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

第1.4版 2025年04月15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	2024-04	_
GCC for Renesas RZ	12.2.1.arm-12-24	_
FSP	2.1.0	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	¥sample¥ap_rzt2_0a_dualcore_sample_CPU0
プロジェクトフォルダ(CPU0)	
DualCore サンプルプログラム	¥sample¥ap_rzt2_0a_dualcore_sample_CPU1
プロジェクトフォルダ(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
	L _{¥ 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

<¥sam	iple フォルダ内>		
	AlphaProject.ap_rzt2_0a.2.1.0.	•••	AP-RZT2-0A パックファイル
	pack		
	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_	•••	AP-RZT2-0A DualCore サンプルプログラム
CPU0 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
CPU0 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)
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
<¥samp	ble¥ap_rzt2_0a_dualcore_sample_CPL	J1¥scrip	ot 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。
- 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。
- 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。

※必ず CPU0 のサンプルをビルドした後 CPU1 のサンプルをビルドしてください。 CPU1 のサンプルのみのビルドは行えません。

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=""></sw2>	
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。

rztfsp_v1_0_0 - e ² studio				-		\times
ファイル(F) 編集(E) ナビゲート(N) 検索(A) プロジェクト(P) Renesas Views 実行(R)	ウィンドウ(W) ヘルプ(H)					
🔦 🎄 🔳 🎄 デバッグ(B) 🗸 💽 ap_rzt2_0a_ssbl_sample_CPU1	Relei 🗸 🌼 🗄 📩 🖷 📳	🗟 📎 🕶 🚳 📼	010 · 🖶			
× = = = = = = = = = = = = = = = = = = =	\$ ⊪ + Ⅲ ≌ ឱ & ₽	🕸 💋 🎓 🔗	• 回 • 同 •	*;- =;* 🖕 🗸	→ +	1
€ 1 ap_rzt2_0a_dual	core_sample_CPU1 Debug	1	Q	😭 🚾 C/C+	+ 🏇 🤊	デバッグ
<u>▶ プロジェクト・エクスプローラー × □ □ </u> 2 ap_rzt2_0a_dual	core_sample_CPU0 Debug		- C (x)	· 変 × »。		- 0
デバッグ(D)		>		<u>∦</u>		eń 8
> デ ap_rzt2_0a_dualcore_sample_CPU0 [Debug] デパッグの構成(B).			ŧ	前	型	
> jej ap_rzt2_0a_dualcore_sample_CPU1 お気に入りの編成(/)					
						_
	進行状 UXモリーノ U」う	テバック・ 近 問題	Debugg 🌸 .		• 🗖 -	- U
CDT ビルド・コンソール Jap rzt2 0a dua	core sample CPU0	× ↔ .	U 🔄 🔛 🖓			
17466 112 60284 7786	2 13026 ap_rzt2_0a_dualc	core_sample_CPU0.e	lf			ĉ
< > > <						>
przt2_0a_dualcore_sample_CPU0		8.				10

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

■ テバッグ機成 構成の作成、管理、および実行	×
	名前(N): [P,rzt2.0a_dualcore_sample_CPU0 Debug メイン 珍 Debugger ● Startup □ 共通 「シ ソース プロジェクト(P): ap_rzt2_0a_dualcore_sample_CPU0 CC++ アプリケーション: Debug/ap_rzt2_0a_dualcore_sample_CPU0.elf 変数(V) プロジェクトの検索(H) 参照(6) を動能に必要に応じてだしド Build Configuration: Select Automatically 〇 自動だルドを有効にする ○ 自動だルドを有効にする ④ ワークスペース協定の使用 ワークスペース協定の構成
18 項目のうち 15 項目がフィルターに一致	前回保管した状態に戻す(V) 適用(Y)
0	デバッグ(D) 閉じる



[Debugger] タブを選択し、 [Debug hardware] を [J-Link ARM] 、
 [Target Device] を「R9A07G075M24_CR52_0」に設定されていることを確認してください。

■ デバック準成 構成の作成、管理、および実行	
マルタスカ こ C(C++ アブリケーション) こ C(C++ アブリケーション) こ ACS Script こ GDB Simulator Debugging こ GDB Simulator Debugging (RH850) こ GDB Simulator Debugging (RH850) こ GDB A(-ドウェア・デバッキング) I Ava アブリケーション I Ava アブリケーション I Ava アブリケーション I Ava アブレット マ Renesas GDB Hardware Debugging □ ap.rt2, 0x dualcore_sample_CPU1 Debug □ p.rt2, 0x dualcore_sample_CPU1 Debug	名範(N): ap_rtt2_0a_dualcore_sample_CPU0 Debug ③ メイン ③ Debug hardware: →Link ARM Target Device: 07G075M24_CR52_0 GOB Settings Connection Settings デパッグ・ツール設定 GOB 接段設定: ④ ローカル GDB サーバーを自動起動 ホスト名または IP アドレス: localhost 〇 リモート GDB サーバーへ接続 GDB ボート番号: 61234 接続タイムアウト(s): 30 GOB GOB GOB 3マンド: arm-none-eabl-gdb 季照
18 項目のうち 15 項目がフィルターに一致	前回保管让た状態に戻す(V) 適用(Y)
0	デパッグ(D) 閉じる

デバッグ設定を新規作成した場合などに、 [Target Device] に「R9A07G075M24_CR52_0] 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。

