# RA ファミリ 開発チュートリアル

6版 2024年12月02日

1. 位	既要 2	•
1.1	概要2	
1.2	使用環境	
1.3	CPU ボード スイッチ設定	r
1.4	各 CPU ボードの資料に関して	;
1.5	各 CPU ボードの Arm TrustZone セキュリティ機能について 6	,
2. 栈	幾能	,
2.1	サンプルプログラムの使用機能	,
3. e	2 studio を用いた動作方法 9	1
3.1	サンプルプログラムの動作方法	)
3.		
	1.1 インホート万法	۱
3.	1.1 インホート万法9 1.2 ビルド方法	
3. 3.	1.1 インホート方法9 1.2 ビルド方法	)
3. 3. <b>4. e</b>	<ol> <li>1.1 インホート方法</li></ol>	<b>)</b> + ,
3. 3. <b>4. e</b> 4.1	1.1 インホート方法	)
3. 3. <b>4. e</b> 4.1 <b>5. R</b>	<ol> <li>1.1 インホート方法</li></ol>	

# 1. 概要

## 1.1 概要

本アプリケーションノートは、弊社製 RA ファミリ CPU ボードを用いて、フレキシブル ソフトウェア パッケージ(FSP)を使用 したプログラムを作成・動作確認するまでのチュートリアル資料です。 各機能に関する詳細は、サンプルプログラムのソースコードやサンプルプログラム解説、ハードウェアマニュアル、回路図を ご覧ください。

## 1.2 使用環境

本アプリケーションノートの解説で用いる開発環境を以下に示します。

### <AP-RA6M-0A/1A>

ソフトウェア	バージョン	備考
e2 studio	V2023-04	_
FSP	V4.5.0	Flexible Support Package
GCC ARM Embedded	V10.3.1.20210824	_
Renesas Flash Programmer	V3.13.02	評価版使用可能

デバッガ	ハードウェアバージョン	備考
J-Link	V10	Segger Microcontroller Systems 社

### <AP-RA8M-0A / AP-RA8D-0A>

ソフトウェア	バージョン	備考
e2 studio	V2023-10	_
FSP	V5.1.0	Flexible Support Package
GCC ARM Embedded	V13.2.1.arm-13-7	_
Renesas Flash Programmer	V3.13.00	評価版使用可能

デバッガ	ハードウェアバージョン	備考
J-Link	V11	Segger Microcontroller Systems 社

<AP-RA8T-0A >

ソフトウェア	バージョン	備考
e2 studio	V2024-07	_
FSP	V5.5.0	Flexible Support Package
GCC ARM Embedded	V13.2.1.arm-13-7	_
Renesas Flash Programmer	V3.16.00	評価版使用可能

デバッガ	ハードウェアバージョン	備考
J-Link	V11	Segger Microcontroller Systems 社

※AP-RA6M-0A/1A, AP-RA8M-0A, AP-RA8D-0A, AP-RA8T-0A, と J-Link を直接接続することはできません。 CPU ボード側(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。

## 1.3 CPU ボード スイッチ設定

CPU ボードを動作させる際は、動作モードに応じてボード上のディップスイッチを設定する必要があります。

### (1) AP-RA6M-0A

出荷時設定:「ボード上の SDRAM を使用する」および「シングルチップモード」



SDRAM ON :ボード上の SDRAM を使用する OFF:ボード上の SDRAM を使用しない



BootMode SCI/USB: SCI ブートモード・USB ブートモード SGL :シングルチップモード (プログラム動作時やデバッガ使用時)

(2) AP-RA6M-1A

出荷時設定:「シングルチップモード」



BootMode SCI/USB : SCI ブートモード・USB ブートモード SINGLE : シングルチップモード (プログラム動作時やデバッガ使用時)

(3) AP-RA8M-0A, AP-RA8D-0A



Ethernet Ethernet を使用する場合: ON CAMERA を使用する場合: OFF



BootMode

SCI/USB: SCI ブートモード・USB ブートモード SGL :シングルチップモード (プログラム動作時やデバッガ使用時) (4) AP-RA8T-0A

出荷時設定:「シングルチップモード」



BootMode SCI:SCIブートモード・USBブートモード SGL:シングルチップモード (プログラム動作時やデバッガ使用時)

# 1.4 各 CPU ボードの資料に関して

資料・サンプルプログラムは、弊社 Web サイトのボード紹介ページで公開されています。

RA ファミリ CPU ボードシリーズ		https://www.apnet.co.jp/product/ra/index.html
AP-RA6M-0A	製品ページ	https://www.apnet.co.jp/product/ra/ap-ra6m-0a.html
AP-RA6M-1A	製品ページ	https://www.apnet.co.jp/product/ra/ap-ra6m-1a.html
AP-RA8M-0A	製品ページ	https://www.apnet.co.jp/product/ra/ap-ra8m-0a.html
AP-RA8D-0A	製品ページ	https://www.apnet.co.jp/product/ra/ap-ra8d-0a.html
AP-RA8T-0A	製品ページ	https://www.apnet.co.jp/product/ra/ap-ra8t-0a.html

1.5 各 CPU ボードの Arm TrustZone セキュリティ機能について

弊社製 RA ファミリ CPU ボードは搭載している CPU によって、Arm TrustZone セキュリティ機能(以下、セキュリティ機能 と記述します)の有無があります。

各 CPU ボードのセキュリティ機能の有無を以下に示します。

セキュリティ機能	CPU ボード名(CPU 種類)	参考資料
なし	AP-RA6M-0A(RA6M3)	_
あり	AP-RA6M-1A(RA6M5)	「RA6M5 グループ ユーザーズマニュアル ハードウェア編」
	AP-RA8M-0A(RA8M1)	「RA8M1 グループ ユーザーズマニュアル ハードウェア編」
	AP-RA8D-0A(RA8D1)	「RA8D1 グループ ユーザーズマニュアル ハードウェア編」
	AP-RA8T-0A(RA8T1)	「RA8T1 グループ ユーザーズマニュアル ハードウェア編」
		> 「セキュリティ機能」章

セキュリティ機能がある CPU のプログラム開発を e2studio で行う際、デフォルトでセキュリティ機能のメモリセキュリティ 属性の設定がビルド時に自動で行われます。(設定内容は Debug フォルダの rpd ファイルとして出力されます)



AP-RA6M-1AのUART サンプルプログラムの場合

また、メモリセキュリティ属性の設定はデフォルトで e2studio のプログラムダウンロード時に自動で CPU に書き込まれます。 ただし、Renesas Flash Programmer(以下、RFP と記述します)でプログラムをダウンロードする際には、セキュリティ 設定をユーザが行う必要があります。

本書の「5.1 Renesas Flash Programmer を使用した書き込み」では、メモリセキュリティ属性の設定を、全領域 非セキュアとする場合の手順も記述してありますので、ご参考ください。

# 2. 機能

# 2.1 サンプルプログラムの使用機能

以下のサンプルプログラムを用いて、各機能の確認をすることができます。 (2024 年 2 月現在の公開データの状況です)

### <AP-RA6M-0A>

ドキュメント番号	サンプルプログラム	機能
AN2003	標準サンプルプログラム	・CAN 通信
		・ネットワーク通信
		・QSPI FlashROM 読み書き
		・UART 通信
		・SD カード読み書き
		・USB ホスト メモリ読み書き
		・USB ファンクション 仮想 COM 通信
AN2004	LCD-KIT サンプルプログラム	・LCD 出力
		・タッチパネル操作
AN2005	WM-RP-10 サンプルプログラム	・無線 LAN 通信
AN2006	wolfSSL サンプルプログラム	・セキュリティ(wolfSSL)ネットワーク通信

### <AP-RA6M-1A>

ドキュメント番号	サンプルプログラム	機能
AN2007	標準サンプルプログラム	・CAN 通信
		・ネットワーク通信
		・UART 通信
		・SD カード読み書き
		・USB ファンクション 仮想 COM 通信
		・無線 LAN
		(ESP32-WROOM-32E)制御

### <AP-RA8M-0A>

ドキュメント番号	サンプルプログラム	機能
AN2009	標準サンプルプログラム	・CAN 通信
		・ネットワーク通信
		・UART 通信
		・SD カード読み書き
		・USB ホスト メモリ読み書き
		・USB ファンクション 仮想 COM 通信
AN2010	Pmod サンプルプログラム	・Pmod 動作
		(GPIO/UART/SPI/I2C)

