release	リリースビルド用フォルダ
├─¥ script	スクリプト用フォルダ
└─¥ src	ソースファイル用フォルダ

2.2　ファイルの構成

本サンプルプログラムは以下のファイルで構成されています。
本節では、サンプルプログラムの作成にあたって追加したファイルについて記述し、自動生成ファイルなどに関しては説明を省略します。

・共通ファイル

<¥sample フォルダ内>

AlphaProject.ap_rzt2_0a.1.2.0.pack	･･･	AP-RZT2-0A パックファイル
ap_rzt2_0a_devsetup.elf.srec	･･･	AP-RZT2-0A Flash 書き込み用モトローラファイル
ap_rzt2_0a_ssbl_sample.bin	･･･	SSBL（シリアル FlashROM ブート用）プログラム
FlashROM_Write(DualCore).bat	･･･	プログラム書き込み用バッチファイル

2.2.1　DualCore サンプルプログラムのファイル構成（CPU0）

<¥sample¥ap_rzt2_0a_dualcore_sample_CPU0¥フォルダ内>

.cproject	･･･	CPROJECT ファイル
.project	･･･	PROJECT ファイル
ap_rzt2_0a.pincfg	･･･	AP-RZT2-0A ピンコンフィグファイル
ap_rzt2_0a_dualcore_sample_CPU0 Debug.jlink	･･･	AP-RZT2-0A DualCore サンプルプログラム J-Link 設定ファイル(RAM デバッグ用)
ap_rzt2_0a_dualcore_sample_CPU0 Debug.launch	･･･	AP-RZT2-0A DualCore サンプルプログラム デバッグおよびランタイム設定ファイル(RAM デバッグ用)
configuration.xml	･･･	FSP コンフィギュレータファイル

<¥sample¥ap_rzt2_0a_dualcore_sample_CPU0¥script フォルダ内>

fsp_ram_execution.ld	･･･	RAM 実行用リンカスクリプトファイル
fsp_rom_execution.ld	･･･	ROM 実行用リンカスクリプトファイル

<¥sample¥ap_rzt2_0a_dualcore_sample_CPU0¥src フォルダ内>

ap_rzt2_xa_xspi.c	･･･	XSPI 通信用ソースファイル
ap_rzt2_xa_xspi.h	･･･	XSPI 通信用ヘッダファイル
hal_entry.c	･･･	アプリケーションソースファイル

2.2.2　DualCore サンプルプログラムのファイル構成（CPU1）

<¥sample¥ap_rzt2_0a_dualcore_sample_CPU1¥フォルダ内>

.cproject	･･･	CPROJECT ファイル
.project	･･･	PROJECT ファイル
ap_rzt2_0a.pincfg	･･･	AP-RZT2-0A ピンコンフィグファイル
ap_rzt2_0a_dualcore_sample_	･･･	AP-RZT2-0A DualCore サンプルプログラム
CPU1 Debug.jlink		J-Link 設定ファイル(RAM デバッグ用)
ap_rzt2_0a_dualcore_sample_	･･･	AP-RZT2-0A DualCore サンプルプログラム
CPU1 Debug.launch		デバッグおよびランタイム設定ファイル(RAM デバッグ用)
ap_rzt2_0a_dualcore_sample_	･･･	AP-RZT2-0A DualCore サンプルプログラム
CPU1 Release.jlink		J-Link 設定ファイル(ROM デバッグ用)
ap_rzt2_0a_dualcore_sample_	･･･	AP-RZT2-0A DualCore サンプルプログラム
CPU1 Release.launch		デバッグおよびランタイム設定ファイル(ROM デバッグ用)
configuration.xml	･･･	FSP コンフィギュレータファイル

<¥sample¥ap_rzt2_0a_dualcore_sample_CPU1¥script フォルダ内>

fsp_ram_execution_cpu1.ld	･･･	RAM 実行用リンカスクリプトファイル

<¥sample¥ap_rzt2_0a_dualcore_sample_CPU1¥src フォルダ内>

hal_entry.c	･･･	アプリケーションソースファイル

3.　動作説明

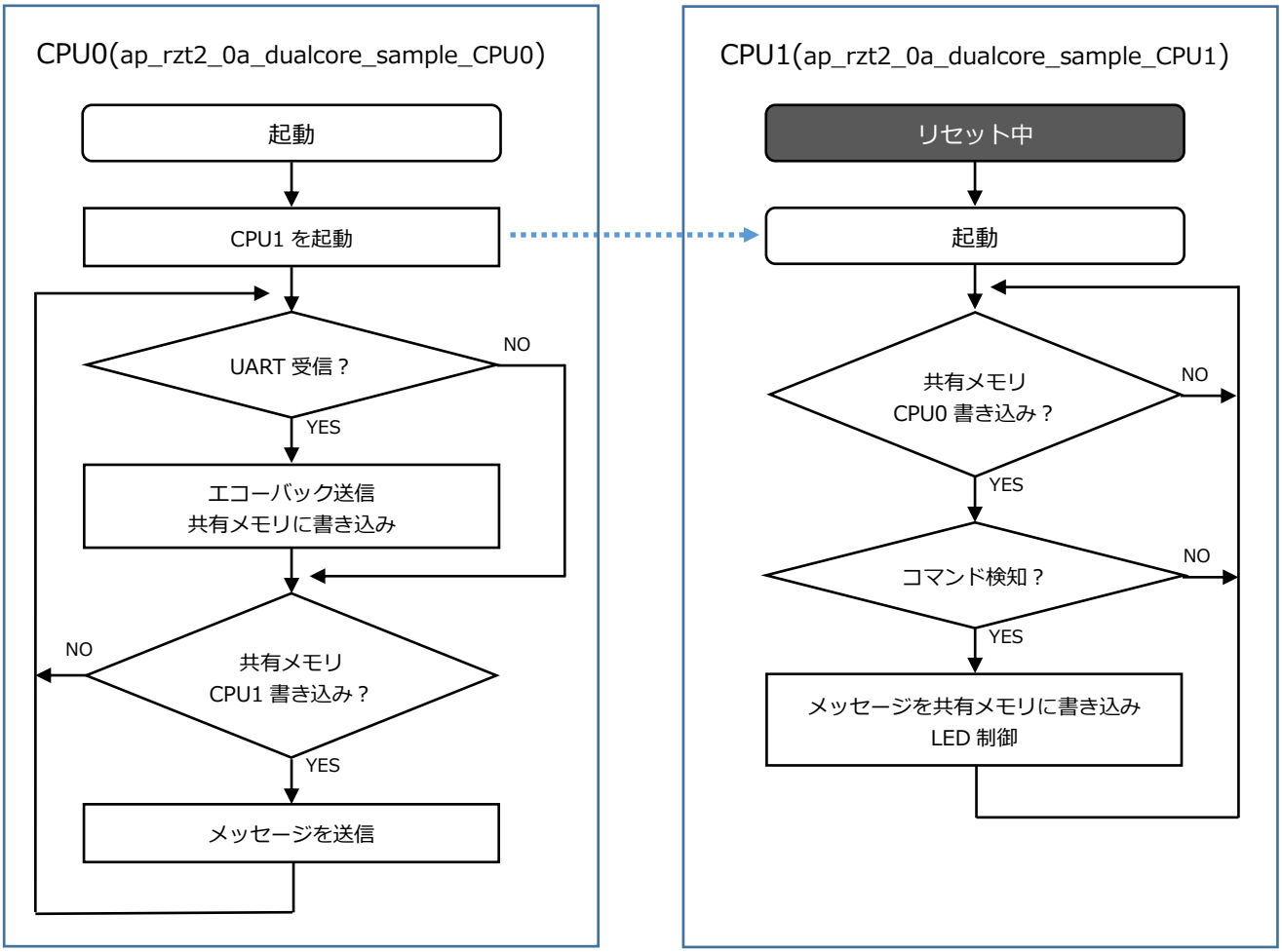
3.1　サンプルプログラムの動作

AP-RZT2-0A に搭載された CPU（R9A07G075M24GBG）は二つの CPU（CPU0、CPU1）を持っています。
通常、電源起動時に CPU0 が起動しますが、CPU0 から CPU1 を起動させることで二つの CPU が同時に機能します。

本サンプルプログラムは二つの CPU を動作させるサンプルプログラムであり、動作を開始すると下記の順に処理を行います。

1. CPU0 が起動します
2. CPU0 が CPU1 を起動させます。
3. CPU0 は UART のエコーバックを行い、受信したデータを共有メモリに書き込みます。
   また、共有メモリを監視し、CPU1 によって書き込まれたデータを UART で送信します。
4. CPU1 は共有メモリを監視し、CPU0 によって書き込まれたデータからコマンド（3.1.2 を参照）を検知した場合は LED の点滅制御を行い、動作したことを示すメッセージを共有メモリに書き込みます。

以下に、上記の動作の流れを図に示します。



3.1.1　サンプルプログラムにおける CPU0 の動き

- SCI0 を用いて通信を行います。

  シリアルの設定は、115200bps、ビット長 8、パリティなし、ストップビット 1、フロー制御なしです。

  動作確認は、ホスト PC 上のターミナルソフト（ハイパーターミナル等）を使用してください。
- エコーバックを行い、受信したデータを共有メモリに書き込みます。
- CPU1 が共有メモリに書き込んだ場合はそのデータを UART で送信します。

3.1.2　サンプルプログラムにおける CPU1 の動き

- 共有メモリを監視し、CPU1 が書き込んだデータから以下のコマンドを検知した場合、LED の点滅制御を行います。

コマンド	LED の制御
*OFF	LED 消灯
*250MS	250ms 周期で LED 点滅
*1000MS	1000ms 周期で LED 点滅

- LED の点滅制御後、共有メモリに以下のメッセージを書込み、処理終了を CPU0 に通知します。

  「¥r¥nCPU1 received command.¥r¥n」

3.2　DualCore サンプルプログラムのデバッグ方法

サンプルプログラムを CPU ボード上で実行するためには、ビルドしたサンプルプログラムの実行ファイルを CPU ボードにダウンロードする必要がありますが、本サンプルプログラムは 2 つのコアで動作するため、各コアに実行ファイルをそれぞれダウンロードする必要があります。

本節では CPU ボードの各コアにサンプルプログラムをダウンロードする方法、ボードのシリアル FlashROM へ書き込んで実行する方法については説明いたします。

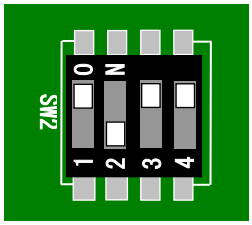
サンプルプログラムのビルド方法や、サンプルプログラムの基本的なダウンロード方法などについては、以下のアプリケーションノートに詳細な手順が記されていますので、こちらもご参考ください。

・AN1647 RZ/T2M 開発チュートリアル

3.2.1　RAM デバッグ方法

1. 「AN1647 RZ/T2M 開発チュートリアル」を参考に、サンプルプロジェクト「ap_rzt2_0a_dualcore_sample_CPU0」と「ap_rzt2_0a_dualcore_sample_CPU1」をビルドしてください。

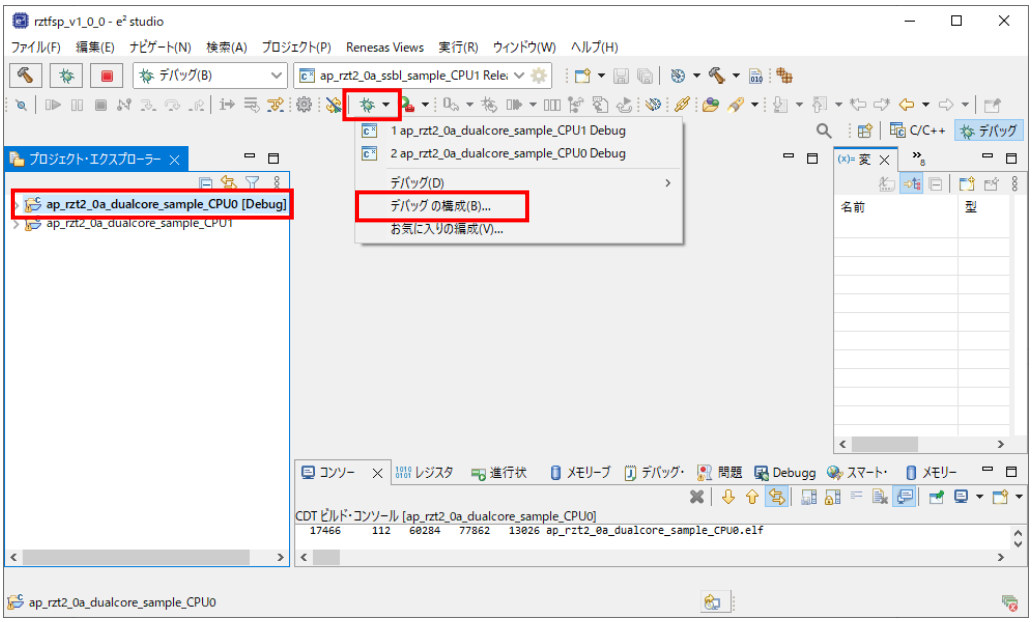
2. ボード上のディップスイッチを設定します。
   サンプルプログラム動作時は、以下の通りに設定してください。



<SW2 設定>	
ブートモード	：シリアル FlashROM ブート/USB ブート/SCI ブートのいずれにも該当しないモード
JTAG Hash モード	：使用する（不問）