(国) デバッグ構成			
構成の作成、管理、および実行			-
Run Break Time Measurement Operating Frequency must not be empty a	and must be between 0.001 & 10000.0MHz		300
Provident internet openanty requerty may not be any y			
	名前(N): ap_rzt2_0a_dualcore_sample_CPU0 Debug		
フィルタ入力	📄 メイン 隊 Debugger 🕨 Startup 🔲 共通 🧤 ソ-	-ス	
 C/C++アブリケーション C/C++リモート・アブリケーション EASE Script G GDB OpenOCD Debugging 	Debug hardware: J-Link ARM \checkmark Target Device:	107G075M24_CR52_0	
GDB Simulator Debugging (RH850)	Lite CEL-Flach	10103	¥ A
GDB ハードウェア・デバッギング	CELStart	0x0	- · · ·
🗾 Java アプリケーション	CFI End	0x0	
💌 Java アプレット	✓ Semihosting		
✓ C [™] Renesas GDB Hardware Debugging	Semihosting breakpoint address		
c* ap_rzt2_0a_dualcore_sample_CPU0 Debug	✓ RTOS		
ap_rzt2_0a_dualcore_sample_CPU1 Debug	デバッグビューにRTOSのスレッド情報を表示する	はい	~
e* ap_rzt2_0a_dualcore_sample_CPU1 Release	多数のスレッドを使用するプロジェクトをデバッグする	いいえ	~
Renesas Simulator Debugging (RX, RL78)	✓ System		
日、リモート Java アフリケーション	Allow caching of flash contents	はい	v
- 起動クループ	> 時間計測		
	フレークまでの実行時間を計測する	1201	~
	Count Every Core Cycle	(du)	~
	Operating Frequency [MHz]		
	~ リセット時の動作		
	リセットコマンドの動作	U ENF	
	リセット後のPC値	$[(t_i)] \rightarrow [(v_i)] \rightarrow [(v_i)]$	変更
	リセット後のCPSR値		~~]
18 項目のうち 15 項目がフィルターに一致		前回保管した状態に戻す(V)	愈用(Y)
(?)		デバッグ(D)	閉じる



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。

	名前(N): ap_rzt2_0a_dualcore_sa	mple_CPU0 Debug			
	📄 メイン 🕸 Debugger 🕟 Star	tup 🦻 🌽 ソース 🔲 井	:通		× 1
 C (C++) (5) - (3) C (C++) (F-ト・アプリケーション 	- イメージとシンボルをロード				
EASE Script GDB OpenOCD Debugging GDB Simulator Debugging	ファイル名 マープログラム・パイナリー 「ap r	ロード・タイプ イメージとシンボル	オフセット	接続時 Yes	追加
 C GDB ハードウェア・デバッギング Java アプリケーション 					編集
Mar Java アプレット で Renesas GDB Hardware Debugging					上へ
ap_rzt2_0a_dualcore_sample_CPU0 Debug ap_rzt2_0a_dualcore_sample_CPU1 Debug ap_rzt2_0a_dualcore_sample_CPU1 Belease					下へ
ap_rzt2.0a_dualcore_sample_CPU1 Release Renesas Simulator Debugging (RX, RL78)	ランタイム・オブション				
■ 起動グループ	□ プログラム・カウンター設定先(16進): □ ブリークポイント設定先: main				
	□ 再開				
	コマンドを実行				
	set \$cpsr=0x200001da				^

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。

〕 値 ¥ 目 ∀ ▼ フィルタ入力	名前(N): ap_rzt2_0a_dualcore_sar	nple_CPU0 Debug tup 時/ ソース 🛄 共	ŧ通		
 C/C++ アブリケーション C/C++ リモート・アブリケーション 	- イメージとシンボルをロード				~
 ■ EASE Script © GD8 OpenOCD Debugging © GD8 Simulator Debugging (RH850) © GD8 /\-ドウェア・デバッギング 	ファイル名 ビ ブログラム・バイナリー [ap_r	ロード・タイプ イメージとシンボル	オフセット 0	接続時 Yes	追加 編集
 Java アブリケーション Java アブリケーション Java アブリケーション ap_rtt2_0a_dualcore_sample_CPUD Debug ap_rtt2_0a_dualcore_sample_CPUD Debug ap_rtt2_0a_dualcore_sample_CPUH Release Reneasa Simulator Debugging (RX, RL78) リモート Java アブリケーション 起動グループ 					除去 上へ 下へ
	ランタイム・オブション □ プログラム・カウンダー設定先(16進): □ ブレークポイント設定先: main □ 再開				
	コマンドを実行 set \$cpsr=0x200001da //				
					~



9. ボードとの接続が完了することを確認します。



- プロジェクト「ap_rzt2_0a_dualcore_sample_CPU1」を対象に、4~7 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」または「CR52_0」→「CR52_1」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。

E 🛪 🖻 🖌 🗖 .	名前(N): ap_rzt2_0a_dualcore_sar	nple_CPU1 Debug			
	📄 メイン 🕸 Debugger 🕨 Star	tup 🔲 共通 🧤 ソ・	-ス		× 1
□ C/C++ リモート・アプリケーション □ C/C++ リモート・アプリケーション □ CASE S+	イメージとシンボルをロード				
GDB OpenOCD Debugging	ファイル名	ロード・タイプ	オフセット	接続時	追加
雪 GDB Simulator Debugging (RH850) 罰 GDB ハードウェア・デバッギング		イメージとシンボル	0	Yes	編集
□ Java アプリケーション ③ Java アプレット					除去
Renesas GDB Hardware Debugging					上へ
ap_rzt2_0a_dualcore_sample_CPU0 Debug					下へ
③ Reneas Simulator Debugging (RX, RL78) ① リモート Java アブリケーション 夏 建語グループ	 ランタイム・オブション コブログラム・カウンター設定先(16 ジブレークボイント設定先: 再開 コマンドを実行 set \$cpsr=0k20001da 	進): main			Ĵ

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。

🖸 起動	b ×
<u>^</u>	前回起動したRenesas GDB デバッグ・セッションがまだ実行中です。 以前のデバッグ・セッションをすべて終了させて新しいデバッグ・セッションを開始しますか? 'No'を選択し、起動中のデバッグ・セッションを停止しない場合は動作が不安定になる可能性が あります。
□常に	停止する はい(Y) いいえ(N) キャンセル