### <AP-RA8D-0A>

ドキュメント番号	サンプルプログラム	機能
AN2009	標準サンプルプログラム	・CAN 通信
		・ネットワーク通信
		・UART 通信
		・SD カード読み書き
		・USB ホスト メモリ読み書き
		・USB ファンクション 仮想 COM 通信
AN2010	Pmod サンプルプログラム	・Pmod 動作
		(GPIO/UART/SPI/I2C)
AN2011	LCD-KIT サンプルプログラム	・LCD 出力
		・タッチパネル操作
AN2013	LCD-KIT+カメラサンプルプログラム	・LCD 出力
		・CMOS カメラ入力

### <AP-RA8T-0A>

ドキュメント番号	サンプルプログラム	機能
AN2015	標準サンプルプログラム	・CAN 通信
		・ネットワーク通信
		・UART 通信
		・USB ホスト メモリ読み書き
		・USB ファンクション 仮想 COM 通信
AN2016	Pmod サンプルプログラム	・Pmod 動作
		(GPIO/UART/SPI/I2C)

3. e2 studio を用いた動作方法

本章では、e2 studio を使用して CPU ボードを動作させる方法を説明します。 e2 studio および FSP(Flexible Software Package)はインストール済みであるものとしますので、 事前にインストールを行ってください。

3.1 サンプルプログラムの動作方法

本節では、弊社で作成した AP-RA6M-0A サンプルプログラムを例に動作方法の説明をいたします。

- 3.1.1 インポート方法
  - ① e2 studio を起動し、ツールバーの [ファイル] → [インポート] を選択します。



② [CMSIS Pack]を選択し [次へ]を選択し、pack ファイルをインポートします。

(本書では例として「sample¥ CustomBSP¥AlphaProject.ap\_ra6m\_0a.4.5.0.pack」をインポートする図を示します) すでに開発環境に pack ファイルをインポート済みである場合は、⑥へお進みください。

ゴンポート	_		×
選択 Import a CMSIS Pack into e2 studio			1
インポート・ウィザードの選択(S): フィルタ入力			
<ul> <li>✓ (ユーー)</li> <li>✓ CMSIS Pack</li> <li>② CMSIS Pack</li> <li>② Reneas &amp; Import Existing C/C++ Project into Workspace</li> <li>④ Reneas CA78K0R (CS+) ブロジェクト</li> <li>③ Reneas CC-RX/CC-RL (CS+) ブロジェクト</li> <li>④ アーカイブ・ファイル</li> <li>④ アーカイブ・ファイル</li> <li>④ ファイル・システム</li> <li>● フォルダーまたはアーカイブ由来のブロジェクト</li> <li>◎ 既存プロジェクトをワークスペースへ</li> <li>□ 設定</li> <li>&gt; ▷ C/C++</li> </ul>			~
(ア)         (ア)         (ト)         (L)         (L) </td <td>E)</td> <td>キャンセ</td> <th>μ</th>	E)	キャンセ	μ

③ [Import RA CMSIS Pack ウィンドウ]が表示されましたら、インポートする pack ファイルを選択してください。

Import CMSIS Pack			×
Import CMSIS Pack			X
Choose CMSIS pack to import			J
Saarife aant film			
			•
Specify device family:			
		~	
(?) < 戻る(B) 次へ(N) > 終了(	<u>F</u> )	キャンセル	



9 pack ファイルを選択後、メッセージ「No Renesas Family selected」が表示されるので、
 Specify device family から「Renesas RA」を選択してください。

Import CMSIS Pack		×
mport CMSIS Pack	ſ	
No Renesas Family selected	L	
Specify pack file:		
C:¥workspace¥ra6m¥AlphaProject.ap_ra6m_0a.4.5.0.pack		
C:¥workspace¥ra6m¥AlphaProject.ap_ra6m_0a.4.5.0.pack Specify device family:		
Ci¥workspace¥ra6m¥AlphaProject.ap_ra6m_0a.4.5.0.pack Specify device family:		···

### ⑤ [終了]を選択してください。

Import CMSIS Pack	_		×
Import CMSIS Pack		ſ	
Choose CMSIS pack to import			
Specify pack file:			
C:¥workspace¥ra6m¥AlphaProject.ap_ra6m_0a.4.5.0.pack			
Specify device family:			
Renesas RA			$\sim$
? <戻3(B) 次へ(N) > 終	了( <u>F</u> )	キャンセ	μ

⑥ もう-度ツールバーの [ファイル] → [インポート] を選択します。

771	「ル(E) 編集(E) ソース(S)	リファクタリング(T)	ナビゲート( <u>N</u> )	検索( <u>A</u> )	プロジェ	クト( <u>P</u> )	Renesas	<u>V</u> iews	実行( <u>R</u> )	ウィン
۵,	新規(N) ファイルを開く(.) ファイル・システムからプロジェク 最近のファイル	A トを開く	lt+シフト+N > 、	Configur	rations	€ -	~ ♂ • [2	לא איי איי	 • : Rº (	П
	閉じる(C) すべて閉じる(L)	Ctr	Ctrl+W l+シフト+W							
	保存(S) 別名保存(A)		Ctrl+S							
B	すべて保管(E) 前回保管した状態に戻す(T)	C	rrl+シフト+S							
	移動(∨) 名前を変更(M)		F2							
8	更新(F) 行区切り文字の変換(D)		F5 >							
8	印刷(P)		Ctrl+P	-						
24 24	インポート(I) エクスポート(O)			<b>B</b> 31	ンソール:	x 🌸	スマート・ブ	ラウザー	-۶۲ 💭	ŀ.∠⊐
	プロパティ(R)		Alt+Enter							
	ワークスペースの切り替え(W) 再開 終了/出口(X)		>							
				_						

⑦ [既存のプロジェクトをワークスペースへ]を選択し[次へ」を選択します。

<mark>e</mark> <sup>2</sup> インポート	_		×
選択 アーカイブ・ファイルまたはディレクトリーから新規プロジェクトを作成します。		Ľ	5
Select an import wizard:			
フィルタ入力			
<ul> <li>□、ファイル・システム</li> <li>□、フィルダー表たはアーカイブ由来のプロジェクト</li> <li>☆ 既存プロジェクトをワークスペースへ</li> <li>□</li> </ul>			^
> > > > > > > > C/C++ > > > > > > S oft > > S VN			
<ul> <li>&gt; Contracting</li> <li>&gt; Cont</li></ul>			
> ひ マスジ > > テーム > >> テ 実行/デバッグ			*
? < 戻る(B) 次へ(N) > 終了(f)	E)	キャンセ	JL

⑧ [ルート・ディレクトリーの選択]を選択し、[参照]からサンプルプログラムのフォルダを選択します。

<mark>e</mark> <sup>2</sup> インポート	_	
プロジェクトのインポート 既存の Eclipse プロジェクトを検索するディレクトリーを選択します。		
● ルート・ディレクトリーの選択(□:		参照( <u>R</u> )
○ アーカイブ・ファイルの選択( <u>A</u> ):	~	参照( <u>R</u> )
プロジェクト( <u>P</u> ):		
	र्च /	べて選択( <u>S</u> )
	選択を	すべて解除( <u>D</u> )
		更新( <u>E</u> )
オフション □ ネストしたプロジェクトを検索( <u>H</u> )		
□ プロジェクトをワークスペースにコピー(_)		
□ ワークスペースに既に存在するフロジェクトを隠す(j)		
	±1	
	77	1757 ( <u>VV</u> )
	Ja Ja	EV(( <u>C</u> )
(N) >         終了(E)		キャンセル

AN2002 RA ファミリ 開発チュートリアル ©2024 Alpha Project Co., Ltd. ⑨ [終了] を選択します。

インポート				×
<b>プロジェクトをインポート</b> 既存の Eclipse プロジェクトを検索	するディレクトリーを選択します。			
<ul> <li>・レート・ディレクトリーの選択(①:</li> <li>・アーカイブ・ファイルの選択(A):</li> <li>プロジェクト(P)・</li> </ul>	C:¥workspace¥ra6m¥ap_ra6	im_Oa_ether	✓ 参照( <u>R</u> )	) <b></b> ()
	le(C:¥workspace¥ra6m¥ap_ra	a6m_0a_ett	すべて選択 選択をすべて解 更新( <u>E</u> )	( <u>S</u> ) (除( <u>D</u> )
オブション オブション ネストレたプロジェクトを検索仕 ブロジェクトをワークスペースにコ 完了次集、新しくインボートは ワークスペースに既に存在する?	) ピー( <u>C)</u> 「プロジェクトを閉じる( <u>o</u> ) 「ロジェクトを開す(j)	>		
- <b>ワ-キング・セット</b> - <b>ワ-キング・セットにプロジェクト</b> ワーキング・セット( <u>0</u> ):	を追加(①	~	新規( <u>W</u> ) 選択( <u>E</u> )	
? < 戻る	<u>B</u> ) 次へ( <u>N</u> ) >	終了( <u>F</u> )	キャンセ	.JL