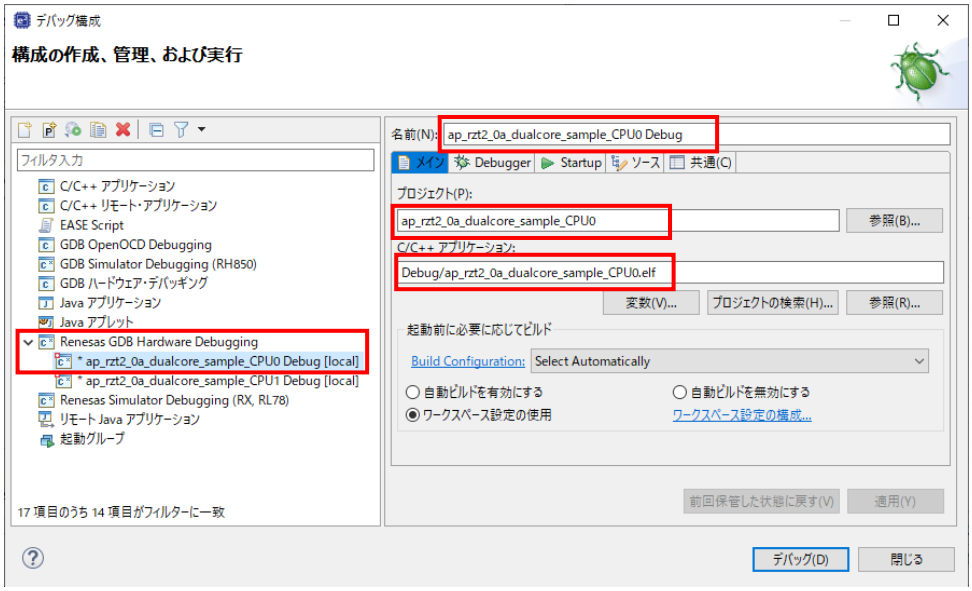
3. ボードに電源を投入してください。

4.　CPU0 プロジェクトを選択し、ツールバーのデバッグアイコンから［デバッグの構成］を開きます。

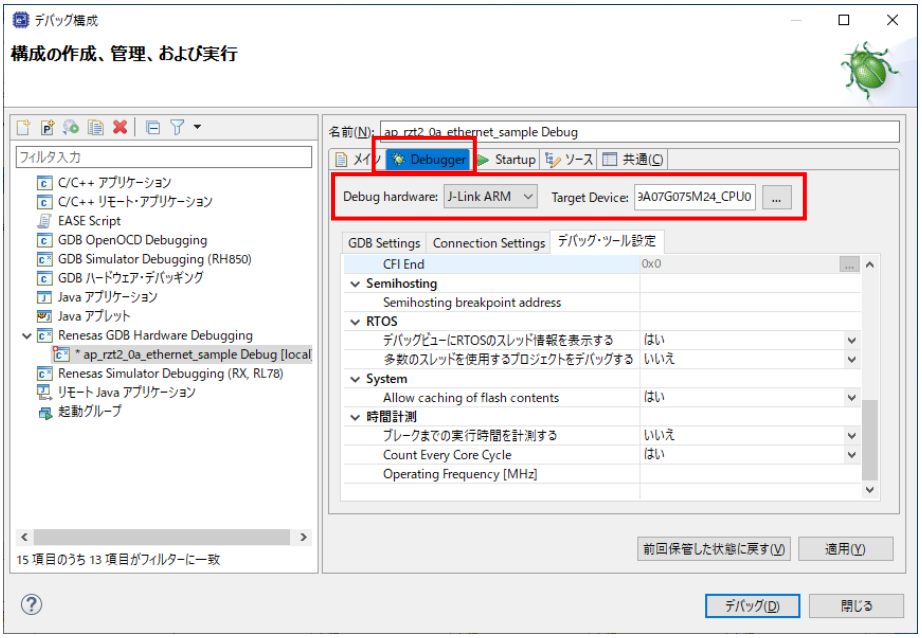


5.　［Renesas GDB Hardware Debugging］のデバッグ設定から［<プロジェクト名> Debug］を選択し、下記の内容になっていることを確認してください。
また、デバッグ設定が見つからない場合はデバッグ設定を新規作成してください。

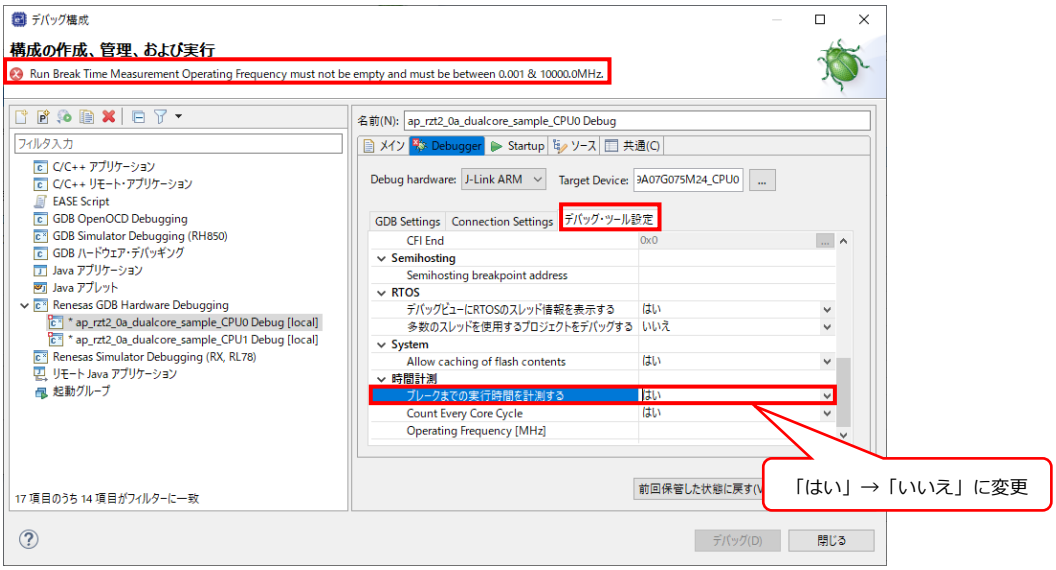
- ［名前］：ap_rzt2_0a_dualcore_sample_CPU0 Debug
- ［プロジェクト］：ap_rzt2_0a_dualcore_sample_CPU0
- ［C/C++アプリケーション］：Debug/ap_rzt2_0a_dualcore_sample_CPU0.elf



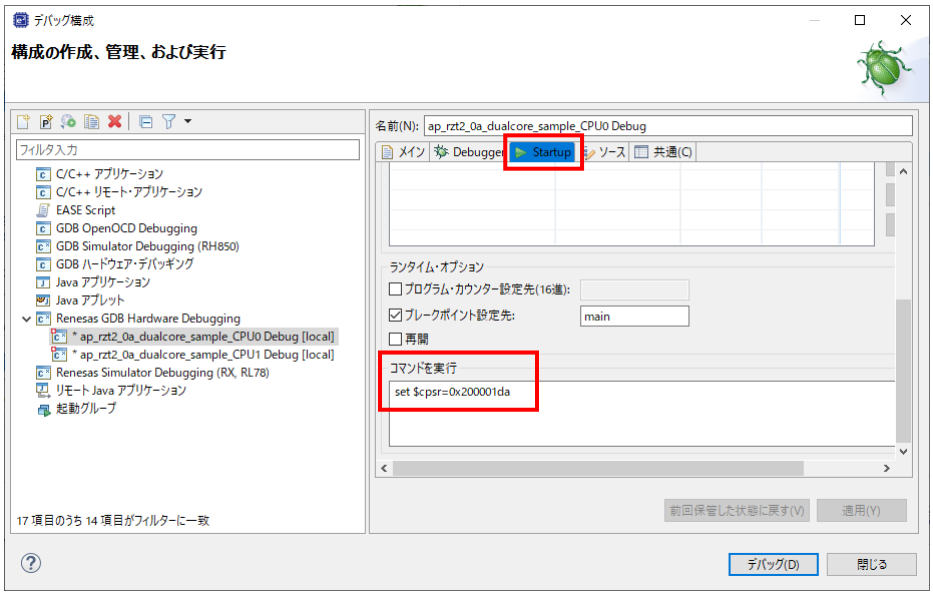
6. ［Debugger］タブを選択し、［Debug hardware］を［J-Link ARM］、
［Target Device］を「R9A07G075M24_CPU0」に設定されていることを確認してください。



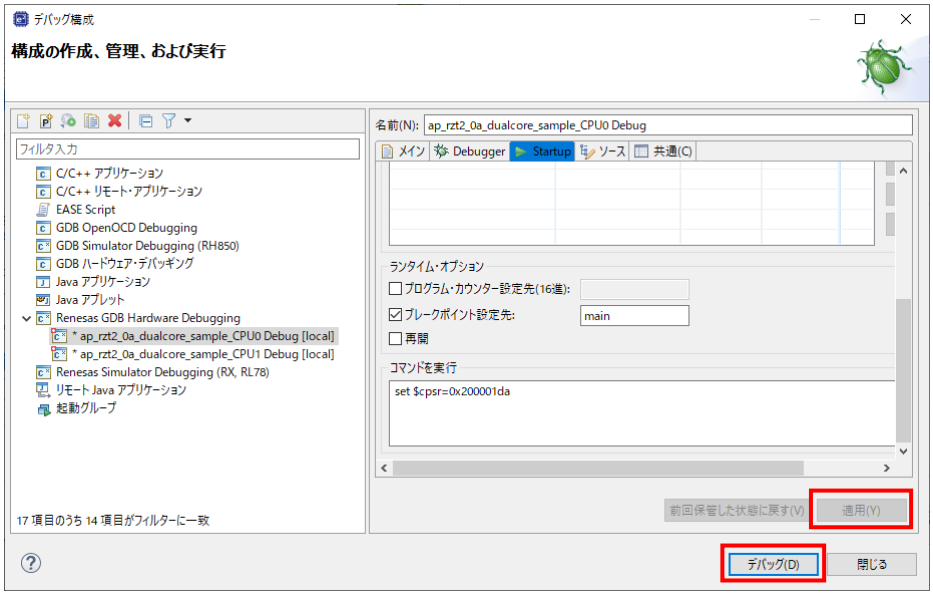
デバッグ設定を新規作成した場合などに、［Target Device］に「R9A07G075M24_CPU0］を選択するとウィンドウ上にエラーメッセージ「Run Break Time ～」が表示されることがあります。
エラーメッセージが表示された場合は、「デバッグ・ツール設定」＞「時間計測」＞「ブレークまでの実行時間を計測する」を「いいえ」に変更してください。



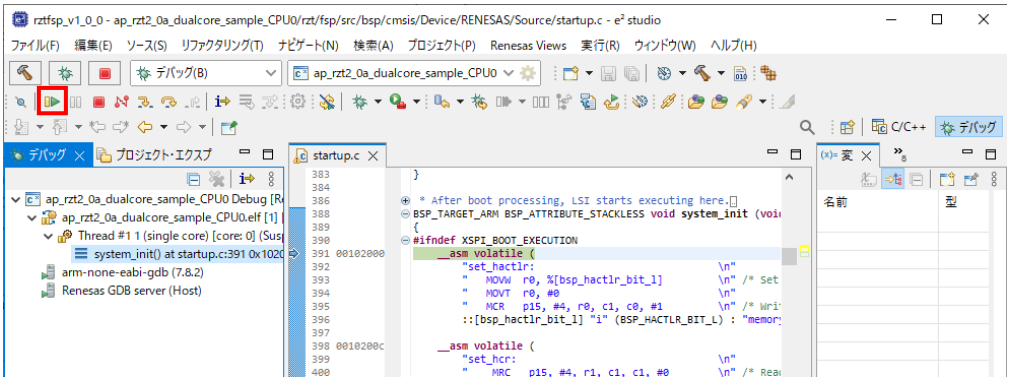
7.　［Startup］タブを選択し、［コマンドを実行］に「set $cpsr=0x200001da」を設定してください。



8.　「適用」ボタンを押して設定を保存し、続けて「デバッグ」ボタンを押します。

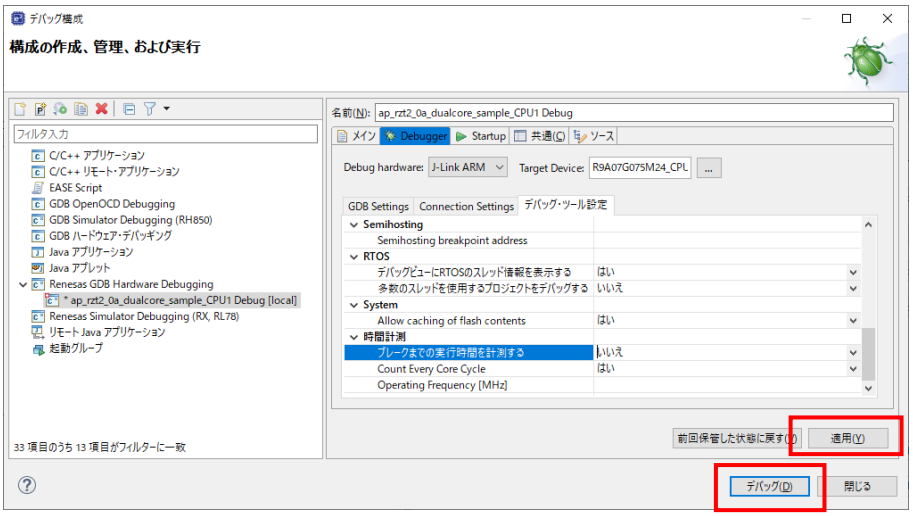


9.　ボードとの接続が完了することを確認します。

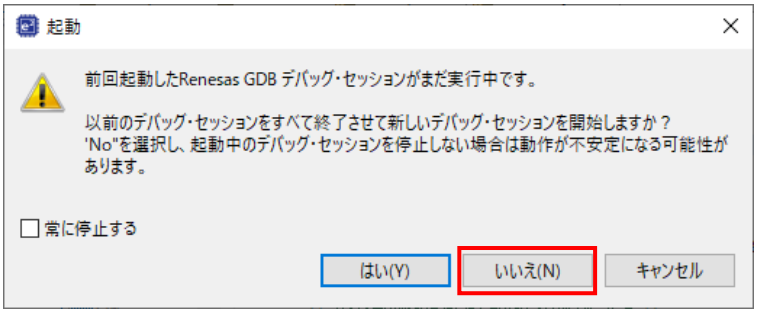


10. プロジェクト「ap_rzt2_0a_dualcore_sample_CPU1」を対象に、4～6 と同様のデバッグ設定を行ってください。
※プロジェクトや Target Device などの設定は「CPU0」→「CPU1」に読み替えてください。