※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。

rztfsp_v1_0_0 - ap_rzt2_0a_dualcore_sample_CPU0/src/hal_entry.c - e ² studio	-		×
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(N	P) Renesas Views 実行(R) ウィンドウ(W) ヘルプ(H)		
💊 🐞 🔳 漆 デバッグ(B) 🗸 🖻 ap_rzt2_0a_dualcore_sample_	CPU1 🗸 🔅 🗄 🖛 🔚 🐚 🛞 🕶 🗞 🕶 🗟 🗄 🏪		
🔌 🗈 🗉 🛢 🕅 R. 👁 Le 🏟 🗮 党 🚳 🗞 🚸 🕶 💁 🕶 🔩 🗸	🍓 🕪 🕶 🕼 🐒 🛃 🕸 🖉 🖉 🤔 🥭 🔗 🕶 🍠		
월 ▼ 福 ▼ 🏷 💞 🗇 ▼ 🖒 ▼ 🛃	Q i 🖻 🖬 Q	/C++ 🎄 デ/	バッグ
💿 דאָטיזע 🔁 אין דער	€ hal_entry.c × € system_init() a »5 □	× =	° 🗆
✓ E [™] ap. rzt2. 0a. dualcore. sample. CPU0 Debug [Renesas GDB Hardware Debugging]	134 0000019c err = R_SCI_UART_Open(&g_uart0_ctrl, &g_u ~	X X	۱ 🎉
Alge ap_rzt2_0a_dualcore_sample_CPU0.elf [1] [cores: 0] Inread #1 1 (single core) [core: 0] (Running)	<pre>135 135 135 135 135 135 135 135 135 135</pre>		
arm-none-eabi-gdb (7.8.2)	137 1 138 /* UARTデーな受信まで待機 */	Q.	
Renesas GDB server (Host)	139 00000250 ⊖ while(s_recv.head == s_recv.foot)	DC 0/0 04	0/4
✓ ☑ ap_rzt2_0a_dualcore_sample_CPU1 Debug [Renesas GDB Hardware Debugging]	141 00000264 R_BSP_SoftwareDelay(10, BSP_DELAY	PC: 0/8 UA:	0/4
Process [1] [cores: 0]	142 143 /8 CDUI(ChurthayTU)(-2-5Mask).abs	88 🛃	
Inread #11 (single core) [core: 0] (Kunning)	144 0000026e ⊖ if(data_receive_flag == 1)	タイプ	^
Renesas GDB server (Host)	145 { 145 accession flag at		ы
			н́г I
	148 /* Read data from the shared	🗌 🧉	ŀν
	149 OTTSET = SHARED_MEMORY_SIZE /	□ ° .	10 1
	assert(ESP_SUCCESS == err); ∨	S DI	,
	< >	709191	• •
	🗐 コン 🗙 🚟 レジ 🖏 進 🔋 Xモ 🗓 デバ 🛃 間 🖳 De 🁒 スマ	🛛 Xモ 🧧	° 🗆
	🗉 🗙 🔆 🗟 🛃 🛃 🚝 🚝	🛃 🗐 👻	📑 👻
	ap_rzt2_0a_dualcore_sample_CPU1 Debug [Renesas GDB Hardware Debugging]		
			0
< >>	<		>
実行中			
	<u></u>		10

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。

※必ず CPU0 のサンプルをビルドした後 CPU1 のサンプルをビルドしてください。 CPU1 のサンプルのみのビルドは行えません。

rztfsp_v1_0_0 - ap_rzt2_0a_dualcore_sample_CPU0/src/ha	al_entry.c - e² studio	- D >	×
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナピゲート(M	l) 検索(A) プロジェクト(P) Renesas Views 実行(R) ウィンドウ(W) ヘルプ(H)		
🔦 🎄 🔳 🎄 デバッグ(B) 🗸 🖻 ap_	rzt2_0a_ssbl_sample Debug 💿 🗸 🔅 🗄 🛨 🐨 🔚 🐚 🛛 🗞 🕶 🚳 🗄 🏪		
🔪 🗈 🗉 🛤 🔍 🕾 . 🤉 iii 🚺	Ø 😂 🖉 🗸	A	
월 ▼ 취 ▼ ∜ ⊄ ♥ ↔ ▼ ↔ ▼ <mark>-1</mark> ユー1	ザプロジェクト:Release ビルド	Q 🗄 😰 🖬 ር/C++ 🎄 テ/(୭	ッグ
🎄 デバッグ 🎦 プロジェクト・エクスプ 🗙	T main.c Ac hal_entry.c X	🗖 (X)=変 🎱 ブ× 🔭 🖳	
	* function * */	^ 💻 🖶 💥 💥 🤯 🗟 🗶 👘	
> 🚰 ap_rzt2_0a_dualcore_sample_CPU0 85	<pre>static uint16_t sci_thread_recv(uint8_t * buf, T_SciLingBuf *</pre>	3 0 A A S 4	
> 😂 ap_rzt2_0a_dualcore_sample_CPU1 86	<pre>static void cpu1_data_load(void); static void cpu1_exec_start(void);</pre>	8	
88 89	<pre>static uint32_t calc_checksum (uint16_t * const p_param); static fsp err t ospi set quad mode enable (xspi ospi instanc</pre>		
	C COLICE K SHAKED WEMOKA EXample abbilcactou		
SSBL -	プロジェクト : Debug ビルド		
101	<pre>uint8_t receive_buffer[SHARED_MEMORY_SIZE] = {0};</pre>		
103	uint8_t send_buf[BUFF_SIZE]; uint32_t send_len = 0;		\sim
105	<pre>s_recv.head = 0;</pre>	¥	×
			_
VE 🗐	9- × 🔠 レジスタ 🖏 進行状 🚺 メモリーフ 🛄 テバック・ 🚉 問題 🔛 Deb	ugg 🐝 スマート・ 🚺 Xモリー	
(終了)	an 1772 Da dualcore sample CPUD Debug [Renesas GDB Hardware Debugging] Ren	esas GDB server (Host) (Terminated 202	¶ ▼ 3/02
-112 3 -	ap_rac_os_daarcorc_sample_er oo bebag (nenesas obb maraware bebagging) nen	233 GDD Jerrer (11037) (Terminated 202	^
	ver for kenesas tarpets.	>	
😂 ap_rzt2_0a_ssbl_sample			