⑩ ナビゲーションウィンドウにサンプルプログラムのプロジェクトが追加されていることを確認します。



以上でプロジェクトのインポートは完了です。

### 3.1.2 ビルド方法

プロジェクトのコンフィギュレータファイルを開きます。



② [BSP] タブを開きます。

③ [BSP]タブで [Board] が「ap\_ra6m\_0a」であることを確認します。

ra_workspace - ap_ra6m_0a_ether_sample/conf	iguration.xml - e² studio		- 🗆 X
ファイル( <u>F</u> ) 編集( <u>E</u> ) ソース( <u>S</u> ) リファクタリング(T) ナ	ビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> ) Renesas <u>V</u> iews 実行( <u>R</u> )	ウィンドウ( <u>W)</u> ヘルプ( <u>H</u> )	
🐔 掾 🔳 🎄 デバッグ(B) 🗸	💽 ap_ra6m_0a_ether_sample Debug 🗸 🌞 🗄 🕇 🖉 🕼	a   🚳 • 🔦 • 🔜 🟪 🔍 🗐 🐝   🏘 •	· 💁 🗸
🔍 - 核 🗰 💷 😭 🖏 🕹 😻 🥖 🕍 🗃 -	🔐 🕶 🖻 🕶 🕝 🕶 🤔 🛷 🕶 📴 👖 🗄 🖢 🛪 🖏 🕶 1	¢ d* (+ + c) + <b>⊡</b>	
		Q i 😭   🔂 c/c+-	🖡 💮 FSP Configuration 🛛 🎄 Debug
P □	🔅 [ap_ra6m_0a_ether_sample] FSP Configuration 🗙		E アウトライン 🛛 🖳 🗆
► 🕏 🏹 🖇 V 🚰 ap_ra6m_0a_ether_sample	Board Support Package Configuration	Generate Project Content	アウトラインを提供するアクティブなエディタ ーはありません。
> Win バイナリー		Restor	
> Bp includes			
> 👝 Debug	Device Selection		
Script ap ra6m 0a ether sample Debug, launch	FSP version: 4.5.0 V	Board Details	
ap_ra6m_0a.pincfg	Board: ap_ra6m_0a 🗸 🔽		
configuration.xml	Device: B7FA6M3AH3CFC		
> ⑦ Developer Assistance			
	RIOS MEERIOS		
	1		
	Summary BSP Clocks Pins Interrupts Event Links Stacks Co	mponents	
	Problems  D ンソール	יעבבדי	. 🔝 😼 🚽 🖻 🕶 🗖 🕶 🗖
	Installing support files for AlphaProject.ap_ra6m	1_0a.3.4.0-ap010000.packInstallation	complete.
			×
J			,
		http://tool-su	ipport5.1/content.xml.xz 💼 🖷

④ [Generate Project Content] をクリックし、自動作成ファイルを出力して設定をプロジェクトに適用します。

ra_workspace - ap_ra6m_0a_ether_sample/conf	lguration.xml - e² studio		- 🗆 X
ファイル(E) 編集(E) ソース(S) リファクタリング(T) つ	・ビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> ) Renesas <u>V</u> iews 実行( <u>R</u> ) ウィンドウ( <u>W</u> ) ヘルフ	<sup>†</sup> ( <u>Н</u> )	
後 第 ■ 株 デバッグ(B)	💽 ap_ra6m_0a_ether_sample Debug 🗸 🌞 🗄 🕶 🔚 🎼   🛞 🕶 🔦 🕶 [	🗟 🐏 🔍 💿 🞉 🗛 🕶	<b>Q</b> -
0. • 🗞 🕪 💷 😭 🖏 🛷 💋 📸 •	🚳 🕶 🔮 🕶 🥭 🛷 🕶 🗑 🔳 🖷 🦉 🕶 🖗 🖛 🏷 マク	* 📑	
		् । 😰   😼 c/c++	🔅 🔅 FSP Configuration 🛛 🐐 Debug
🔁 วีอจระวรหารวสมาย-รา- 🛛 👘 🗖	[ap_ra6m_0a_ether_sample] FSP Configuration ×		P ロ
► 🕏 🏹 🖇	Board Support Package Configuration	Generate Project Content	アウトラインを提供するアクティブなエディタ ーはありません。
> 続 バイナリー > 前 Includes		ta Restore	
> 😕 src	Device Selection		
> 🔁 Debug	Roard Datails		
ap_ra6m_0a_ether_sample Debug.launch	FSP version: 4.5.0 V		
ap_ra6m_0a.pincfg	Board: ap_ra6m_0a 🗸 🚵		
ige configuration.xmi i ra_cfg.txt	Device: R7FA6M3AH3CFC		
> ⑦ Developer Assistance	RTOS: FreeRTOS		
	<	>	
	Summary BSP Clocks Pins Interrupts Event Links Stacks Components		
	🖹 Problems 📃 ヨンソール 🕴 🦦 スマート・ブラウザー 👊 スマート・マニュアル	<u>∎</u> ,	🚮 😼 📑 🗉 😁 🗖
	RA FSP		
	Installing support files for AlphaProject.ap_ra6m_0a.3.4.0-ap01	0000.packInstallation	complete.
	<		>
		http://tool-su	pport5.1/content.xml.xz 💼 🖷

⑤ ツールバーからビルドアイコンを選択します。

ビルドが成功すると、¥Debug ワークフォルダにオブジェクトファイルが生成されます。

			_
📴 ra_workspace - ap_ra6m_0a_ether_sample/configuration.xml - e² studio	-	D X	(
ファイル(E) 編集(E) ソース(S) リファクタリング(T) ナビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト(P) Renesas <u>V</u> iews 実行(R) ウィンドウ( <u>W) ヘルプ(H</u> )			
🔨 🕸 🔳 🎋 F/(7/6) 🗸 🖬 ap_ra6m_0a_ether_sample Debug 🗸 🌼 📑 🕶 🗟 🐚 😽 🐝 🖗 🌾 🖗 😽 🐇 👘 🔅 🌾 🖗	<b>4</b> -		
[ 및 * 物 때 때 旨 집 값 [ ♥ ] #   # * @ * @ * @ * [# # *] # [ = = = = [ ] * [ ] * 한 라 다 다 다 다 다 다 다 다 다 다 다 다 다 다 다 다 다 다			
Q. IB 园 C/C++	FSP Configuration	🎄 Debu	g
🚹 プロジェクト・エクスプローラー 🔯 🗧 📮 ra6m_0a_ether_sample) FSP Configuration 🗙 📃 🗖	🏝 アウトライン 😂	- 6	=
🗉 🕏 🏹 🕴 Board Support Package Configuration 📀 🤈	アウトラインを提供するアク	ティブなエディ	19
v 🗭 ap_ra6m_0a_ether_sample Generate Project Content =	-はありません。		
> 🕅 Includes 🔤 🤯			
> 😂 src Device Selection			
> Debug			
ap_ra6m_0a_ether_sample Debug.launch			
ti ap_ra6m_0a.pincfg Board: ap_ra6m_0a ✓ 🚵			
≧ ra_cfg.txt Device: R7FA6M3AH3CFC			
> ⑦ Developer Assistance RTOS: FreeRTOS ~			
Summary BSP Clocks Pins Interrupts Event Links Stacks Components			-
	11 🖻   🔜 🖻 🗕 🕻	<u>9</u> • − t	3
Installing support files for AlphaProject.ap_ra6m_0a.3.4.0-ap010000.packInstallation co	complete.		^
			$\sim$
		-	
http://tool-supp	port5.1/content.xml.	xz 🔜 🖗	5

e2 studio の詳細な使用方法に関しては、 e2 studio のマニュアルを参照してください。

### 3.1.3 デバッグ方法

 「3.1.2 ビルド方法」を参考に、プロジェクトをビルドしてください。 (例で使用する画像では、プロジェクト名は「ap\_ra6m\_0a\_ether\_sample」となっています。)

BootMode

SCI/USB-SGL

シングルチップモード

JSW2 : SGL

ZMSD

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、BootMode を「SGL(SINGLE)」に設定します。