11. 「適用」ボタンを押して設定を保存し、続けて「デバッグ」ボタンを押します。

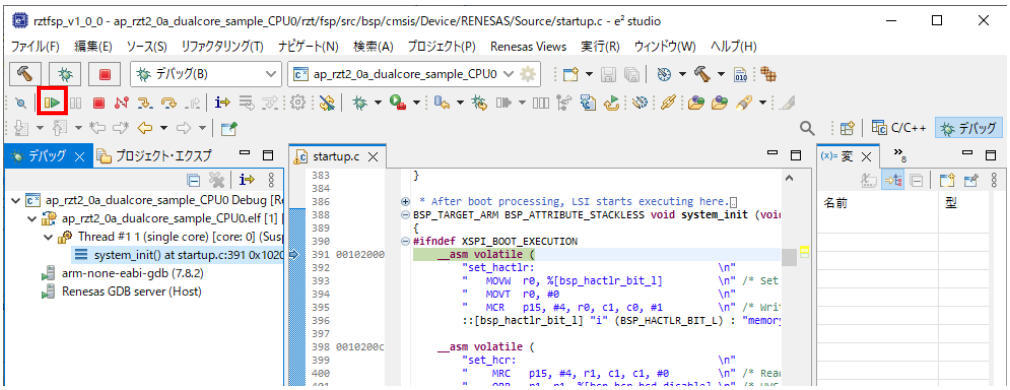


以下のメッセージが表示されるので、「いいえ」を選択します。



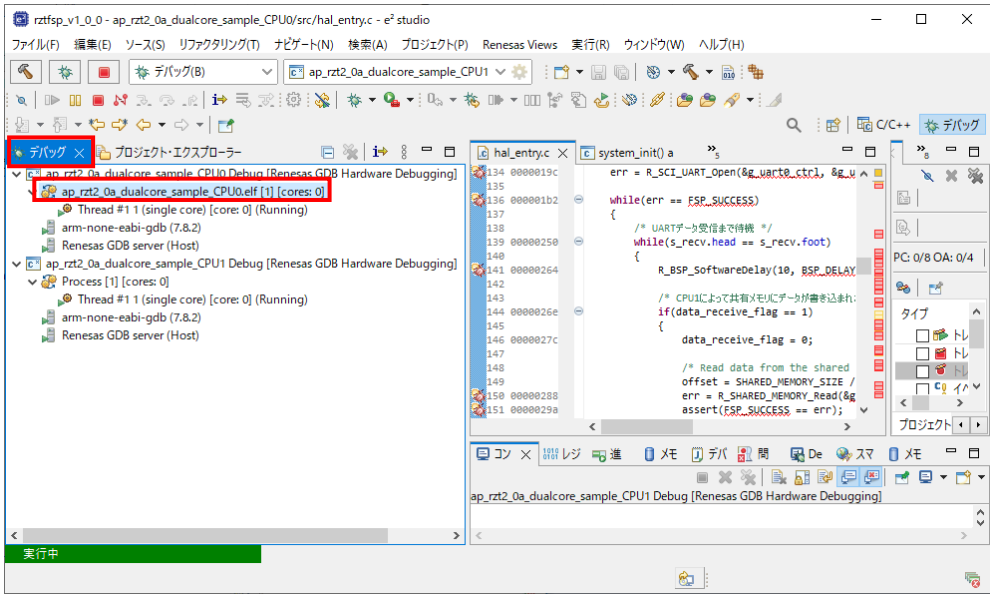
※ウィンドウが表示されない場合は、e2studio のワークスペースを作り直すか、e2studio のユーザマニュアルをご参考ください。

12. ボードとの接続が完了することを確認します。

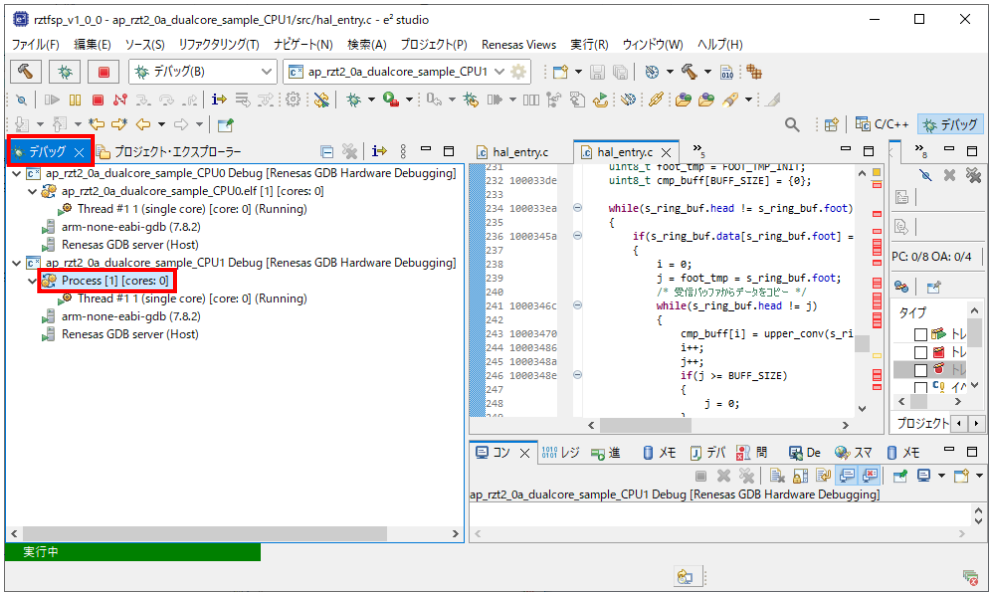


13. 以降、CPU0/CPU1 はそれぞれのプログラムに従って動作します。

CPU0 のプログラムの停止やブレークポイントの設定などのデバッグ操作を行う場合は、デバッグウィンドウの[ap_rzt2_0a_dualcore_sample_CPU0 Debug]内の[ap_rzt2_0a_dualcore_sample_CPU0.elf [1] [core: 0]]を選択した上で操作してください。



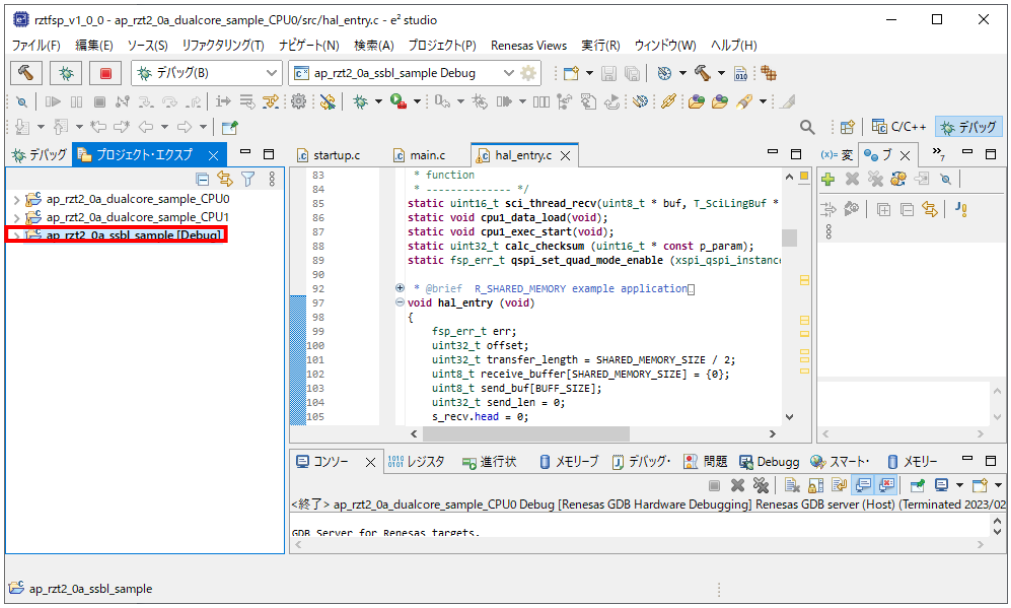
CPU1 のプログラムの停止やブレークポイントの設定などのデバッグ操作を行う場合は、デバッグウィンドウの[ap_rzt2_0a_dualcore_sample_CPU1 Debug]内の[Process [1] [core: 0]]を選択した上で操作してください。



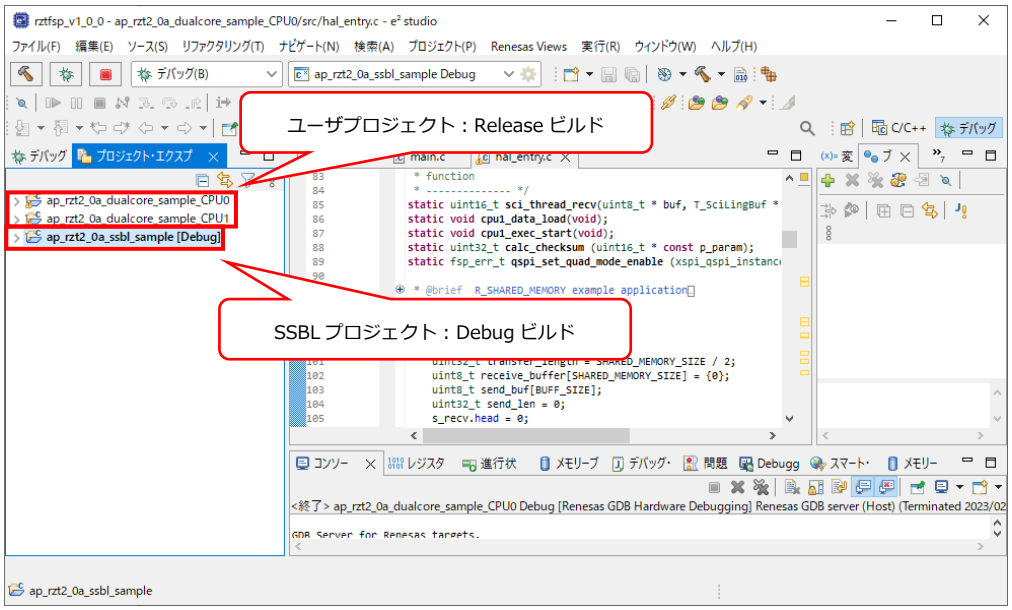
3.2.2　ROM デバッグ方法

本項ではボード上のシリアル FlashROM に書き込まれた DualCore サンプルプログラムをデバッグする手順を示します。
後述の「4. シリアル FlashROM への書き込み」を参考に、あらかじめプログラムをシリアル FlashROM へ書き込んでから読み進めてください。

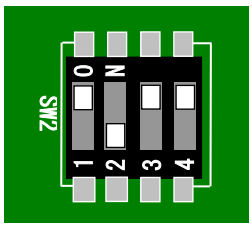
また、本項では本サンプルプログラムとは別に、ボード付属の SSBL サンプルプログラムも必要です。
事前に本サンプルプログラム「ap_rzt2_0a_dualcore_sample_CPU0」「ap_rzt2_0a_dualcore_sample_CPU1」とは別に、SSBL サンプルプログラム「ap_rzt2_0a_ssbl_sample」のインポートも行ってください。



1. ROM デバッグを行うユーザプロジェクトには「Release」ビルド、「SSBL サンプルプロジェクト」には「Debug」ビルドをそれぞれ「AN1647 RZ/T2M 開発チュートリアル」を参考にビルドします。



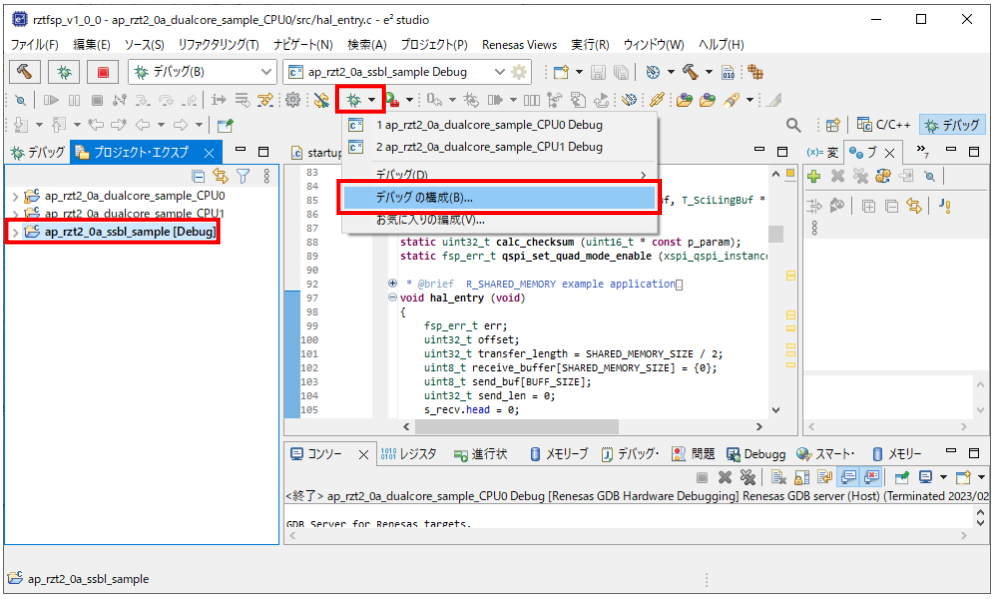
2. ボード上のディップスイッチを設定します。
   AP-RZT2-0A・AP-RZT2-1A 共通で以下の通りに設定してください。



<SW2 設定>	
ブートモード	：シリアル FlashROM ブート/ USB ブート/SCI ブートの いずれにも該当しないモード
JTAG Hash モード	：使用する（不問）

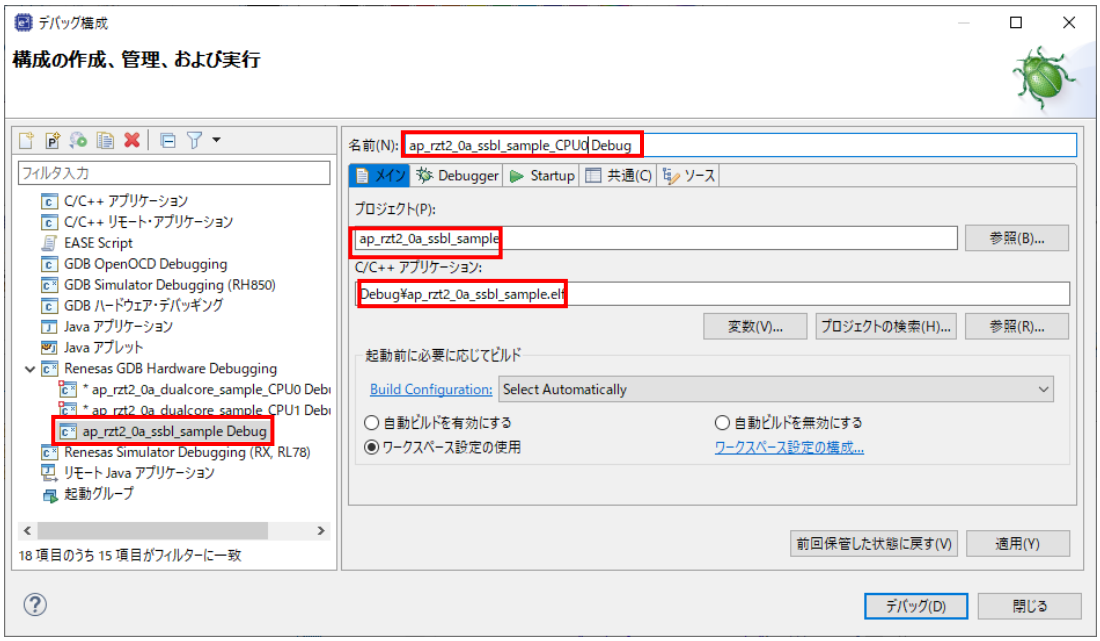
3. ボードに電源を投入してください。

4. SSBL プロジェクトを選択し、ツールバーのデバッグアイコンから［デバッグの構成］を開きます。

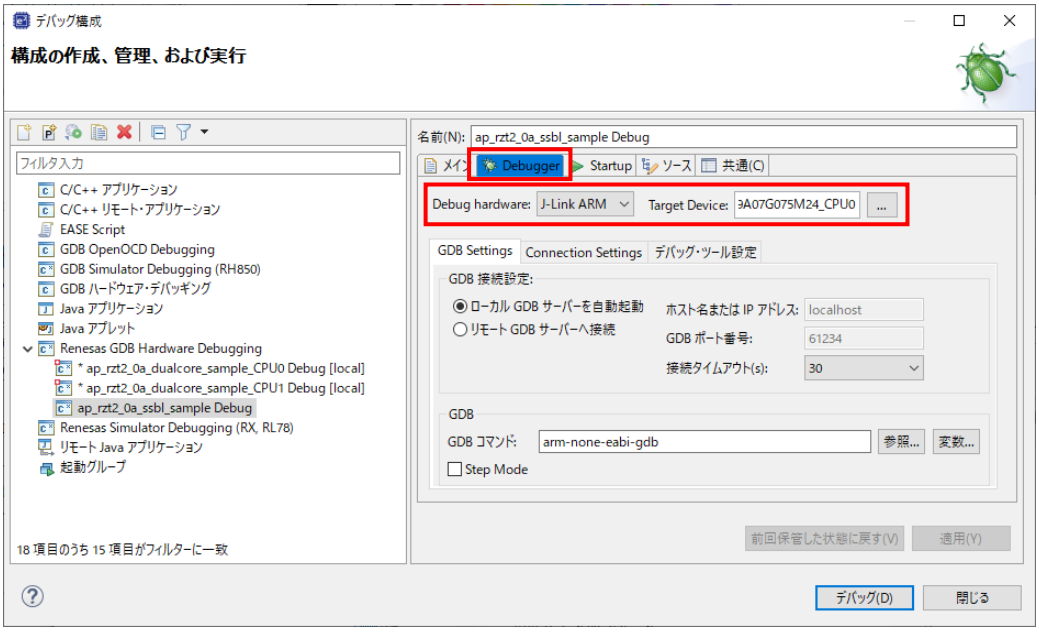


5. ［Renesas GDB Hardware Debugging］のデバッグ設定から［ap_rzt2_0a_ssbl_sample Debug］を選択し、下記の内容に設定してください。
また、デバッグ設定が見つからない場合はデバッグ設定を新規作成してください。