AN1651 AP-RZT2-OA (RZ/T2M CPU BOARD) DualCore サンプルプログラム解説 ©2023 Alpha Project Co., Ltd. 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=""></sw2>	
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 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

(ハクスカ (ハクス (ハク ((((((((名前(N): ap_rtt2_0a_ssbl_sample Debug スイン 体 Debugger ▶ Startup □ 共通 ちッソ プロジェクト(P): ap_rtt2_0a_ssbl_sample C/C++ アブリケーション: Debug¥ap_rtt2_0a_ssbl_sample.eff を動前に必要に応じてビルド <u>Build Configuration</u> : Select Automatically ○ 自動ビルドを有効にする	-ス 参照(B) 変数(V) プロジェクトの検索(H)、参照(R) 〇 自動ビルドを告効にする
rc_y ye_reccossusanijet bebugg © Renesas Simulator Debugging (RX, RL78) 型 リモート Java アプリケーション ■ 起動グループ		
酒日のうち 16 酒日がフィルターに一致		前回保管した状態に戻す(V) 適用(Y

[Debugger] タブを選択し、 [Debug hardware] を [J-Link ARM] 、
 [Target Device] を [R9A07G075M24_CR52_0] に設定されていることを確認してください。

🐻 デバッグ機成	- 🗆 X
構成の作成、管理、および実行	TO T
 ・・アブリケーション ・ ・ ・	名範(N): ap_rt2_0a_scbl_sample Debug → X12 [弦 Debugaer] → Startup [共通 ち ソース Debug hardware: -Link ARM ▼ Target Device: R9A07G075M24_CR5 GDB Settings Connection Settings デパッグ・ツール設定 GDB 接続設定: ④ ローカル CDB サーパーを自動起動 ホスト名または IP アドレス: localhost 〇 リモート GDB サーパーへ接続 GDB ポート番号: 61234 接続タイムアウト(s): 30 ▼ GDB Step Mode ▲ Additional GDB Server Arguments ▲
19 項目のうち 16 項目がフィルターに一致	前回保管した状態に戻す(V) 週用(V)
?	デバッグ(D) 閉じる

デバッグ設定を新規作成した場合などに、 [Target Device] に「R9A07G075M24_CR52_0] 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。

	名前(N): ap_rzt2_0a_ssbl_sample Debug		
ルタ入力	📄 メイン 🏁 Debugger 🕨 Startup 🔲 共通 🧤 ソ	-7	
 C (C++ アブリケーション C (C++ リモート・アブリケーション E ASE Script GFR One OCD Debugging 	Debug hardware: J-Link ARM V Target Device:	R9A07G075M24_CR5	
GDB Simulator Debugging (RH850)	GDB Settings Connection Settings 77.99*7-10	10.2	
GDB ハードウェア・デバッギング	Use CFI-Flash	0.0	~
lava アプリケーション	CFI Start	0x0	
ava 77/0%	CFIEnd	0x0	
Renesas GDB Hardware Debugging	Seminosting		
ap rzt2 0a dualcore sample CPU0 Debug	Seminosting breakpoint address		
ap rzt2 0a dualcore sample CPU1 Debug		(†1)	
ap rzt2 0a dualcore sample CPU1 Release	アハックビューにKIOSのスレット信報を表示する タキャストールドを使用するプロジョクレキデバルグナス	1917	
ap rzt2 0a ssbl sample Debug	多数のスレットを使用するフロシェクトをナバックする	5 002	`
Renesas Simulator Debugging (RX, RL78)	V System	(†1)	
リモート Java アプリケーション	Allow caching of flash contents	140'	1
↓ 記動グループ	▼ 時間計測 ゴレークまでの実行時間を計測する	(†1)	
	Count Fuero Core Curle	(1)	
	Count Every core cycle	1010	
	Utwicking requency [WH2]		
	マ クビクド時の動作		
	リビットコイントの動力ト	560F	_ ·

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。

📴 デバッグ構成

イメージとシンボルをロード ファイル名 ジ ブログラム・バイナリー [ap_r	ロード・タイプ イメージとシンボル	オフセット	接続時	
		0	Yes	追加 福集 除去
ランタイム・オプション □ プログラム・カウンター設定先(16 □ プレークポイント設定先: □ ★ ##	進): main			
	ランタイム・オブション □ プログラム・カウンター設定先(16 ☑ ブレークボイント設定先: □ ★ ##	ランタイム・オブション □ プログラム・カウンター設定先(16進): □ ブレークボイント設定先: main □ 〒 ##	ランタイム・オブション □ プログラム・カウンター設定先(16進): □ ブレークボイント設定先: main	ランタイム・オブション □ プログラム・カウンター設定先(16進): □ ブレークポイント設定先: main □ ★ ##



– 🗆 X

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。

🙆 ダウンロード・モジュールの	の追加		×
ダウンロード・モジュール名の	指定:		
変数 プ	ロジェクトの検索	ワークスペース	ファイル・システム
		OK	キャンセル

🗐 ダウンロード・モジュールの追加	_		×
ワークスペースのリソースを選択			
✓			^
> 🥭 .settings			
> 🗁 Debug			
> 🔁 ra			
V 🦳 Release			
> 🔁 rzt			
> 🔁 rzt_gen			
> - src			
ap_rzt2_0a_dualcore_sample_CP00.bin			
ap rzt2 0a dualcore sample CPU0.elf.	in		
ap rzt2 0a dualcore sample_ci Obiena	D		
a makefile	·		
makefile.init			
bjects.mk			
o sources.mk			~
OK		キャンセ	JV