<AP-RA6M-0A>



JSW1 : ON ボード上の SDRAM を使用する





JSW1:SINGLE





JSW1

Ethernet を使用する場合: ON CAMERA を使用する場合: OFF



JSW2 : SGL シングルチップモード

<AP-RA8T-0A>



JSW1:SGL シングルチップモード



- ③ ボードに電源を投入してください。
- ④ プロジェクトを選択し、メニューバーから [デバッグの構成]を開きます。

ra_workspace - ap_ra6m_0a_ether_sample/cont	iguration.xml - e² studio		-	
ファイル(E) 編集(E) ソース(S) リファクタリング(T) :	・ビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> ) Renesas <u>V</u> iews 実行( <u>R</u> ) ウィンドウ( <u>W</u> ) ヘルプ( <u>H</u> )			
🐔 🔅 🔳 🔅 デパッグ(B) 🗸	💽 ap_ra6m_0a_ether_sample Debug 🗸 🌼 🗄 🖬 🔚 🎼 🐘 😵 💌 🚳 🗄	\$e   ≈   ⊕   <b>≈  </b> † •	Q	
0 🗞 🕪 💷 😭 🖏 🐼 🖉 💼 •	88 ▼ 🖻 ▼ 🞯 ▼ 🎒 🛷 ▼ 🗟 🗉 🖷 🖞 ▼ 🖓 ▼ ∜⊃ 라 ◆ → ↔ ▼		(起動履歴なし)	
		Q । 😰   📴	<i>∓/(₩/I(</i> D)	> Jebug
陷 วือўторытрадонан 🙁 🗉	@ [ap_ra6m_0a_ether_sample] FSP Configuration ⋈	=	デバッグの構成(B)	° 🗆
↓ 🔁 ap_ra6m_0a_ether_sample [Debug]	Board Support Package Configuration	Generate Project Content	お気に入りの編成(V) ーはありません。	エディタ
> 続 バイナリー > 前 Includes		👿 Restore		
> 🚰 ra	Device Selection			
> 🔑 ra_gen	Device Selection			
> 📂 src > 🗁 Debug	FSP version: 4.5.0 V Board Details			
> 🧀 ra_cfg	Board: ap ra6m 0a			
> 🗁 script				
ap_ra6m_0a_ether_sample Debug.launch	Device: R/FA6MI3AH3CFC			
ар_гаатол.pinctg	RTOS: FreeRTOS 🗸			
in ra_cfg.txt				
> ⑦ Developer Assistance				
	<	>		
	Summary BSP Clocks Pins Interrupts Event Links Stacks Components			
	Problems ロソール 窓 (楽) スマート・ブラウザー 印刷 スマート・マニュアル ※	0 0 🔁 🖬 🖬 =	▶ 🛃 🚽 🗨 🕶	🔁 🕶 🗖
	CDT ビルド・コンソール [ap_ra6m_0a_ether_sample]			
	'Finished building: ap_ra6m_0a_ether_sample.siz'			^

- ⑤ [Renesas GDB Hardware Debug] の[<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。
  - [名前]: <プロジェクト名> Debug (AP-RA6M-1Aの場合: <プロジェクト名> Debug\_Flat)
  - [プロジェクト]: <プロジェクト名>
  - [C/C++アプリケーション]: Debug¥ <プロジェクト名>.elf

📴 デバッグ構成		— 🗆 X
構成の作成、管理、および実行	Ŧ	- Total
📑 🖻 🔕 🗎 🗙 🖻 🗡 🗸	名前( <u>N</u> ): ap_ra6m_0a_ether_sample Debug	
7ィルタ入力	📄 メイン 🎋 Debugger 🔈 Startup 🔲 共通(C) 🎙	/ ソース
C C/C++ アプリケーション	_プロジェクト( <u>P</u> ):	
EASE Script	ap_ra6m_0a_ether_sample	参照( <u>B</u> )
GDB OpenOCD Debugging	C/C++ アプリケーション:	
C GDB バードウェア・デバッギング	Debug¥ap_ra6m_0a_ether_sample.elf	
」 Java アプリケーション		変数( <u>V</u> ) プロジェクトの検索( <u>H</u> ) 参照( <u>R</u> )
✓ C <sup>™</sup> Renesas GDB Hardware Debu	起動前に必要に応じてビルド	
c× ap_ra6m_0a_ether_sample	Build Configuration: Use Active	~
■ Kenesas Simulator Debugging ■ リモート Java アプリケーション	○ 自動ビルドを有効にする	○ 自動ビルドを無効にする
🔂 起動グループ	● ワークスペース設定の使用	<u>ワークスペース設定の構成</u>
15 項目のうち 13 項目がフィルターに一致		前回保管した状態に戻す( <u>V</u> ) 適用( <u>Y</u> )
?		デ/(ッグ( <u>D</u> ) 閉じる

- ⑥ [Debugger] タブを選択し、 [Debug hardware] が [J-Link ARM] 、 [Target Device] が「R7FA6M3AH」 (※) に 設定されていることを確認してください。
  - ※ AP-RA6M-1Aの場合は [CPU] から「R7FA6M5BH」
     AP-RA8M-0Aの場合は [CPU] から [R7FA8M1AH]
    - AP-RA8D-0A の場合は [CPU] から [R7FA8D1BH]

AP-RA8T-0A の場合は [CPU] から [R7FA8T1AH] を選択してください。

📴 デバッグ構成	— 🗆 X
構成の作成、管理、および実行	
<ul> <li>ご 診 ŵ 論 X □ マ マ</li> <li>フィルタ入力</li> <li>C (C++ アブリケーション</li> </ul>	名前(N): ap_ra6m_0a_ether_sample Debug
<ul> <li>C/C++ リモート・アブリケーション</li> <li>EASE Script</li> <li>GDB OpenOCD Debugging</li> <li>GDB Simulator Debugging (R)</li> <li>GDB Simulator Science (R)</li> </ul>	Debug hardware: J-Link ARM v Target Device: R7FA6M3AH GDB Settings Connection Settings デバッグ・ツール設定 GDB 接続設定:
C GUB /-F)エア・テパッキンク J Java アブリケーション ■ Java アブレット ▼ C Renesas GDB Hardware Debux □ C ap ra6m Oa ether sample	<ul> <li>● 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</li> <li>○ リモート GDB サーバーへ接続 GDB ポート番号: 61234</li> </ul>
<ul> <li>配 Renesas Simulator Debugging</li> <li>リモート Java アプリケーション</li> <li>記 起動グループ</li> </ul>	GDB GDB Jマンド: arm-none-eabi-gdb 参照 変数 Step Mode
< > 15項目のうち13項目がフィルターに一致	前回保管Lた状態に戻す(V) 適用(Y)
?	デパッグ( <u>D)</u> 閉じる

⑦ [デバッグ]を選択します。

📴 デバッグ構成	- D X
構成の作成、管理、および実行	·
	名前(M): [ap_ra6m_0a_ether_sample Debug
<ul> <li>C/C++アブリケーション</li> <li>C/C++アブリケーション</li> <li>C/C++リモート・アブリケーション</li> <li>E ASE Script</li> <li>GDB OpenOCD Debugging</li> <li>GDB Simulator Debugging (Ri</li> <li>GDB /-ドウェア・デバッギング</li> <li>Java アブリケーション</li> <li>Java アブリケーション</li> <li>Renesas GDB Hardware Debug</li> <li>C Renesas Simulator Debugging</li> <li>UモートJava アブリケーション</li> <li>転動グルーブ</li> </ul>	<ul> <li>アメイン vs Debugger ▶ Startup □ 共通() レックス</li> <li>Debug hardware: J-Link ARM ∨ Target Device: R7FA6M3AH</li> <li>GDB Settings Connection Settings デパッグ・ツール設定</li> <li>GDB 接続設定:</li> <li>③ 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</li> <li>〇 リモート GDB サーバーへ接続 GDB ポート番号: 61234</li> <li>GDB</li> <li>GDB コマンド: arm-none-eabi-gdb 参照 変数</li> <li>Step Mode</li> </ul>
< > 15 項目のうち 13 項目がフィルターに一致	前回保管した状態に戻す(⊻) 適用(⊻)
?	デパッグ( <u>D)</u> 閉じる