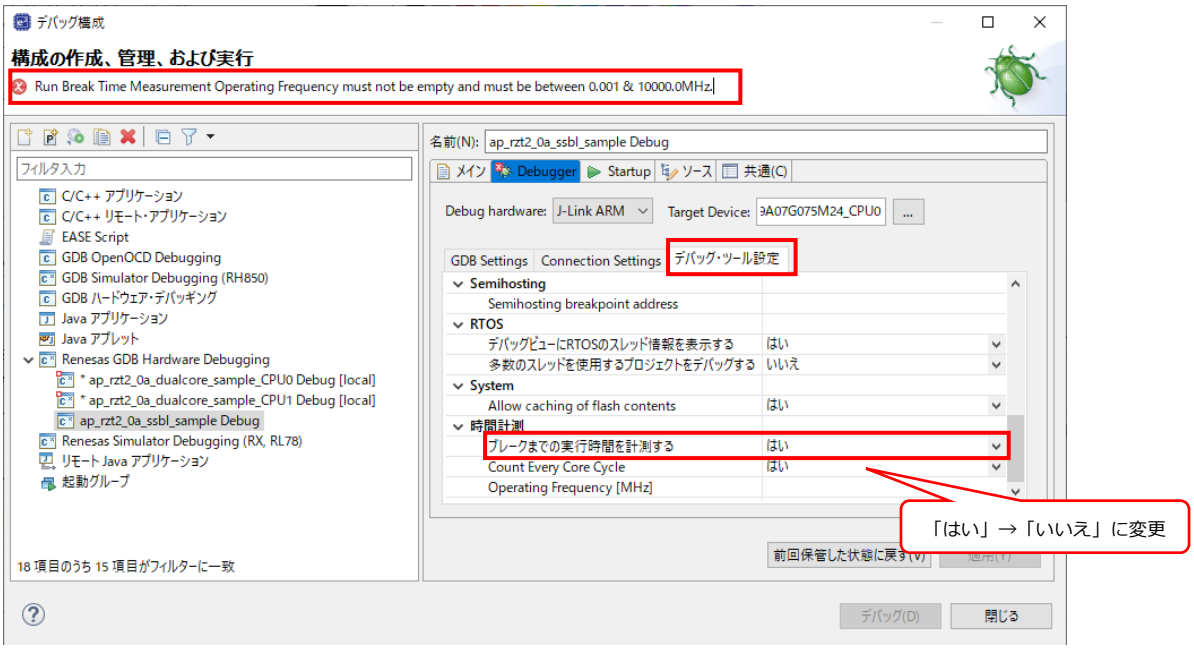
- [名前]：ap_rzt2_0a_ssbl_sample_CPU0 Debug
- [プロジェクト]：ap_rzt2_0a_ssbl_sample
- [C/C++アプリケーション]：Debug¥ ap_rzt2_0a_ssbl_sample.elf



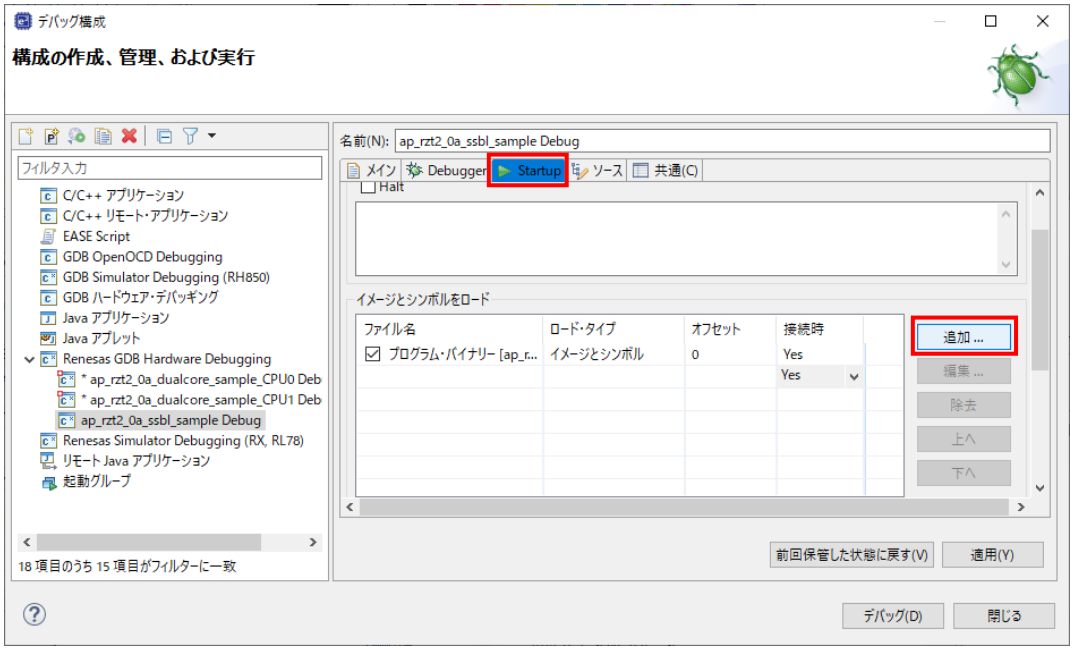
6. ［Debugger］タブを選択し、［Debug hardware］を［J-Link ARM］、
［Target Device］を［R9A07G075M24_CPU0］に設定されていることを確認してください。



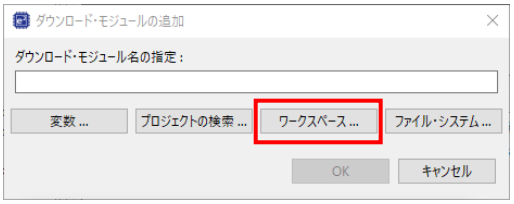
デバッグ設定を新規作成した場合などに、［Target Device］に「R9A07G075M24_CPU0］を選択するとウィンドウ上にエラーメッセージ「Run Break Time ～」が表示されることがあります。
エラーメッセージが表示された場合は、「デバッグ・ツール設定」＞「時間計測」＞「ブレークまでの実行時間を計測する」を「いいえ」に変更してください。

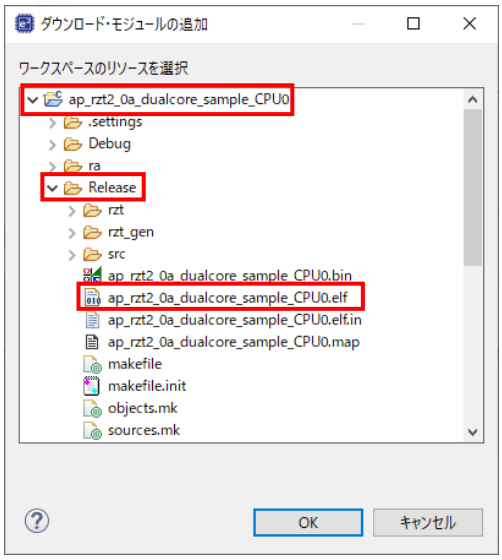


7.　［Startup］タブを選択し、［イメージとシンボルをロード］の「追加」ボタンを押してください。

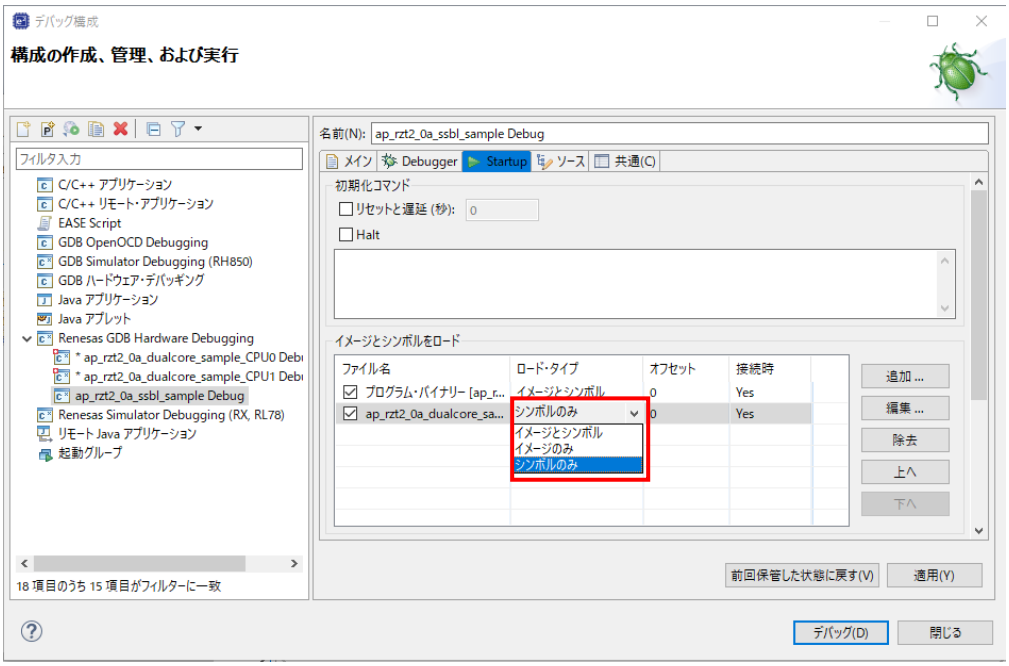


8.　「ワークスペース」を押し、「ap_rzt2_0a_dualcore_sample_CPU0」プロジェクトの「Release¥ap_rzt2_0a_dualcore_sample_CPU0.elf」を選択してウィンドウを閉じてください。

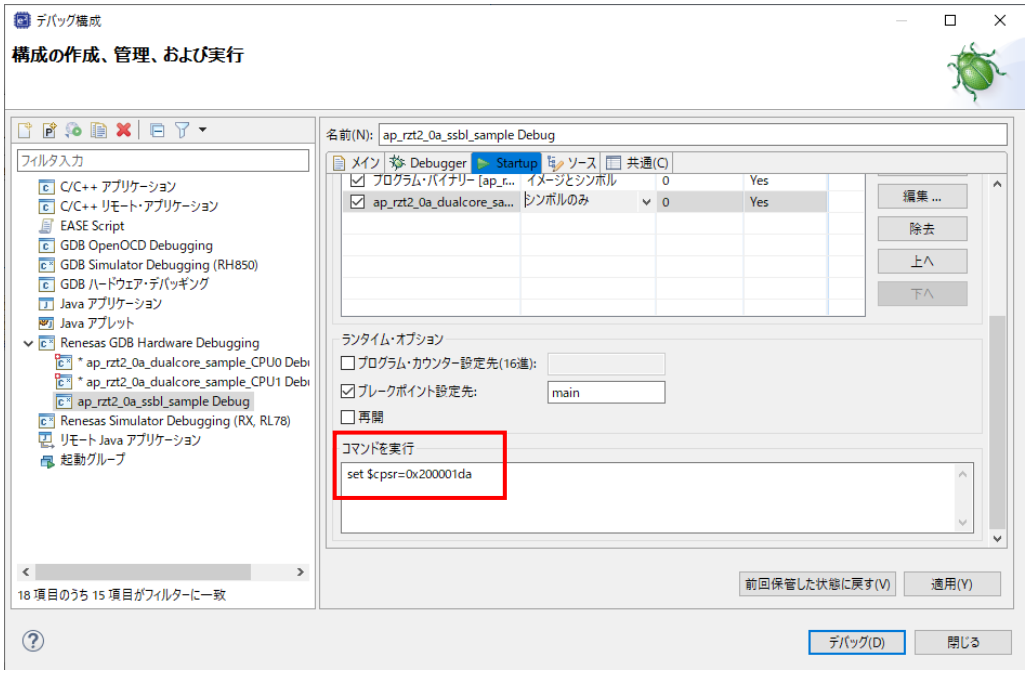




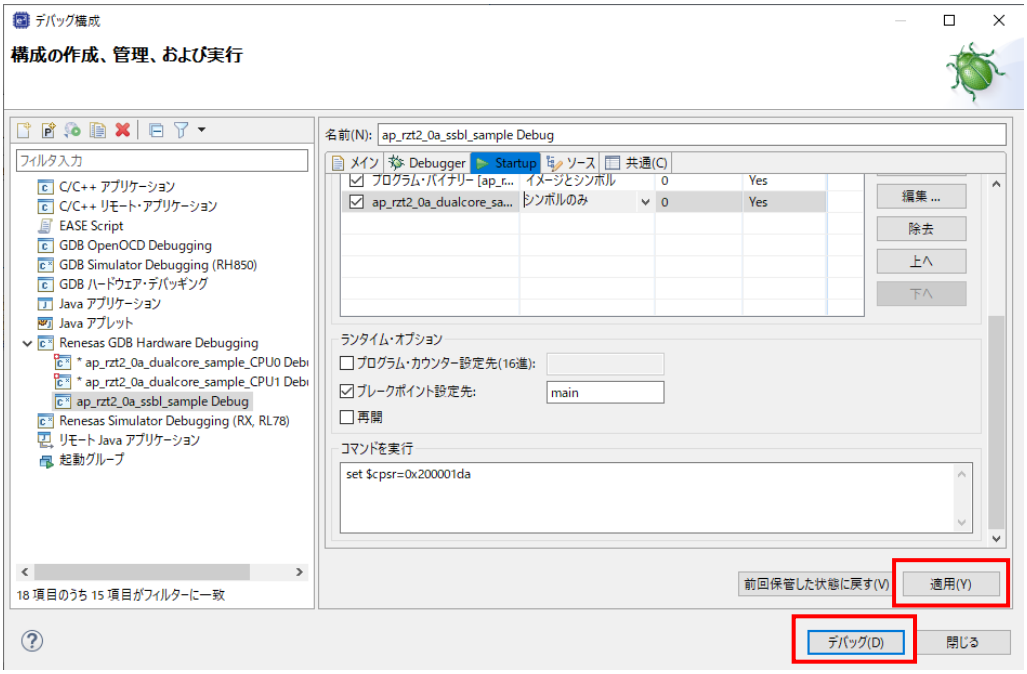
9.　［イメージとシンボルをロード］に追加できた項目の「ロード・タイプ」を「シンボルのみ」に変更してください。