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。

🖻 🐢 🗎 🗶 🕞 🏹 👻	名前(N): ap rzt2 0a ssbl sample Debug					
(ルタ入力	📄 メイン 🏂 Debugger 📄 Startup 🔲 共通 🍕 ソー	ح				
C/C++ アプリケーション				~		
C/C++ リモート・アプリケーション	イメージとシンボルをロード					
EASE Script GDB OpenOCD Debugging GDB Simulator Debugging GDB Simulator Debugging	ファイル名 ロード・タイプ	オフセット	接続時	追加		
GDB ハードウェア・デバッギング	✓ JUJJA (1) J= [ap_n リンボルのみ	0	Yes	編集		
] Java アプリケーション		-		除去		
Renesas GDB Hardware Debugging				ΕΛ		
ap_rzt2_0a_dualcore_sample_CPU0 Debug				<u> </u>		
ap_rzt2_0a_dualcore_sample_CPU1 Release ap_rzt2_0a_dualcore_sample_CPU1 Release ap_rzt2_0a_ssbl_sample_Debug						
Renesas Simulator Debugging (RX, RL78)	- ランダイム・オンジョン □ プログラム・カウンター設定先(16弾):					
。リモート Java アプリケーション 1 記動グループ	□ ブレークポイント設定先: main					
	コマンドを実行					
	set \$cpsr=0x200001da			~		
				~		

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

デバッグ爆成 構成の作成、管理、および実行					×
 ・ ・ ・	名前(N): ap_rzt2_0a_ssbl_sample []	Debug Tup II 共通 時 ソース	1		
 C GDB OpenOCD Debugging E GDB Simulator Debugging (RH850) E GDB /-ト³エア・デバッギング Java アブリケーション Ø Java アブリット ✓ C Renesas GDB Hardware Debugging E ap_rrt2_0a_dualcore_sample_CPU0 Debug E ap_rrt2_0a_dualcore_sample_CPU1 Debug 	J71ル名 図 プログラム・パイナリー [ap_r 図 ap_rzt2_0a_dualcore_sa	ロート・ダイノ イメージとシンボル シンボルのみ	オノゼット 0 ・ 0	接続時 Yes Yes	追加
 (c) ap_rtt2_0a_dualcore_sample_CPU1 Kelease (c) ap_rtt2_0a_ssbl_sample Debug (c) Renesas Simulator Debugging (RX, RL78) (c) リモート Java アブリケーション (c) 起動グループ 	- ランタイム・オブション □ プログラム・カウンター設定先(16) ☑ ブレークポイント設定先: □ 再開 □マッドを実行 set \$cpsr=0x200001da	#): main			
19 項目のうち 16 項目がフィルターに一致				前回保管し	た状態に戻す(V) 適用(V) デパッグ(D) 閉じる

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。

		: ap_rzt2_0a_ssbi_sample (ソ	Debug tup 🔲 共通 🤤 🤅	ノース			~
 C/C++ アノリリーション C/C++ リモート・アプリケーション C/C++ レモート・アプリケーション 	-1×-	ジとシンボルをロード					
』 EASE Script] GDB OpenOCD Debugging] GDB Simulator Debugging (RH850)] GDB ハードウェア・デバッギング	77 12	イル名 プログラム・バイナリー [ap_r ap_rtt2_0a_dualcore_ca	ロード・タイプ イメージとシンボル シンボルのみ		オフセット の	接続時 Yes Ver	追加 編集
」 Java アプリケーション ④ Java アプレット ④ Renesas GDB Hardware Debugging ⑥ ap_rzt2_0a_dualcore_sample_CPU0 Debug		ap_rzcz_ua_uuarcore_sa	2210000	•			除去 上へ
C ap_rt2_0a_dualcore_sample_CPU1 Release C ap_rt2_0a_ssbl_sample_Debug P cenesas Simulator Debugging (RX, RL78) リモート Java アプリケーション ま数グループ	- 524 [] 7 [] 7 [] 7	ロイム・オブション ログラム・カウンター設定先(16 レークボイント設定先: 5開	進): main				
	ראב - Set	バを実行 \$cpsr=0x200001da					^ ~



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()"」の直前 でブレークした場合、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

デバッグ掲載 構成の作成、管理、および実行	×
○ ● ● ● ● ● ○ > ● <th>名前(N): ap_rtt2_0a_dualcore_sample_CPU1 Release プロジェクト(P): ap_rtt2_0a_dualcore_sample_CPU1 文化 第 Debugger >>>>>>>>>>>>>>>>>>>>>>>></th>	名前(N): ap_rtt2_0a_dualcore_sample_CPU1 Release プロジェクト(P): ap_rtt2_0a_dualcore_sample_CPU1 文化 第 Debugger >>>>>>>>>>>>>>>>>>>>>>>>
?	デバ(ッグ(D) 閉じる

15. [Debugger] タブを選択し、 [Debug hardware] を [J-Link ARM] 、 [Target Device] を [R9A07G075M24_CR52_1] に設定されていることを確認してください。

デバッグ機成 構成の作成、管理、および実行	×
 ・ ・ ・	名前(N): ap_rrt2_0a_dualcore_sample_CPU1 Release → X12
19 項目のうち 16 項目がフィルターに一致	前回保管した状態に戻す(V) 適用(Y)
0	デバッグ(D) 閉じる