⑧ ボードとの接続が完了したらプログラムを実行し、サンプルプログラムを動作させてください。



⑨ プログラムの動作が確認できましたら、CPU ボードへのプログラムのダウンロードも完了しています。

プログラムの動作については、デバッグしているサンプルプログラムのアプリケーションノートをご参照ください。

# 4. e2 studio を用いた新規プロジェクト作成方法

本章では、e2 studio で CPU ボード用プログラムを作成する方法を説明します。 プロジェクトコンフィギュレータの設定を行うプロジェクトとして、「4.1 UART プログラムの作成例」にて、UART を使用 したプログラムの作成方法を説明します。 なお、本書で扱っている FSP は バージョン 4.5.0 です。異なるバージョンの FSP をご使用の場合は、本書で解説する項目・ 設定等が異なる場合がございます。ご注意ください。

また、本章では、図中の選択・実行する必要のある箇所を<mark>赤色枠</mark>の内容の確認をする箇所を<mark>橙色枠のので</mark>で 示します。

## 4.1 UART プログラムの作成例

本節では、AP-RA6M-0A で動作する UART のプログラムを例に、プロジェクトの作成方法を説明します。

① e2 studio のメニューから、 [ファイル] - [新規] - [Renesas C/C++ Project] - [Renesas RA] を選択します。





② [Renesas RA C Executable Project] を選択し、 [次へ] を押します。

New C/C- Templates for	++ Project	_		×
All C/C++	Renesas RA C/C++ Project Create an executable or static library for Renesas RA.	C/C++ pr	roject	
?	< 戻る( <u>B</u> ) 次へ( <u>N</u> ) > 終了(	<u>F</u> )	キャンセ	zJL

③ [Project name] に任意のプロジェクト名を入力します(ここでは「uart\_sample」とします)。
 入力後、 [次へ] を押して次へ進みます。

Renesas RA C/C++ Project	_		×
Renesas RA C/C++ Project Project Name and Location			\$
Project name			
Luart_sample マーデフォルト・ロケーションの使用(D)			
ロケーション(L): E:¥workspace¥WMRP_20211014¥ra_workspace¥uart_sample ファイル・システムを選択(Y): デフォルト ~		参照( <u>R</u> )	
You can download more Renesas packs here			
(ア)         (ア)         ※了(E)		キャンセ	IL

#### ④ AP-RA6M-0A/1A の場合

Device Selection 設定では、[FSP version] から[4.5.0]、[Board]から[Custom User Board]、 [CPU] から [R7FA6M3AH3CFC] (※) をそれぞれ設定します。

Toolchain 設定では「GNU ARM Embedded」を選択のうえ、 [Toolchain version] に[10.3.1.20210824]を設定します。

Debugger 設定では、 [Debugger] には [J-Link ARM] を設定します。

設定後、 [次へ] を押します。

※ AP-RA6M-1A の場合は [CPU] から [R7FA6M5BH3CFB] を選択してください。

AP-RA8M-0A、AP-RA8D-0Aの場合

Device Selection 設定では、[FSP version] から[5.1.0]、[Board]から[Custom User Board]、

[CPU] から [R7FA8M1AHDCBD] (※) をそれぞれ設定します。

Toolchain 設定では「GNU ARM Embedded」を選択のうえ、 [Toolchain version] に[13.2.1.arm-13-7]を設定します。 Debugger 設定では、 [Debugger] には [J-Link ARM] を設定します。

設定後、 [次へ] を押します。

※ AP-RA8D-0A の場合は [CPU] から [R7FA8D1BHDCBD] を選択してください。

#### AP-RA8T-0A の場合

Device Selection 設定では、[FSP version] から[5.5.0]、[Board]から[Custom User Board]、

[CPU] から [R7FA8T1AHECFB]を設定します。

Toolchain 設定では「GNU ARM Embedded」を選択のうえ、 [Toolchain version] に[13.2.1.arm-13-7]を設定します。 Debugger 設定では、 [Debugger] には [J-Link ARM] を設定します。

設定後、 [次へ] を押します。

Renesas RA C/C++ Project			_		Х
Renesas RA C/C++ Project Device and Tools Selection					\$
Device Selection         FSP Version:       4.5.0         Board:       Custom User Board (Any Device)         Device:       R7FA6M3AH3CFC         Language: <ul> <li>C C C++</li> </ul>	Board Description Device Details TrustZone Pins Processor	No 176 Cortex-M4			
Toolchains GNU ARM Embedded GCC ARM A-Profile (AArch64 bare-metal) 9.2.1.20191025	Debugger J-Link ARM				~
?	< 戻る( <u>B</u> ) 次	^( <u>N</u> ) >	終了( <u>F</u> )	キャンセ	JL

⑤ (AP-RA6M-1A/AP-RA8M-0A/AP-RA8D-0A/AP-RA8T-0A の場合) プロジェクトタイプを設定します。

サンプルでは [Flat (Non-TrustZone) Project] を選択します。

(セキュリティ機能がない CPU ボードの場合は表示されませんので⑥へお進みください)

Renesas RA C/C++ Project		×
Renesas RA C/C++ Project Project Type Selection		Ź
<ul> <li>Project Type Selection</li> <li>Flat (Non-TrustZone) Project         <ul> <li>Remesas RA device project withdut TrustZone separation</li> <li>All code, data and peripheral settings will be configured in this project</li> <li>Renesas RA device will remain in secure mode</li> <li>EDMAC RAM buffers will automatically be placed in non-secure RAM</li> </ul> </li> <li>TrustZone Secure Project</li> <li>Renesas RA device project for TrustZone secure execution</li> <li>All code, data and peripherals placed in this project will be initialized as secure</li> <li>Secure project settings such as TrustZone partitions, linker maps and a list of secure peripherals will be passed to a selected non-secure project</li> <li>After initialization, a call to the non-secure startup handler will be made</li> <li>TrustZone Non-secure Project</li> <li>Renesas RA device project for TrustZone non-secure execution</li> <li>All code, data and peripherals placed in this project will be initialized as non-secure</li> <li>Must be associated with a secure project or secure execution</li> <li>All code, data and peripherals placed in this project will be initialized as non-secure</li> </ul>		
?	< 戻る( <u>B</u> ) 次へ( <u>N</u> ) > 終了( <u>B</u> ) キャンセ	セル

⑥ ビルド生成物と RTOS 設定を設定します。

サンプルでは [Build Artifact Selection] に [Executable] 、 [RTOS Selection] に [No RTOS] を選択します。

😰 Renesas RA C/C++ Project	_		×
Renesas RA C/C++ Project Build Artifact and RTOS Selection			2
Build Artifact Selection	RTOS Selection		
Project builds to an exet Project builds to a static library file     Executable Using an RA Static Library     Project builds to an executable file     Project uses an existing RA static library project			
?	< 戻る(B) 次へ(N) > 終了(F)	キャンさ	211

#### ⑦ プロジェクトを作成します。

プロジェクトのテンプレートは [Bare Metal - Minimal] を選択し、 [終了] を押してプロジェクトを作成します。

🕲 Renesas RA C/C++ Project					×
Renesas RA C/C++ Project				_	$\diamond$
Project Template Selection					_
Project Template Selection					
Bare Metal - Minimal     Bare metal FSP project that includes BSP. This project v     [Renesas.RA.3.4.0.pack]	vill initialize clocks, pins	s, stacks, and the	C runtime enviro	nment.	
Code Generation Settings					
Use Renesas Code Formatter					
?	< 戻る( <u>B</u> ) 次	ζ∧( <u>N</u> ) >	終了( <u>F</u> )	キャンセ	JL

⑧ プロジェクトが作成されました。
 以降では、RA プロジェクトコンフィギュレータでプロジェクトの設定を行います。
 プロジェクトのコンフィギュレータファイルを開いてください。



⑨ [Stacks] タブを開きます。この画面で、HAL ドライバやフレームワークの追加や設定をします。
 [New Stack] を押して、 [Driver] - [Connectivity] - [UART Driver on r\_sci\_uart] を選択します。



WART のモジュールが追加され、下図の橙色の枠のように、UART Driver が画面に表示されます。
 [g\_uart0 UART Driver on r\_sci\_uart]のモジュールをクリックすると、そのプロパティがプロパティビューに表示されます。

tacks Configuration		Generate Project Conter
Threads	HAL/Common Stacks	🔊 New Stack > 👙 Extend Stack > 🙀 Remove
Common     Gioport I/O Port Driver on r_sio     guarto UART Driver on r_sci_uc	<ul> <li> <sup>₽</sup> g_ioport I/O Port Driver on r_ioport         <sup>1</sup> <sup>1</sup></li></ul>	Guardo UART Driver on r_sci_uart      G      G      Add DTC Driver for     Transmission     [Recommended but     optional]      Guardo UART Driver on r_sci_uart
Dbjects 🐑 New Object > 💼 Remove		