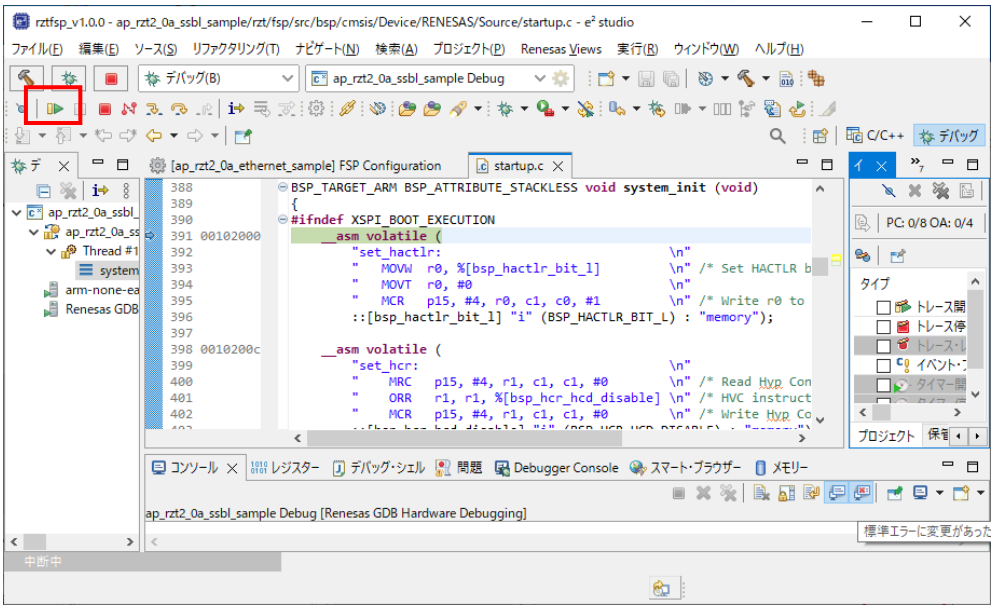
10. ［コマンドを実行］に「set $cpsr=0x200001da」を設定してください。



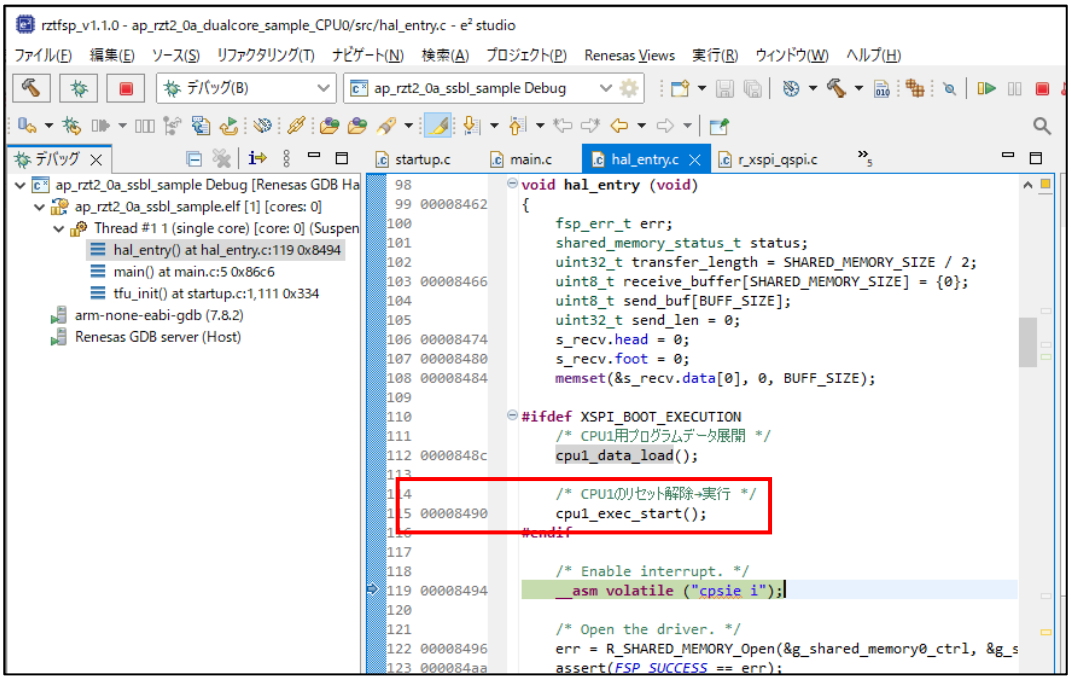
11. 「適用」ボタンを押して設定を保存し、続けて「デバッグ」ボタンを押します。



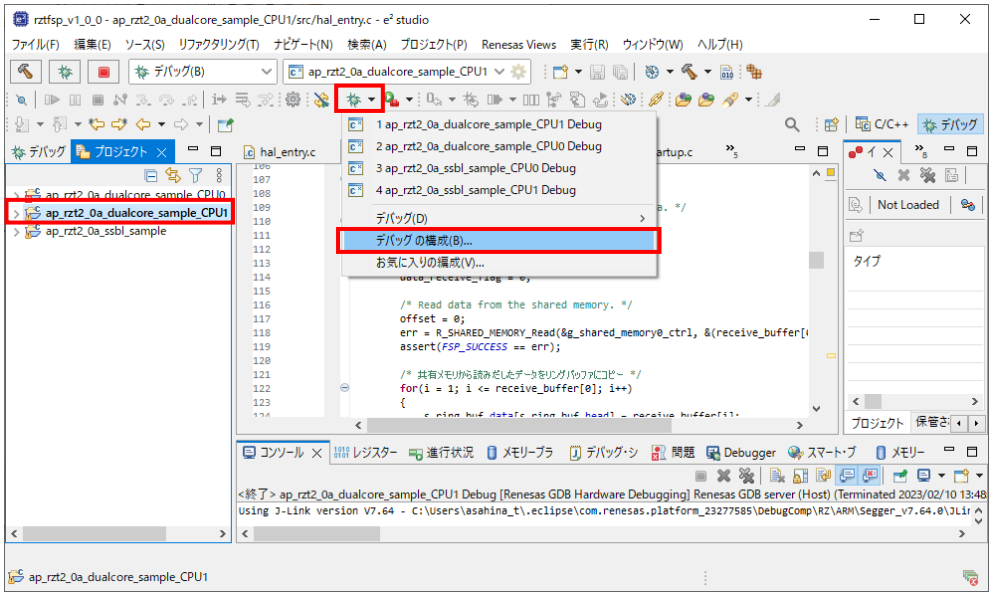
12. ボードとの接続が完了したらプログラムを「再開」ボタンを押し、サンプルプログラムを動作させてください。



13. CPU0 のプログラムが「<ap_rzt2_0a_dualcore_sample_CPU0¥src¥hal_entry.c>内の"cpu1_exec_start()"」を実行完了した以降、CPU1 のデバッグは可能です。

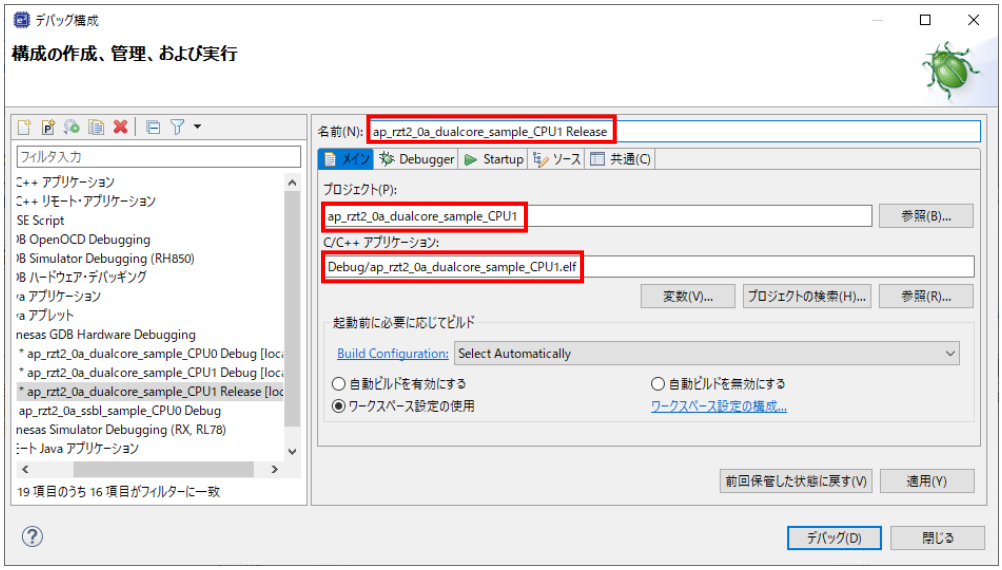


プロジェクト「ap_rzt2_0a_dualcore_sample_CPU1」を選択し、ツールバーのデバッグアイコンから［デバッグの構成］を開きます。



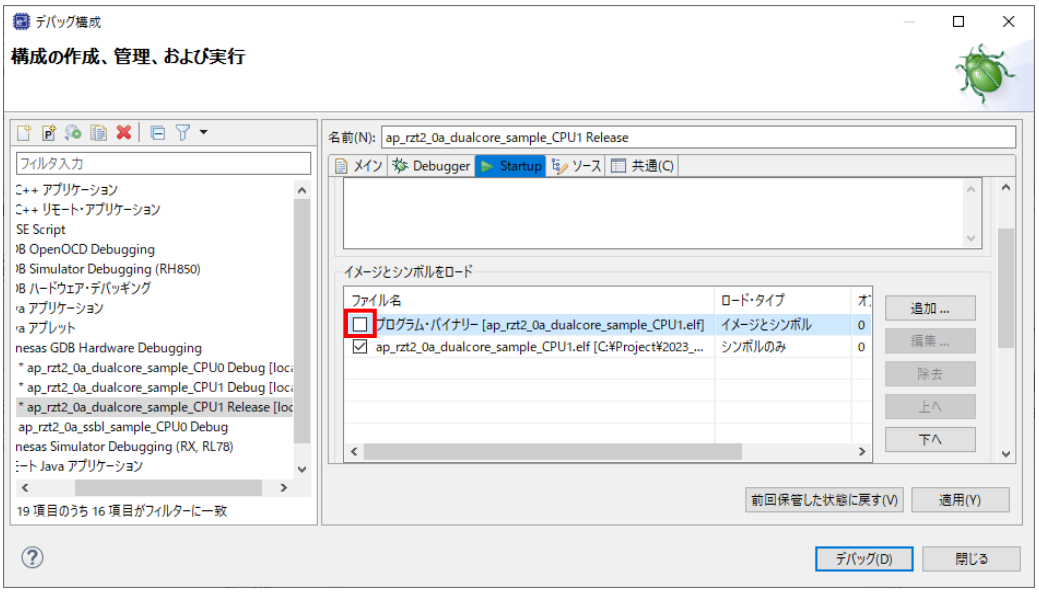
14. ［Renesas GDB Hardware Debugging］を右クリックして新規構成を作成し、下記の内容に設定します。

- ［名前］：ap_rzt2_0a_dualcore_sample_CPU1 Release
- ［プロジェクト］：ap_rzt2_0a_dualcore_sample_CPU1
- ［C/C++アプリケーション］：Debug/ap_rzt2_0a_dualcore_sample_CPU1.elf

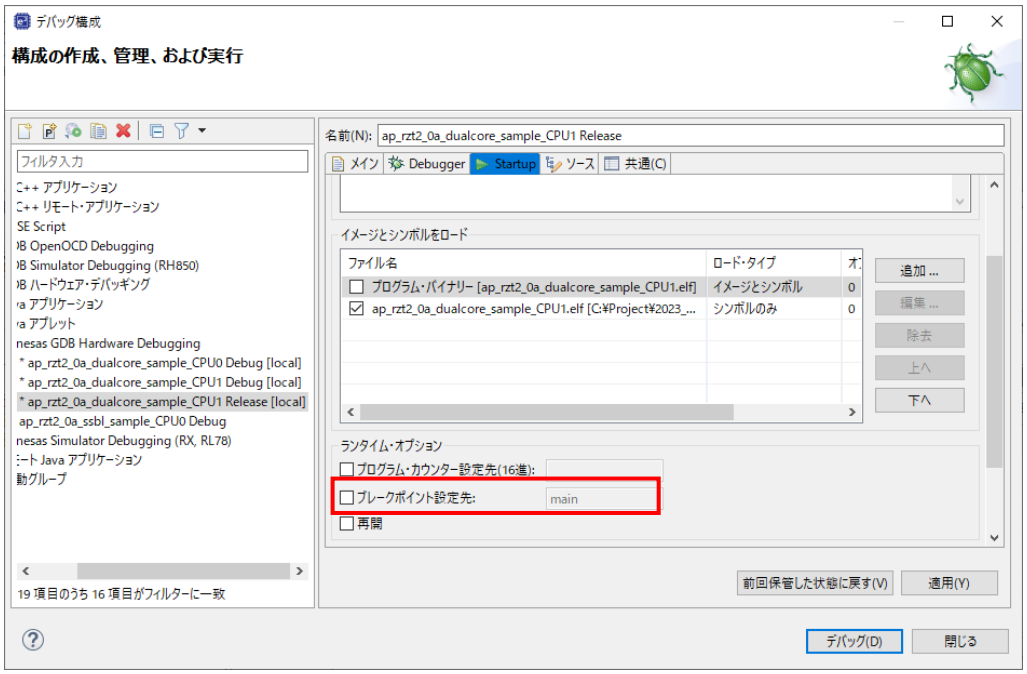


15. 「ap_rzt2_0a_dualcore_sample_CPU1 Release」で 6～9 と同様のデバッグ設定を行ってください。
※プロジェクトや Target Device などの設定は「CPU0」→「CPU1」に読み替えてください。