16. プログラム・バイナリーの「ロード・タイプ」を「シンボルのみ」に変更してください。

2 🕫 🗎 🗶 🖻 🏹 🗸	名前(N): ap_rzt2_0a_dualcore_sample_CPU1 Re	lease		
イルタ入力	📄 メイン 🏂 Debugger 🕟 Startup 🔲 共通	₩ <u>/</u> ソース		
 C/C++ アプリケーション C/C++ リモート・アプリケーション ■ FASE Function 	イメージとシンボルをロード			~
 GBB OpenOCD Debugging GDB OpenOCD Debugging (RH850) GDB N→ドウェア・デバッギング Java アプリケーション Java アプリケーション Renesas GDB Hardware Debugging ap_rt2_0a_dualcore_sample_CPU0 Debug ap_rt2.0a_dualcore_sample_CPU1 Debug ap_rt2.0a_dualcore_sample_CPU1 Release 	ファイル名 □ ブログラム・パイナリー [ap_r シンボルのみ	オフセット 0	接続時 Yes	道加… 編集… 除去 上へ 下へ
 ご ap.rt2_Qa_ssbl_sample Debug ご Renesas Simulator Debugging (RX, RL78) ユ リモート Java アブリケーション ス 起動グループ 	ランタイム・オブション ブログラム・カウンター設定先(16道): ブレークポイント設定先: 両期 コマンドを実行 「at Sense 0: 2000014b			
	set \$cpsr=0x200001da			

17. ブレークポイント設定先のチェックを外します

■ デバッグ機成 構成の作成、管理、および実行					- • ×
 アオルタスカ C C/C++ アブリケーション C C/C++ アブリケーション EASE Script C GDB OpenOCD Debugging G GDB Simulator Debugging (RH850) G GDB Simulator Debugging (RH850) G GDB J/ードウェア・デバッギング Java アプリケーション Java アプリケーション Java アプリケーション G Renesas GDB Hardware Debugging G ap.rt22, 0a. dualcore_sample_CPU1 Debug G ap.rt22, 0a. dualcore_sample_CPU1 Release C ap.rt22, 0a. dualcore_sample_CPU1 Release C ap.rt22, 0a. dualcore_brample C Renesas Simulator Debugging (RK, RL78) リモート Java アプリケーション ま動グ/ルーブ 	名前(N): <u>ap.rt2 0a.dualcore sat</u> スイソ 体 Debugger ▶ Stat イメージとシンボルをロード ファイル名 ビ ブログラム・パイナリー [ap_r ランダイム・オブション □ ブログラム・カウンター設定先116 □ ブレークボイント設定先: □ 再開 コマンドを実行 set \$cpsr=0x200001da	mple_CPU1 Release tup ① 共通 50 ロード・タイプ シンボルのみ 違): main	ソース オフセット 0	接続時 Yes	✓ 道加… 通本 通本 「 」
19 項目のうち 16 項目がフィルターに一致				前回保管	た状態に戻す(V) 適用(Y) デバッグ(D) 閉じる



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

° ፼ ∞ @ ¥ = 7 •	名前(N): ap_rrt2_0a_dualcore_san	nple_CPU1 Release tup 🔲 共通 🦃	y-z		2
 C/C++ アノリケーション C/C++ リモート・アプリケーション 	イメージとシンボルをロード				
 ■ EASE Script GDB OpenOCD Debugging GDB Simulator Debugging (RH850) GDB /\- トゥコア・ブ(ッギング Java アブリケーション Java アブリット Renesas GDB Hardware Debugging a_p.rtzLoa_dualcore_sample_CPU0 Debug a_p.rtzLoa_dualcore_sample_CPU1 Debug 	ファイル名 ☑ プログラム・パイナリ- [ap_r	ロード・タイプ シンボルのみ	オフセット 0	接続時 Yes	追加… 編集… 除去 上へ 下へ
[C] ap.rz2 Ua.dualcore_sample_UPU1 Kelease [C] ap.rz2 Ua.sbl.sample Debug IR Renesas Simulator Debugging (RX, RL78) III リモート Java アプリケーション 尾 起動グループ	 ランタイム・オブション □ブロクラム・カウンター設定先(16) □ブレークボイント設定先: □再開 □マンドを実行 Set \$cpsr=0x20001da 	進): [main			
				前回保管した	と状態に戻す(V) 適用(Y)

19. 「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。

	名前(N): ap_rzt2_0a_dualcore_sar	mple_CPU1 Release			
レタ入力	📄 メイン 🏇 Debugger 🕟 Star	tup 🔲 共通 🦤	ソース		
 C/C++ アブリケーション C/C++ リモート・アプリケーション 	イメージとシンボルをロード				*
EASE Script GDB OpenOCD Debugging	ファイル名	ロード・タイプ	オフセット	接続時	追加
≤″ GDB Simulator Debugging (RH850) 〒 GDB ハードウェア・デバッギング	✓ 70/54·/(1+)-[ap_r	シンホルのみ	0	Yes	編集
」 Java アプリケーション Ti Java アプレット					除去
Renesas GDB Hardware Debugging					上へ
est ap_rzt2_0a_dualcore_sample_CPU0 Debug ap_rzt2_0a_dualcore_sample_CPU1 Debug					下へ
ap_rzt2_0a_dualcore_sample_CPU1 Release ap_rzt2_0a_ssbl_sample Debug	ニョンタイト・オゴション				
Renesas Simulator Debugging (RX, RL78)	□ プログラム・カウンター設定先(16	進):			
a ジェート Java アフリケーション 11 起動グループ	□ ブレークポイント設定先:	main			
	□再開				
	コマンドを実行				
	set \$cpsr=0x200001da				^
					*



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。
- 20. 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。



21. 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	0×00000000
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	0x1000000

• 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"2024年6月14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。

^	名前	更新日時	種類サ	イズ	
4	ap_rzt2_0a_ethernet_sample	2023/02/17 10:04	ファイル フォルダー		
3	ap_rzt2_0a_uartcan_sample	2023/03/01 19:04	ファイル フォルダー		
	ap_rzt2_0a_usbf_sample	2023/03/01 19:04	ファイル フォルダー		
	ap_rzt2_0a_usbh_sample	2023/03/01 19:04	ファイル フォルダー		
	Z AlphaProject.ap_rzt2_0a.1.1.0.pack	2023/02/22 16:51	PACK ファイル	40 KB	
-	ap_rzt2_0a_devsetup.elf.srec	2023/02/13 14:41	SREC ファイル	109 KB	
	😹 ap_rzt2_0a_ssbl_sample.bin	2023/02/13 16:03	BIN ファイル	11 KB	
	■ FlashROM_Write.bat history.txt → U → -L →	C:¥AlphaProject¥workspace A 共有 表示			
	$\leftarrow \rightarrow \cdot 1$	> PC > OS (C:) > AlphaPro	oject > workspace		
	名前	ń ^	更新日時	種類	サイズ
		ap_rzt2_0a_ssbl_sample.bin	2023/02/13 16:03	BIN ファイル	
•		device_setup.py	2023/02/13 12:35	PY ファイル	
-	E 🖺	parameter_generator.py	2022/10/03 15:23	PY ファイル	