プロパティが見つからない場合は、ツールバーの [ウィンドウ] - [ビューの表示] - [プロパティー] で表示できます。

ra_workspace - uart_sample/configuration.xml - e <sup>2</sup> studies	•				
ファイル( <u>E</u> ) 編集( <u>E</u> ) ナビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> )	Renesas <u>V</u> iews 実行( <u>R</u> )	ウィンドウ( <u>W)</u> トルプ( <u>H</u> )			
🔏 🎄 🔳 🎋 デバッグ(B) 🗸 💽 uar	t_sample Debug_Flat	新規ウィンドウ(N)	1	- 🔨 - 🗟 🐂 🛞 💸	🎄 🕶 💁 🕶 🔍 😽
		エディター	>		
월 * 월 * <sup>1</sup> 2 다 · · · · · · · · · · · · · · · · · ·		外観	>		
🔐 🚹 วัดษัรวรหายังวินัก-วิ- 🗙 📃 🖻	# *[uart_sample] FSP	ドューの表示(V)	> 🔍	スマート・ブラウザー	1
	Stacks Configura	パースペクティブ(R)	>	プロジェクト・エクスプローラー	
✓				ゴロパティー	
> 前 Includes		ナビゲーション(G)	> 😐	/u/()1-	
> 😂 ra	Threads	設定(D)		その他(o)	Alt+シフト+Q,Q
> 🔑 ra_gen	HAL/Common	av.AE(F)	100	-	

① [g\_uart0 UART Driver on r\_sci\_uart]のモジュールをクリックすると、そのプロパティがプロパティビューに表示されます。

ra_workspa	ace - uart_sample/configuration.xml - e <sup>2</sup> st	tudio		- 🗆 ×
ファイル( <u>F</u> ) 編集	集(E) ナビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>I</u>	<u>P</u> ) Renesas <u>V</u> iews 実行( <u>R</u> ) ウィンドウ( <u>W</u> ) ヘルプ( <u>H</u> )		
«	📕 🎄 デバッグ(B) 🗸 💽 เ	uart_sample Debug_Flat 🛛 🗸 🌞 🗄 🖶 😭 🕇	🛞 = 🔦 = 🗟 : 🏪 : 🚳 : 🗞   🎄 =	🂁 ▾: 🔍 ▾ 🎋 🕪 💷 😭 🖏 🕹 : 🏈 : 💋 : 🔗 ▾
山・利・	()			Q : 😭 🐻 C/C++ 🛞 FSP Configuration 🔅 Debug
a 🔲 วีย/(ร	1- 23	<b>1</b> 8 <b>1</b>	*[uart_sample] FSP Configuration ×	
🔁 g_uart0	UART Driver on r_sci_uart		Stacks Configuration	Generate Project Content
Settings	プロパティ	值		
APLInfo	✓ Common		New Thread	HAL/Common Stacks 🛛 🖗 New Stack > 🚔 Extend Stack > 🙀 Remove
	Parameter Checking	Default (BSP)	Remove	
	FIFO Support	Disable		port I/O Port 🛛 🕂 g_uart0 UART Driver on r_sci_uart
	DTC Support	Disable	V N HAL/Common	er on r_ioport
	Flow Control Support	Disable	g_loport I/O Port Driver on	A
	<ul> <li>Module g_uart0 UART Driver on r_sci</li> </ul>		g_uarto UART Driver on r_so	U
	> General			
	> Baud			😭 Add DTC Driver for 🛛 😭 Add DTC Driver for
	> Flow Control			Transmission Reception [Not
	> Extra			[Recommended but recommended]
	> Interrupts			optionalj
	✓ Pins		< >	
	TXD_MOSI	<unavailable></unavailable>		
	RXD_MISO	<unavailable></unavailable>	New Object >	
	CTS_RTS_SS	<unavailable></unavailable>	Objects Remove	
			a, nemove	
				< >
			Summary BSP Clocks Pins Interrupts B	Event Links Stacks Components
Probler	ns 🖾 🦦 スマート・ブラウザー	₽ % =	Pin Conflicts ロンソール 83	■ X 💥 🖳 🖬 🖗 🖅 🖤 🗖 🕶 🖛 🗖

#### AP-RA6M-0A/1A の場合

表示されたプロパティのうち、[General]内の 設定[Name]を[g\_uart4]、[Channel]を[4]にそれぞれ変更してください。

#### AP-RA8M-0A、AP-RA8D-0A の場合

表示されたプロパティのうち、[General]内の 設定[Name]を[g\_uart1]、[Channel]を[1]にそれぞれ変更してください。

### AP-RA8T-0A の場合

表示されたプロパティのうち、[General]内の 設定[Name]を[g\_uart0]、[Channel]を[0]にそれぞれ変更してください。

Pイル( <u>F</u> ) 編	集( <u>E)</u> ナビゲート( <u>N</u> ) 検索( <u>A</u> ) プロジェク	ト( <u>P</u> ) Renesas <u>V</u> iews 実行( <u>R</u> ) ウィンドウ( <u>W</u> ) ヘルプ( <u>H</u> )
*	● 茶 デバッグ(B) ~ C	🛚 uart_sample Debug_Flat 🛛 🗸 🔅 📑 😁 🖛 🔚 🐚
- 81 -	5 ♂ ⇔ → →   🛤	
	1- X	<b>1</b> 8 <b>1 1</b>
g_uart0	UART Driver on r_sci_uart	
Cattleres	プロパティ	值
Settings	× Common	
API Into	Parameter Checking	Default (BSP)
	FIFO Support	Disable
	DTC Support	Disable
	Flow Control Support	Disable
	✓ Module g_uart0 UART Driver on r_s	ci
	✓ General	
	Name	g_uart4
	Channel	4
	Data Bits	8bits
	Parity	None
	Stop Bits	1bit
	> Baud	
	> Flow Control	
	> Extra	
	> Interrupts	
		due susilable t
		<ur> <li><ur> <li><ur> <li><ur></ur></li></ur></li></ur></li></ur>
	E T L BRINT	

### アプリケーションノート AN2002

② [Pins] タブを開きます。この画面で、UART 端子の設定をします。

AP-RA6M-0A/1A の場合

[Peripherals]- [Connectivity:SCI]-[SCI4]を選択した後、

[Pin Group Selection]から[\_A only]、[Operation Mode]から[Asynchronous UART]を選択します。

### AP-RA8M-0A、AP-RA8D-0A の場合

[Peripherals]- [Connectivity:SCI]-[SCI1]を選択した後、

[Pin Group Selection]から[\_B only]、[Operation Mode]から[Asynchronous UART]を選択します。

### AP-RA8T-0A の場合

[Peripherals]- [Connectivity:SCI]-[SCI0]を選択した後、

[Pin Group Selection]から[\_C only]、[Operation Mode]から[Asynchronous UART]を選択します。

🔅 *[uart_sample] FSP Configuration 🗙					
Pin Configuration					Generate Project Content
Select Pin Configuration		📑 Export to C	SV file 🗄	Configure Pir	n Driver Warnings
R7FA6M3AH3CFC.pincfg	✓ Manage configurations	🗹 Gene	erate data:	g_bsp_pin_cf	9
Pin Selection $\exists \exists \exists \exists \exists \exists z \exists z$	Pin Configuration				😲 Cycle Pin Group
Type filter text	Name	Value	Lock	Link	
A af Paripharals	Pin Group Selection	_A only			
Monitoring:CAC	Operation Mode	Asynchronous UART			
Analog:ADC	✓ Input/Output			$\langle - \rangle$	
Analog:ADC	TXD_MOSI	✓ P205	<b>_</b>		
Analog DAC12	RXD_MISO	✓ P206	i i i	$\Rightarrow$	
Connectivity:CAN	SCK	None		$\Rightarrow$	
Connectivity:ETHERC	CTS_RTS_SS	None		$\Rightarrow$	
ConnectivityIIC	SDA	None		$\Rightarrow$	
Connectivity:SCI	SCL	None		$\Rightarrow$	
SCIO					
SCI1					
SCI2					
SCI3	<				>
✓ SCI4	Module name: SCI4				
SCI5	Usage: When using Sim	ple I2C mode, ensure port pin	s output ty	pe is n-ch oper	n drain.
SCI6 ¥	When switching	between I2C and other mode	s, first disal	ble.	
端子機能 端子番号	· L				
Summary BSP Clock Pins Interrupts Ever	nt Links   Stacks   Components				