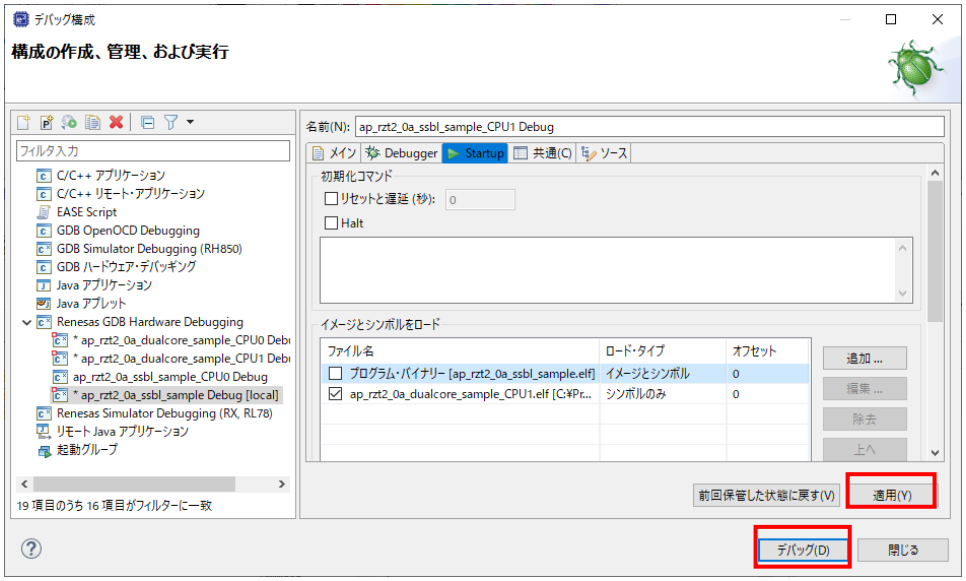
16. 「Debug」タブ内の「イメージとシンボルをロード」内の「プログラム・バイナリー[ap_rzt2_0a_dualcore_sample_CPU1.elf]」のチェックを外します。



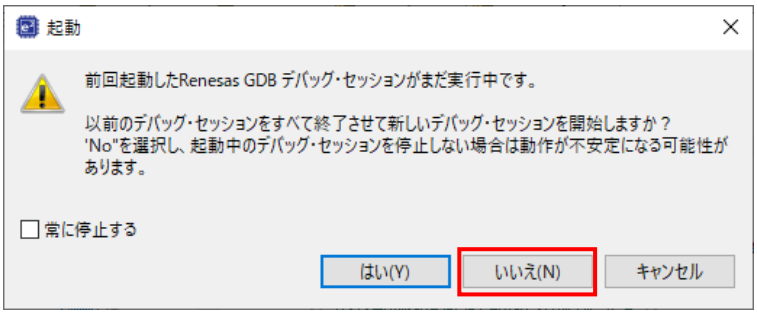
17. 「ブレークポイント設定先」のチェックを外します。



18. 「適用」ボタンを押して設定を保存し、続けて「デバッグ」ボタンを押します。

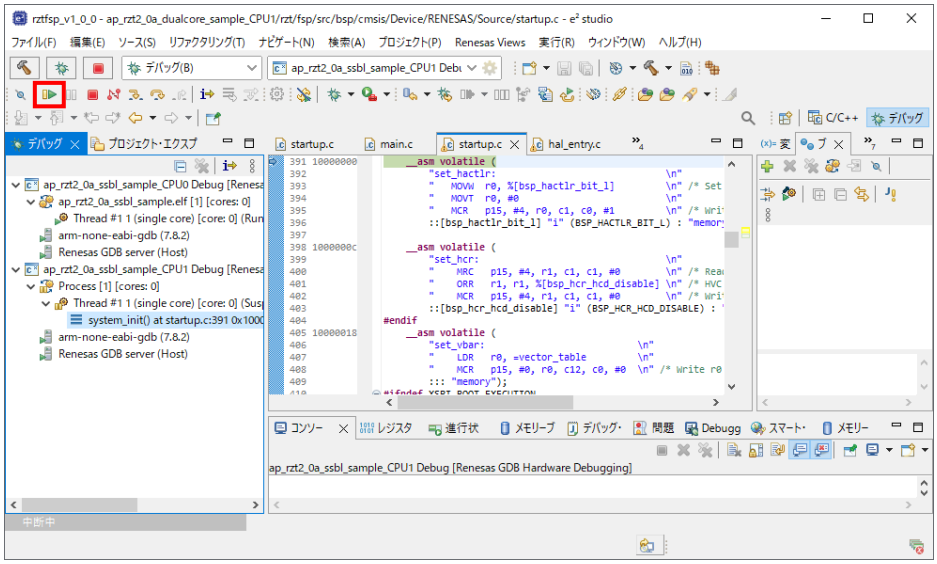


以下のメッセージが表示されるので、「いいえ」を選択します。

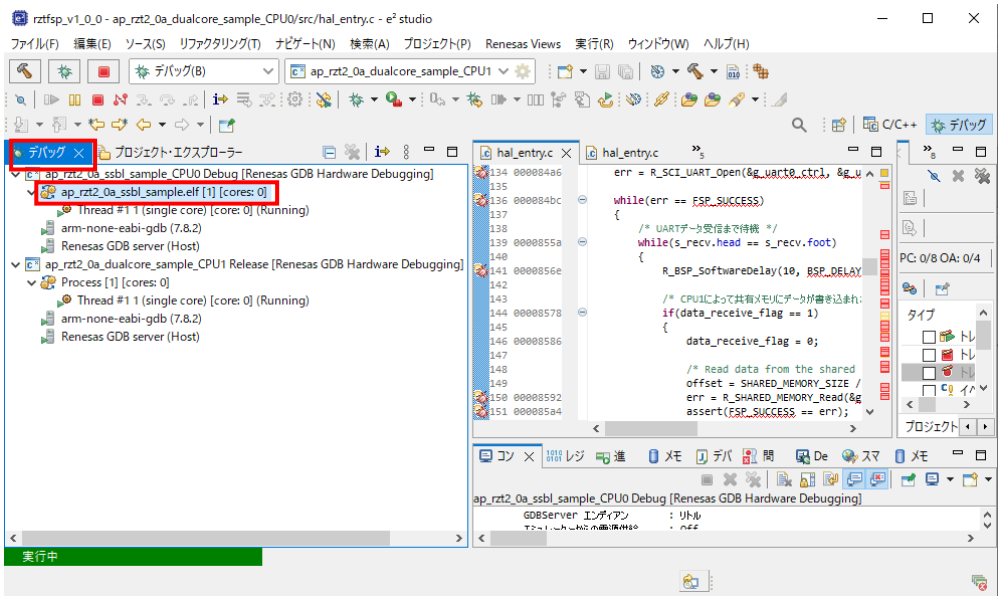


※ウィンドウが表示されない場合は、e2studio のワークスペースを作り直すか、e2studio のユーザマニュアルをご参考ください。

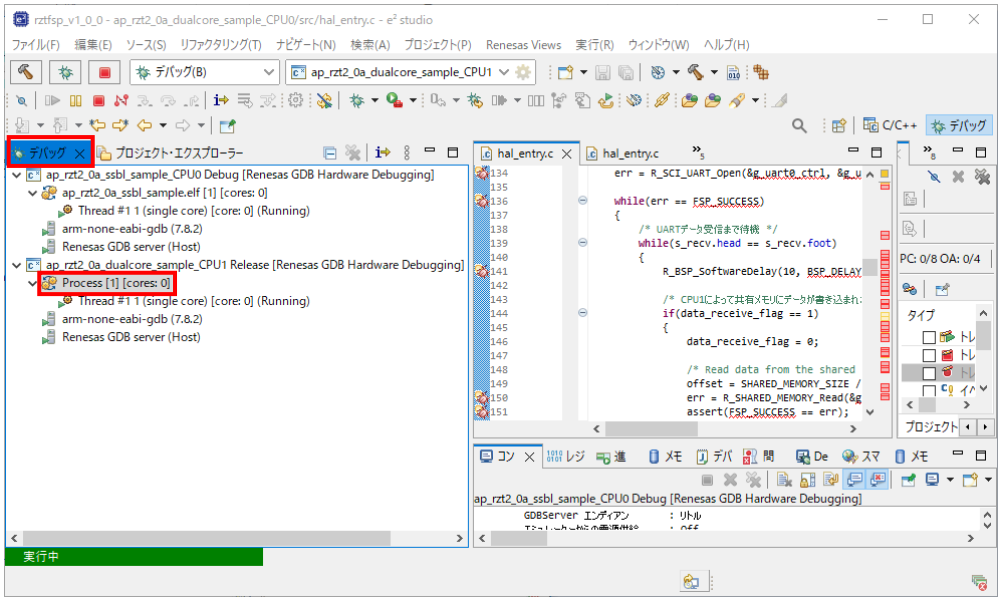
19. ボードとの接続が完了したらプログラムを「再開」ボタンを押し、サンプルプログラムを動作させてください。



20. 以降、CPU0/CPU1 はそれぞれのプログラムに従って動作します。
CPU0 のプログラムの停止やブレークポイントの設定などのデバッグ操作を行う場合は、デバッグウィンドウの[ap_rzt2_0a_ssbl_sample_CPU0 Debug]内の[ap_rzt2_0a_ssbl_sample.elf [1] [core: 0]]を選択した上で操作してください。



CPU1 のプログラムの停止やブレークポイントの設定などのデバッグ操作を行う場合は、デバッグウィンドウの[ap_rzt2_0a_dualcore_sample_CPU1 Debug]内の[Process [1] [core: 0]]を選択した上で操作してください。



4.　シリアル FlashROM への書き込み

AP-RZT2-0A は、シリアル FlashROM ブートを用いることで、電源起動後に自動でプログラムをシリアル FlashROM から RAM へ読み出して実行することができます。
本章ではシリアル FlashROM ブートを行うために必要な、プログラムをシリアル FlashROM へ書き込む手順を説明いたします。

シリアル FlashROM への書き込みにはサンプルプログラムに付属する書き込み用バッチファイルを使用します。
書き込み用バッチファイルは、DualCore サンプルプログラム（AP-RZT2-0A のみ）用ファイルとそれ以外のサンプルプログラム用ファイルの 2 種類用意しています。
※DualCore サンプルプログラムは他サンプルプログラムと異なり、CPU コア別に複数のユーザプログラムを操作する必要があるため、別途用意しております。

本書では前者の DualCore サンプルプログラム用の書き込み用バッチファイルを基に説明を行います。
ただし、DualCore サンプルプログラム以外の共通書き込み用バッチファイルを使ったときの作業と重複する内容については一部記述を省略している箇所があるため、「AN1647 RZ/T2M CPU BOARD 開発チュートリアル」も併せてご参照ください。

注意事項

- シリアル FlashROM の書き込むデータ構成について

  書き込み用バッチファイルを使うことで、以下の表に従ってプログラムをシリアル FlashROM に書き込みます。
  ユーザにてプログラムの配置先などを変更した場合には適宜値を読み替えてご参考ください。

項目	設定値
SSBL プログラムのローダ用パラメータの書き込み先アドレス	0x60000000
SSBL プログラムの書き込み先アドレス	0x60000050
CPU0 プログラムの書き込み先アドレス	0x6004D000
CPU0 プログラムのローダ用パラメータの書き込み先アドレス	0x60007080
CPU0 プログラムの展開先先頭アドレス、および、実行開始アドレス	0x00000000
CPU1 プログラムの書き込み先アドレス	0x600CD000
CPU1 プログラムのローダ用パラメータの書き込み先アドレス	0x60007090
CPU1 プログラムの展開先先頭アドレス、および、実行開始アドレス	0x00180000

- python スクリプトについて

  書き込み用バッチファイルはルネサス エレクトロニクス社が公開する「RZ/T2M グループ デバイスセットアップ for フラッシュブート サンプルプログラム」の"2022 年 10 月 7 日"版データを参考に作成しています。
  データの版数が異なる場合、バッチファイル内のコマンドや手順が正常に動作しない場合がございます。
  サンプルプログラム付属のバッチファイルを用いてシリアル FlashROM への書き込みがうまくいかない場合、「RZ/T2M グループ デバイスセットアップ for フラッシュブート サンプルプログラム」の資料をご参照いただき、適宜修正してバッチファイルをお使いください。

4.1　事前準備

4.1.1　Python のインストール

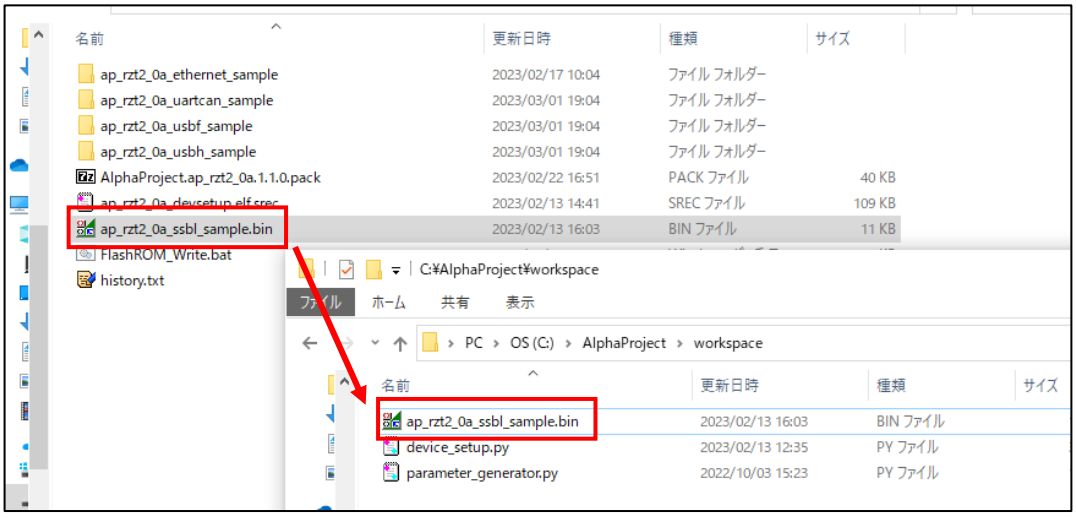
シリアル FlashROM への書き込みに Python を使うことなどは共通の書き込み用バッチファイルと同様です。
「AN1647 RZ/T2M CPU BOARD 開発チュートリアル」の「5.1.1 Python のインストール」を参考に作業を行ってください。