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。

∽ ssbl_sample.bin ssbl_sample.elt 쨆 Stirling - E:¥workspacei	更新日時 2023/02/27 15:56 2023/02/27 15:56 2023/02/27 15:56 2023/02/27 15:56 2023/02/27 15:56	種類 ファイル フォルダー ファイル フォルダー ファイル フォルダー BIN ファイル ELF ファイル	サイズ 11 KB 300 KB		
ssbl_sample.bin ssbl_sample.elf 盤 Stirling - E:¥workspacei	2023/02/27 15:56 2023/02/27 15:56 2023/02/27 15:56 2023/02/27 15:56 2023/02/27 15:56	ファイル フォルダー ファイル フォルダー ファイル フォルダー BIN ファイル ELF ファイル	11 KB 300 KB		
ssbl_sample.bin ssbl_sample.elf 器 Stirling - E:¥workspacei	2023/02/27 15:56 2023/02/27 15:56 2023/02/27 15:56 2023/02/27 15:56	ファイル フォルダー ファイル フォルダー BIN ファイル ELF ファイル	11 KB 300 KB		
ssbl_sample.bin ssbl_sample.elf Stirling - E:¥workspace3	2023/02/27 15:56 2023/02/27 15:56 2023/02/27 15:56	ファイル フォルダー BIN ファイル ELF ファイル	11 KB 300 KB		
ssbl_sample.bin ssbl_sample.elf Stirling - E:¥workspace	2023/02/27 15:56 2023/02/27 15:56	BIN ファイル ELF ファイル	11 KB 300 KB		
ssbl_sample.elf	2023/02/27 15:56	ELF ファイル	300 KB		
Stirling - E:¥workspace					
E:¥workspace¥e2studio	6 ≝E E L⊗ ;= AA 😨	ip_rzt2_0a_ssbl_sample¥	Debug¥ap_rzt2_0a_	_ssbl_s 🗖 🖸	
ADURESS 00 01 00000000 83 07 00000010 02 12 00000020 10 1F 00000030 10 1F 00000030 10 1F 00000040 13 00 00000050 6F 00	U2 U3 U4 U5 U6 U7 U OB E3 00 00 40 E3 3 81 E3 11 1F 81 EE 3 1F EE 02 10 81 E3 0 OF EE 6F F0 7F F5 0 80 E3 00 F3 6E E1 0 60 F1 00 3A 10 00 6	8 09 0A 0B 0C 0 0 0F 81 EE 11 1 4 00 9F 85 10 0 1 10 81 E3 4F F 0 00 0F E1 1F 0 8 10 9F E5 01 F 0 20 10 00 00 B	D UE UF U12 F 91 EE F OC EE 0 7F F5 0 CO E3 3 2E E1 F 70 47	23456789ABCU @.0 √√0 .糸 .浴 .×溶	上 へ 宅 · · · · · · · · · · · · · · · · · · ·
	ファイル(F) 編集(E) 検索・街 □	77/4(F) 編集(E) 検索・移動(S) 設定(O) 9/0+ ⁷ 9(W ○ (年 1) (1) (1) (1) (1) (1) (1) (1) (1) (1)	77/fu(F) 攝集(E) 検索・移動(S) 設定(O) 9/2/F 9(W) ^0/5/ (H) ○ (□) (□) (□) (□) (□) (□) (□) (□) (□) (□)	77/fu(F) 攝集(E) 検索・移動(S) 設定(O) 9/2/F 9(W) ^M7 (H) □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	27/fu(f) 攝集(E) 検索・移動(S) 設定(O) 9/2/F 9(W) ペルプ (H)



(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未尾に 「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(Dual Core).bat」を移動します。