③ 今回は特に設定する必要はありませんが、続けて他のモジュールを設定することもあります。
 適宜必要な設定を全て行った後に、[Generate Project Content] をクリックしてください。
 自動作成ファイルが出力されて設定がプロジェクトに適用されます。



ゆ プロジェクトの「src」フォルダに、アプリケーション部分を作成します。

$\leftarrow \rightarrow$	× ↑ 🔤 <	« ra_workspace > uart_s	sample > src	5 V
4	名前	^	更新日時	種類
	hal_ent	try.c	2021/11/03 11:55	C ファイル

⑮ 以上で UART のプロジェクトが完成しました。ビルド後、動作を確認してください。

ra_workspace - uart_sample/configuration.xml - e <sup>2</sup>	studio				_		×
ファイル(E) 編集(E) ソース(S) リファクタリング(I) ナビゲ	ート( <u>N</u> ) 検索( <u>A</u> ) プロジェクト( <u>P</u> ) Renesas	<u>V</u> iews 実行( <u>R</u> ) ウィンドウ( <u>W) ヘルプ(H</u> )					
🔦 🏘 🔳 🎄 รี่เก็ษย์(B) 🗸 💽	] uart_sample Debug_Flat 🛛 🗸 🔅	🗂 🕶 🔚 🐚   🛞 🕶 🔦 🕶 🚵   🏪   🔌	@   🎉   🎋 🕇	• 💁 • 🗄 🖓	***********	8	ø
📸 • 🚳 • 🖻 • 🎯 • 🎒 🛷 • 📴 👖	실 ▼ 취 ▼ th c? ( <b>) ▼</b> c) ▼   =	1	Q	i 😭 🛛 🖬 C/	'C++ 🔅 FSP Configu	ration 💠 l	Debug
🔁 プロジェクト・エクスプローラー 🛛 🖳 🗖	🔅 [uart_sample] FSP Configuration 🗙				語 アウトライン 🛙	8	
	Stacks Configuration		Generate Pro	ject Content	アウトラインを提供する; ありません。	"クティブなエテ	ディターは
<ul> <li>&gt; (a) Includes</li> <li>&gt; (b) Includes</li> <li>&gt; (c) ra</li> <li>&gt; (c) ra,gen</li> <li>&gt; (c) Debug</li> <li>&gt; (c) ra,efg</li> <li>&gt; (c) script</li> <li>(c) ra,efg</li> <li>&gt; (c) script</li> <li>(c) ra,efg</li> <li>&gt; (c) Developer Assistance</li> </ul>	New Thread Remove Remove HAL/Common G g_jopot VO Port Driver on g_uart4 UART Driver on r_s	HAL/Common Stacks   New Stack >   Cont I/O Port or on r_ioport  Add DIC Driver for Transmission [Recommended but optional]	Extend Stack > a	Remove			
	Summary BSP Clocks Pins Interrupts	<     Event Links Stacks Components		>			
	🖹 Problems 🖾 📮 ארעב 🖓 אדי 🖓 איז	-ト・ブラウザー 👊 スマート・マニュアル				78	
	0項目		10 c =	1		-	
	8C X11/87C 9H		99-X	NX.	07-939	<u>49</u>	
	<						>
			Ć				<b>1</b>

5. Renesas Flash Programmer を使用した書き込み

本章では、CPU ボードに Renesas Flash Programmer を使用してプログラムを書き込む方法を説明します。

- 5.1 USB インタフェースを使用した書き込み方法
  - ① CPU ボードの設定を SCI/USB ブートモードに変更し、CPU ボードと PC を USB ケーブルで接続します。

<AP-RM6A-0A>



AP-RA6M-0AのJSW1の設定は不問です。

<AP-RM6A-1A>



JSW1:SCI/USB

<AP-RM8A-0A, AP-RA8D-0A>



JSW2 : SCI/USB

#### <AP-RA8T-0A>



JSW1 : SCI

- ② CPU ボードの電源を入れます。
- ③ Renesas Flash Programmer を起動します。
- ④ RFP を起動すると、以下のようなウィンドウが表示されますので、「ファイル(F)」メニューの「新しいプロジェクトを 作成(N)…」を選択します。

🌠 Renesas Flash Programmer V3.08.01 (無償版)	- 🗆 X
ファイル(E) ヘルプ(H)	
新しいプロジェクトを作成(N)	
プロジェクトを開く( <u>O</u> )	
プロジェクトを保存( <u>S</u> )	
イメージファイルを保存(」)	
ファイルチェックサム( <u>C</u> )	
ファイルパスワード設定(P)	
終了(X)	参照( <u>B</u> )
スタート( <u>S</u> )	
Renesas Flash Programmer V3.08.01 [1 Jan 2021] (無(質版)	
	ステータスとメッセージのクリア( <u>C</u> )

⑤ 新しいプロジェクトの作成ウィンドウが表示されますので、プロジェクトの設定を行います。
 設定後、[接続(O)]を押して接続を開始してください。

```
<プロジェクト情報>
```

・ マイクロコントローラ(M)	:	[RA] を選択します。
・ プロジェクト名(N)	:	任意のプロジェクト名を設定します。
		(ここでは例として、「ap_ra6m_rfp」を入力しています。)
・ 作成場所(F)	:	[参照(B)]ボタンを押し、任意の保存先を設定します。
		(ここでは例として「C:¥workspace¥ra6m」を選択しています。)
<通信>		
・ ツール(T)	:	[COM port] を選択します。
ツール詳細(D)		[ツール詳細(D)] ボタンを押して設定ウィンドウを開きます。
		COM の一覧から [RA USB Boot(CDC)] を選択し、 [OK] を押します。
		(COM ポートの番号は環境により異なります。)

COM port が動作しない場合、USB ブート用ドライバが誤認識されている可能性があります。 詳細は RFP のユーザーズマニュアルをご覧ください。

第しいプロジェクトの作成 – ×
プロジェクト情報         マイクロコントローラ(M):       RA         プロジェクト名(N):       ap_ra6m_rfp         作成場所(F):       C:¥workspace¥ra6m       参照(B)
通信 ツール(I): COM port 〜 インタフェース(I): 2 wire UART 〜 ツール詳細( <u>D</u> ) 番号: COM1
<u> 接続(0)</u> キャンセル(0)
ビール詳細 (COM port) ー ○ ×     ツール選択 リセット設定     COM1: i通信ポート     GOM3 + UCD Serial Port     COM10: RA USB Boot(CDC)     COM10: RA USB Boot(CDC)     Extra LSB Boot(CDC)     Extra LSB Boot(CDC)     Extra LSB Ext

⑥ RFPのメイン画面が表示されます。

プログラムファイルの[参照]ボタンをクリックし、書き込むファイルを選択します。 FlashROM に書き込むファイルは、実行したいビルドモードのフォルダ(Debug)内の**.srec ファイル**を指定します。 (図は、AP-RA6M-1A 用 UART サンプルプログラムの.srec ファイルです)

	🕻 Renesas Flash Programmer V3.08	.01 (無償版)		_	×
	ファイル(F) デバイス情報(D) ヘルフ	プ(H)			
	操作 操作設定 ブロック設定 フラッ	シュオブション 接続設定	ユニークコード ユー	ザー鍵	
	プロジェクト情報 現在のプロジェクト: ap_ra6m_rfg マイクロコントローラ: R7FA6M5E プログラムファイル	orpj IH3CFB		<mark>  参照_(B)</mark>	
📕 プログラJ	ムファイルを指定してください。				×
$\leftarrow \rightarrow$	PC > OS (C:) > workspace >	sample > Debug >		✓ ່ບ ,○ Debu	ugの検索
整理 ▼	新しいフォルダー				::: • 🔟 ?
4 ^	名前 ^	更新日時	種類	サイズ 大きさ	
	ra	2021/04/20 16:20	ファイル フォルダー		
4	ra_gen	2021/04/20 16:20	ファイル フォルダー		
	src	2021/04/20 16:20	ファイル フォルダー		
F	ap ra6m 1a sci uart sample.sbd	2021/04/14 12:59	SBD ファイル	4 KB	
a.	ap_ra6m_1a_sci_uart_sample.srec	202 /04/14 12:59	SREC ファイル	29 KB	
•					
	ファイル名(N): ap_ra6m_1a_sci_uart_	sample.srec		~ プログラムフ	アイル (*.hex, *.mot, *.s 〜
				開く( <u>C</u>	) キャンセル