4.1.2　書き込み時の PC⇔ボード間通信プログラムの用意

シリアル FlashROM への書き込みに Python を使うことなどは共通の書き込み用バッチファイルと同様です。
「AN1647 RZ/T2M CPU BOARD 開発チュートリアル」の「5.1.2 書き込み時の PC⇔ボード間通信プログラムの用意」を参考に作業を行ってください。

4.1.3　書き込むプログラムの用意

①　Python ワークフォルダへ、各サンプルプログラムに付属する「ap_rzt2_0a_ssbl_sample.bin」ファイルを移動します。



ここで移動した「ap_rzt2_0a_ssbl_sample.bin」は、シリアル FlashROM ブート時に最初に起動させる SSBL サンプルプログラムのデータです。
SSBL サンプルプログラムが起動すると、ユーザプログラム（ap_rzt2_0a_ethernet_sample など）をシリアル FlashROM から RAM へと展開し、その後ユーザプログラムの実行を開始します。

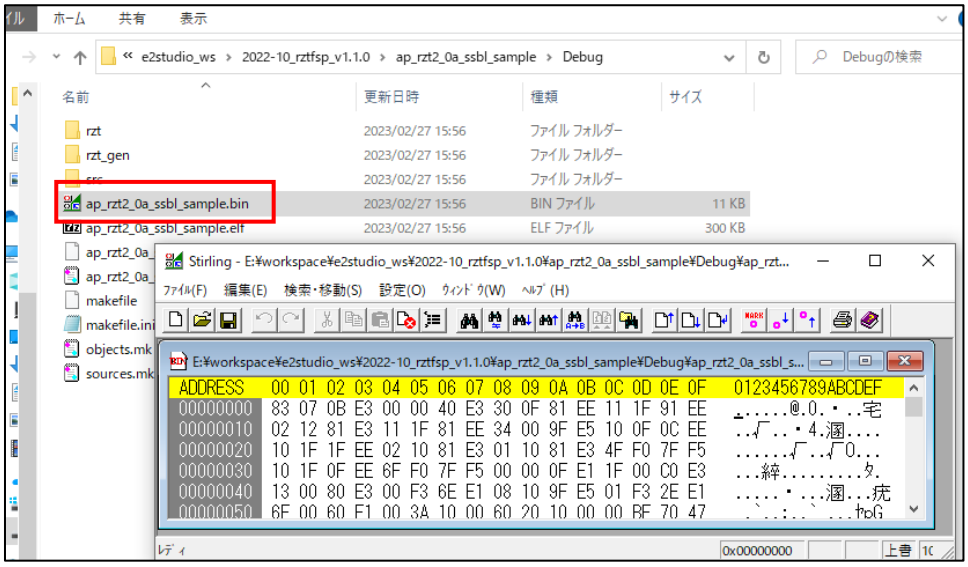
シリアル FlashROM ブート時の動作についてはルネサス社の「RZ/T2M グループ ユーザーズマニュアル ハードウェア編」内「3.5.1 ブート機能」をご参照ください。
文中のローダプログラムが SSBL サンプルプログラムに該当します。

CPU のブート機能を使ってユーザプログラムを実行するのではなく、一旦 SSBL サンプルプログラムを動作させて、ユーザプログラムをシリアル FlashROM から RAM へのプログラムデータの展開・実行する手順を踏むことにより、ユーザプログラムは CPU のブート機能の制限に関係なく、任意のプログラムを作成することができます。
CPU のブート機能の制限については「RZ/T2M グループ ユーザーズマニュアル ハードウェア編」内「3.5.3 ローダプログラム」に記載のローダプログラムの条件をご参照ください。

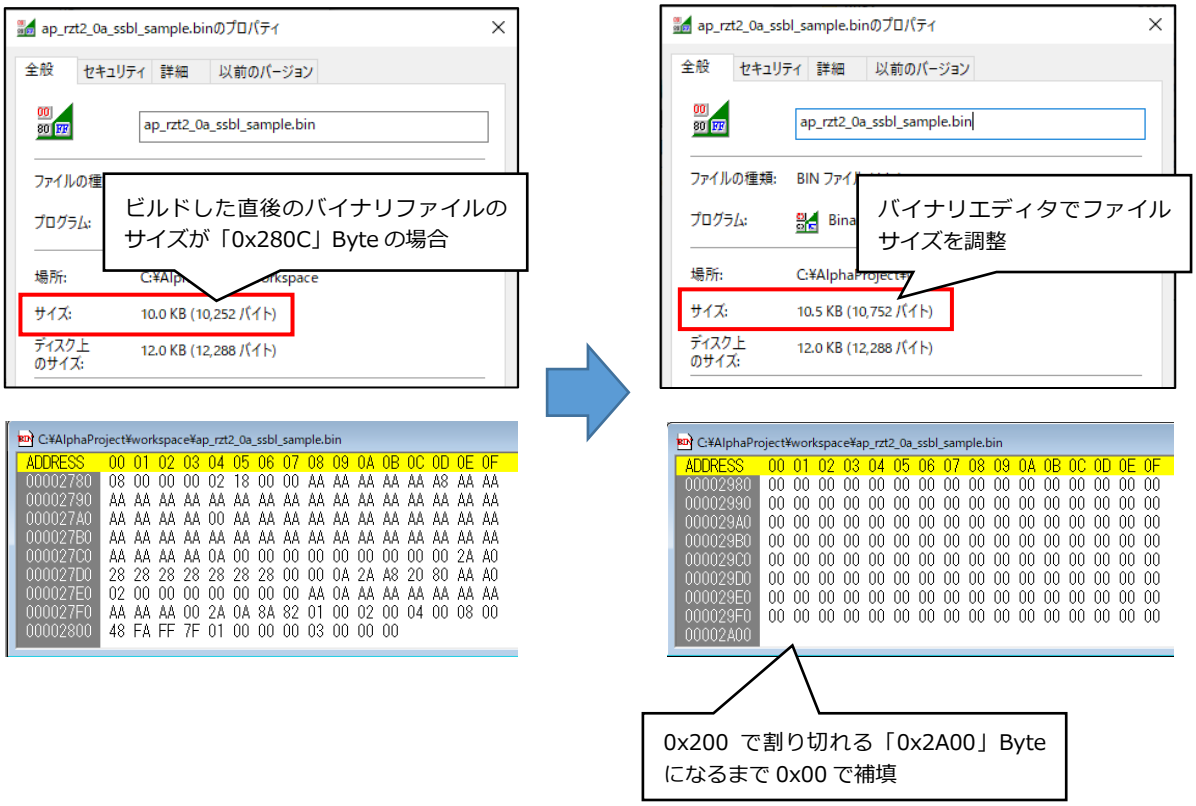
なお、「ap_rzt2_0a_ssbl_sample.bin」は SSBL サンプルプログラムを Debug ビルドで出力したデータから一部変更を加えています。
もし SSBL サンプルプログラムをユーザ自身でカスタマイズした場合は、以下のファイル変更を行ってください。

(1)　SSBL サンプルプログラムの Debug フォルダ内にあるバイナリファイルをバイナリエディタで開きます。

SSBL プログラム：　ap_rzt2_0a_ssbl_sample¥Debug¥ap_rzt2_0a_ssbl_sample.bin
※SSBL サンプルプログラムの Debug ビルドで生成できるバイナリファイルを使用します。



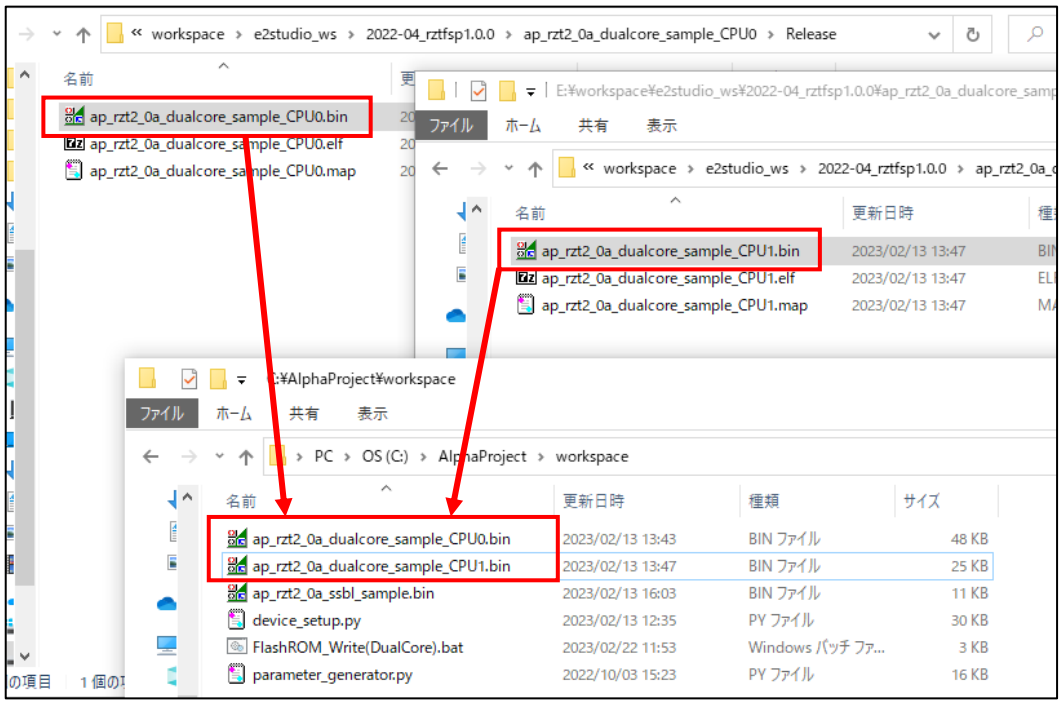
(2)　SSBL プログラム（ap_rzt2_0a_ssbl_sample.bin）のデータサイズが 0x200 単位になるようプログラム末尾に「0x00」などのダミーデータを追加します。その後、上書き保存してください。



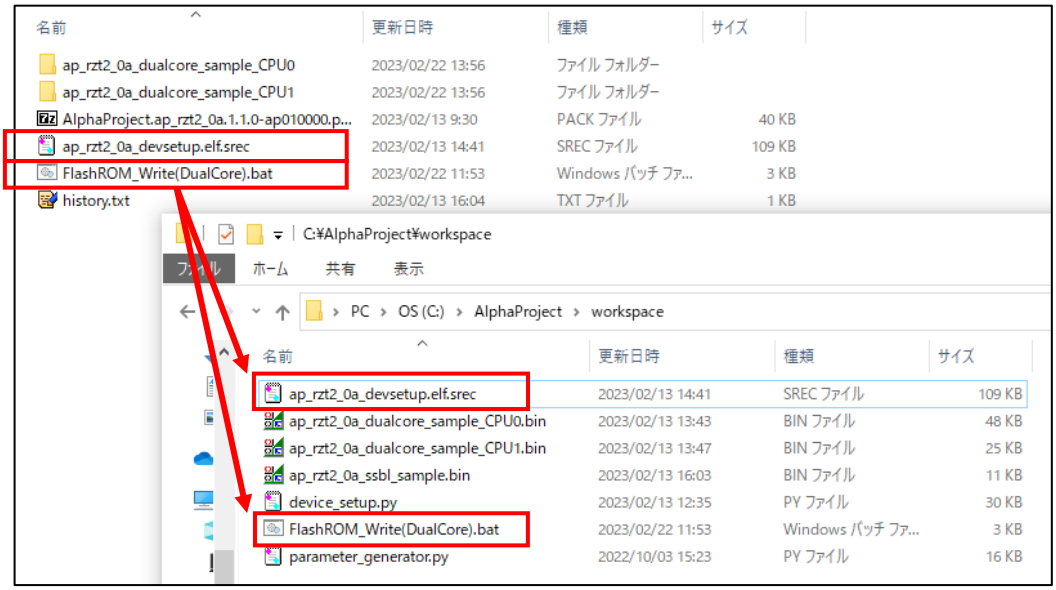
各サンプルプログラムに付属する「ap_rzt2_0a_ssbl_sample.bin」は、SSBL サンプルプログラムの Debug フォルダ内にある「ap_rzt2_0a_ssbl_sample.bin」から上記の変更を加えたものです。

②　Python ワークフォルダへ、DualCore サンプルプログラムで生成した以下のプログラムを移動します。

CPU0 ユーザプログラム：　ap_rzt2_0a_dualcore_sample_CPU0¥Release¥ap_rzt2_0a_dualcore_sample_CPU0.bin
CPU1 ユーザプログラム：　ap_rzt2_0a_dualcore_sample_CPU1¥Release¥ap_rzt2_0a_dualcore_sample_CPU1.bin
※サンプルプログラムの場合は Release ビルドで生成できるバイナリファイルを使用します。



③　Python ワークフォルダへ、サンプルプログラム付属の「ap_rzt2_0a_devsetup.elf.srec」と「FlashROM_Write(DualCore).bat」を移動します。



4.2　シリアル FlashROM への書き込み

書き込み用バッチファイルを使用して、CPU ボードのシリアル FlashROM にプログラムを書き込みます。

4.2.1　ボード設定

シリアル FlashROM への書き込みに SCI 通信を使う方法と USB 通信を使う方法があることは共通の書き込み用バッチファイルと同様です。

「AN1647 RZ/T2M CPU BOARD 開発チュートリアル」の「5.2.1 ボード設定」を参考に作業を行ってください。

4.2.2　書き込み手順

①　SCI ブート、あるいは、USB ブートモードに設定した CPU ボードに電源を入れます。

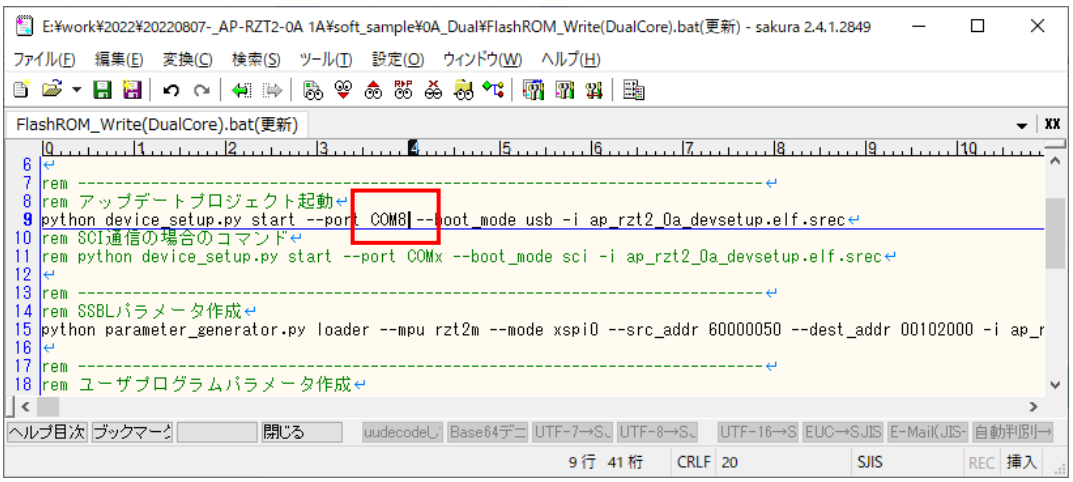
②　デバイスマネージャーを使用し、PC とボードを結ぶ COM ポートの番号を確認してください。