名前 ^	更新日時	種類	サイズ		
ap_rzt2_0a_dualcore_sample_CPU0	2023/02/22 13:56	ファイル フォルダー			
ap_rzt2_0a_dualcore_sample_CPU1	2023/02/22 13:56	ファイル フォルダー			
Z AlphaProject.ap_rzt2_0a.1.1.0-ap010000.p	2023/02/13 9:30	PACK ファイル	40 KB		
ap_rzt2_0a_devsetup.elf.srec	2023/02/13 14:41	SREC ファイル	109 KB		
FlashROM_Write(DualCore).bat	2023/02/22 11:53	Windows バッチ ファ	3 KB		
🗃 history.txt	2023/02/13 16:04	TXT ファイル	1 KB		
ガイル 小一ム 共有	表示				
← · · · · · · · · · · · · · · · · · · ·	PC > OS (C:) > AlphaProjec	ct > workspace 更新日時	種	頬	サイズ
← · · · · · ← · · · · · · · · · · · · ·	PC > OS (C:) > AlphaProjec ^ a_devsetup.elf.srec	t > workspace 更新日時 2023/02/13 14:4	種 1 SR	頬 EC ファイル	サイズ 109 KB
← · · · · · · · · · · · · · · · · · · ·	PC > OS (C:) > AlphaProjec ^ a_devsetup.elf.srec a_dualcore_sample_CPU0.bin	t > workspace 更新日時 2023/02/13 14:4 2023/02/13 13:4	種 1 SR 3 BII	頬 EC ファイル N ファイル	サイズ 109 KB 48 KB
← · · · · · ← · · · · · · · · · · · · ·	PC > OS (C:) > AlphaProjec ^ a_devsetup.elf.srec a_dualcore_sample_CPU0.bin a_dualcore_sample_CPU1.bin	tt > workspace 更新日時 2023/02/13 14:4 2023/02/13 13:4 2023/02/13 13:4	種 1 SR 3 BII 7 BII	類 EC ファイル N ファイル N ファイル	サイズ 109 KB 48 KB 25 KB
 ← · · · ↑ ▲ · · · ↑ ▲ · · · · · · · · · · · · · · · · · · ·	PC > OS (C:) > AlphaProjec 	tt > workspace 更新日時 2023/02/13 14:4 2023/02/13 13:4 2023/02/13 13:4 2023/02/13 16:0	種 1 SR 3 BII 7 BII 3 BII	類 EC ファイル N ファイル N ファイル N ファイル	サイズ 109 KB 48 KB 25 KB 11 KB
 ← · · · ↑ ▲ âŋ ▲ êŋ ▲ êŋ	PC > OS (C:) > AlphaProjec ^ a_devsetup.elf.srec a_dualcore_sample_CPU0.bin a_dualcore_sample_CPU1.bin a_ssbl_sample.bin tup.py	tt > workspace 更新日時 2023/02/13 14:4 2023/02/13 13:4 2023/02/13 13:4 2023/02/13 16:0 2023/02/13 12:3	種 1 SR 3 BII 7 BII 3 BII 5 PY	類 EC ファイル N ファイル N ファイル N ファイル ファイル	サイズ 109 KB 48 KB 25 KB 11 KB 30 KB
← · · · · ◆ F	PC > OS (C:) > AlphaProjec Adevsetup.elf.srec a_dualcore_sample_CPU0.bin a_dualcore_sample_CPU1.bin a_ssbl_sample.bin tup.py I_Write(DualCore).bat	tt > workspace 更新日時 2023/02/13 14:4 2023/02/13 13:4 2023/02/13 13:4 2023/02/13 16:0 2023/02/13 12:3 2023/02/22 11:5	種 1 SR 3 BII 7 BII 3 BII 5 PY 3 Wi	頃 EC ファイル > ファイル > ファイル > ファイル ファイル ファイル ndows バッチ ファ	サイズ 109 KB 48 KB 25 KB 11 KB 30 KB 3 KB

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。
- ・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「Send program data. (S3)」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

l	GS C:¥Windows¥system32¥cmd.exe	—	
	SCI Download mode. Send program data. (S0) Load Program to BTCM Send program data. (S3) Start Boot Program on BTCM parameter_ap_rzt2_0a_dualcore_sample_CPU0.bin parameter_ap_rzt2_0a_dualcore_sample_CPU1.bin 1 個のファイルをコピーしました。 C:¥workspace¥RZT2V210¥Python_work_new(dualcore)¥device_setup.py : error: The specified COM port is invalid C:¥workspace¥RZT2V210¥Python_work_new(dualcore)¥device_setup.py : error: The specified COM port is invalid		A× A×
	C:¥workspace¥RZT2V210¥Python_work_new(dualcore)¥device_setup.py : error: The specified COM port is invalid C:¥workspace¥RZT2V210¥Python_work_new(dualcore)¥device_setup.py : error: The specified COM port is invalid 続行するには何かキーを押してくだざい	: CON : CON	l× l×

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。

- I 🎝	🚽 🗧 🛛 C:¥AlphaProject¥workspace			
ファイル	ホーム 共有 表示			
$\leftarrow \rightarrow$	~ ↑ 📑 > PC > OS (C:) > AlphaProject > works	pace		
4^	名前 ^	更新日時	種類	サイズ
	🗒 ap_rzt2_0a_devsetup.elf.srec	2023/02/13 14:41	SREC ファイル	10
	ap_rzt2_0a_dualcore_sample_CPU0.bin	2023/02/13 13:43	BIN ファイル	4
	ap_rzt2_0a_dualcore_sample_CPU1.bin	2023/02/13 13:47	BIN ファイル	2
	ap_rzt2_0a_ssbl_sample.bin	2023/02/13 16:03	BIN ファイル	1
<u> </u>	ap_rzt2_0a_ssbl_sample_xspi0.bin	2023/02/22 14:13	BIN ファイル	1
	🖺 device_setup.py	2023/02/13 12:35	PY ファイル	З
1	FlashROM_Write(DualCore).bat	2023/02/22 14:12	Windows バッチ ファ	
	Br parameter_ap_rzt2_0a_dualcore_sample.bin	2023/02/22 14:13	BIN ファイル	
	parameter_ap_rzt2_0a_dualcore_sample_CPU0.bin	2023/02/22 14:13	BIN ファイル	
	parameter_ap_rzt2_0a_dualcore_sample_CPU1.bin	2023/02/22 14:13	BIN ファイル	
	parameter_generator.py	2022/10/03 15:23	PY ファイル	1

メッセージ「Send program data. (S3)」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。
 本再後、ファイルを伊クレて問じます。

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。
- - SCI 通信の場合

C:¥Windows¥system32¥cmd.exe
SCI Download mode. Send program data. (SO) Load Program to BTCM
Send program data. (S3) Start Boot Program on BTCM parameter_ap_rzt2_0a_dua core_sample_CPU0.bin
parameter_ap_rzt2_0a_dua1core_samp1e_UPU1.bin 1個のファイルをコビーしました。
writeflash : Setup success. writeflash : Setup success. writeflash : Setup success
writeflash:Setup success. 続行するには何かキーを押してください...

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=""></sw2>	
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)

アプリケーションノート AN1651

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	R9A07G075M24_CR52_0		
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	: シリアル FlashROM ブート/
SW2 設定>	USB ブート/SCI ブートの
ブートモード	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 JTAG Hash モード</sw2>	: シリアル FlashROM ブート : 使用する(不問)	
ZWS 2 - ZWS 4 - C	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。

ALPHAPROJECT

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: guery@apnet.co.jp