- ⑦ ファイルの指定後、一度 CPU ボードの電源を切断し、再投入します。
   この後は CPU のセキュリティ機能に関する設定に続きます。
   セキュリティ機能がない CPU ボードの場合は、⑧~⑩のステップは不要ですので⑪にお進みください。
- ⑧ [操作設定] タブを開いて、[コマンド] 内の [フラッシュオプション書込み] と [フラッシュオプションベリファイ] の それぞれにチェックマークを付けて有効化します。

🌠 Renesas Flash Programmer V3.08.01 (無償版)	– 🗆 X
ファイル( <u>F)</u> デバイス情報( <u>D</u> )ヘルプ( <u>H</u> )	
操作 「操作設定」フロック設定 フラッシュオブション 接続語	役定 ユニークコード ユーザー鍵
<ul> <li>コマンド</li> <li>河 消去(£)</li> <li> 書き込み(£)</li> <li> ペリファイ(少)</li> <li> フラッシュオブションペリファイ(少)</li> <li> チェックサム(\$)</li> </ul>	消去オブション(① プロック選択消去 ~ 書き込みとベリファイオブション □ 書き込み前に消去(B) デバイスからリードしてベリファイ ~ チェックサム計算方式(M)
0×FF補完	CRC-32方式 エラー設定 ビデバイス範囲外エラーを有効にする(B)

⑨ [フラッシュオプション] タブを開いて、設定「DLM」の [設定オプション] を [設定する]、 [遷移先] を開発で指定する値に設定します。

(ここでは例として、[遷移先]を [SSD] に設定しています)

· · · · · · ·						
定 フラッシュオフション	接続設定	ユニークコード	ユーザー鍵			
	· ···································	<i>t</i> 3				
	SSD	90				
						_
	何もし	ない				
Key						
	設定	する				
	0					
	0					
Data Flash Secure [KB] SRAM Secure [KB]						
	U					
	(=+ )	+				
	1960	(a(L)				
	Key	読定: SSD 何もし Key 意定: 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	設定する SSD 何もしない Key 設定する 0 0 0 0 0 0 0 0 0 0 0 0 0	設定する SSD 何もしない Key 読定する 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	設定する SSD 何もしない Key 設定する 0 0 0 0 0 0 0 0 0 0 0 0	設定する SSD 何もしない Key 読定する 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

① [フラッシュオプション] タブ内の設定「Boundary」の [設定オプション] を [設定する] に
 設定し、さらに、ユーザが作成したプログラムに従ったセキュア領域をそれぞれ設定します。
 (ここではメモリの全領域をませたコアとする例として [Code Flash Secure [KB]] ~ [SPAM NSC [KB]

(ここではメモリの全領域を非セキュアとする例として、[Code Flash Secure [KB]] ~ [SRAM NSC [KB]]まで すべてを [0] に設定しています)

2	Renesas	Flash	Programmer	V3.08.01 (無償版)					-		×
ファ	イル( <u>E</u> )	デバ・	イス情報( <u>D</u> )	ヘルプ( <u>H</u> )							
操作	操作	設定	ブロック設定	フラッシュオプション	接続設定	22-2	コード	ユーザー鍵			
× .	DLM					_					
1	設定オブシ	ヨン			設定す	้จ					
	遷移先				SSD						
× .	DLM KC	ys			(at.1.1	50					
設定4フンヨノ 19もしんし Encounted SECDEG Key											
	Encrypted SCOBBG Key										
	Encrypte	1 RM	A Kev	,				_			
	Boundar	у									
	設定オプション				設定す	3					
	Code Flash Secure [KB]			0	0						
	Code Flash NSC [KB]			0	0						
	Data Flash Secure [KB]			0							
	SRAM Secure [KB]				0						
	SRAM N	50 [K	B]		0						
× :	ゼキュリフ	<b>'</b>			/ <b>T</b> + 1 +						
	設定オノソヨン			1960/	36.1						
	↑Л共11ビニ)·	マントタ	光止		いいえ						

(1) [操作]タブを開いて [スタート] ボタンを押して書き込みを開始します。

📓 Renesas Flash Programmer V3.08.01 (無值版) - 🗌 🗙							
ファイル( <u>F</u> ) デバイス情報( <u>D</u> )	ヘルプ( <u>H</u> )						
操作 提作設定 ブロック設定	フラッシュオブション 接続設定 ユニークコード ユーザー鍵						
プロジェクト情報       現在のプロジェクト       コロジェクト情報       現在のプロジェクト       マイクロコントローラ:       R7FA6M5BH3CFB       プログラムファイル       C*workspace¥sample¥Debug¥ap_ra6m_1a_sci_uart_sample_srec       ORC-32:       1D4DA07A       フラッシュ操作       (当夫)>       (当夫)>							

- 12 プログレスバーが表示され、書き込みの進捗に合わせて変化します。
- ③ プログレスバーの表示が消え、メイン画面に [正常終了] と表示されれば、正常に書き込みが終了しています。
   CPU ボードの電源を切り、その後 RFP を終了します。
   以上で FlashROM への書き込みは終了です。

🌠 R	enesas Flash	Programmer	V3.08.01 (無償版)				—		×
ファイ	ル( <u>E</u> ) デバ・	イス情報( <u>D</u> )	ヘルプ( <u>H</u> )						
操作	操作設定	ブロック設定	フラッシュオブション	接続設定	ユニークコード	ユーザー鍵			
7	ロジェクト情報 現在のプロジ: マイクロコントロ	i ェクト: ap_ra コーラ: R7F#	ı6m_rfprpj A6M5BH3CFB						
プ	ログラムファイノ	4							
	C:¥workspac	e¥sample¥De	ebug¥ap_ra6m_1a_s	ci_uart_samp	le.srec		4	<u>参照…(B</u> )	
					CRO	>-32 : 1D4DA	107A		
7	ラッシュ操作								
	消去 >> 書き	(込み >> ベリ)	ファイ >> フラッシュオフ	ジョン書き込	み>> フラッシュ	オブションベルニ	774		
			スタート(	( <u>S</u> )			正常	終了	
[Flash	h Options]								^
ベリファ1 [Flash オプション ツールが	イを実行します h Options] ン情報:DLMJ	२ <b>न</b> -५							
採作が	成功しました	0							- 6
									~
						ステー	タスとメッセ	2ージのクリ	7( <u>C</u> )

④ CPUボードをサンプルプログラム動作時のボード設定に変更し、再度電源を投入してサンプルプログラムが動作することを確認します

<AP-RA6M-0A>



JSW1:ON ボード上の SDRAM を使用する



JSW2 : SGL シングルチップモード

# <AP-RA6M-1A>



JSW1:SINGLE

#### <AP-RA8M-0A, AP-RA8D-0A>



JSW1

Ethernet を使用する場合: ON CAMERA を使用する場合: OFF



JSW2:SGL シングルチップモード

<AP-RA8T-0A>



JSW1 : SGL

シングルチップモード

再度同じ設定でファイルを書き込む際は、同じプロジェクトを開くことで®から作業を開始できます。 その他の機能など RFP の詳細につきましては、RFP のユーザーズマニュアルを参照してください。

注意:プロジェクトは、設定時に接続していた CPU に対応しています。異なる CPU の書き込みには使用できません。 例えば、AP-RA6M-1A と接続して作成した RFP のプロジェクトを使用して他の RA ファミリが実装されたボードに 書き込むことはできませんので、使用する CPU に合わせたプロジェクトを新規作成してください。

# 改定履歴



版数	更新日	改定内容
1版	2020/02/21	新規作成
2版	2020/07/22	FSP のバージョン 1.2.0 に対応
3版	2021/11/10	FSP のバージョン 3.4.0 に対応
		AP-RA6M-1A に関する記述追加
		「3.1 サンプルプログラムの動作方法」の pack ファイルインポート
		方法を更新
4版	2023/10/02	住所の更新
5版	2024/02/07	AP-RA8M-0A, AP-RA8D-0A に関する記述追加
5.1版	2024/03/01	AP-RA6M-0A/1A FSP のバージョン 4.5.0 に対応
6版	2024/12/02	AP-RA8T-0A に関する記述追加
C NA	202 1/ 12/ 02	

# 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

・RA ファミリおよび RA6M3, RA6M5, RA8M1, RA8D1, RA8T1 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。

- ・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。
- ・Renesas Flash Programmer 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。
- ・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: query@apnet.co.jp