※使用する COM ポートの取り違えにご注意ください。

※COM ポートが現れない場合は、ボードのスイッチ設定や USB ケーブルの接続を見直してください。

③　書き込み用バッチファイル「FlashROM_Write(DualCore).bat」をテキストエディタで開きます。

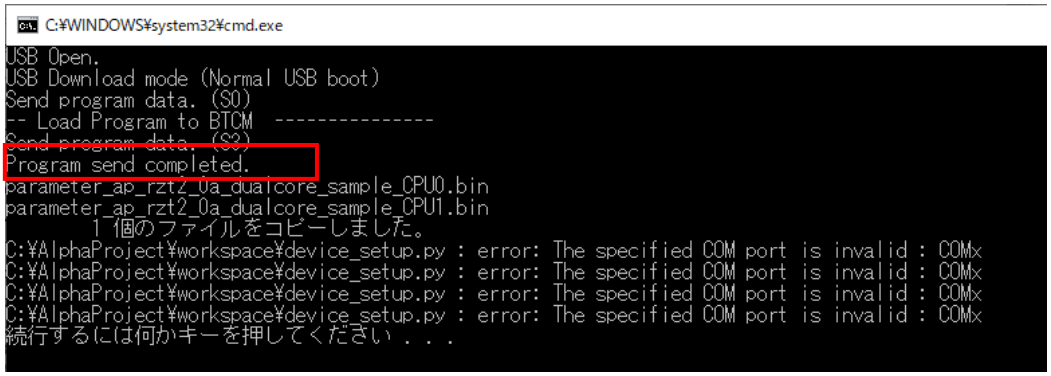
④　「python device_setup.py start～」の行中の COM 番号を②で確認した COM ポート番号に変更してください。
変更後、ファイルを保存して閉じます。



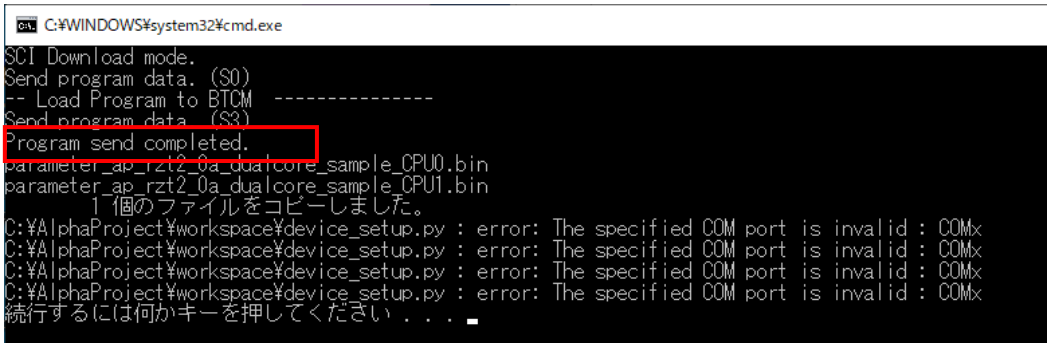
⑤　書き込み用バッチファイルをダブルクリックして実行します。

⑥　書き込み用バッチファイルが起動して次のような表示になることを確認します。
エラーメッセージが表示されていますが「Program send completed」までが表示されれば問題ありません。

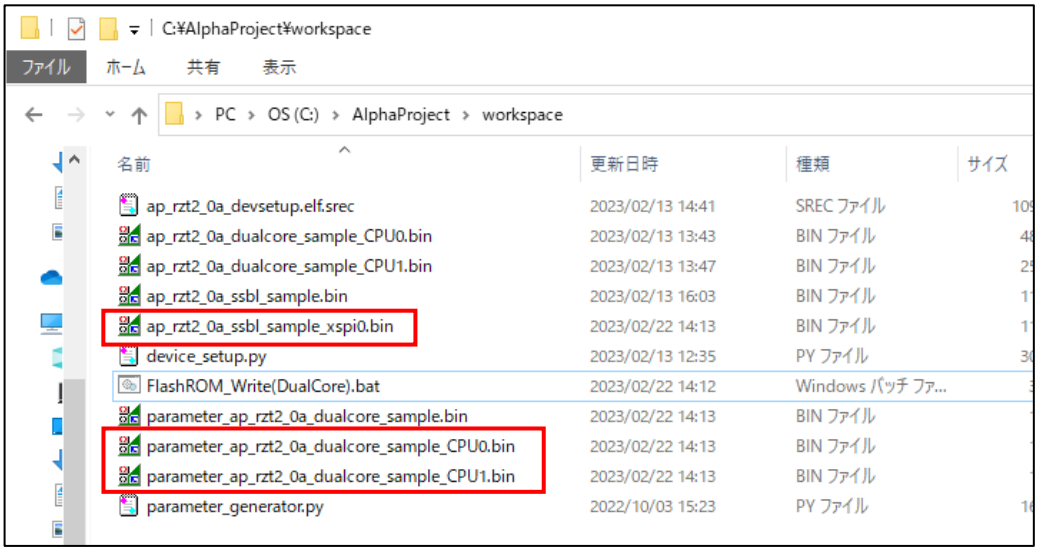
- USB 通信の場合



- SCI 通信の場合



また、バッチファイルを実行すると python ワークフォルダに「ap_rzt2_0a_ssbl_sample_xspi0.bin」や「parameter_<ユーザプログラム>.bin」のファイルが新規作成されます。

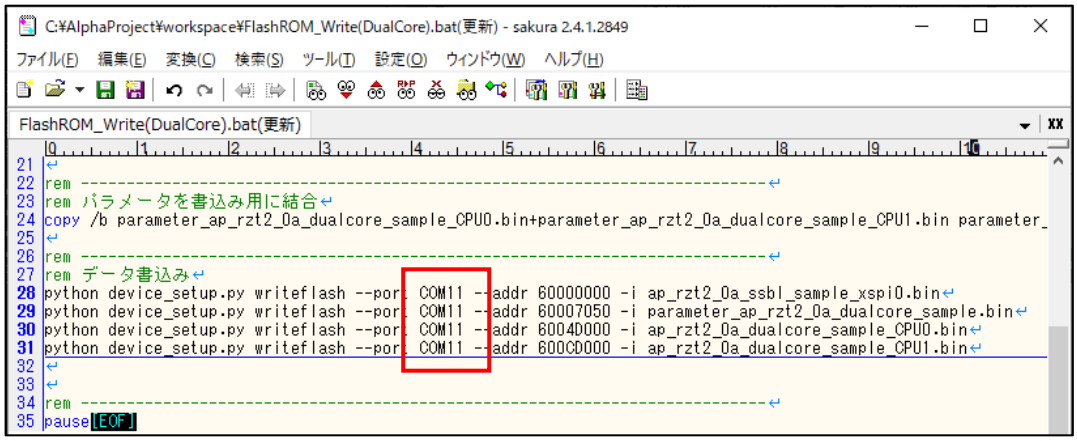


メッセージ「Program send completed」が表示されない場合、バッチファイル内のコマンド「python device_setup.py start ～」が正常に動作していません。
また、ファイル「ap_rzt2_0a_ssbl_sample_xspi0.bin」などが作成されない場合、バッチファイル内のコマンド「python parameter_generator.py ～」が正常に動作していません。

各コマンドが正常に動作しない場合は、python スクリプトの入手元であるルネサス社の「RZ/T2M グループ デバイスセットアップ for フラッシュブート サンプルプログラム」の資料をご確認いただき、各 python スクリプトの仕様に従ってバッチファイル内のコマンドを修正してください。

⑦　CPU ボードの電源を落とさずに、再度デバイスマネージャーを使用し、PC とボードを結ぶ COM ポートの番号を確認してください。
※SCI 通信の場合は COM ポートに変わりはないはずですが、USB 通信の場合 COM ポートが変わることがあります。

⑧　再度書き込み用バッチファイルをテキストエディタで開き、「python device_setup.py writeflash～」の行中の COM 番号を⑦で確認した COM ポート番号に変更してください。このとき、④で変更した「python device_setup.py start～」の行中の COM ポート番号は変更しないでください。
変更後、ファイルを保存して閉じます。

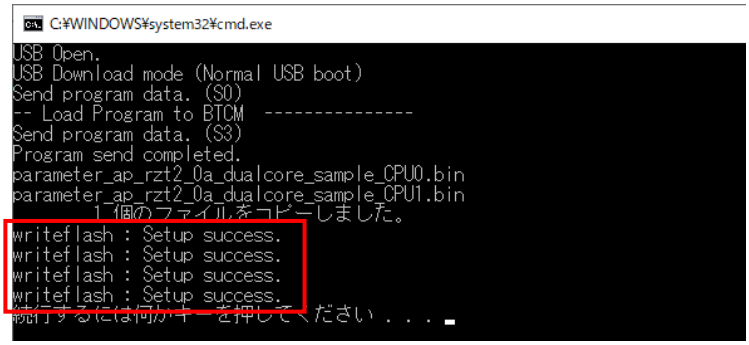


⑨　CPU ボードの電源を切り、再度電源を入れて再起動します。

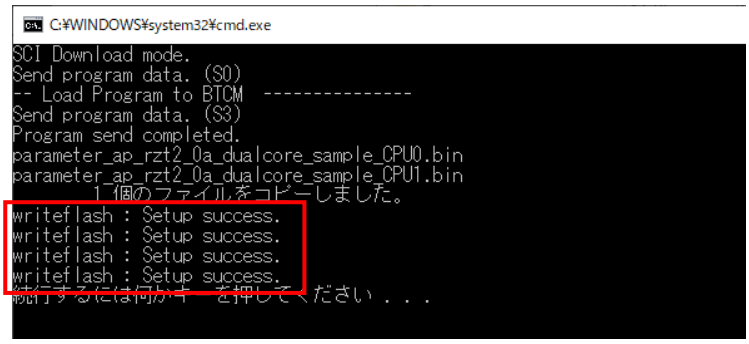
⑩　再度書き込み用バッチファイルをダブルクリックして実行します。

⑪　書き込み用バッチファイルが起動して次のような表示になることを確認します。
「writeflash：Setup success.」が 4 行表示されればシリアル FlashROM への書き込みは成功です。

- USB 通信の場合



- SCI 通信の場合



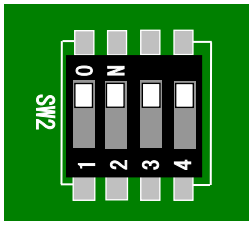
以上でシリアル FlashROM への書き込みは終了です。

4.3　シリアル FlashROM ブートの実行

「4.2　シリアル FlashROM の書き込み」を行った後、以下のスイッチ設定を行うことで、シリアル FlashROM ブートを行うことができます。

ボードに電源を投入し、プログラムの動作についてご確認ください。

SW2 設定



<SW2 設定>	
ブートモード	: シリアル FlashROM ブート
JTAG Hash モード	: 使用する（不問）

5.　開発環境使用時の各設定値

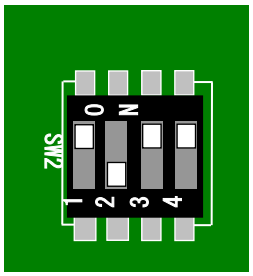
開発環境を使用する際の、AP-RZT2-0A 固有の設定を以下に示します。

※プロジェクト「ap_rzt2_0a_dualcore_sample_CPU1」を使用する際は、各設定値を「CPU0」→「CPU1」に読み替えてください。

ビルド・動作確認方法	
項目名	設定値
サンプルプログラムフォルダ	sample¥ap_rzt2_0a_dualcore_sample_CPU0
プロジェクト	ap_rzt2_0a_dualcore_sample_CPU0
デバッグ時のボード設定	「4.1 スイッチ設定」参照
デバッグ用出力フォルダ	/ ap_rzt2_0a_dualcore_sample_CPU0/Debug
デバッグ用実行ファイル	ap_rzt2_0a_dualcore_sample_CPU0.elf
Debug hardware	J-Link ARM
Target Device	R9A07G075M24GBG_CPU0
SerialFlash 書き込み用フォルダ	ap_rzt2_0a_dualcore_sample_CPU0¥Release
書き込みファイル	ap_rzt2_0a_dualcore_sample_CPU0.bin

5.1　スイッチ設定

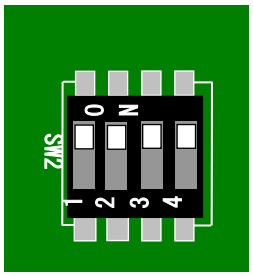
・SW2



<SW2 設定>
ブートモード　：シリアル FlashROM ブート/
USB ブート/SCI ブートの
いずれにも該当しないモード
JTAG Hash モード　：使用する（不問）

Fig4.1-1 デバッグ時のボード設定

・SW2



<SW2 設定>
ブートモード　：シリアル FlashROM ブート
JTAG Hash モード　：使用する（不問）

Fig5.1-2 シリアル FlashROM ブート時のボード設定

ご注意

・本文書の著作権は株式会社アルファプロジェクトが保有します。
・本文書の内容を無断で転載することは一切禁止します。
・本文書に記載されているサンプルプログラムの著作権は株式会社アルファプロジェクトが保有します。
・本サンプルプログラムで使用されているミドルウェアおよびドライバの著作権はルネサス エレクトロニクス株式会社が保有します。
・本文書に記載されている内容およびサンプルプログラムについてのサポートは一切受け付けておりません。
・本文書の内容およびサンプルプログラムに基づき、アプリケーションを運用した結果、万一損害が発生しても、弊社では一切責任を負いませんのでご了承ください。
・本文書の内容については、万全を期して作成いたしましたが、万一ご不審な点、誤りなどお気付きの点がありましたら弊社までご連絡ください。
・本文書の内容は、将来予告なしに変更されることがあります。

商標について

・RZ および RZ/T2M は、ルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
・Arm®は Arm Ltd.の登録商標です。
・e2 studio は、ルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。
・J-Link は、SEGGER Microcontroller GmbH & Co. KG の登録商標もしくは商標です。
・Flexible Software Package は、ルネサス エレクトロニクス株式会社の登録商標、商標または商品名称です。

・その他の会社名、製品名は、各社の登録商標または商標です。

ALPHA PROJECT
株式会社アルファプロジェクト
〒431-3114
静岡県浜松市中央区積志町 834
https://www.apnet.co.jp
E-Mail: query@apnet.co.jp